CÉCI HPC Training 2025

UNIVERSITÉ DE NAMUR

Who I am

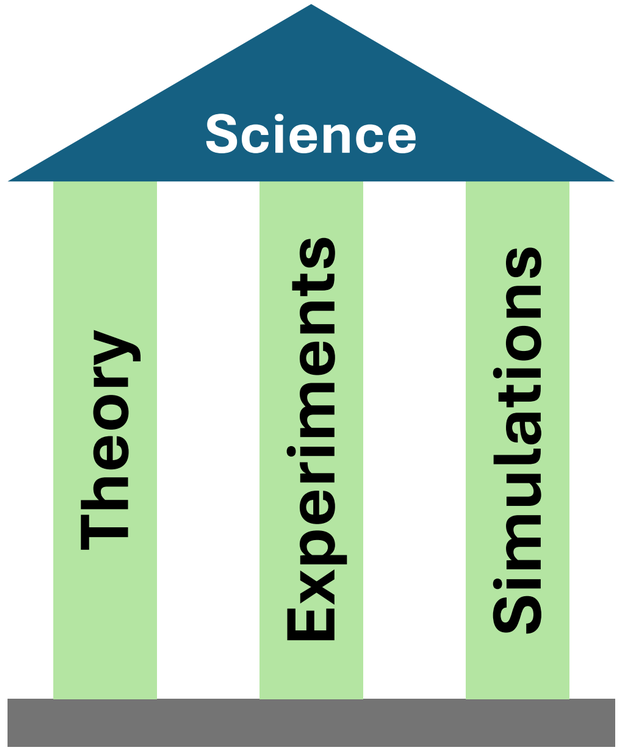
- Background: Master of CS (UNamur 2002)
- HPC tech lead since 2005
- Based in the Faculty of Sciences at UNamur

The Pillars of Science

- The three Pillars of Science
  - Pillar 1: Theory (Models)
  - Pillar 2: Experiment
  - Pillar 3: Numerical simulations

- Numerical simulations is an alternative to scale models and lab experiments
  - Faster
  - Cheaper
  - More flexible

→ Thanks to efficient numerical methods, **High Performance Computing (HPC)** can solve extremely detailed models within reasonable timeframes

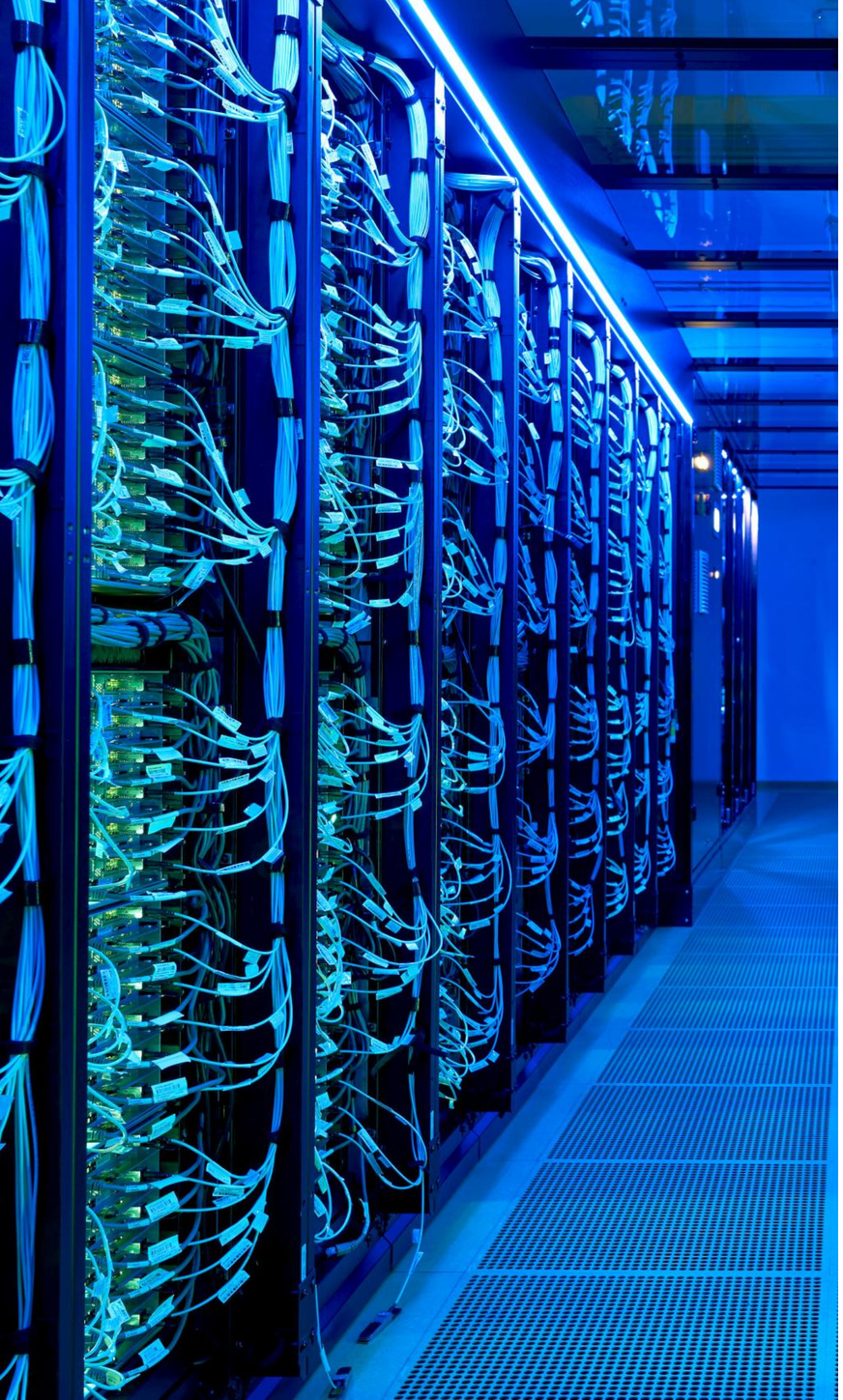


What is High Performance Computing ?

- High Performance Computing (HPC) is the method by which scientists and engineers solve complex problems using software that require high bandwidth, low latency networking and high computing capabilities

- High Performance Computing is
  - An effective algorithm and an effective implementation
  - Computational power
  - Memory
  - Storage

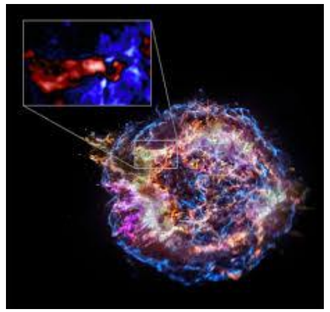
→ For HPC we use **supercomputers**



Why do you need HPC?

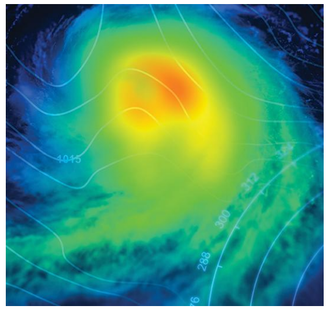
- A simulation that requires a lot of memory
- A program that could use many cores
- A program that takes very long to run
- To get access to specialized hardware
  - Accelerator (GPU, FPGA, …)
  - Large memory
  - High-speed interconnect for parallel runs

HPC use cases

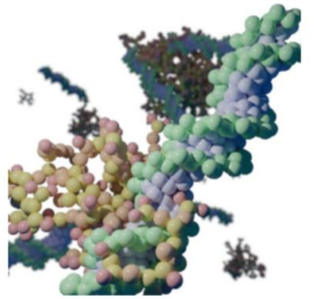


Space science

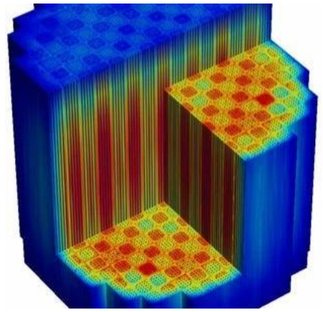
Earth science



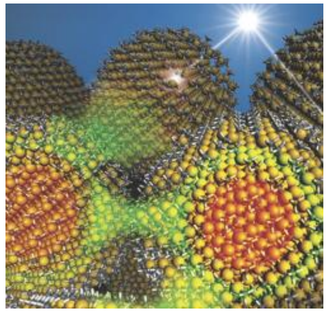
Atmospheric modeling



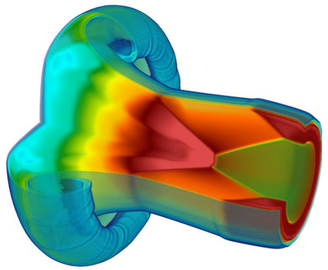
Life science



Nuclear science



Materials science



Engineering

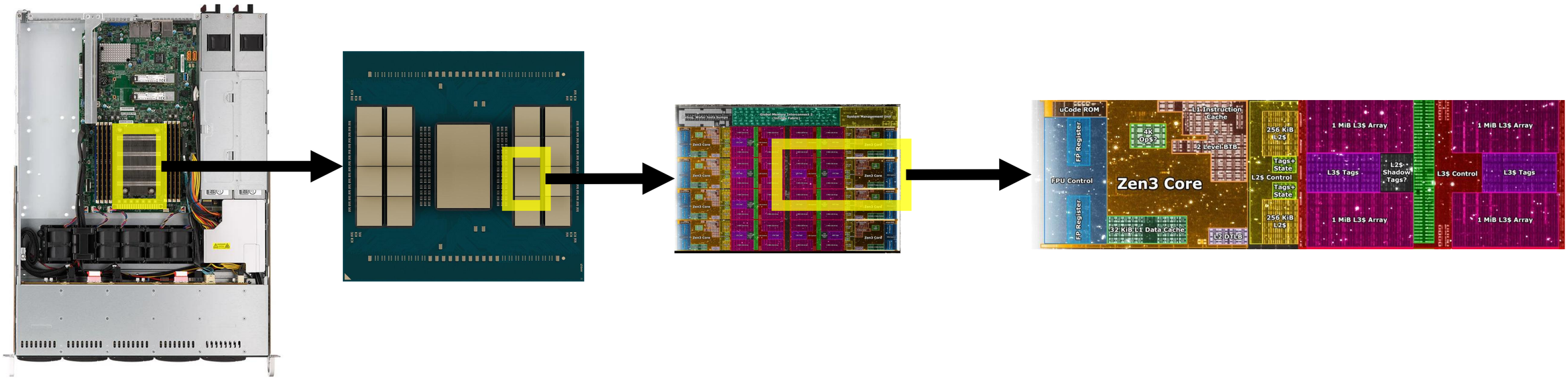
Entertainment industry

Terminology

- **Cluster:** a set of connected nodes that work together.
- **Node:** a computer (server). It usually has one or more multi-core processors.
- **CPU:** the primary computational engine of an HPC node. CPUs typically have multiple cores.
- **Core:** each independent central processing units within a CPU.
- **Memory (RAM):** volatile memory shared between all cores on a node.
- **Drive storage:** devices (spindles or SSD) that hold data persistently, even when power is off. Can be directly attached to a node or shared in storage systems.
- **Network:** connections linking together nodes and storage systems.

Multi-core processor (CPU)

- **Multi-core processor (CPU)** = single computing component with cores
- **Core** = independent processing units, which read and execute program instructions



Compute node **Chip** **Chiplet** **Core**

Mesure supercomputer power

- **FLOPS** = **FL**oating-point **O**perations **P**er **S**econd
- Primary measure of supercomputer power, other operations are irrelevant
- Supercomputer performance is measured using specialized benchmarks that test how many floating-point operations can be performed per second
  - LINPACK
  - HPCG
- Actual performance can vary based on hardware, software optimization and the type of calculation

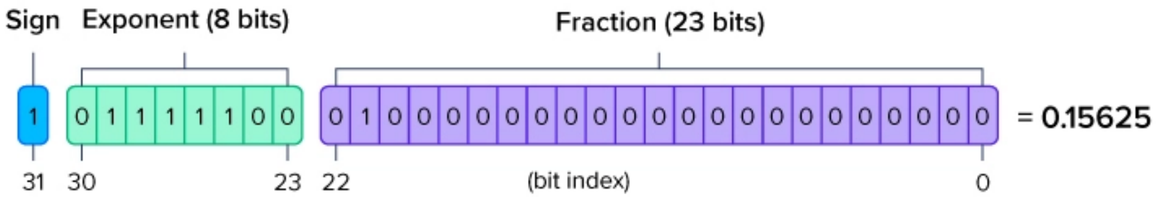
From TeraFLOPS to ExaFLOPS

- **TeraFLOPS** = one trillion (10^{12}) floating-point operations per second
- **PetaFLOPS** = one quadrillion (10^{15}) floating-point operations per second
- **ExaFLOPS** = one quintillion (10^{18}) floating-point operations per second

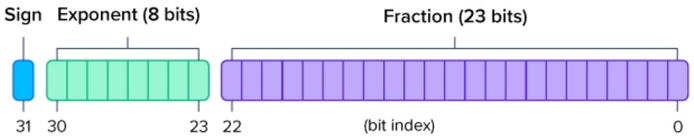
- iPhone 16 Pro: 2.3 TeraFLOPS
- NVIDIA RTX 4090: 82 TeraFLOPS
- Tianhe-3 Supercomputer: 1.7 ExaFLOPS

Floating Point number

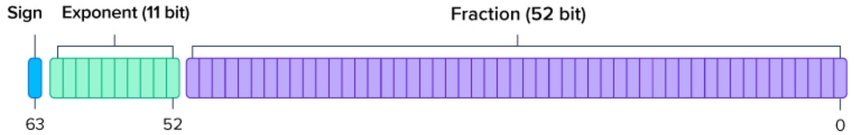
- Representation of a number in binary



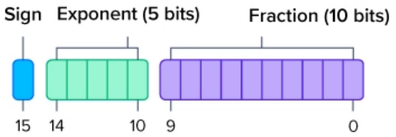
- FP32 (Single Precision):



- FP64 (Double Precision):



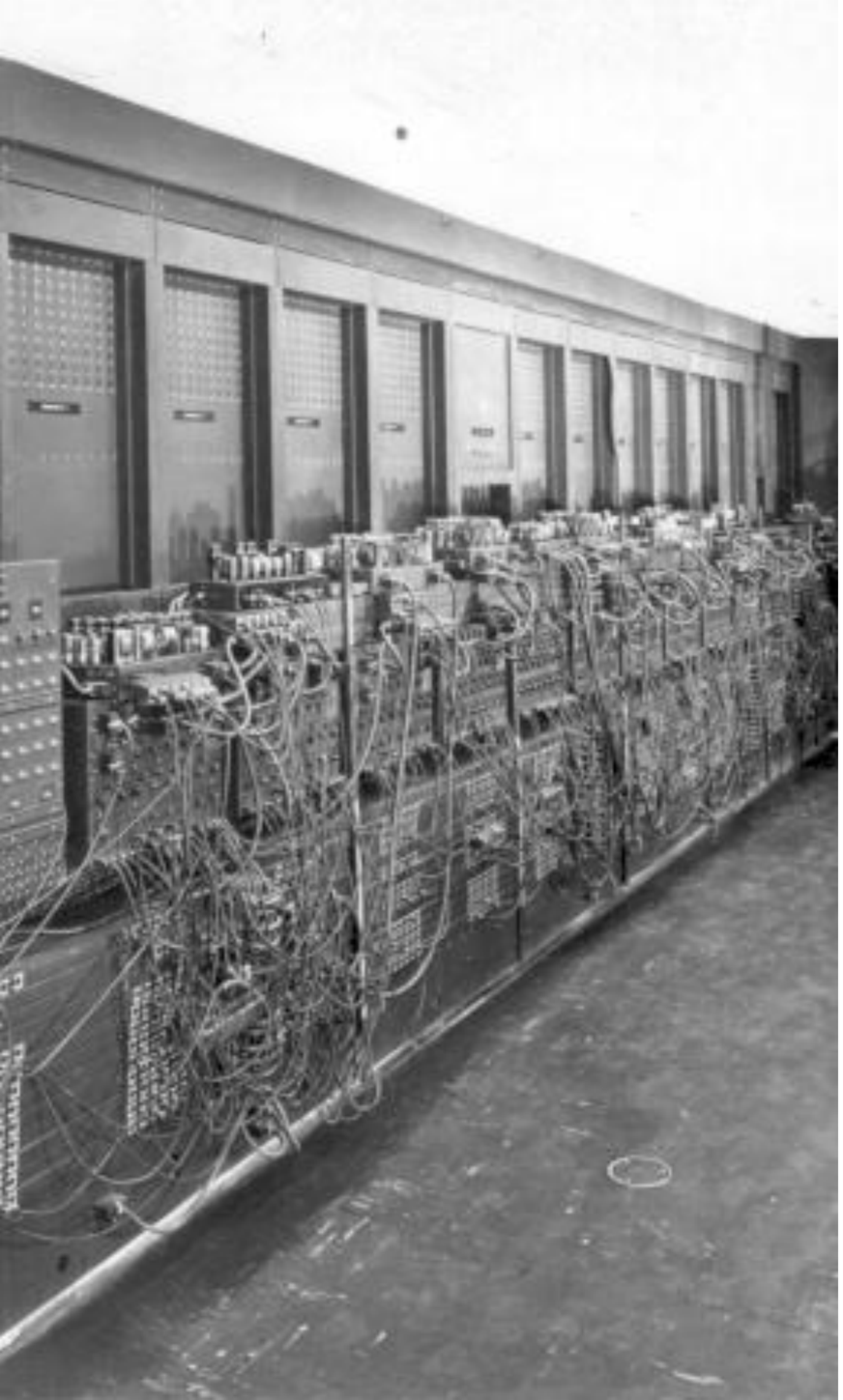
- FP16 (Half Precision):



- Benchmarks for HPC use Single Precision (32-bit) or FP32

Until the 60s: the human computers

- Before the supercomputer, “human computers” were often utilized to solve mathematical problems and process data
- Hidden Figures (2016) movie
- Performance: about 0.05 FLOPS per human



1940s - 1960s: the first supercomputers

- Driven by military needs
  - Artillery tables
  - Code breaking
  - Aircraft, submarine, etc. design
  - Nuclear weapon design

- ENIAC, 1943
  - “Electronic Numerical Integrator and Computer”
  - First stored-program electronic computer
  - University of Pennsylvania
  - In operation until 1955
  - 50 FLOPS

1975 - 1990: the Cray era

- Vector Processors
  - Designed for operations on data arrays rather than single elements
- Shared memory multiprocessing
  - Small number (up to 8) processors with access to the same memory space
  - Issue with memory contention
- Supercomputers was custom-built systems and very expensive
- Cray 1A (1976)
  - up to 8 MB RAM
  - up to $8 million price tag
  - 5-ton with Freon cooling system
  - 160 MegaFLOPS

1990 - 2010: the cluster era

- The age of effective parallel computers begins
  - Overcome the memory contention
  - Distributed memory computers

- Intel iPSC/1 “Hypercube” (1985)
  - up to 128 nodes
  - 80286 processor + 80287 math co-processor
  - 512K RAM
  - 2 MegaFLOPS
  - $500,000 price tag (128 nodes version)

1990 - 2010: the cluster era

- "Beowulf" clusters
  - Named from an old English poem
  - Build using commodity "off-the-shelf" components

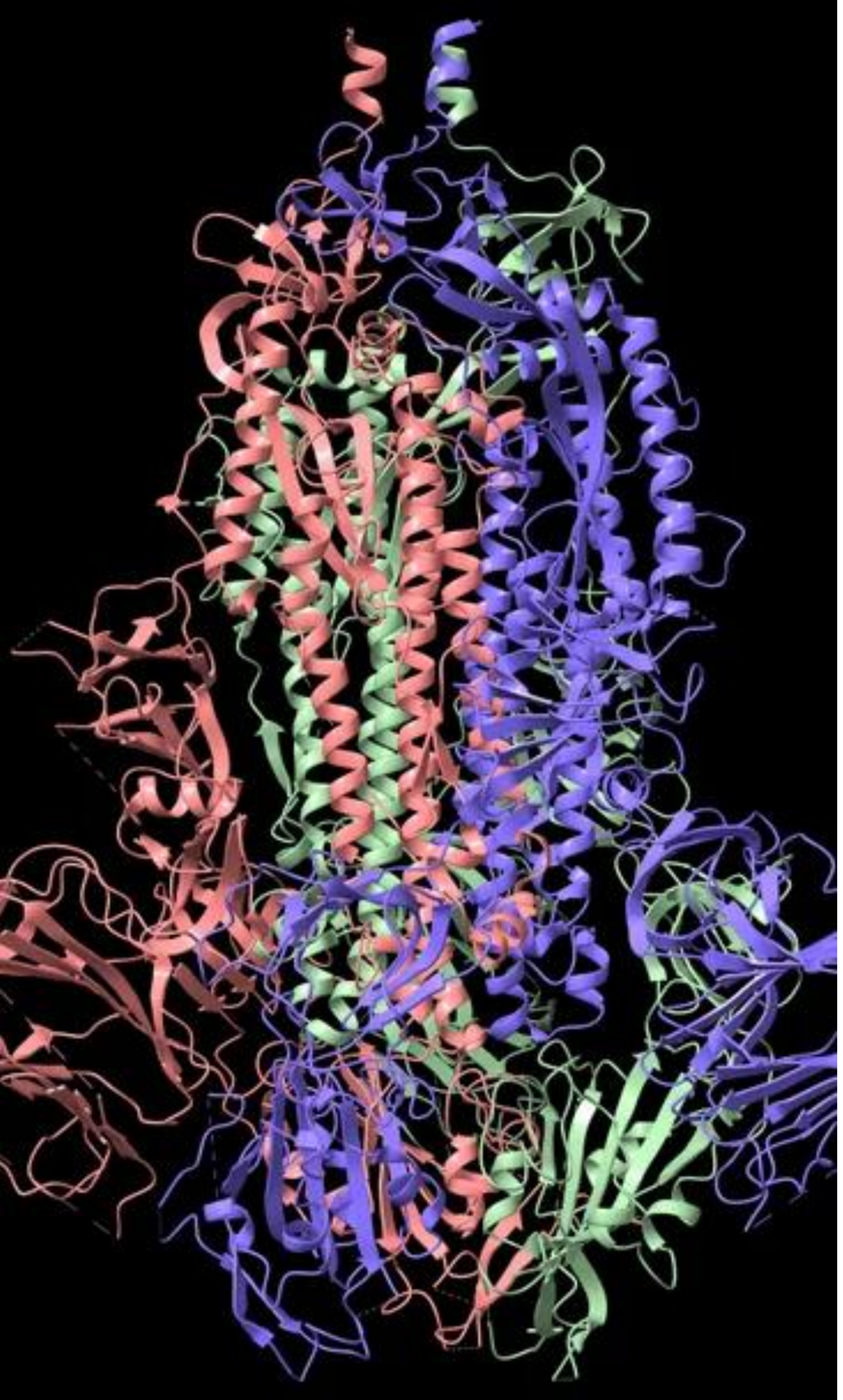
- First Beowulf computer cluster at NASA Goddard Space Flight Center (1994)
  - 16x Intel 486DX PCs
  - 16 MB RAM
  - 1 GigaFLOPS
  - $50,000 price tag

2000 – 2020: the GPU and hybrid era

- Accelerator–based architectures
  - CPU and accelerators coprocessing
  - Offload accelerator-intensive workloads
- Very large number of nodes each with:
  - Many-core processors
  - RAM shared by all cores
  - Accelerators (mainly GPUs)
- Computation speed often limited by data-movement rate
- Hybrid programming on multicore architectures is more complex

2000 – 2020: the GPU and hybrid era

- Roadrunner (Los Alamos – 2018)
  - 122,400 cores
  - $100 million cost
  - First supercomputer to reach one PetaFLOPS
  - TOP500 #1 from June 2008 to June 2009
  - 2 MW power



2020 – now: the age of AI

- Rise of AI/ML lead to a surging demand for computer power

- New classes of workloads appeared
  - Large neural networks, huge training/inference data, massive model databases
  - Require not just faster hardware, but new software frameworks, optimized data access, and integration of HPC + AI workflows
  - Example: AlphaFold2

TOP500.org

- Maintains a list of fastest supercomputers in the world
- Based on the LINPACK benchmark program
- A new list comes out every June and November

- El Capitan (Livermore, USA)
  - #1 (June 2025)
  - 11 millions cores
  - 1.7 ExaFLOPS
  - $600 million cost
  - 30 MW power

Parallel computing

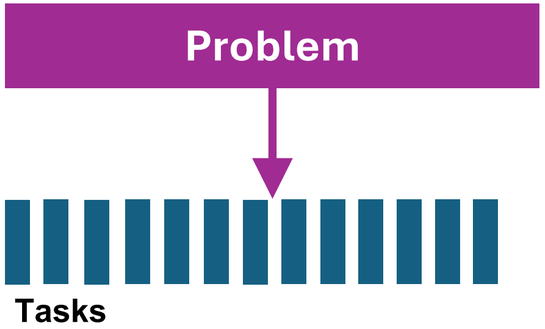
- Sequential systems are painfully slow
  - Calculations make take days, weeks, years …
  - More cores can get job done faster
- The key of HPC is **parallelism**
- The concept is simple:

Parallelism = use multiple cores for a single problem

From problem to solution

- Analyze the problem:
  - split problem in independent tasks
  - find solutions for each task

- Problems have certain amount of potential parallelism.



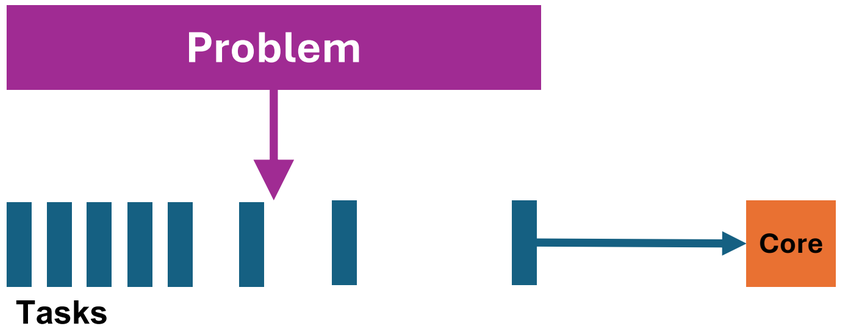
Sequential execution

- Tasks are not all independent

OR

- No parallel resources are available
  - Single core execution
  - Memory constraint
  - Limited communication bandwidth

→ Sequential execution



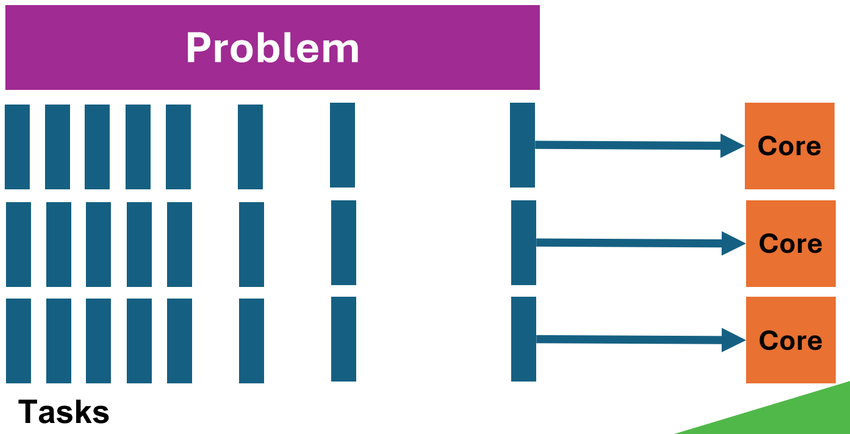
Embarrassingly parallel

- Tasks independent, no order

AND

- Parallel resources are available
  - A lot of cores
  - Large memory
  - Enough communication bandwidth

→ Parallel execution



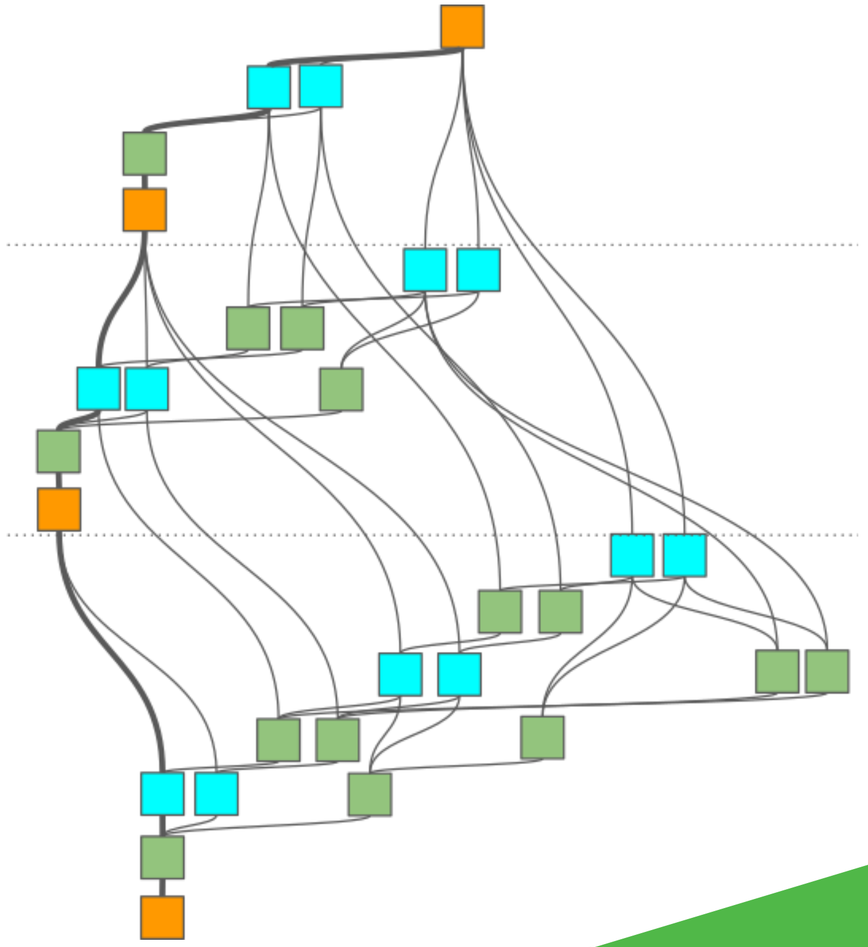
In the real world

- Tasks are not all independent

AND

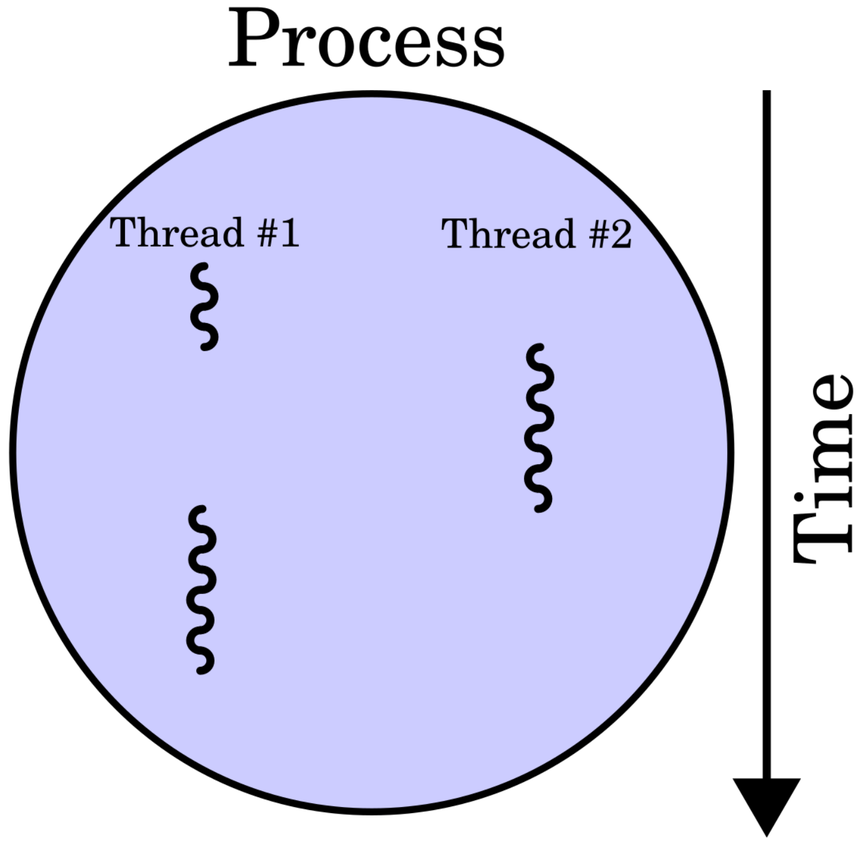
- Resources are limited
  - Multi/many cores machine
  - Limited memory
  - Limited communication bandwidth

→Look for a good compromise



Process and thread

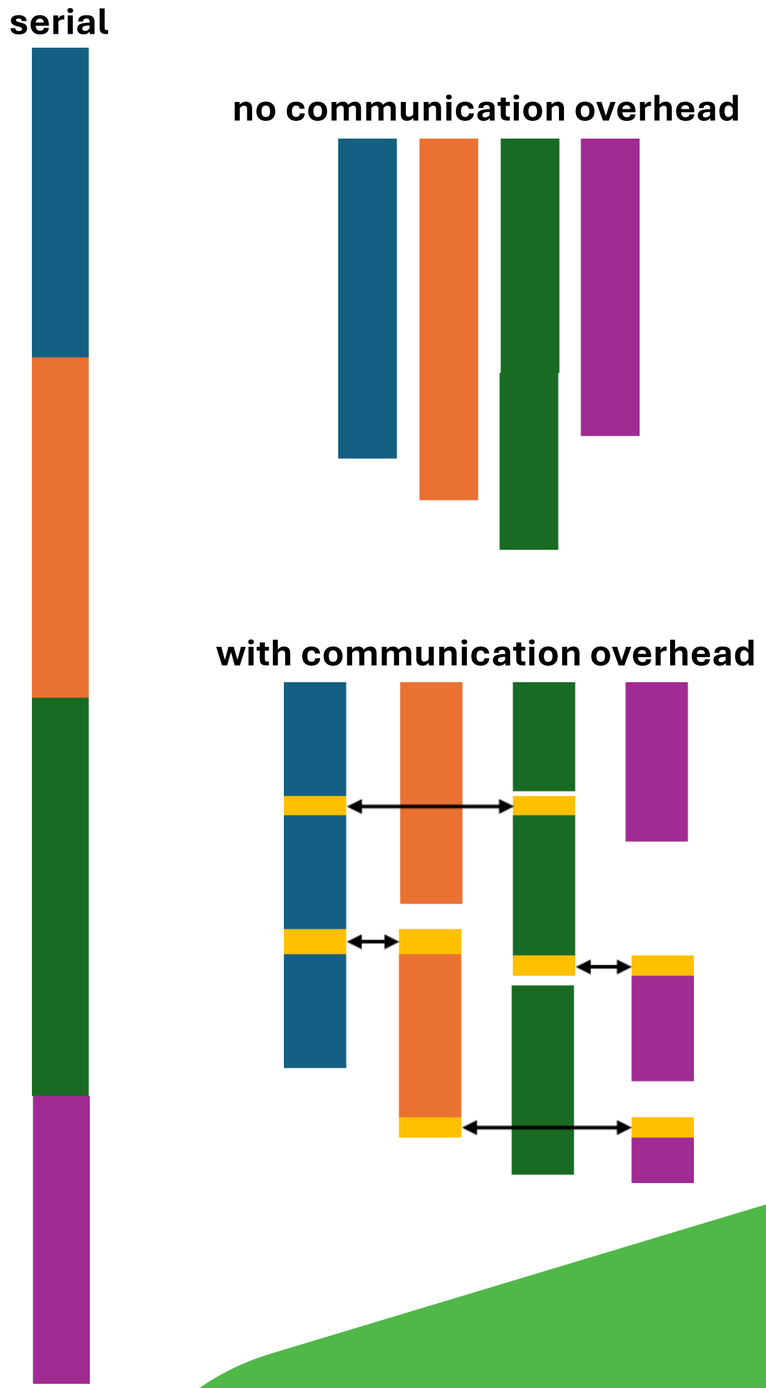
- Software concepts
- A thread is essentially just **an execution of a sequence of instructions**
- A process is a **collection of one or more threads**
- **A parallel program will execute threads in parallel**
- In HPC, one thread is executed on one core



Parallel Overhead

- Parallelization comes with some **costs:**
  - Communication between the threads
  - Synchronization
  - Resources management

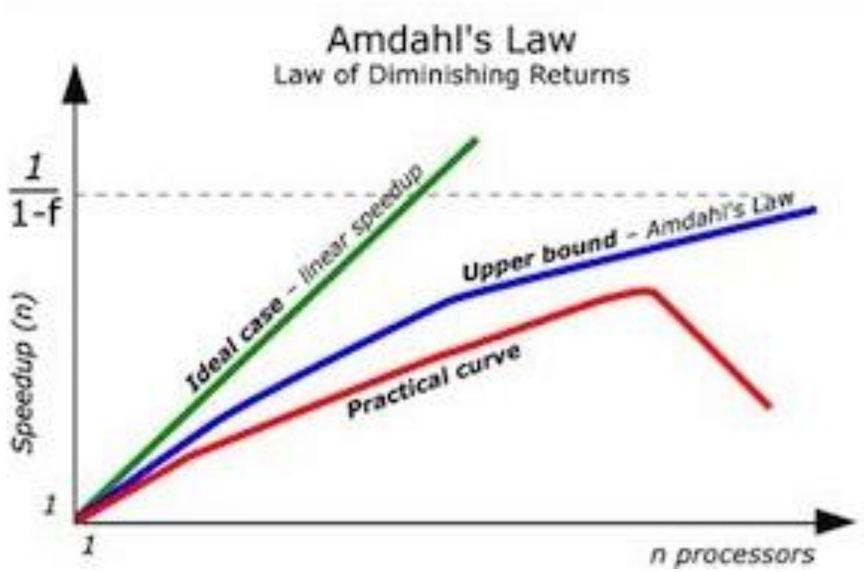
→ To get efficient parallelization, you need to **minimize the overhead**



Amdahl's Law

- Use to estimate the parallel program performance speedup when using several cores

- In the real world, speedup is limited by:
  - the serial part of the program, whose time remains constant
  - the hardware factors (communication bandwidth, memory speed, …)
  - the parallel overhead



$$\text{Speedup} = \frac{\text{Sequential execution time}}{\text{Parallel execution time}}$$

Implicit Parallelism

- Done by the compiler
- Automatically detects potential parallelism in the software

- Pro
  - Frees the developper from the details of parallel execution
  - Flexible solution

- Cons
  - Not always efficient

Explicit Parallelism

- Done by the developer
- Annotate the tasks for parallel execution or explicitly assign tasks to cores
- Control the execution and synchronization points

- Pro
  - Can be very efficient
- Cons
  - Developper are responsible for all details
  - Developper must have deep knowledge of the computer architecture to achieve maximum performance.

Parallel Programming Models

- Two dominant parallel explicit programming models:

Shared memory

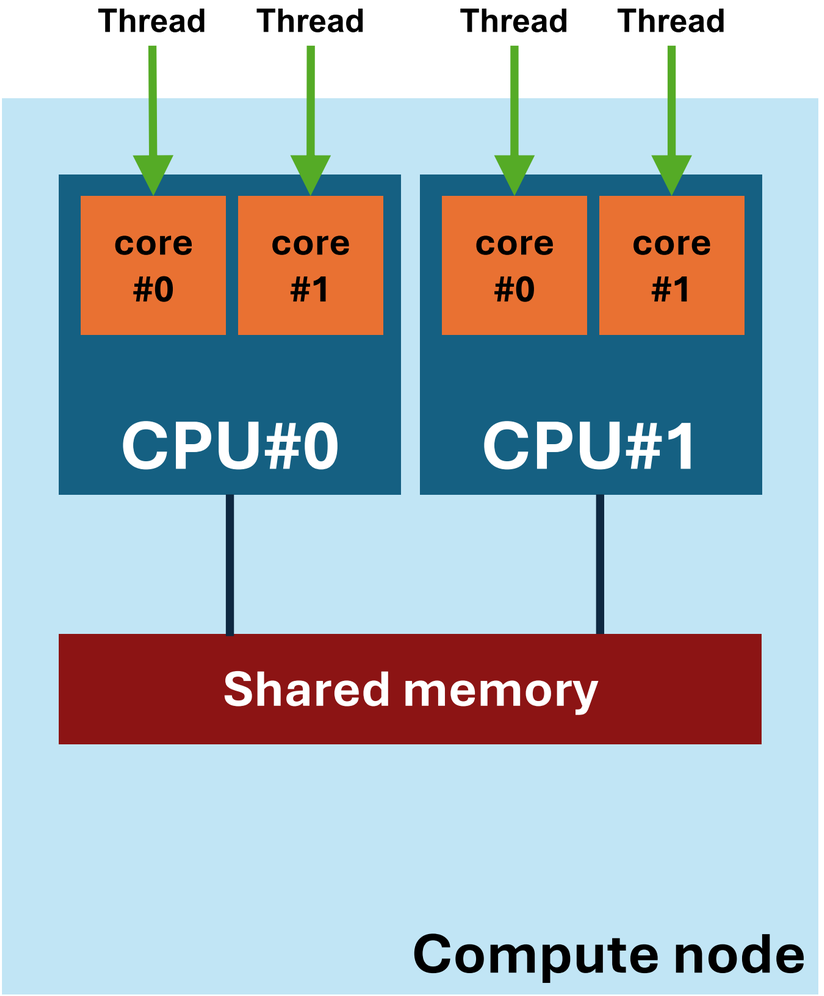
Distributed memory (Message Passing)

Shared memory

- Multiple threads operate independently but share the same memory
  - Single address space
  - No need to transfer data from one thread to another
  - Changes in a memory location effected by one thread is visible to all other threads
  - OpenMP

→ execution one a single node!

- Pro
  - Data sharing between threads is fast
  - "Easy" to use
- Con
  - Scalability is not good
  - Programmer is responsible for specifying synchronization

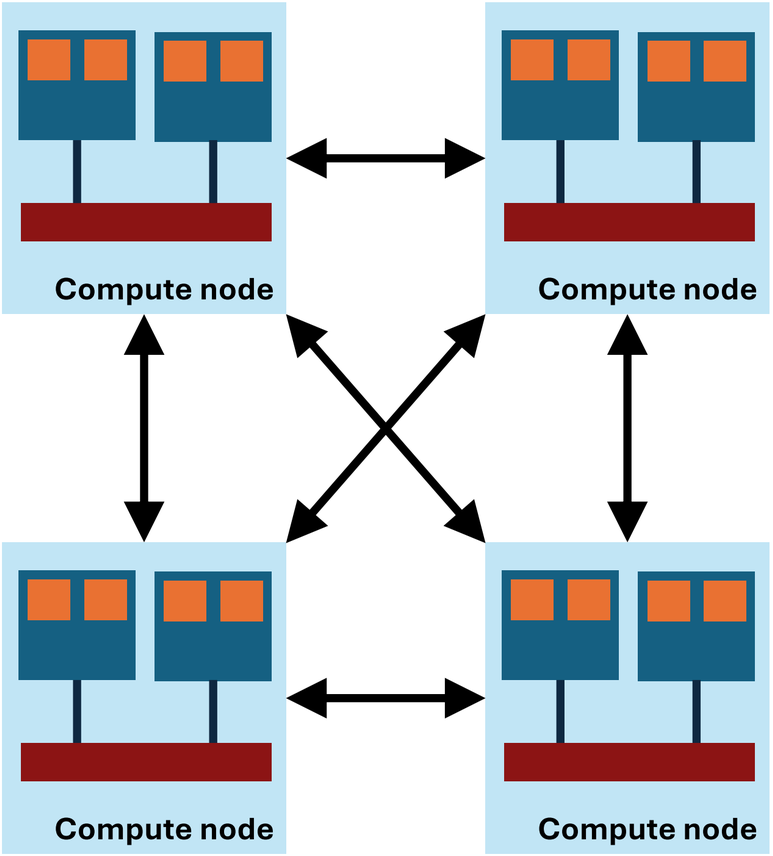


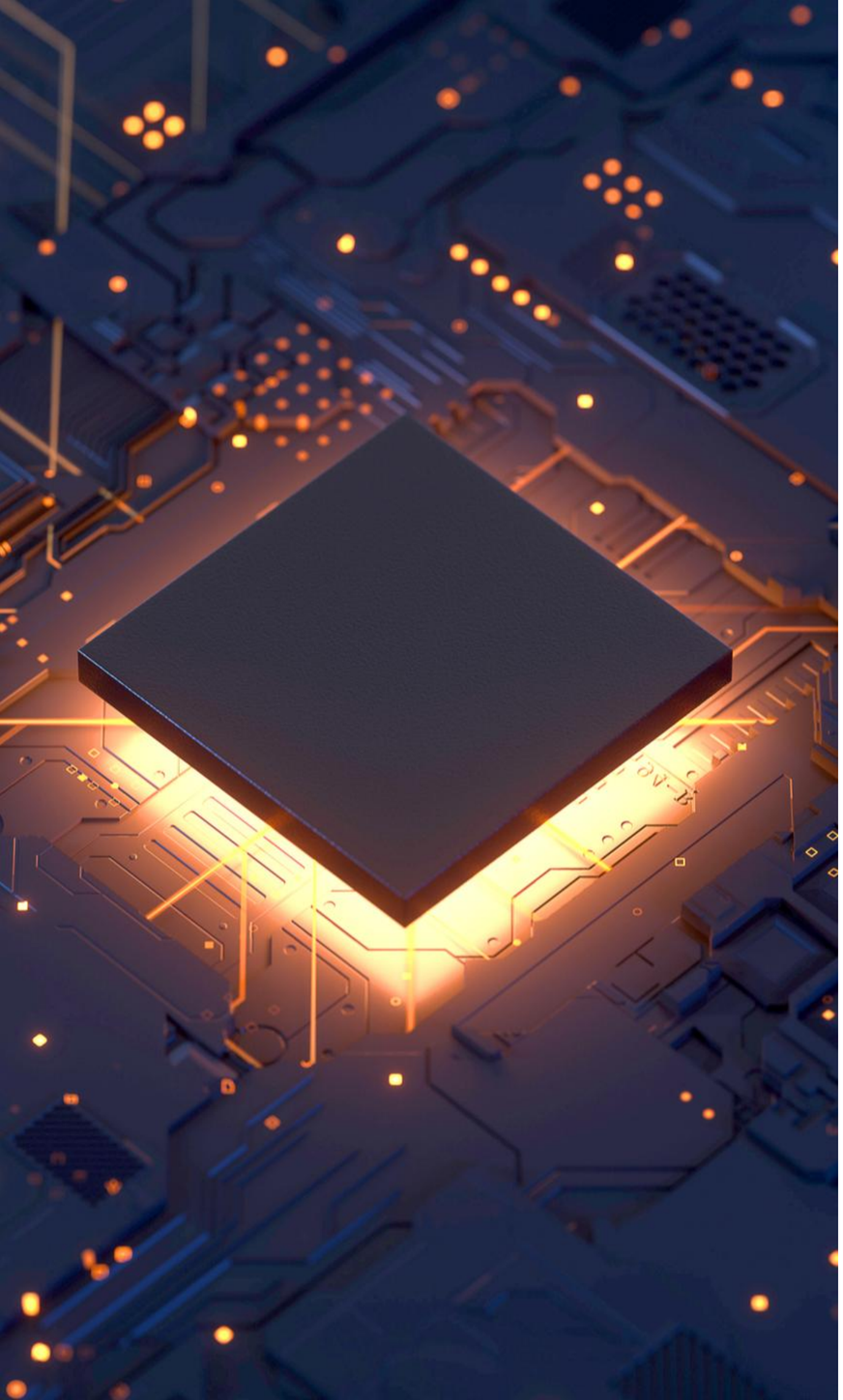
Distributed memory : Message Passing

- All data is private to the processes
  - Separate address spaces
  - Data is shared by exchanging messages
  - Data transfer requires cooperative operations by each process
  - MPI is the "de facto" industry standard for message passing

→ *can* run on multiple nodes

- Pro
  - Good scalability
- Con
  - Complex
  - Convert a program to this model require a complete rewrite





Accelerated computing

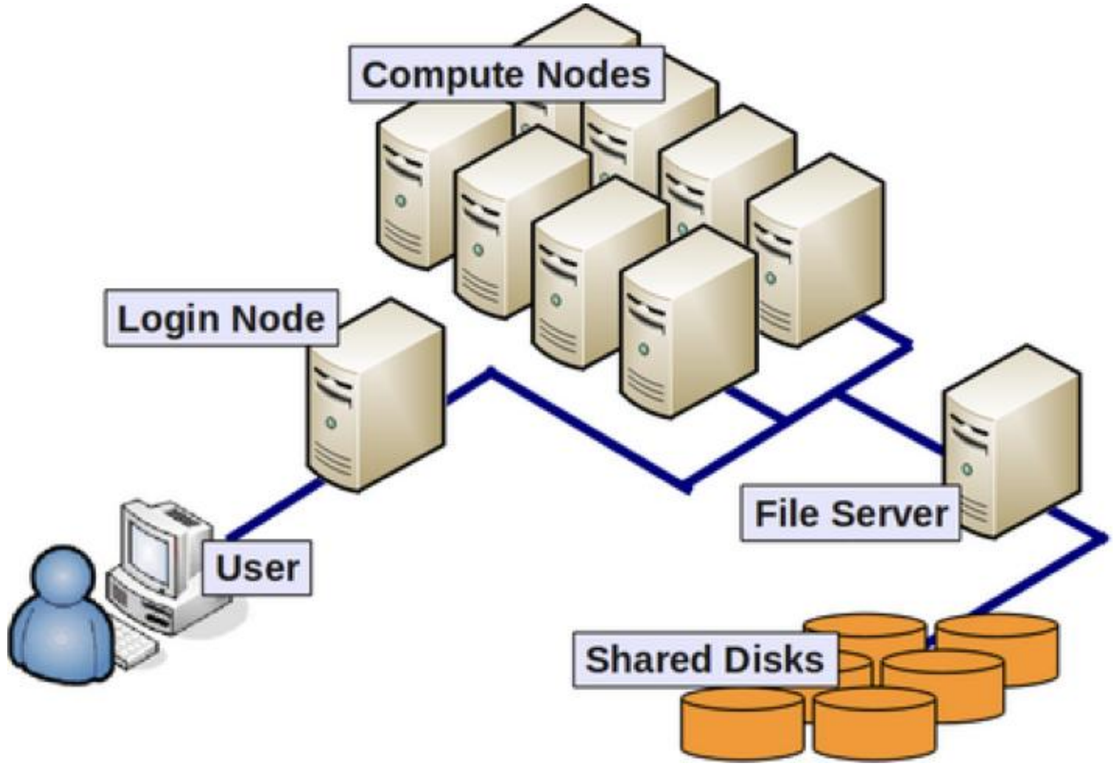
- Use of specially designed hardware and software to speed computing tasks
  - Up to several order of magnitude speedup in some cases
- Mostly GPUs (graphics processing units) but also
  - Application-Specific Integrated Circuits (ASICs)
  - Field Programmable Gate Arrays (FPGAs)
- Require the use of dedicated software stacks
- They can make your code run faster but can be hard to program

Parallel programming in practice

- Parallel programming needs **compilers and toolboxes**
- Many scientific software comes now in a parallel version
- Not so easy to set-up
- On HPC cluster, the job scheduler take care of asking resources correctly

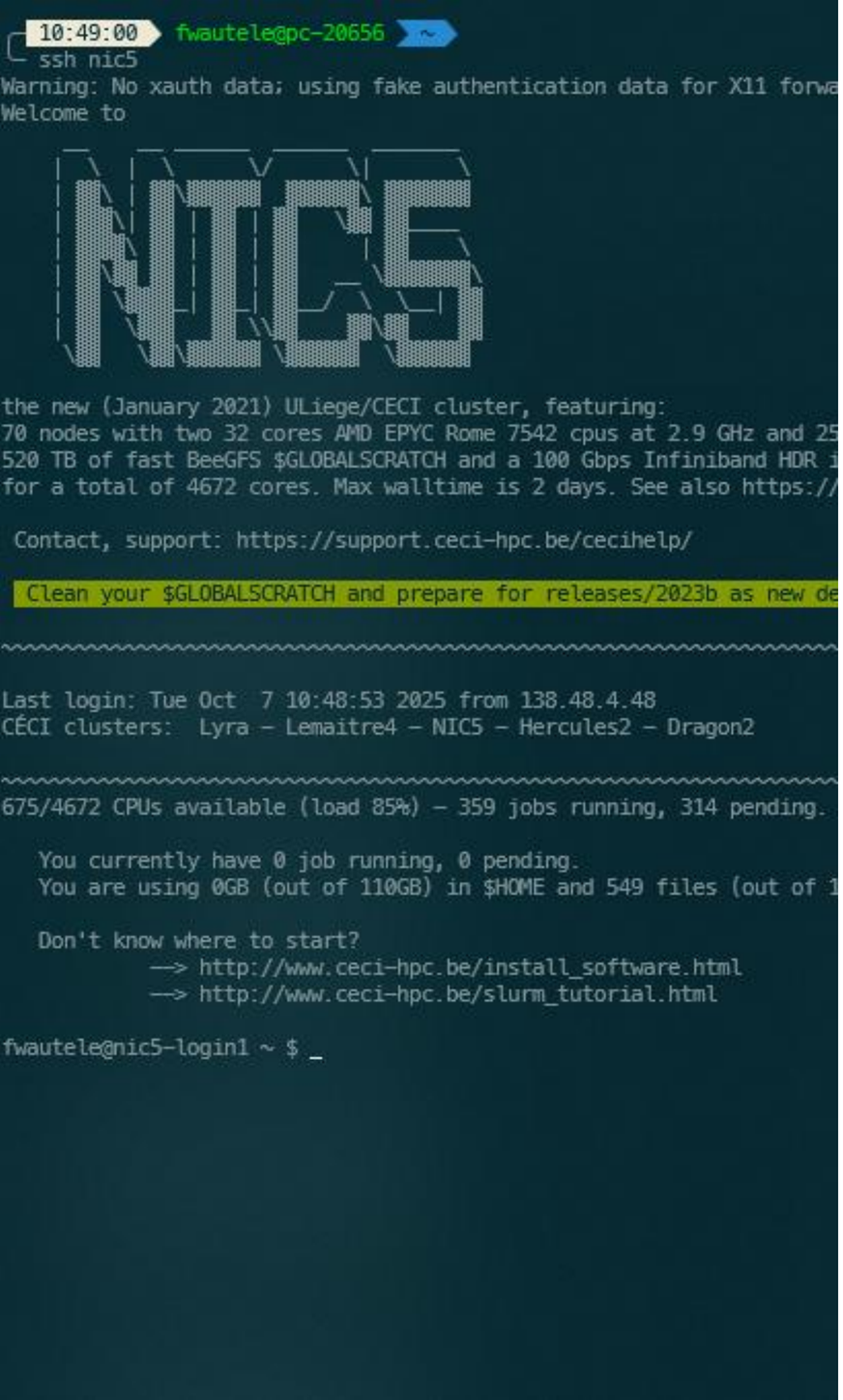
HPC cluster

- One or several login nodes
- Many compute nodes
- One or several shared storage systems
- A high-performance network
- A software stack
  - Scientific software
  - Compilers and libraries



HPC cluster

- HPC cluster is like to your personal computer
  - Hardware is very similar to personal computer
  - Can run the same software than your personal computer
- But not is a multiusers environment
  - You will need to connect and log in
  - You share storage space
  - You share computational resources
- Policies in place to ensure the fair sharing of resources
  - All program execution must be submitted to the job scheduler



Accessing the HPC

- You will need to log in using your CÉCI credentials
- You cannot access local data
  - You must move your data to/from your personal computer
- The application are submitted to a **job scheduler**
  - Batch jobs
- Control the system via **text commands**
  - Old-fashioned
  - You can do (almost) everything from the command line
  - It is the way to submit jobs for batch execution

Login node

- Can be use to:
  - Develop your code
  - Manage your files
  - Check your storage usage
  - Manage your jobs
  - To pre-/post-process your data/jobs
  - To visualize your data

- Login nodes are shared resources

→**NEVER** run programs on the login node

Computation heavy job MUST be submitted to the scheduler and not run directly.

Command line

- HPC clusters usually don't have graphical desktop environments installed
- Users' interaction with the HPC clusters by typing commands in a terminal
- A software (”shell”) interprets these commands and communicates with the operating system
- Powerful and flexible
- A bit of practice is needed

Concept of “modules”

- Large software stack are often installed on HPC cluster
- Different users may need different tools, or different versions of the same software
- To avoid conflicts between the different software or different versions of the same software

- Modular approach:
  - Software packages/tools are available as “modules”
  - Environment variables are set accordingly
  - Conflicts are avoided

Batch processing

- Unattended execution is the typical HPC usage

- The job scheduler is a program that manages unattended program execution (batch processing)
  - checks the resources available
  - priorities the jobs and control the execution order
  - schedules the execution of job on compute node when requested resources are available

→ prepare your program so that it can run without any intervention

Job scheduler

- You will need to specifies the resources needed
  - # of nodes/cores
  - amount of memory
  - expected run time (wall-clock time)
  - GPUs
  - …

- This is done in a **job script**

- The job scheduler relies on these information

→BE ACCURATE! a misuse of the resources may affect all the users

Job script

```
#!/bin/bash
#SBATCH --job-name=demo
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --partition=gpu
#SBATCH --gres="gpu:1"

ml purge
ml LIST_THE_MODULES_YOU_NEED_HERE

srun aprogram < ~/mydata001
```

Job parameters + Needed resources

Modules

Job steps

Parallel jobs

- You need to specify the number of nodes and cores you need
- A shared memory program (e.g. openMP) requires cores on **one single node**
- A distributed memory program (e.g. MPI) *can* run **on multiple nodes**

Submit your job

- From the directory where you have your files, type:

```
sbatch run.sh
```

- Where run.sh is your job file
- Then, the scheduler reads the options (#SBATCH), and assigns the job to the queue
- Your job can start when the requested resources are available

Summary

- Prepare:
  - Prepare all the program needs for the execution
  - Estimate the resources you need
  - Write the job script
- Run:
  - Submit your job script
  - Check your job status
- Wait patiently for the results

HPC ecosystem

Tier 0
“Super-supercomputers”

Tier 1
National/regional centers

Tier 2
Local centers

HPC ecosystem

Tier 0

“Super-supercomputers”

EuroHPC Joint Undertaking

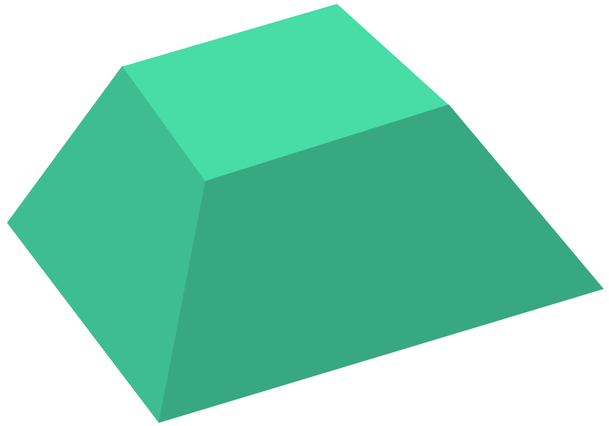
PRACE

LUMI

- 360,000+ cores
- up to 4TB memory
- 10,000+ GPU
- 100+ PB storage
- #9 Top500
- 379 PetaFLOPS

EuroHPC Joint Undertaking

HPC ecosystem in Europe



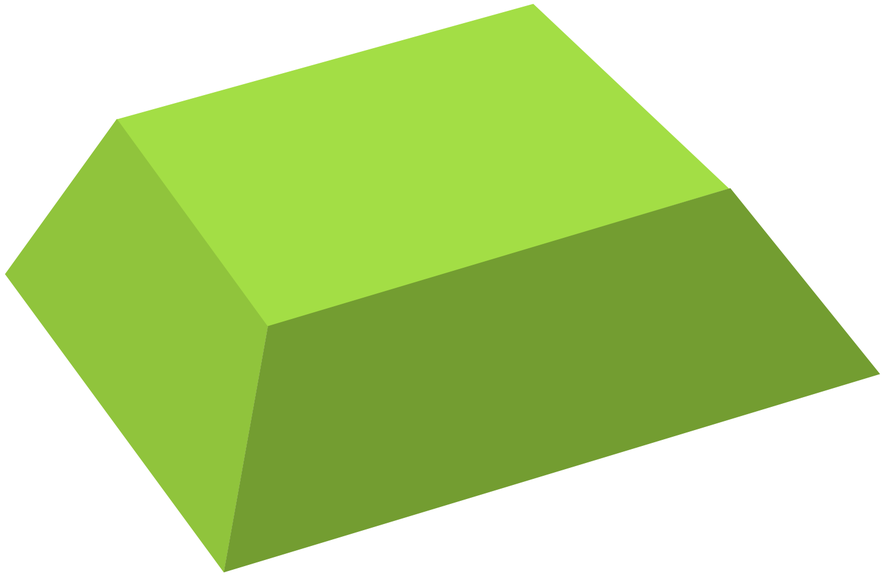
Tier 1
National/regional centers

Cenaero

Lucia

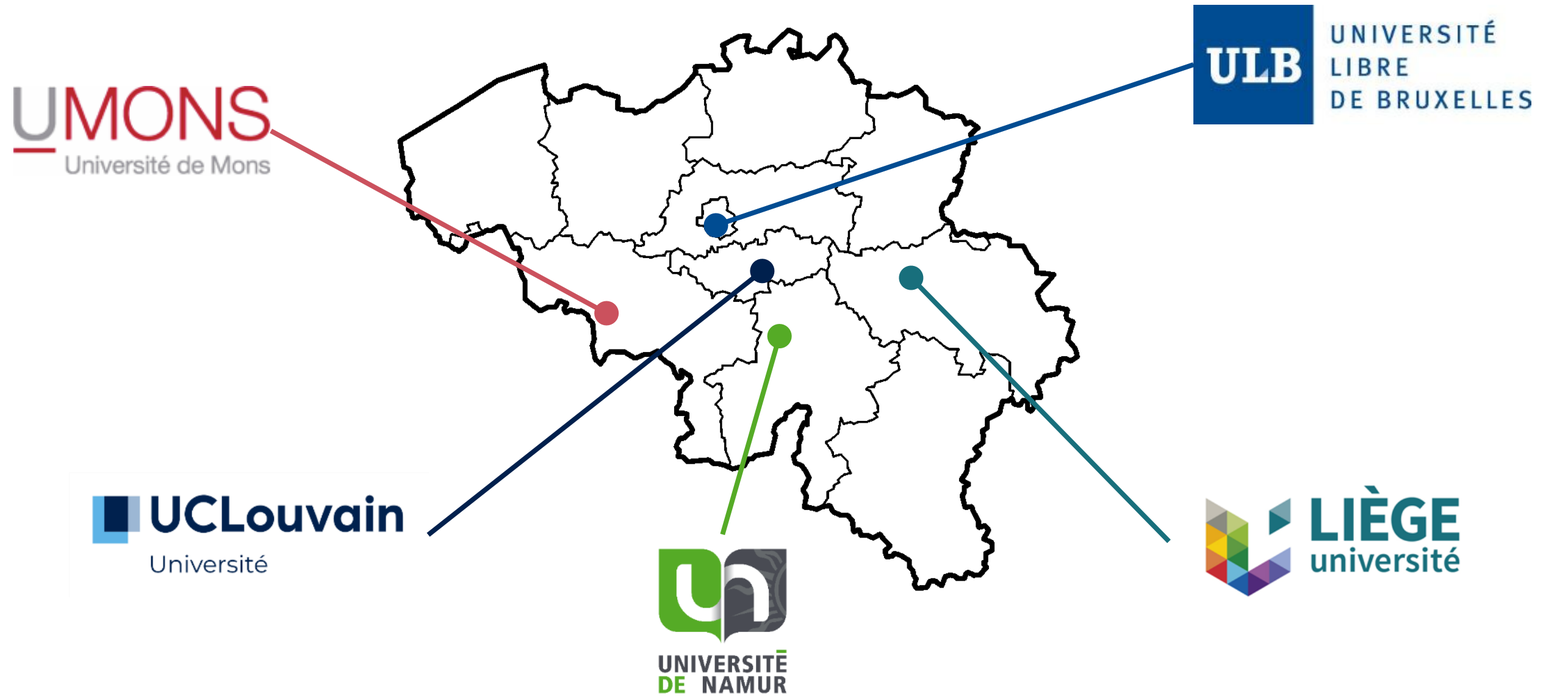
- 38,000 cores
- up to 4 TB memory
- 200+ GPU
- 7+ TB storage
- 4 PetaFLOPS

HPC ecosystem in Europe

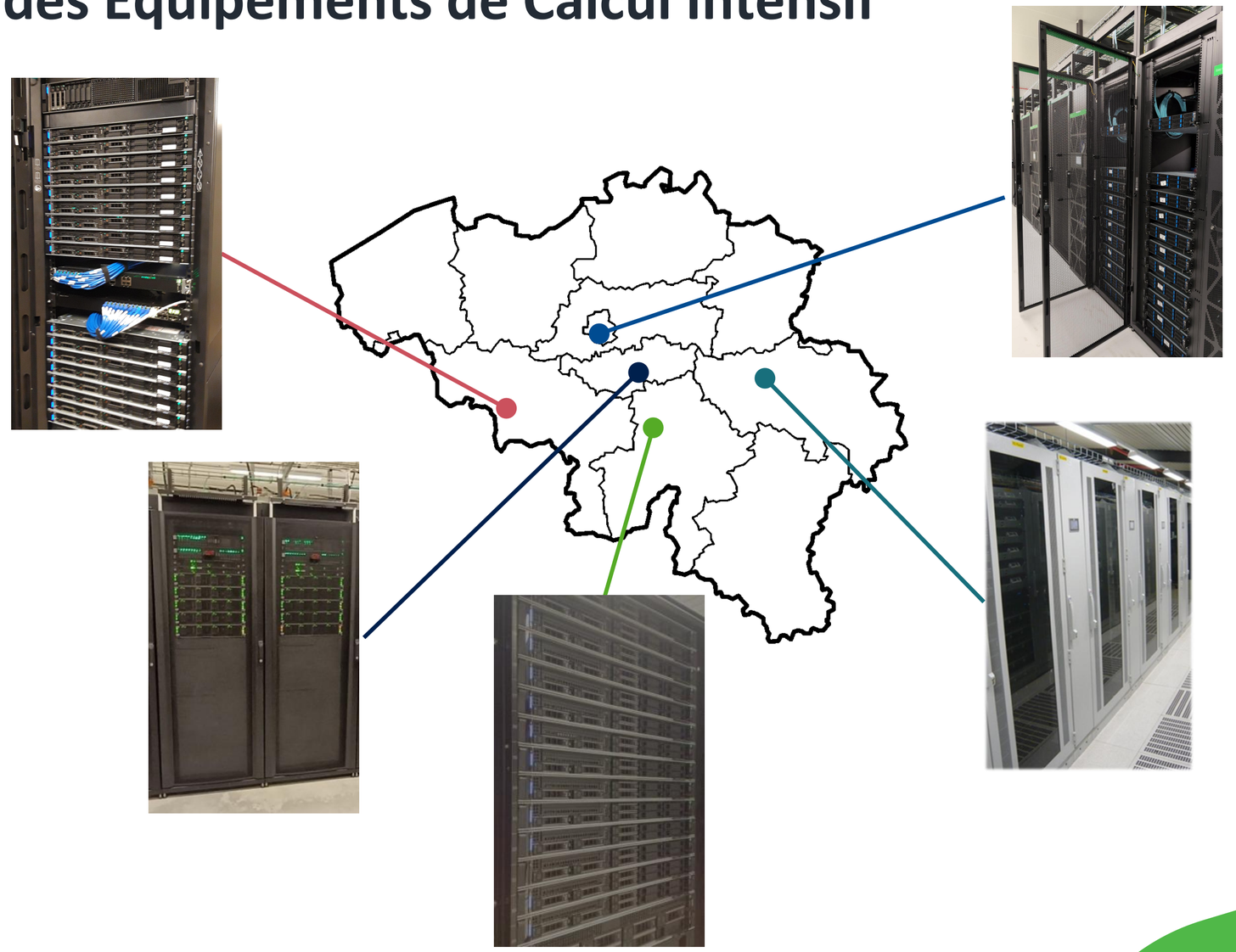


Tier 2
Local centers

Consortium des Équipements de Calcul Intensif



Consortium des Équipements de Calcul Intensif



NIC5 @ LIÈGE université

- 4672 cores
- 1 TB memory
- 250 TB storage

- Suitable for:
  - Parallel jobs
  - I/O intensive jobs
  - Short duration jobs

Lemaitre 4 @ UCLouvain Université

- 5120 cores
- 768 memory max
- 144 TB storage

- Suitable for
  - Parallel jobs
  - I/O intensive jobs
  - Short duration jobs

Hercules 2 @ UNIVERSITÉ DE NAMUR

- 1152 cores
- 2 TB memory
- 250 TB storage
- 16 GPU

- Suitable for:
  - Single node job only
  - High memory jobs
  - Long duration jobs (15 days)
  - Single precision GPU job

Dragon 2 @UMONS Université de Mons

- 592 cores
- 384 GB memory max
- 100 TB storage
- 4 GPU

- Suitable for:
  - Single node job only
  - Long duration jobs (21 days)
  - Double precision GPU jobs

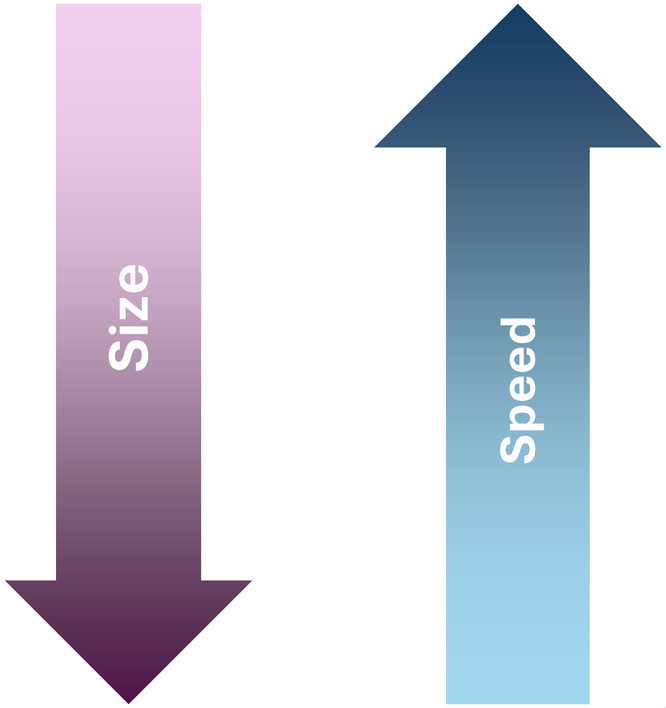
Lyra @ ULB UNIVERSITÉ LIBRE DE BRUXELLES

- For data-intensive jobs
  - BigData
  - Machine Learning
  - Deep Learning
  - AI
  - High Performance Data Analytics
- 46 NVIDIA GPUs
- Expected Q4 2024

Storage

- HPC cluster offer several types of storage space with varying speeds and sizes.

- **Scratch Space**
  - High speed storage
  - Ephemeral
  - Can be node-attached or shared
- **Homedir**
  - Your data, shared among a cluster
- **CÉCI Common Storage**
  - Shared among all CÉCI clusters
  - Can be used to share files between users using “CÉCI projects”



CÉCI Support and Services

- Basic support
  - Helpdesk (https://support.ceci-hpc.be/cecihelp/)
  - Monitoring and reporting (https://www.ceci-hpc.be/status.html)

- Application support
  - Installation
  - Workflows
  - Best practices

- Training
  - Documentation
  - Youtube channel
  - Scheduled training

CÉCI training

- Learning how to use HPC infrastructure
- Learning how to program on HPC cluster
- Going Parallel
- Scripting Language for HPC
- Data on HPC and Data Management

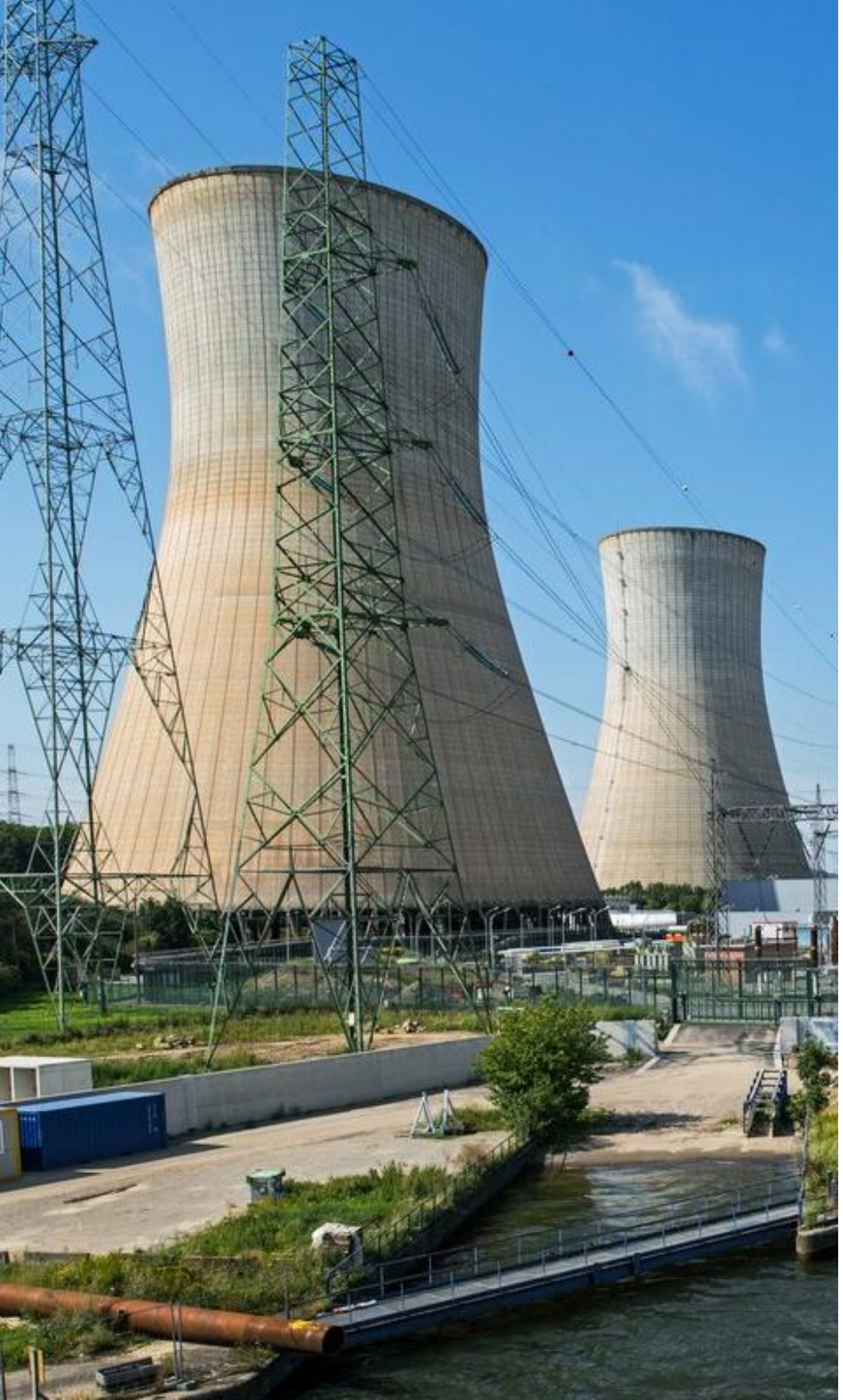
Become a CÉCI user in two steps (almost)

1

Go to http://www.ceci-hpc.be

2

Click here



Supercomputers use a lot of power!

- The #1 TOP500 supercomputer uses enough energy to power 24,000 U.S. homes!
- And this is nothing compared to the number of other personal computers, laptops, tablets, mobile phones, and servers that are performing computations 24/7…
- Lawrence Berkeley National Laboratory projected that datacenters could consume up to 12% of US electricity by 2028

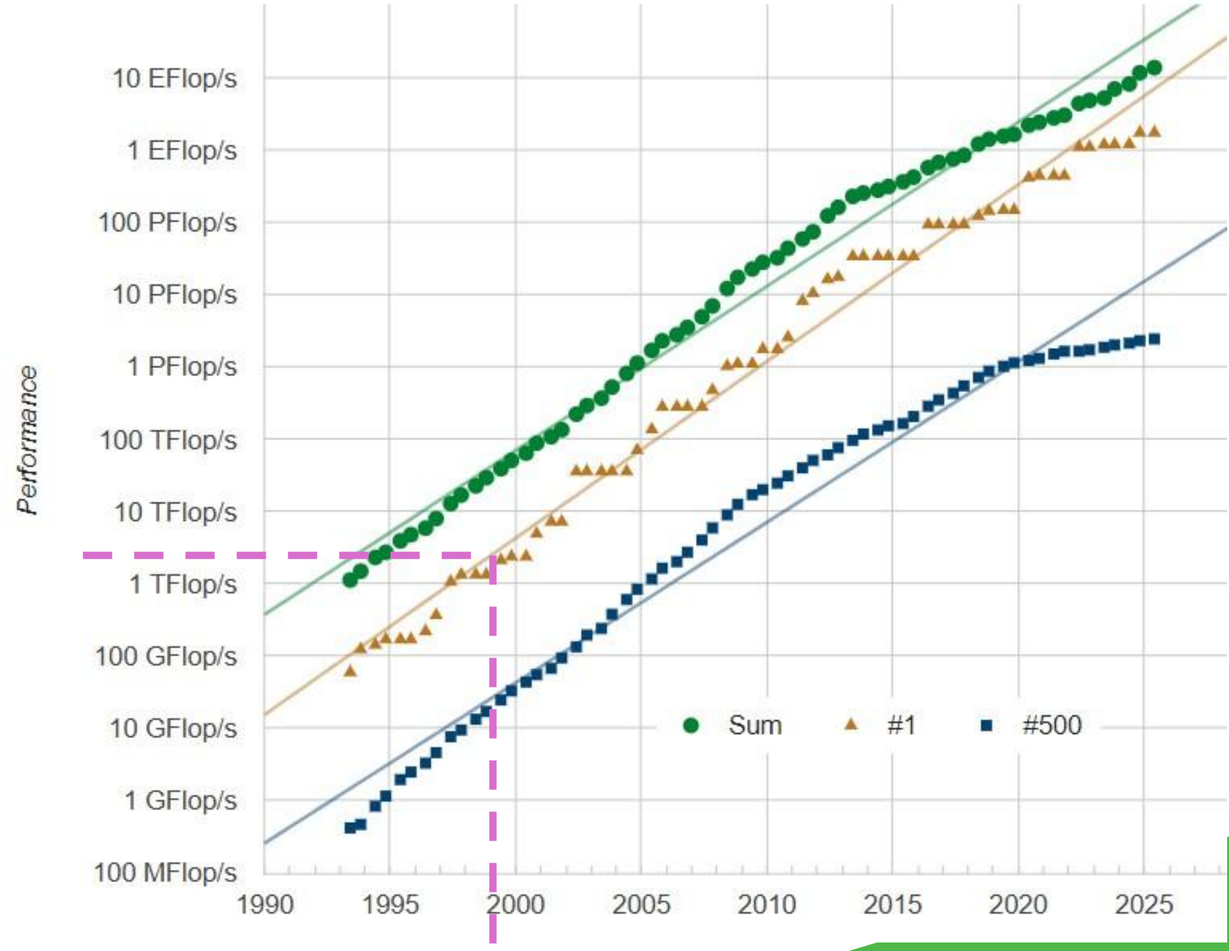
Green500

- Rank supercomputers in terms of energy efficiency
- Performance per Watt (GFLOPS/W)
- https://www.top500.org/lists/green500

- Green500 #1: Jedi
  - Jülich Supercomputing Centre (JSC), Germany
  - 20,000 cores, 4.5 PetaFLOPS
  - 73 GFlops/watts
  - For comparison: #1 Top500 : 59 GFlops/watts

HPC is ahead of the curve

	ASCI Red Supercomputer	iPhone 16 Pro Smartphone
launch	1999	2024
TeraFLOPS	3.1	2.3
fits in	a large building	your pocket
power draw	850 kW	1 W
unit cost	$130 million	$1,300



Conclusions

- HPC provides the ability solve bigger problems in the same amount of time and/or you can solve the same sized problems in less time

- HPC can solve extremely detailed mathematical and physical models within **reasonable timeframes**