High Performance Computing for AI models

Dr Nikos Bakas

December 18, 2025

Contents

Part 1

General Concepts of High Performance Computing (HPC)

1.1 Definition of HPC

High Performance Computing (HPC) refers to the practice of aggregating computing power in a way that delivers **much greater performance** than one could get out of a typical desktop computer or workstation in order to solve large problems in **science, engineering, and business**.

Figure 1.1: ARIS - HPC Infrastructure in Greece https://www.hpc.grnet.gr/

HPC systems can process data and perform **complex calculations** at **extremely high speeds**.

1.2 The Advent of Clustering: Late 1990s and 2000s

- The Beowulf **Beowulf cluster**, uses **inexpensive, commodity hardware** connected by a network.

- The first Beowulf cluster **(1994)** comprised **16 i486 DX4 processors** and reached **500 MFLOPS**.

Taking an **Intel Core i9 processor with 10 cores**, a conservative clock speed of 3.0 GHz, and assuming it can perform 16 FLOPS per cycle per core, the theoretical peak performance would be: $10\,\text{cores} \times 3.0\,\text{GHz} \times 16\,\frac{FLOP/cycle}{core} =$ **480 GFLOPS**

Figure 1.2: The Borg, a 52-node Beowulf cluster used by the McGill University pulsar group to search for pulsations from binary pulsars. Copyrighted free use, https://commons.wikimedia.org/w/index.php?curid=119482

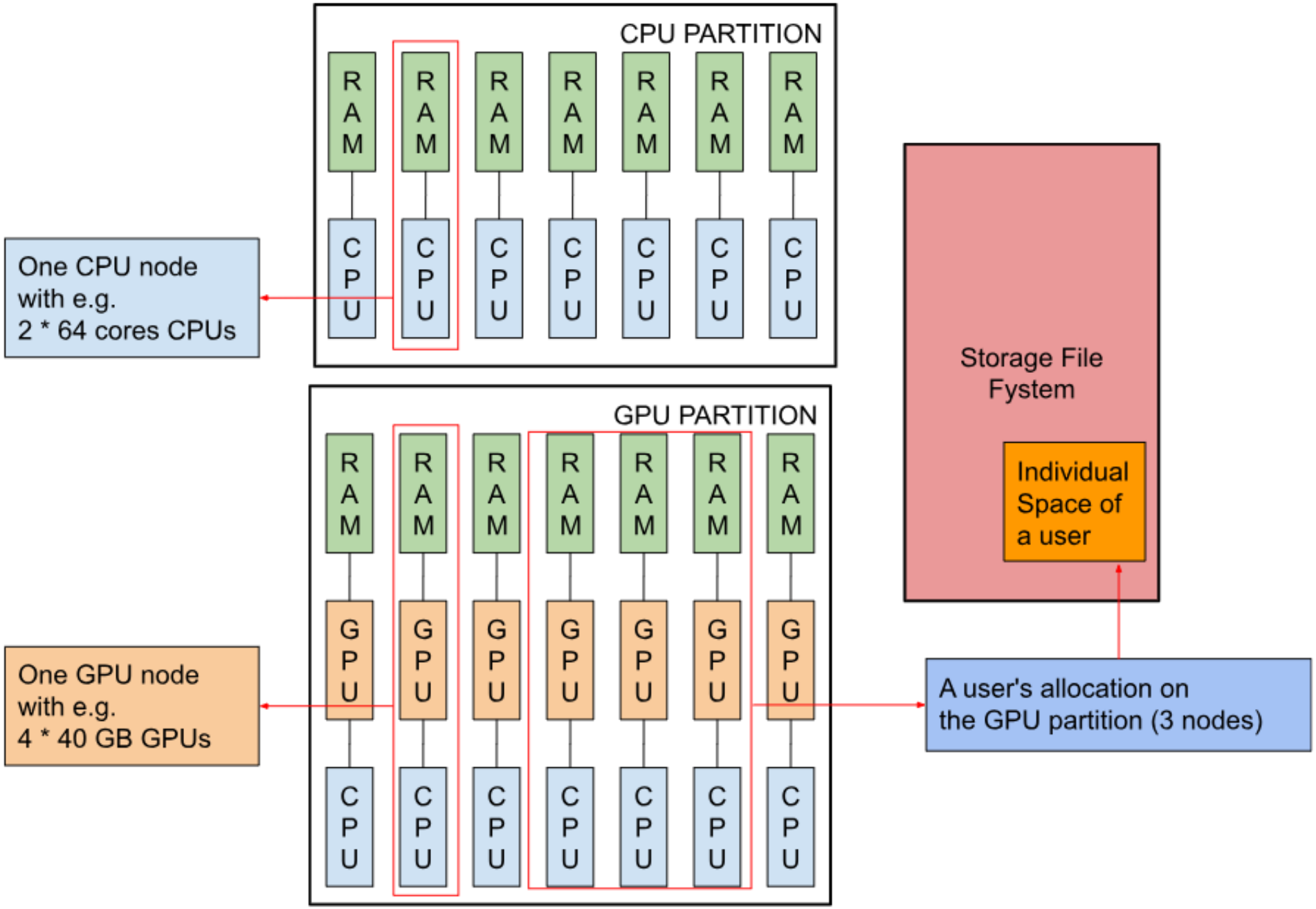
1.3 Components of an HPC System

An HPC system typically involves:

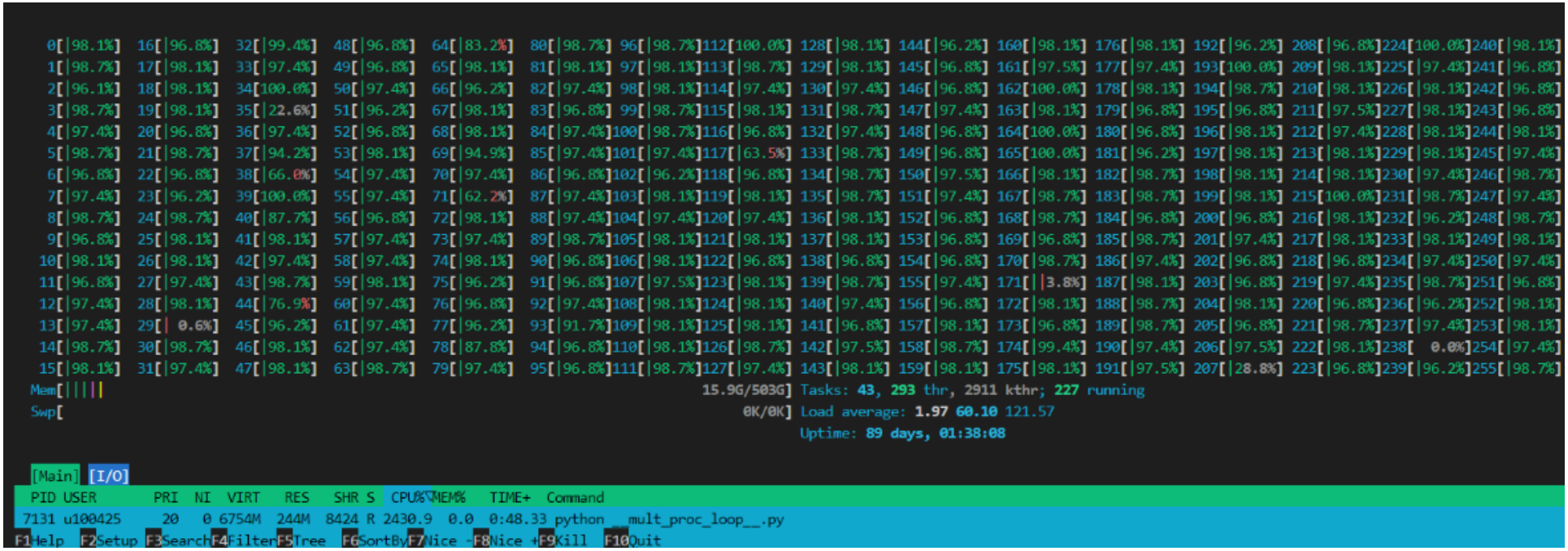
- **Compute Nodes:** These are the individual servers that provide the processing power. Each node contains one or more CPUs (Central Processing Units) or GPUs (Graphics Processing Units), and often both.
- **Networking:** A high-speed network interconnects the compute nodes, allowing for rapid communication and data transfer between nodes.
- **Storage:** Fast, large-capacity storage systems are required for input/output (I/O) operations, storing the data that is generated and used by applications running on the HPC system.
- **Software:** This includes the operating systems, programming models, and applications and tools specific to HPC tasks.

1.4 CPUs, GPUs and Nodes

- A cluster divided into **CPU** and **GPU** partitions
- A CPU node highlighted - multiple **CPU nodes** exist within the CPU partition.
- User's allocation in the **GPU partition** is indicated, showing a combination of RAM and GPU resources assigned to a user.


CPU PARTITION
RAM
CPU
One CPU node
with e.g.
2 * 64 cores CPUs
GPU PARTITION
GPU
One GPU node
with e.g.
4 * 40 GB GPUs
Storage File
Fystem
Individual
Space of
a user
A user's allocation on
the GPU partition (3 nodes)

Threads on 1 node of MeluXina supercomputer
https://docs.lxp.lu/system/overview/



The screenshot was taken using the **htop** tool, which is a command-line utility for monitoring Linux processes.

1.5 Moore's Law

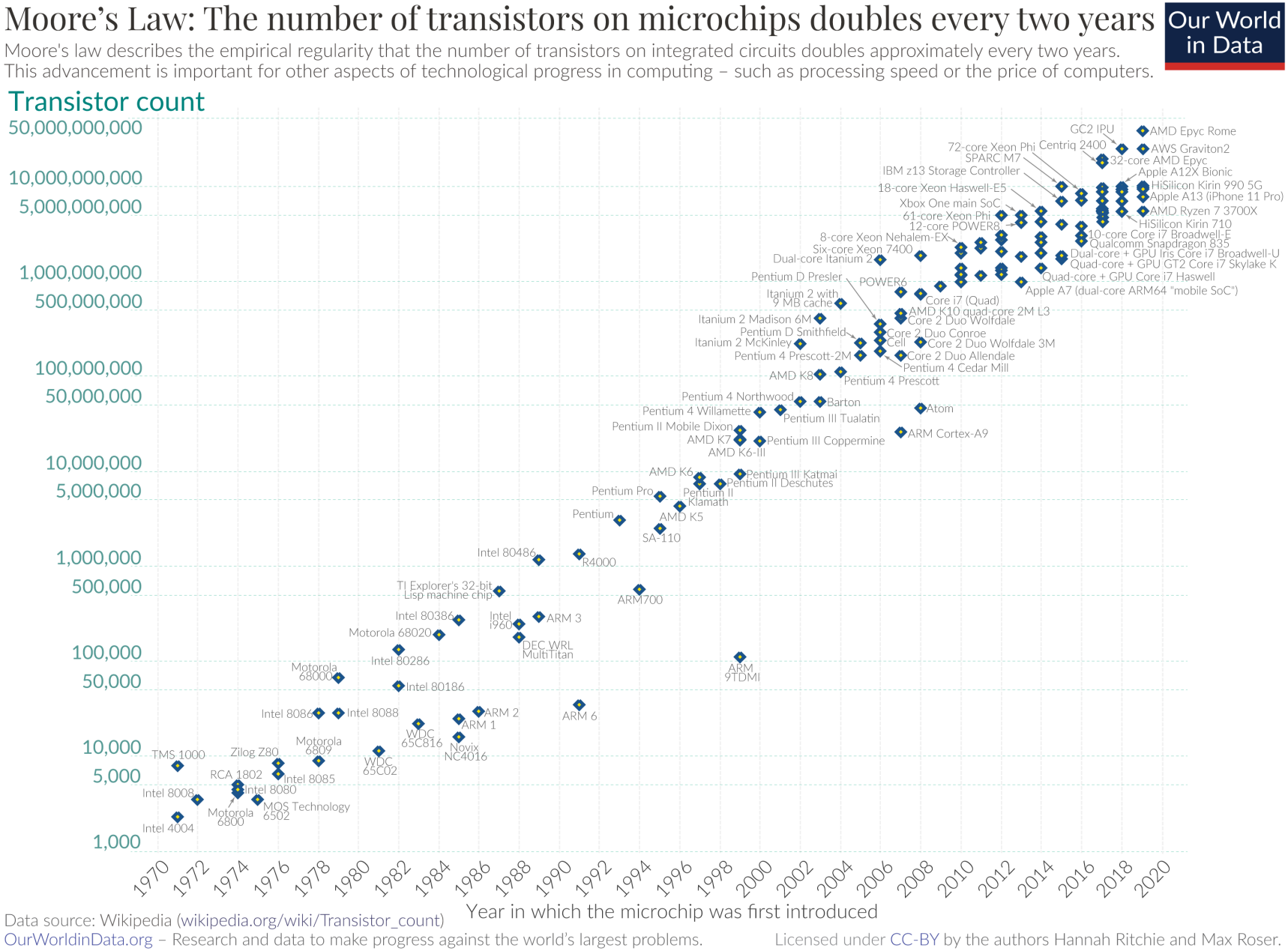
A **transistor** is a semiconductor device used to amplify or switch electronic signals and electrical power. It is one of the basic building blocks of modern electronic devices. In essence, transistors can be understood **as a type of switch** that controls the flow of electricity in a circuit.

An **integrated circuit** (IC), sometimes called a **microchip**, is a semiconductor wafer on which thousands or millions of tiny resistors, capacitors, and transistors are fabricated. An integrated circuit can function as an amplifier, oscillator, timer, microprocessor, or even computer memory.

Gordon E. Moore, a co-founder of the semiconductor company **Intel**, is the person behind the formulation of Moore's Law.

In 1965, Moore observed that the number of transistors on integrated circuits had doubled every year since their invention and predicted that this trend would continue into the foreseeable future.

A decade later, in **1975**, based on the changing pace of technology, he revised his projection, estimating **a doubling approximately every two years**.



Hannah Ritchie and Max Roser, https://creativecommons.org/licenses/by/4.0
https://ourworldindata.org/uploads/2020/11/Transistor-Count-over-time.png

1.6 Sample Slurm Script

*Sample slurm script to allocate 4 nodes, with 2 * 64 cores each, with hyperthreading (2 threads per core).* Check also https://slurm.schedmd.com/sbatch.html.

```
#!/bin/bash

#SBATCH --job-name=python_job          # Descriptive job name
#SBATCH --nodes=4                      # Request 4 compute nodes
#SBATCH --ntasks-per-node=64           # 64 tasks per node
#SBATCH --cpus-per-task=2              # Allocate 2 CPUs per task (for
 ↪  hyperthreading)
#SBATCH --time=04:00:00                # Set a job time limit of 4
 ↪  hours
#SBATCH --partition=highmem            # Submit to a partition suitable
 ↪  for your needs
#SBATCH --mem=512GB                    # Example: Request 512GB memory
 ↪  per node

# Load necessary modules
module load python/3.9  # Example: Load Python 3.9 module
# Run your Python script
mpirun -np 256 python my_python_script.py
```

Part 2

Scaling

2.1 Why Parallelization Matters

The following is an example code for the brightness increase of 1_000_000 Images:

```
using Base.Threads, Images

image_paths = ["image1.jpg", "image2.jpg", ...,
 ↪  "image1_000_000.jpg"]

brightness_increase = 50  # Define your brightness increase

@threads for path in image_paths
        image = load_image(path)  # Load the image
        adjusted_image = clamp.(image .+ brightness_increase, 0,
         ↪  255)  # Adjust brightness
        save_image(adjusted_image, "adjusted_$path")  # Save the
         ↪  adjusted image
end
```

The code comprises commands for:

- **Parallelism:** Distributes image processing across threads, making 1,000,000 images as fast to process as one, assuming sufficient resources.
- **Vectorized Computations:** Modern processors use vectorized instructions (SIMD) to operate on multiple data points simultaneously within a single instruction.

Computational Complexity: Turning an $O(n)$ operation into an effective $O(1)$ operation (assuming perfect parallelization zero overhead).

- **Perfect Parallelization is Impractical**: Real-world scenarios often have dependencies between loop iterations or require synchronization, making perfect parallelization difficult to achieve.
- **Overhead Always Exists** (Almost): Even the most optimized systems have some overhead associated with thread management. While it might be minimal, it's not truly zero.

2.2 Speedup

Speedup in the context of HPC is a metric used to quantify the performance improvement of a parallel system compared to a serial one. It is typically expressed as:

$$\text{Speedup} = \frac{\text{Execution Time on Single Processor}}{\text{Execution Time on Multiple Processors}}$$

An **ideal speedup** is linear, meaning the speedup is **equal to the number of processors**. This, however, is rarely the case due to **overheads** inherent in parallel systems.

Example:

Let's say a computational task runs for **80 hours** on a single processor. Running the same task on a parallel system with **4 processors** takes **22 hours**. Thus, the speedup is:

$$\text{Speedup} = \frac{\mathbf{80}}{\mathbf{22}} \approx \mathbf{3.64} < \mathbf{4}$$

This represents a **sub-linear speedup**, indicating that there are likely some inefficiencies in the parallel processing.

2.3 Parallelization Efficiency

Parallelization efficiency, also known as **parallel efficiency**, is a measure of how efficiently a computational task runs in parallel compared to serially. It is defined as the ratio of the speedup achieved to the number of processors used:

$$\text{Parallelization Efficiency} = \frac{\text{Speedup}}{\text{Number of Processors}}$$

Values of parallelization efficiency range from **0** (no benefit from parallelization) to **1** (perfect linear speedup). In reality, efficiencies are less than **1** due to **overheads like communication between processors, synchronization, and non-uniform memory access delays**.

Example:

If a task takes 100 hours to complete on a single processor and 15 hours to complete on 8 processors, the speedup is:

$$\text{Speedup} = \frac{100}{15} \approx \mathbf{6.67}$$

The parallelization efficiency would be:

$$\text{Parallelization Efficiency} = \frac{6.67}{8} \approx \mathbf{0.83}$$

Here, an efficiency of **0.83** means the system is utilizing the parallel processors fairly efficiently, but there is still some room for improvement.

2.4 Floating Point Operations Per Second (FLOPS)

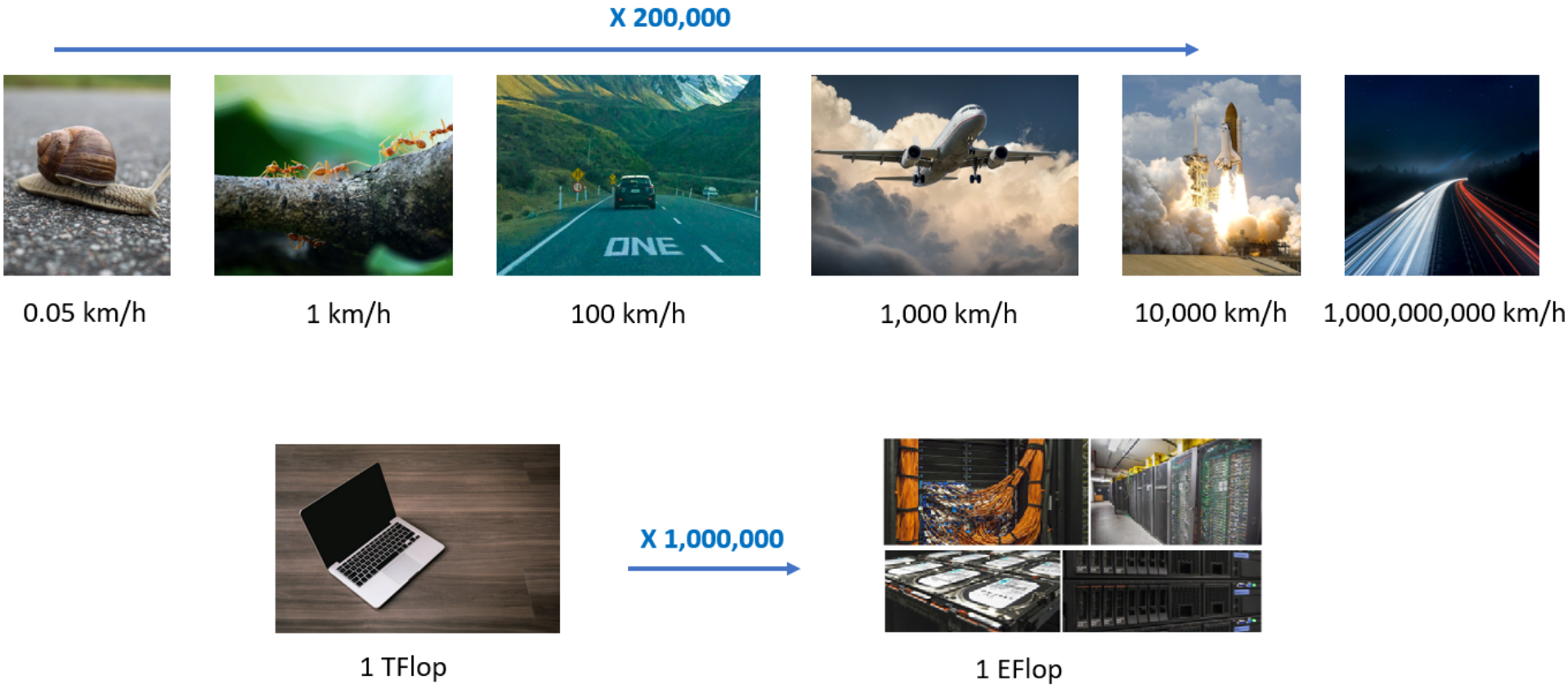
There are several metrics to evaluate HPC performance:

FLOPS (Floating Point Operations Per Second): A measure of a computer's performance, especially in fields of scientific computations that require floating-point calculations. Example: A supercomputer with 1 Peta FLOP can perform 10^{15} floating-point operations per second.

Operations	Name	Abbreviation
1	FLOPS	FLOPS
10^3	Kilo FLOPS	KFLOPS
10^6	Mega FLOPS	MFLOPS
10^9	Giga FLOPS	GFLOPS
10^{12}	Tera FLOPS	TFLOPS
10^{15}	Peta FLOPS	PFLOPS
10^{18}	Exa FLOPS	EFLOPS

Table 2.1: Magnitudes of FLOPS (Floating Point Operations Per Second)

FLOPS are widely used to assess state-of-the-art Supercomputers.



- This speed comparison highlights a snail (0.05 km/h) scaling up to the speed of a rocket (10,000 km/h), increasing by 200,000 times.
- Similarly, the computational power comparison between a laptop (1 TFlop) and an exascale supercomputer (1 EFlop) is even higher, being 1,000,000 times more powerful.

Part 3

Programming Models in HPC

HPC applications are often developed using programming models that support parallelism:

- **OpenMP (Open Multi-Processing):** An application programming interface (API) that supports multi-platform shared-memory multiprocessing programming.
- **MPI (Message Passing Interface):** Used for programming parallel computers. It involves processes sending and receiving messages to achieve parallelism.
- **GPGPU (General-Purpose computing on Graphics Processing Units):** Utilizes GPUs to perform computation in applications traditionally handled by the CPU.

These models come with their own set of libraries and compilers that are designed to optimize performance by taking full advantage of the hardware capabilities present within HPC environments.

Part 4

State of the art machines

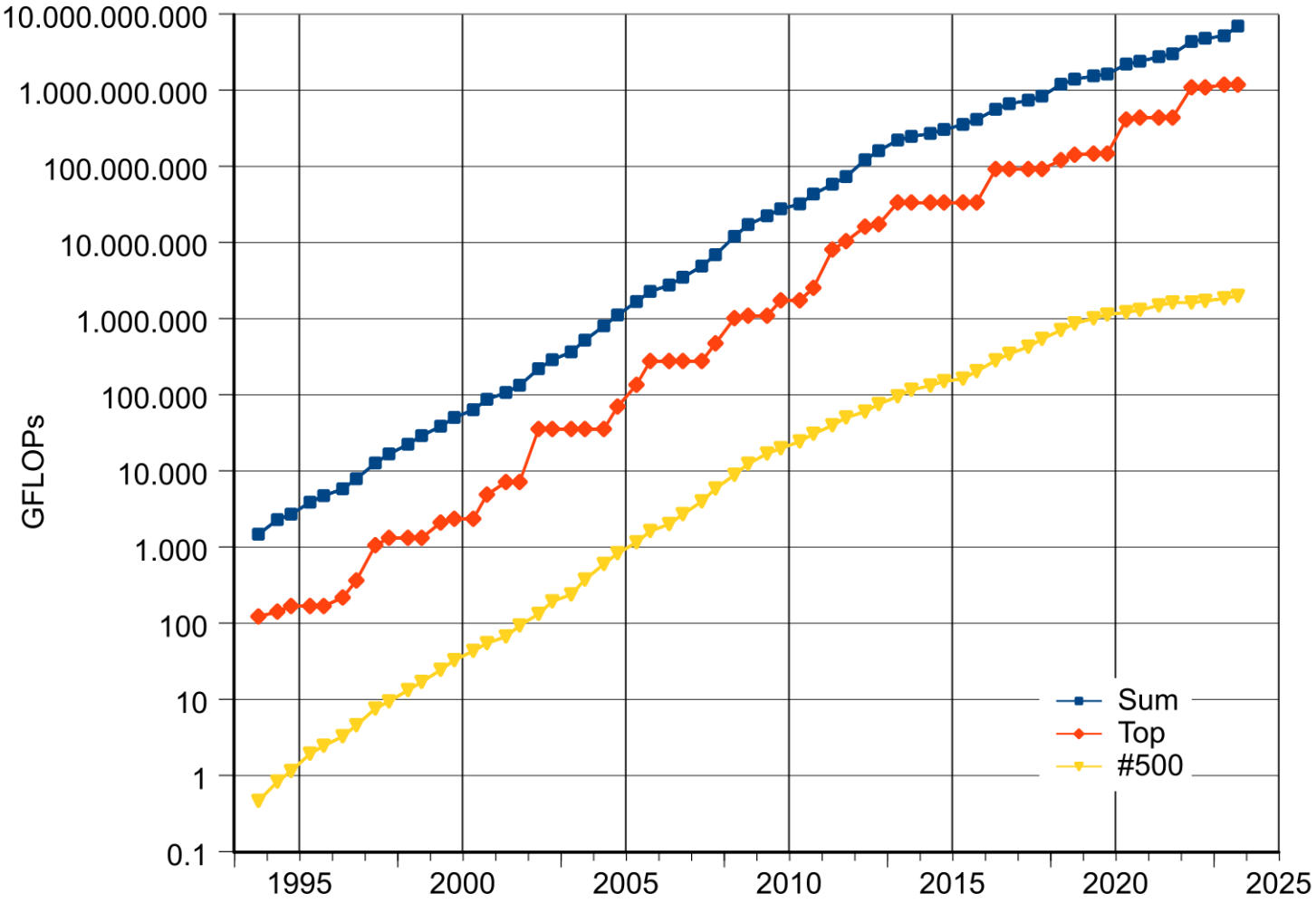
4.1 The Top 500 list

- The Top 500 list is a ranking of the **world's 500 most powerful non-distributed computer systems**. The list is compiled twice a year.
- **Performance** of the supercomputers on the Top 500 list **is measured using the LINPACK Benchmark**. This benchmark tests the system's ability to solve a dense system of linear equations, providing a measure of the computer's floating-point rate of execution.

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,039,616	1,742.00	2,746.38	29,581
2	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	2,055.72	24,607
3	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
4	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
5	**HPC6** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, RHEL 8.9, HPE Eni S.p.A. Italy	3,143,520	477.90	606.97	8,461

https://top500.org/ Nov. 18, 2024

4.1.1 Exponential Growth



https://en.wikipedia.org/wiki/TOP500#/media/File:Supercomputers-history.svg

4.2 Top 8 European Supercomputers

The European High Performance Computing Joint Undertaking (EuroHPC JU) is a joint initiative between the EU, European countries and private partners to develop a World Class Supercomputing Ecosystem.
https://eurohpc-ju.europa.eu/index_en

1. **LUMI (CSC, Finland)**
   - LUMI-C: 1536 nodes, 128 cores/node, 256-1024 GB RAM/node
   - GPU: 2560 nodes, 64 cores/node, 4 GPUs, 128 GB GPU-RAM
   - Visualization: 64 nodes, 1 GPU, 48 GB GPU-RAM
   - Peak Performance: 550 petaflops
   - URL: https://www.lumi-supercomputer.eu/lumis-full-system-architecture-revealed/
2. **Leonardo (Cineca, Italy)**
   - Booster Module: 3456 nodes, 32 cores/node, 512 GB RAM/node, 4 GPUs, 64 GB GPU-RAM
   - Data Centric Module: 1536 nodes, 112 cores/node, 512 GB RAM/node
   - Peak Performance: 323.4 petaflops
   - URL: https://leonardo-supercomputer.cineca.eu/hpc-system/

3. **MareNostrum 5 (Barcelona Supercomputing Center, Spain)**
   - General Purpose Partition: 6408 nodes, 112 cores/node, 256 GB RAM/node
   - Accelerated Partition: 1120 nodes, 64 cores/node, 512 GB RAM/node, 4 GPUs, 64 GB GPU-RAM
   - Peak Performance: 314 petaflops
   - URL: https://www.bsc.es/innovation-and-services/marenostrum/marenostrum-5
4. **MeluXina (LuxProvide, Luxembourg)**
   - Cluster: 573 nodes, 128 cores/node, 512 GB RAM/node
   - Accelerator-GPU: 200 nodes, 64 cores/node, 512 GB RAM/node, 4 GPUs, 40 GB GPU-RAM
   - Large memory: 20 nodes, 128 cores/node, 4096 GB RAM/node
   - Peak Performance: 18.29 petaflops
   - URL: https://docs.lxp.lu/system/overview/
5. **Karolina (IT4I, Czech Republic)**
   - CPU: 828 nodes, 128 cores/node, 256-24000 GB RAM/node
   - GPU: 72 nodes, 8 GPUs, 40 GB GPU-RAM
   - Peak Performance: 15.69 petaflops
   - URL: https://www.it4i.cz/en/infrastructure/karolina

6. **Vega (IZUM, Slovenia)**
   - GPU partition: 60 nodes, 128 cores/node, 512 GB RAM/node, 4 GPUs, 40 GB GPU-RAM
   - CPU node Standard: 768 nodes, 128 cores/node, 256 GB RAM/node
   - CPU node Large Memory: 192 nodes, 128 cores/node, 1000 GB RAM/node
   - Peak Performance: 10.05 petaflops
   - URL: https://doc.vega.izum.si/architecture/
7. **Deucalion (Guimarães, Portugal)**
   - ARM cluster: 1632 nodes, 48 cores/node
   - X86 cluster: 500 nodes, 48+ cores/node
   - Accelerated partition: 33 nodes
   - Peak Performance: 10 petaflops
   - URL: https://macc.fccn.pt/resources#deucalion
8. **Discoverer (Sofia Tech Park, Bulgaria)**
   - CPU: 1128 nodes, 128 cores/node, 256 GB RAM/node
   - CPU-Fat: 18 nodes, 128 cores/node, 1000 GB RAM/node
   - Peak Performance: 5.94 petaflops
   - URL: https://docs.discoverer.bg/resource_overview.html

LUMI
FINLAND

LEONARDO
ITALY

MELUXINA
LUXEMBOURG

KAROLINA
CHECH REPUBLIC

DISCOVERER
BULGARIA

VEGA
SLOVENIA

DEUCALIO
PORTUGAL

MARENOSTRUM 5
SPAIN

4.3 ARIS - HPC Infrastructure in Greece

Compute Nodes

Table 4.1: Nodes Summary

Node Type	Count	Accelerator	Memory	Cores
THIN nodes	48	w/o	512 GB	128@2.45 GHz [1]
GPU nodes	3	4 x 80GB[2]	512 GB	128@2.45 GHz
FAT nodes	16	w/o	1024 GB	128@2.45 GHz [1]

https://doc25.aris.grnet.gr/system/hardware/

Figure 4.1: ARIS - HPC Infrastructure in Greece

[1] two sockets
[2] NVIDIA Ampere A100

Table 4.2: GPU Nodes Technical Information

Architecture	x86-64
System	Dell PowerEdge XE8545
Total number of nodes	3
Total number of cores	384
Total number of GPUs	12
Total amount of RAM [TByte]	1.5
Total Linpack Performance [TFlop/s]	240
Components	
Processor Type	AMD EPYC 7763
Nominal Frequency [GHz]	2.45
Processors per Node	2
Cores per Processor	64
Cores per Node	128
Hyperthreading	OFF
Accelerators	
Accelerator type	GPU – NVIDIA Ampere A100
Accelerators per node	4
Accelerator memory [GByte]	80
Memory	
Memory per Node [GByte]	512

Figure 4.2: ARIS - HPC Infrastructure in Greece

https://www.hpc.grnet.gr/

4.4 Daedalus - EuroHPC supercomputer in Greece

The way is open to building a EuroHPC world-class supercomputer in Greece

A hosting agreement has been signed between the EuroHPC Joint Undertaking and the National Infrastructures for Research and Technology (GRNET) in Greece, where DAEDALUS, a new EuroHPC supercomputer, will be located.

- Total sustained performance (Rmax): 89,2Petaflops.
- Expected to be listed within the top 30 positions on the TOP500 list and top 7 among the EuroHPC JU systems.
- 11PB storage (10PB HDD capacity + 1PB NVMe Flash performance)
- Lavrion Technological and Cultural Park (TCPL) https://eurohpc-ju.europa.eu/way-open-building-eurohpc-world-class-supercomputer-greece-2022-11-28_en
- June 11, 2024: GRNET S.A. conducts a Public Consultation on the Open Tender Announcement Issue https://grnet.gr/2024/06/11/public-consultation-lavrio-daedalus/

Part 5

HPC for AI models

5.1 Big Data

Large language models can be trained on datasets containing trillions of words, which translates to roughly **tens of trillions of tokens**.

- **100 tokens are approximately equal to 75 words**. Tokens can be a whole word, but they can also be smaller parts of words or even punctuation marks.

- **Wikipedia** comprises approximately **3 billion** tokens.

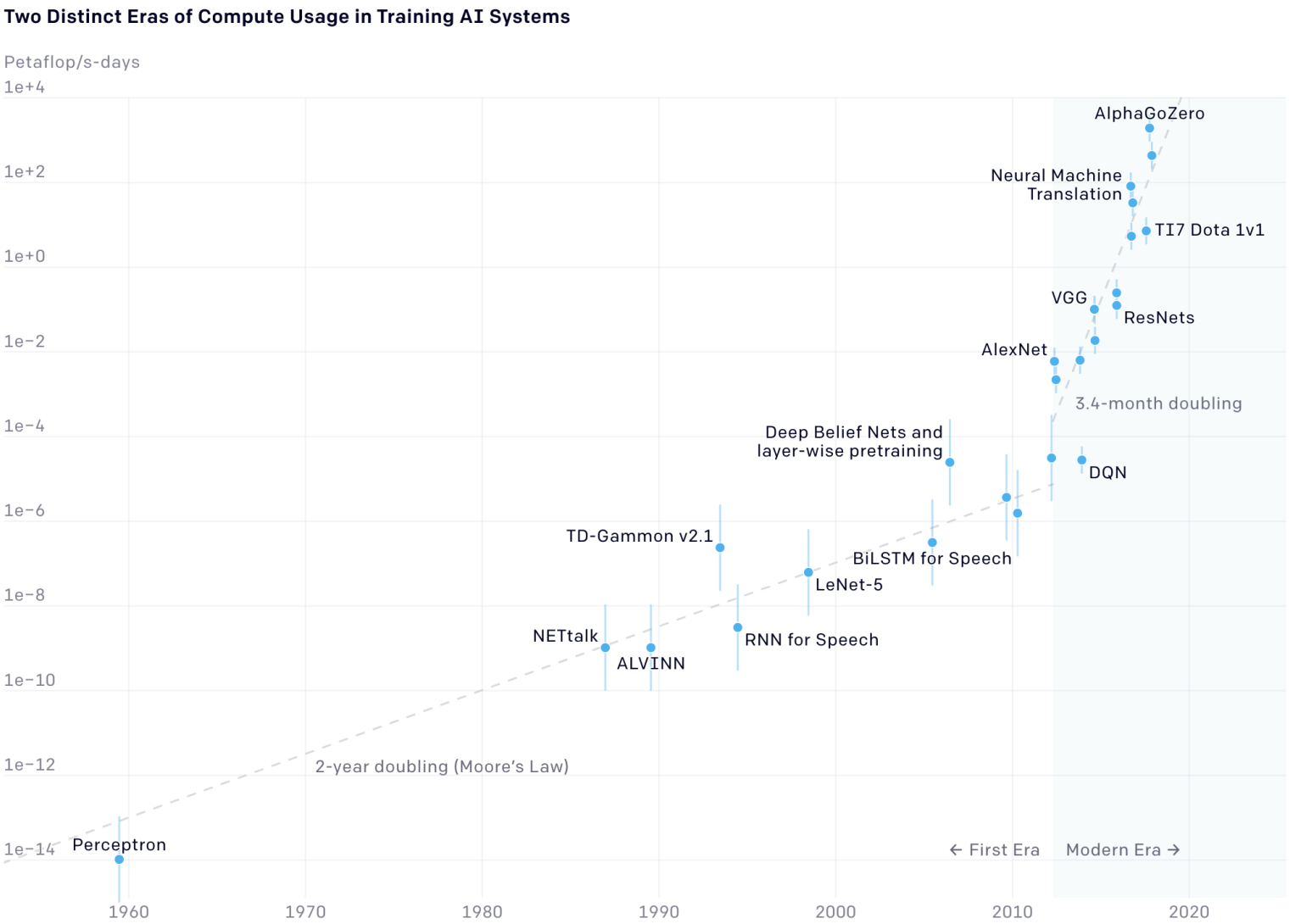
- **The British Library** has around 14 million books. Assuming an average book has 50,000 tokens (words x 1.3 tokens/word), we get a very rough estimate of **700 billion tokens** (14 million books * 50,000 tokens/book). The stacks already exceed ∼750 km of shelving and grow by another 9.6 km every year.

5.2 This is a small amount of books!

Figure 5.1: Library shelves generated with https://chat.openai.com/

5.3 AI and compute

Since **2102** we observe a **3.4-month doubling** in computing power used to train AI models. https://openai.com/research/ai-and-compute. Petaflops are 10^{15} FLOPS.



5.4 Parallel Stochastic Gradient Descent (PSGD)

Parallel Stochastic Gradient Descent

- For each GPU (in Parallel)
  - Pick a random data-point
    - Compute the gradient
- Mix the Gradients

  average, ensembles, vector

  summation, AdaSum, etc.
- Update the Weights for all

$\forall$ epoch n, do:

$\forall$ batch i, do in parallel:

$\forall$ GPU g, do:

$$\Delta L_i = \frac{\partial L_i}{\partial w}$$

$$\Delta \mathrm{L} = \frac{1}{b} \sum_{i=1:b} \Delta L_i$$

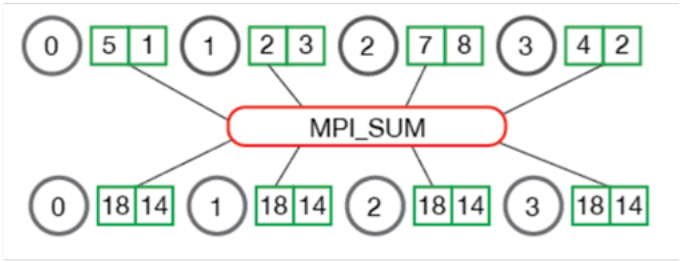
MPI allreduce

$$w_{n+1} = w_n - \gamma \Delta \mathrm{L}$$

https://mpitutorial.com/tutorials/mpi-reduce-and-allreduce/

In General (not always)

- ✓ Larger batch size is faster
- ✓ Smaller batch size more accurate
- ✓ Batch must fit into memory
- ✓ More nodes if GPU cannot handle size
- ✓ The weights' mix cause some accuracy loss



MPI_Reduce
+
MPI_Bcast

5.5 Distributed Data Parallel

Let there be p processes / GPUs with ranks $r \in \{0, \ldots, p-1\}$. Each rank holds an identical copy of the parameters w.

Initialization (once). Rank 0 initializes $w^{(0)}$, then all ranks synchronize:

$$w^{(0)} \leftarrow \mathrm{Bcast}\left(w^{(0)}\right).$$

Per-iteration objective. At iteration t, each rank r draws a *local mini-batch* $B_r^{(t)} = \{(x_{r,j}, y_{r,j})\}_{j=1}^{b}$ (typically via a `DistributedSampler` so data are sharded across ranks). Define the local loss

$$\ell_r^{(t)}(w) = \frac{1}{b} \sum_{j=1}^{b} \ell(f(x_{r,j}; w),\, y_{r,j}),$$

and the global (effective) batch size $B = pb$ with global loss

$$\ell^{(t)}(w) = \frac{1}{p} \sum_{r=0}^{p-1} \ell_r^{(t)}(w) = \frac{1}{B} \sum_{r=0}^{p-1} \sum_{j=1}^{b} \ell(f(x_{r,j}; w),\, y_{r,j}).$$

Gradient computation and synchronization. Each rank computes its local gradient

$$g_r^{(t)} = \nabla_w \ell_r^{(t)}\left(w^{(t)}\right),$$

then DDP synchronizes gradients with an *all-reduce* (sum) and averages:

$$\bar{g}^{(t)} = \frac{1}{p}\,\mathrm{AllReduce}_{\mathrm{sum}}\left(g_r^{(t)}\right) = \frac{1}{p}\sum_{r=0}^{p-1} g_r^{(t)} = \nabla_w \ell^{(t)}\left(w^{(t)}\right).$$

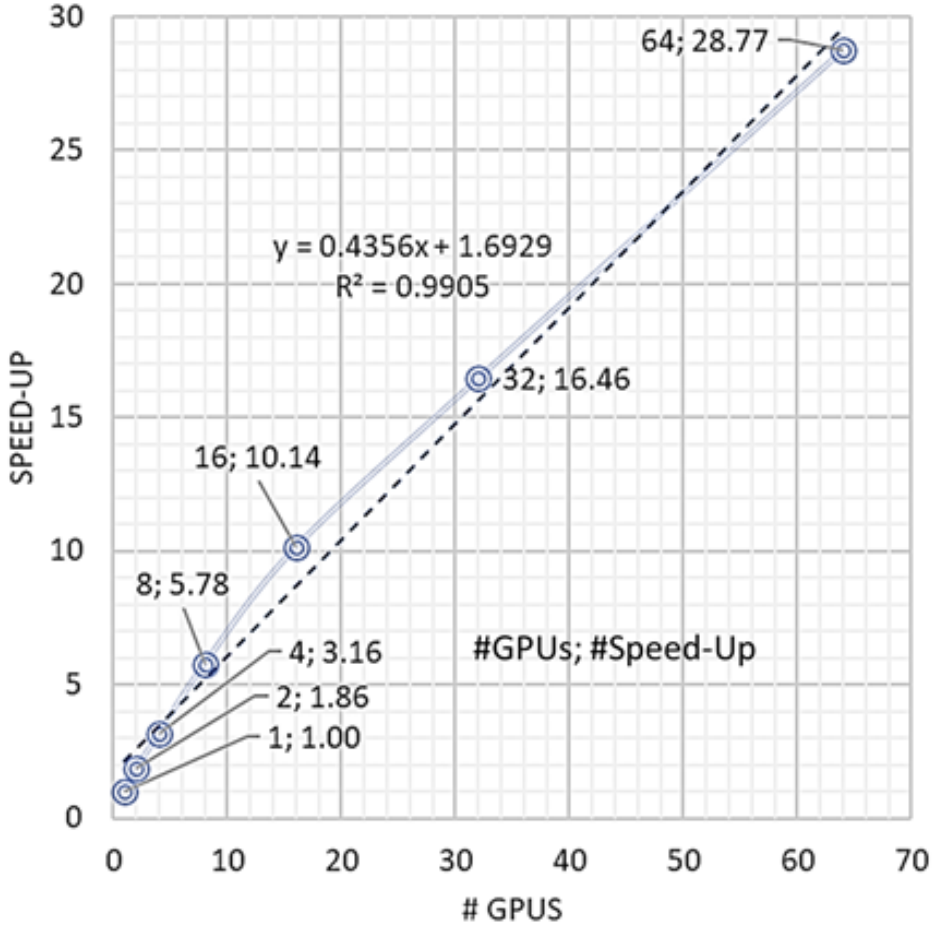
Parameter update (done locally on every rank). After synchronization, every rank applies the same optimizer step, e.g. SGD:

$$w^{(t+1)} = w^{(t)} - \eta\,\bar{g}^{(t)}.$$

(With Adam/other optimizers: $w^{(t+1)} = \mathrm{OptStep}(w^{(t)}, \bar{g}^{(t)})$.)

Note. In practice DDP performs the all-reduce *bucket-wise* during backprop to overlap communication with computation, but mathematically it is the same $\bar{g}^{(t)}$ above.

5.6 Trainind Deep Neural Networks on Distributed GPUs



# GPUs	train minutes (10 epochs)	Speed-Up	% valid. Accuracy	Accuracy Loss
1	44.82	1.00	89.00	0.00%
2	24.07	1.86	89.14	0.16%
4	14.18	3.16	89.11	0.12%
8	7.76	5.78	88.90	-0.11%
16	4.42	10.14	88.46	-0.61%
32	2.72	16.46	88.69	-0.35%
64	1.56	28.77	86.40	-2.92%

- image size: 512x512
- train batch size: 52 (per GPU)
- learning rate: 1e-4
- weight decay:1e-6

https://www.meetup.com/PyData-Cyprus/events/276154247/

PyTorch + Horovod

Nikolaos Bakas et al. (2021). "Performance and scalability of deep learning models trained on a hybrid supercomputer: Application in the prediction of the shear strength of reinforced concrete slender beams without stirrups". In: *8th International Conference on Computational Methods in Structural Dynamics and Earthquake Engineering*. COMPDYN 2021. URL: https://2021.compdyn.org/

5.7 Hyperparameter Tuning of ML Algorithms

Figure 5.2 depicts the utilization of threads on a single node of the MeluXina supercomputer. We see that the node comprises 256 threads, the full utilization of which is the aim.

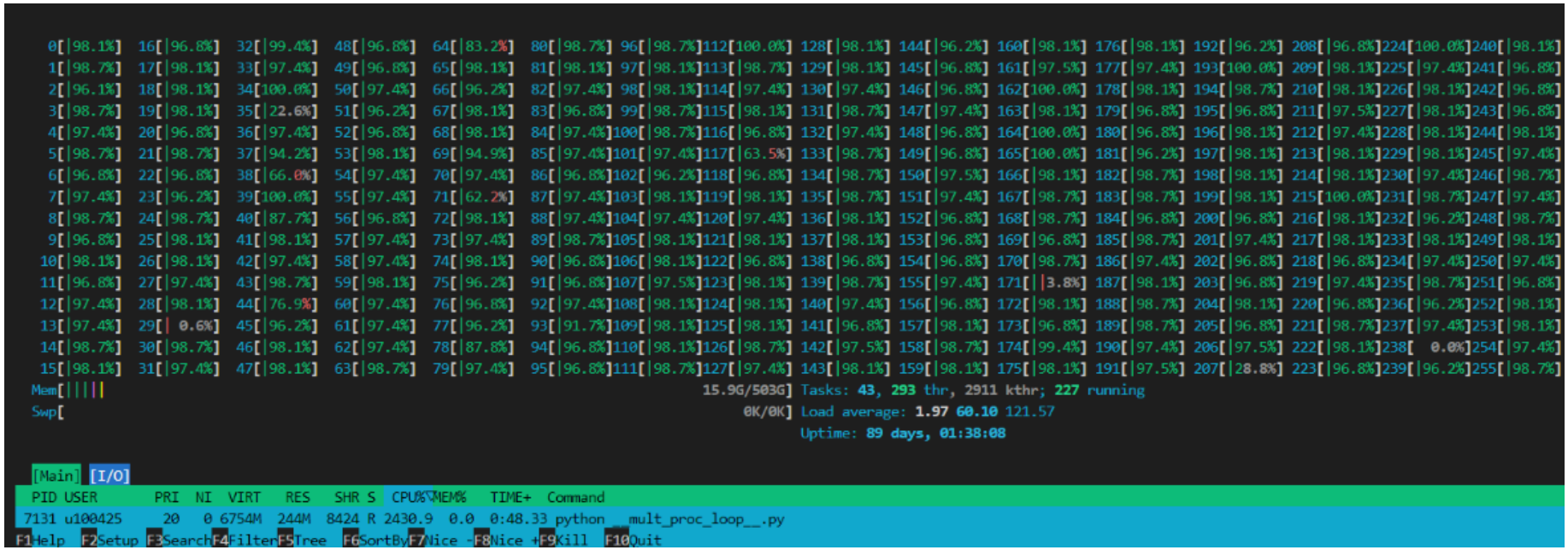


Figure 5.2: Threads at MeluXina supercomputer https://docs.lxp.lu/system/overview/

For each training, we use cross-validation
We train in parallel, one model per:

- 1 thread or ,
- "few" threads if the potential models are less than the available threads.

E.g. in Figure 5.3, 8 threads are utilized to train one deep learning model.

For each training, we use cross-validation

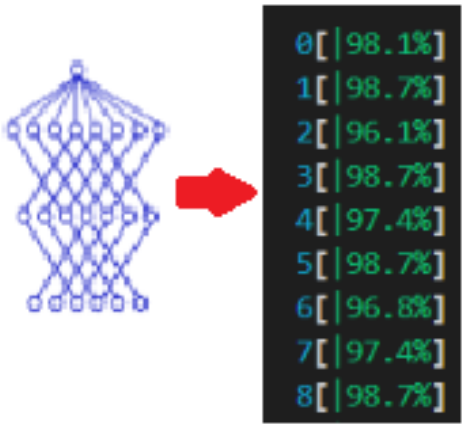


Figure 5.3: 8 threads utilized to train one deep learning model

Table 5.1: Compute time for the Steel SSI & Fundamental period datasets

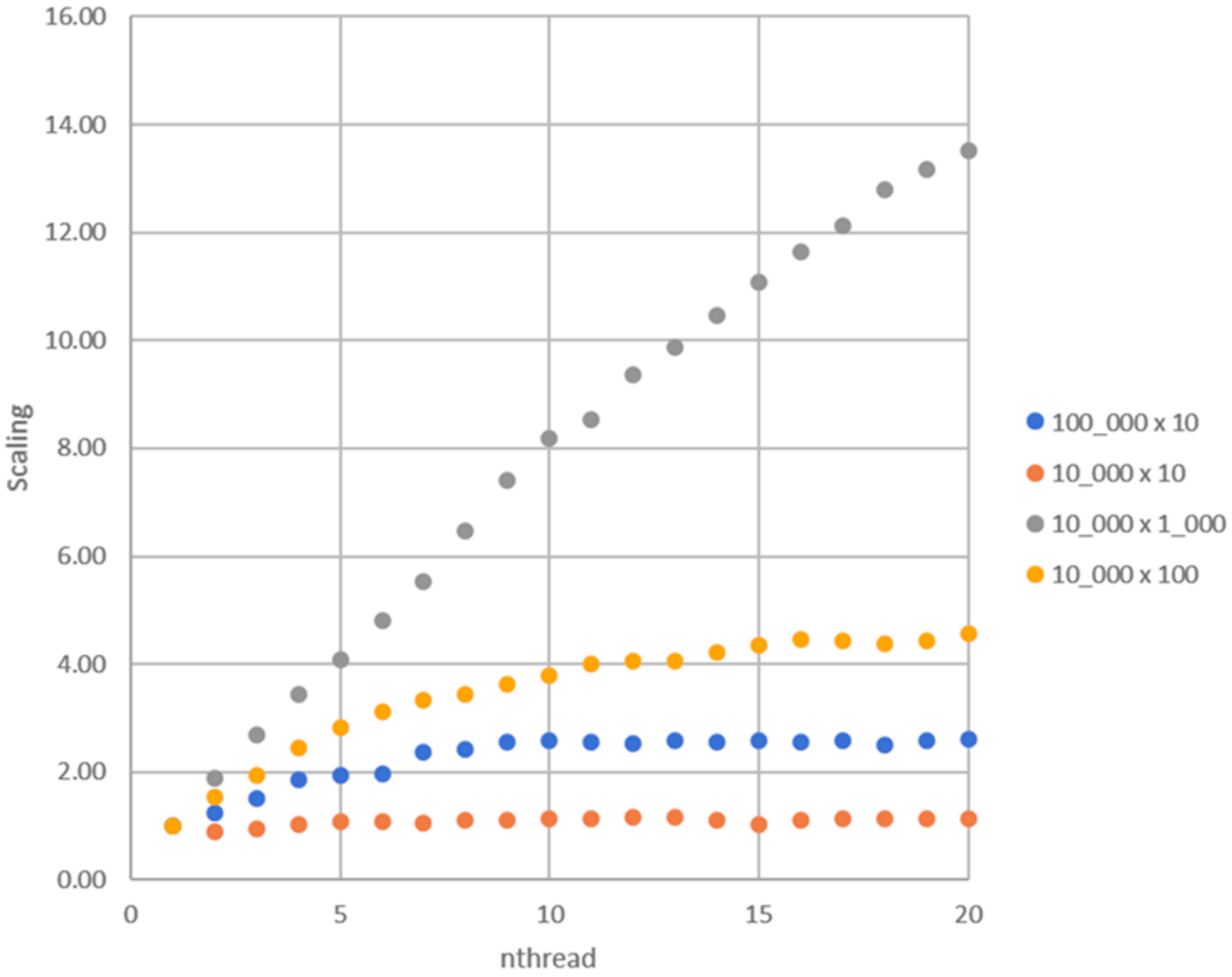
Dataset	**Samples**	**Features**	**# seconds 1st tune**	**# seconds 2nd tune**	**#seconds final train**	**# total seconds**	**Threads**
Steel ssi	98 308	6	57.51	361.45	21.16	440.12	256
Steel ssi	98 308	6	1,897.37	2,207.39	21.39	4,126.15	8
Fund. period	10 000	6	14.33	27.37	1.62	43.32	256
Fund. period	10 000	6	314.65	25160	1.63	567.88	8
Fund. period*	10 000	6	45.84	64.07	2.25	112.16	256
Fund. period*	10 000	6	1,260.14	2,742.67	1.63	4,004.44	8

Dewald Gravett et al. (2021). "New Fundamental Period Formulae for Soil-Reinforced Concrete Structures Interaction using Machine Learning Algorithms And Anns". In: *Soil Dynamics and Earthquake Engineering*. URL: https://www.sciencedirect.com/science/article/pii/S0267726121000786

George Markou et al. (Jan. 2024). "A general framework of high-performance machine learning algorithms: application in structural mechanics". en. In: *Computational Mechanics*. ISSN: 0178-7675, 1432-0924. URL: https://link.springer.com/10.1007/s00466-023-02386-9

5.7.1 Scaling of XGBoost

Figure 5.4: XGBoost compute times, for various datasets' size. The experiments are repeated 5 times, and the computation times have been averaged.



5.7.2 Scaling Results

Number of Models	Compute Time (seconds)	Equivalent without praellelism (seconds)	Equivalent without praellelism (hours)	Number of Threads	Scaling
1	15	15	0.004	8	1
100	45	1500	0.417	255	33.33
1000	140	15000	4.167	255	107.14
10000	1105	150000	41.667	255	135.75

A. M. v. d. Westhuizen, G. Markou, N. Bakas, and M. Papadrakakis, "Developing an artificial neural network model that predicts the fundamental period of steel structures using a large dataset," in *KEYNOTE Talk: 9th International Conference on Computational Methods in Structural Dynamics and Earthquake Engineering*. COMPDYN 2023, Athens, Greece, 12-14 June 2023. [Online]. Available: https://2023.compdyn.org/

Part 6

Apply for Access at EuroHPC JU

6.1 EuroHPC JU AI Factories Access Calls for Industrial Innovation

6.1.1 Playground Access

- **Target**: Industry (SMEs, startups), new or entry-level users of HPC/AI Factories.
- **Resources & duration**: Small allocations on a single system for **1-3 months** for **5,000 GPU hours**.
- **Process**: Permanently open, FIFO; short application; eligibility + technical check; access typically within 2 working days.

6.1.2 Fast Lane Access

- **Target**: Industry users already familiar with HPC.
- **Resources & duration**: Up to about **50,000 GPU hours** on a single system, for **1-3 months**.
- **Process**: Permanently open, FIFO; short proposal; eligibility + technical check; access typically within 4 working days.

6.1.3 Large Scale Access

- **Target**: Industry users with high-impact AI projects needing $>$ **50,000 GPU hours**.
- **Resources & duration**: Very large allocations on a single system for **3, 6, or 12 months**, for More than **50,000 GPU hours**.
- **Process**: Continuous call with frequent cut-off dates; full proposal + technical and peer-review evaluation; decision within about 10 working days after each cut-off.

https://www.eurohpc-ju.europa.eu/ai-factories/ai-factories-access-calls_en

6.2 EuroHPC AI Factory Fast Lane call

6.2.1 The Project

- Project details
- Project title
- Project summary (abstract)
- Keywords
- Proposal for civilian purposes
- Is any part of the project confidential?
- Artificial Intelligence (AI) technology
- AI set of technologies selection
- Application Domain
- Project duration (1, 2, 3 months)

6.2.2 Project Lead and Organisation information

6.2.3 Team Members Information

6.2.4 AI Factory selection

- AI Factory Selection
- Code(s) used
- Maximum number of GPUs
- Total storage required (GB)
- Total amount of data to transfer to/from (GB)

6.2.5 Code Details and Feasibility

- Code details
- Name and version of the code
- Webpage and other references
- Description of the code
- Scalability and performance
- Describe the scalability and performance of the application
- Explain how the optimization work proposed will contribute to future large scale applications

6.2.6 Ethics Self-Assessment

- How does your project ensure ethical principles and addresses potential societal impacts associated with the development and deployment of AI technologies
- How your system ensures that end-users have the ability to control vital decisions about their own lives
- Privacy & Data Governance
  - Does your proposal involve handling of personal data? (Yes/No)
  - Please describe how data is collected and processed from the aspect of lawfulness, fairness and transparency
  - What measures (such as anonymization, pseudonymisation, encryption, and aggregation) you took to safeguard the rights of data subjects?
  - Please describe the measures you employ to prevent data breaches and leakages
- Fairness: How do you ensure avoiding algorithmic bias, in input data, modelling and algorithm design?
- Individual, and Social and Environmental Well-being
- Transparency: Are the end-users aware that they are interacting with an AI system? (Yes/No)
- How the participants and/or end-users will be informed about interacting with an AI system, and about its purpose, capabilities, limitations, benefits and risks
- Accountability: Please describe how your system ensures that potential ethically and socially undesirable effects will be detected, stopped, and prevented from reoccurring

6.3 Frequently Asked Questions (FAQ)

Which organisations are eligible for access to EuroHPC machines?

Any European organisation is eligible for access to perform Open Science research (the results of the work are made available for open access). This includes public and private academic and research institutions, public sector organisations, industrial enterprises and SMEs.

What is the cost?

Currently access is free of charge.

What are the participation conditions?

Participation conditions depend on the specific access call that a research group has applied. In general users of EuroHPC systems commit to:

- acknowledge the use of the resources in their related publications,
- contribute to dissemination events,
- produce and submit a report after completion of a resource allocation.

More information on participation conditions can be found in the call's Documents section.

https://eurohpc-ju.europa.eu/access-our-supercomputers/access-policy-and-faq_en

6.4 Resources

- EuroCC Sweden: How to apply for access to EuroHPC JU supercomputers https://www.youtube.com/watch?v=g5jOio006-E
- ENCCS (NCC Sweden): "How to use the PRACE-calls portal - Application to JU supercomputers" Seminar https://www.youtube.com/watch?v=N1QqMh7HOmQ
- NCC Greece: https://eurocc-greece.gr/how-to-apply-for-access-to-eurohpc-ju-supercom
- HPC wiki: https://hpc-wiki.info/hpc/HPC_Wiki