



INTRODUCTION TO HIGH-PERFORMANCE COMPUTING (HPC)

6 March 2026 - Guest lecture for HOWEST (Campus Bruges Station)



whoami

`kenneth.hoste@ugent.be` (*email*)

`@boegel` (*GitHub, Slack*)

`@boegel@mast.hpc.social` (*Mastodon*)

`@boegel.bsky.social` (*BlueSky*)

`linkedin.com/in/kenneth-hoste`

- Masters (2005) & PhD (2010) in Computer Science from Ghent University
- PhD topic: machine learning applied to software performance, compilers, ...
- Joined HPC-UGent team in October 2010
- **Main tasks: user support & training, software installations, EuroHPC projects**
- Likes family, beer, loud music, free & open source software, dad jokes, stickers, ...
- Doesn't like CMake, SCons, Bazel, TensorFlow, OpenFOAM, PyTorch, ...

LIVE POLL

- A couple of poll questions will be raised during the presentation
- Please join in (anonymously), no login required
- Use your smartphone or browser
- Visit menti.com + use 7224 4307, or scan this QR code:



menti.com (7224 4307)

OUTLINE

- What are supercomputers? What are they used for? How do they work?
- HPC infrastructure in Flanders
- Important concepts in HPC: multi-core, MPI, Amdahl's law, ...
- **Hands-on exercises** on the HPC-UGent infrastructure: logging in, running jobs, ...
- Extra material (if we have time):
 - Trends in HPC: GPUs, AI, cloud computing, rise of Arm & RISC-V
 - Quantum computing in a nutshell



TERMINOLOGY YOU WILL LEARN ABOUT

- Supercomputer, nodes, processors, cores, interconnect, FLOPS
- High-Performance Computing (HPC), scientific computing
- Parallel computing, shared memory vs distributed memory computing
- OpenMP, threads, MPI, processes, CPUs, ...
- Accelerated computing, GPUs
- Cloud computing
- Quantum computer, qubits



[menti.com \(7224 4307\)](https://menti.com/72244307)

POLL

How familiar are you with supercomputing?

- I have no idea what it is, or why I should care.
- I have a vague idea of what a supercomputer is.
- I know what a supercomputer is, but never used one.
- I have used a supercomputer already.
- I frequently use a supercomputer.
- I am an HPC system administrator
- I have my own supercomputer.





WHAT IS A SUPERCOMPUTER?

WHAT IS HIGH-PERFORMANCE COMPUTING (HPC)?

Harnessing the power of multiple interconnected cores/nodes/processing units.



“Rack” with desktop PCs in a basement
(observed at UGent in 2010 era)



HPC-UGent Tier-2 infrastructure
(ca. 2018)



LUMI, pre-exascale EuroHPC supercomputer
Finland (2022)

Over 450,000 CPU cores
+ 12,000 AMD MI250X GPUs
~531 PFLOPS
~145 million Euro

WHAT IS A SUPERCOMPUTER?

- A bunch of "normal" computers: servers, a.k.a. (worker)nodes
- **Fast local network** to link them together ("interconnect"): $> 100 \text{ Gbit/sec}$, $< 1\mu\text{s}$
- Large amount of **shared storage** (disks, SSDs, maybe tape,...): petabytes
- Can be considered one big computer (in some sense)
- Used for **resource-intensive workloads** (compute, memory, storage, ...)
- Sometimes with special-purpose accelerator hardware (like GPUs)
- Usually in a dedicated (part of a) datacenter

TERMINOLOGY: PERFORMANCE IN FLOPS

- Performance of supercomputers are usually measured in FLOPS:
64-bit floating-point operations per second
 - Example of one FLOP: $1.34 \times 54.97 = 73.6598$
- This metric is used to create the Top 500 list of supercomputers since 1993
- 1 megaFLOPS (MFLOPS): 1 million FLOPS (10^6)
- x1000 each generation: gigaFLOPS, teraFLOPS, petaFLOPS, exaFLOPS, zettaFLOPS
- Current generation of (largest) supercomputers: in order of **exaFLOPS** (10^{18})



<https://top500.org>

1 EFLOP = 1,000,000,000,000,000,000 64-bit floating-point operations per second

For comparison: there are about 7.5×10^{18} grains of sand on Earth

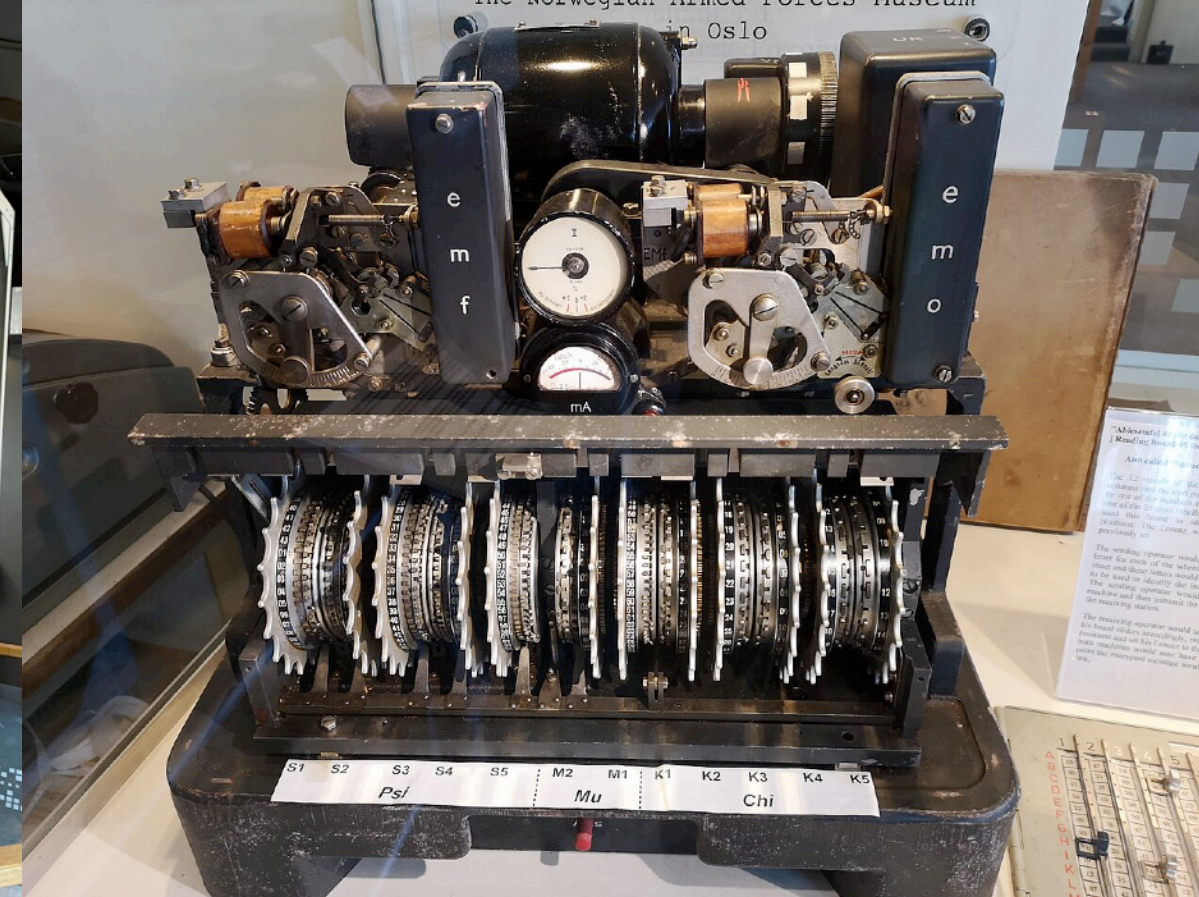
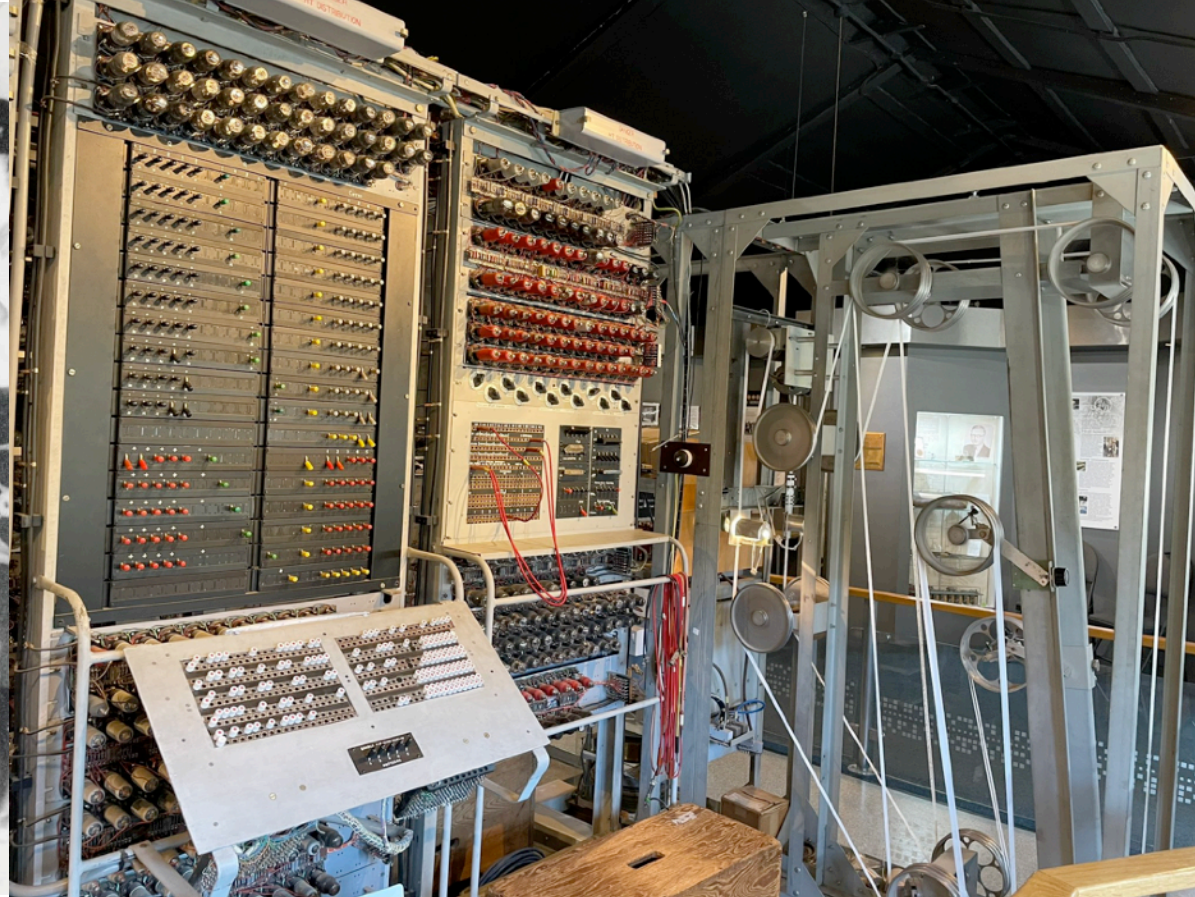
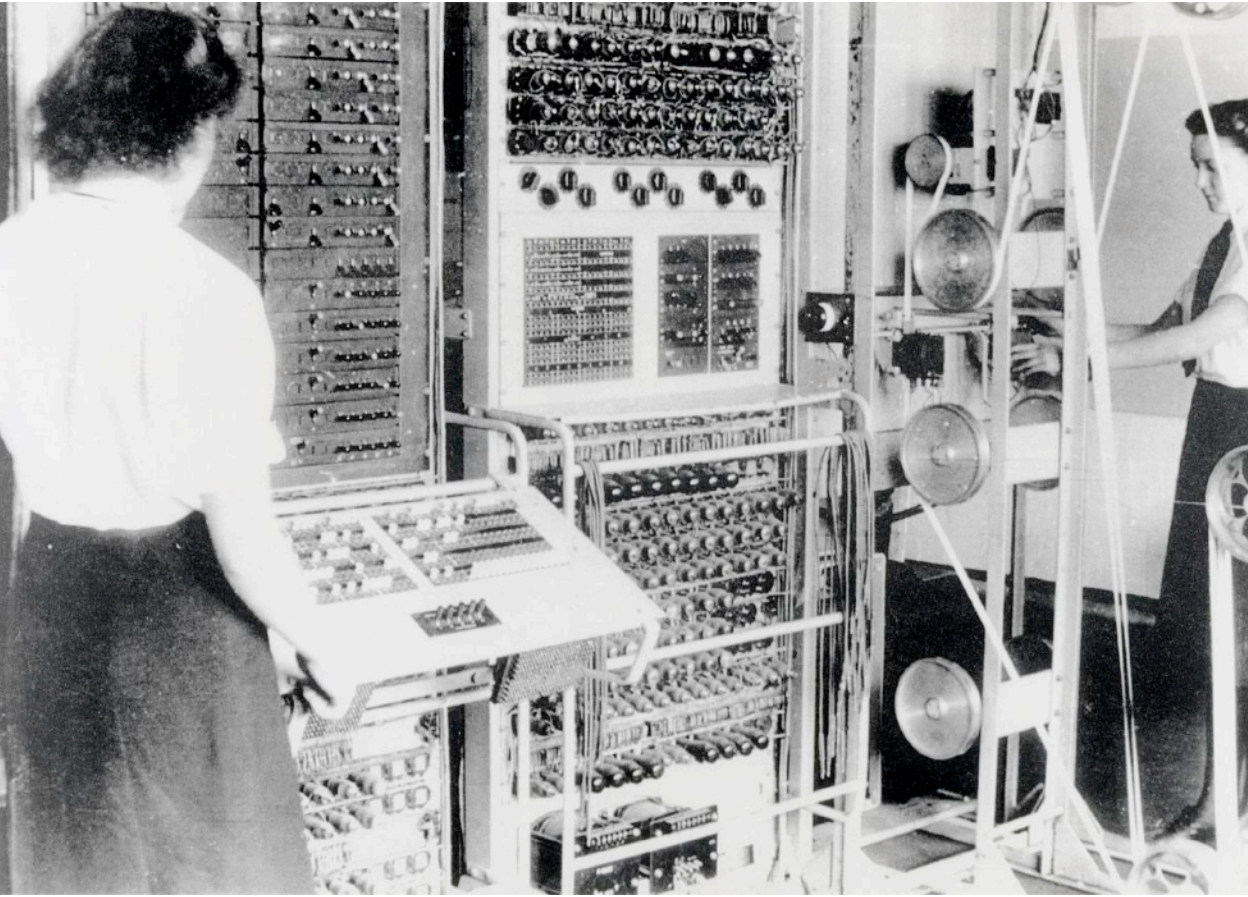


Where is the current fastest supercomputer in the Top 500 located?

- Europe
- Russia
- United States
- Japan
- China
- Somewhere else...



[menti.com \(7224 4307\)](https://menti.com/72244307)



Colossus (1943)

First “digital programmable computer”

Processed ~25,000 characters of input per second (5-way parallel),
no concept of memory or storage yet, used *paper tape* as input

Used during WWII in codebreaking (Lorenz cipher)

Existence was secret until mid-1970s

See it *in operation* at National Museum of Computing @ Bletchley Park (close to London)



CDC 6600 (1964)

"first supercomputer"

3 MFLOPS, up to 982 kilobytes of memory



Cray-2 (1985)

1.9 GFLOPS (first gigascale system)

4 vector processors, 256 MWords of memory



ASCI Red, US (1997)

12 years after 1+ GFLOPS

1.6 TFLOPS (first terascale system)

9,298 processors (76 nodes)

1,212 GB of RAM memory



Jaguar, US (2009)

12 years after 1+ TFLOPS...

1.75 PFLOPS (first petascale system)

18,688 AMD Opteron nodes (224,256 cores)

360TB of memory, 10PB of storage



Titan, US (2013)

17.58 PFLOPS - 18,688 nodes (~300,000 CPU cores)

Each node: 16-core AMD Opteron CPU + **1 NVIDIA Tesla GPU**

~700TB of memory (CPU+GPU), 40PB of storage



Piz Daint, Lugano - Switzerland (2016)

~25 PFLOPS

~7,500 Intel Xeon nodes, ~133,000 cores + 5,704 NVIDIA P100 GPUs

8.8 PB shared storage, Cray Aries interconnect

#3 in Top500 (June 2017)



MareNostrum 4, Barcelona (2017)

11.15 PFLOPS , 1.3 MW power

3,456 Intel Xeon nodes (165,888 cores)

14 PB shared storage, Omnipath interconnect

#13 in Top500 (June 2017)



画像の無断使用・無断転載を禁じます

©RIKEN



画像の無断使用・無断転載を禁じます

©RIKEN

Fugaku, Japan (2020)

#1 in Top500 in June 2020

158,976 nodes (7.6 million cores), **442 PFLOPS**

Fujitsu A64FX CPUs (Arm 64-bit)

Torus fusion interconnect, 150 PB of shared storage



El Capitan, US (2024)

Current #1 in Top 500 of supercomputers (since Nov'24)

~1 million CPU cores (AMD Genoa) + ~43,800 AMD MI300A GPUs

Slingshot interconnect (12.8 Tbits/sec)

2.82 exaFLOPS (peak)

First exascale system (in US)

Total cost: 500 million US \$

15 years after 1+ PFLOPS

Well, actually...



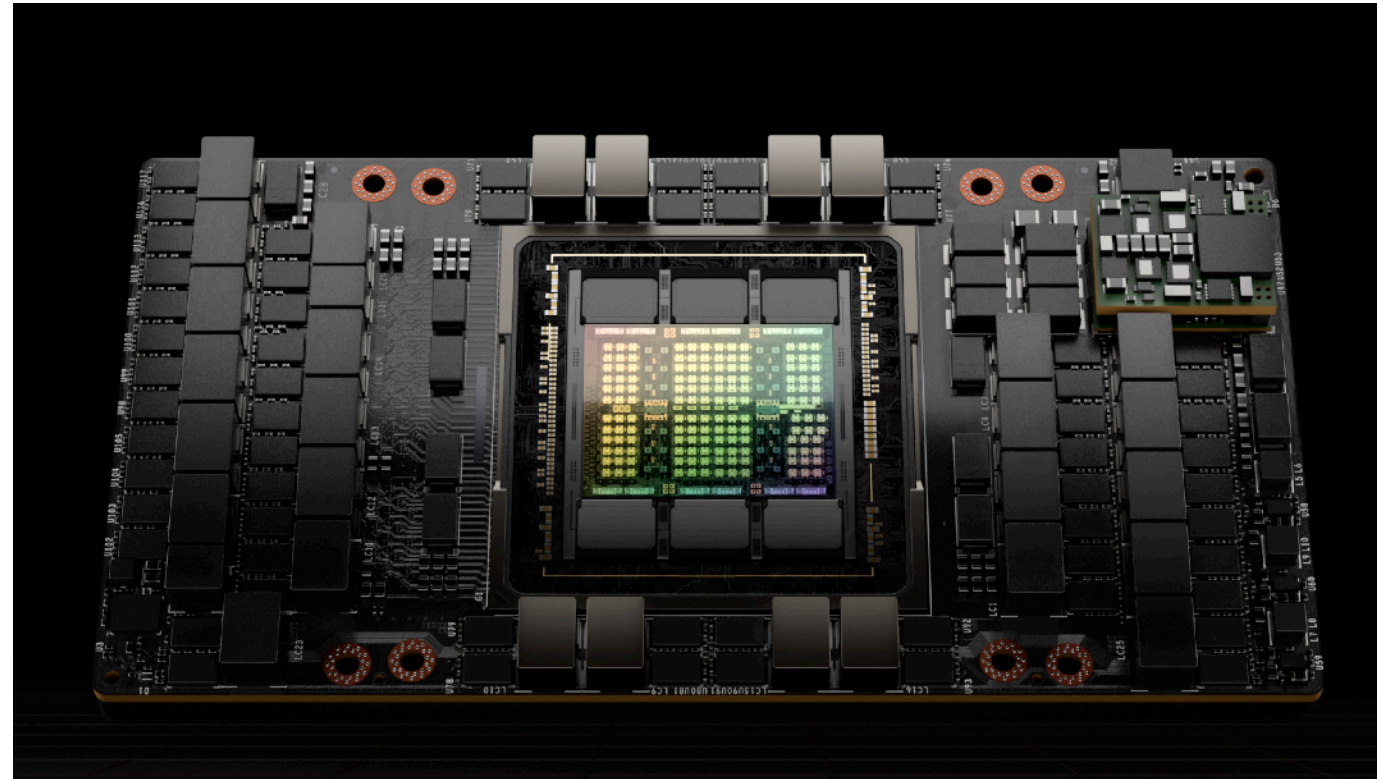
NVIDIA V100 GPU (2018)

~\$10,000

7.8 TFLOPS

Equivalent with #1 supercomputer in 2000!





NVIDIA H100 GPU (2023)

~\$25,000

34 TFLOPS

Equivalent with #1 supercomputer in June 2004!





iPhone X (2017)

~500 EUR

Includes NPU (Neural Processing Unit) of ~600 GFLOPS

Equivalent with #1 supercomputer in 1996!



POLL



Vlaanderen
is computing

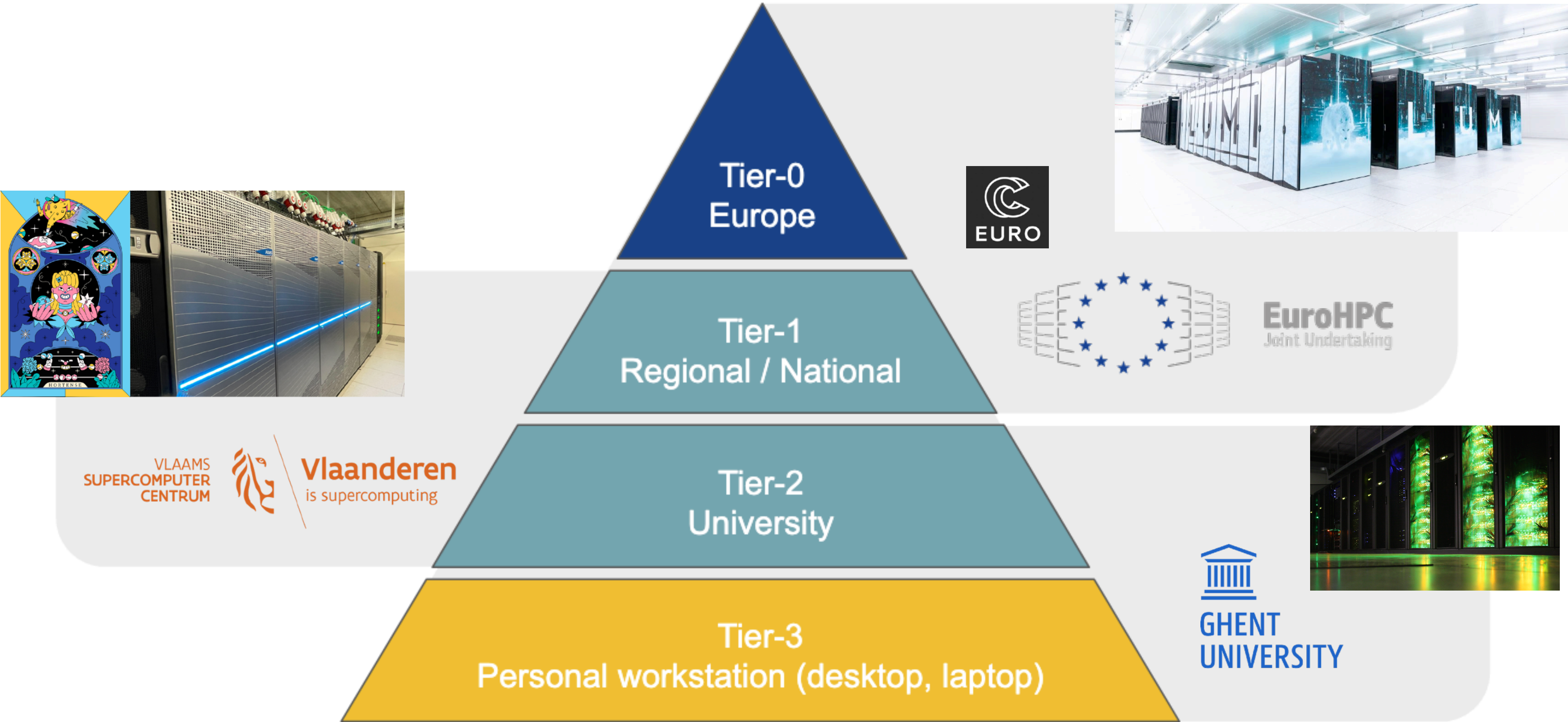
Where is the fastest supercomputer in Flanders?

- There is none
- Antwerp
- Bruges
- Brussels
- Ghent
- Leuven
- Somewhere else



[menti.com \(7224 4307\)](https://menti.com/72244307)

EUROPEAN TIERS OF SCIENTIFIC COMPUTING





HPC-UGent (since 2008)

Current Tier-2 infrastructure:

- 3(+1) CPU clusters, 3 GPU clusters, 1 debug/interactive cluster
- ~22,000 CPU cores in total, 3(+1) different generations of AMD CPUs
- 76 (+16) GPUs - NVIDIA V100, A100 (+ H100)
- All clusters with Infiniband interconnect
- ~4PB of shared storage in total

FLEMISH SUPERCOMPUTER CENTRE (VSC)

- Partnership between different Flemish universities
- Virtual centre with hubs in Ghent, Leuven, Brussels, Antwerp
- Shared infrastructure: account management, cloud, Tier-1, ...
- Funded by Flemish government (FWO) + universities
- More info: <https://www.vscentrum.be>



Vlaanderen
is computing



VSC TIER-2 INFRASTRUCTURE



Vlaanderen
is computing



More info: docs.vscentrum.be/compute.html

VSC TIER-1 COMPUTE “HORTENSE”



Vlaanderen
is computing

- Hosted, operated, and supported by HPC team at Ghent University since 2021
- 2x 384 CPU-only nodes (128-core AMD Rome or Milan CPUs) + 40 GPU nodes (4x NVIDIA A100)
- **Over 100,000 CPU cores + 160 GPUs in total!**
- High-speed Infiniband network (HDR-100) + 6PB of dedicated scratch storage



- Project-based access (free of charge, funded by FWO)
- 3 cut-off dates per year for submitting project proposals
- Project duration is typically 8 months
- 500k - 5M core hours (CPU-only) or 1k - 25k GPU hours

VSC TIER-1 COMPUTE “SOFIA”



Vlaanderen
is computing



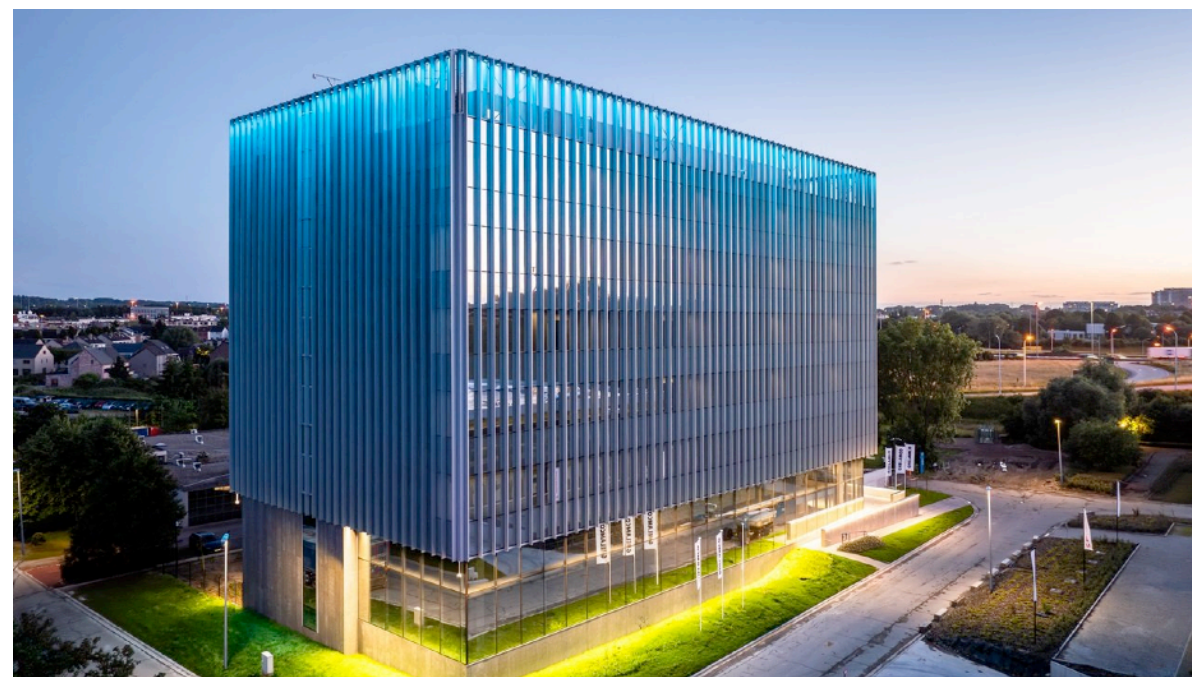
- Installed at Green Energy Park in Zellik, operated by VUB, almost production-ready
- Investment of 8.6 million Euro
- ~25,000 CPU cores (with 2GB or 8 GB RAM per core)
- **168 NVIDIA H200 GPUs**
- NVIDIA Quantum-2 InfiniBand network at 200 Gbps
- 4 visualisation nodes
- 4,460 TB of scratch storage + 24 TB NVME
- Peak power consumption of 315 kW,
200 kW water-cooled for improved energy efficiency



VSC TIER-1 COMPUTE “SOFIA”



Vlaanderen
is computing



EVOLUTION IN SCIENTIFIC COMPUTING: BIGGER, FASTER



Margaret Hamilton (1969)

Standing next to source code she developed for the Apollo moon mission



Katie Bouman (2019)

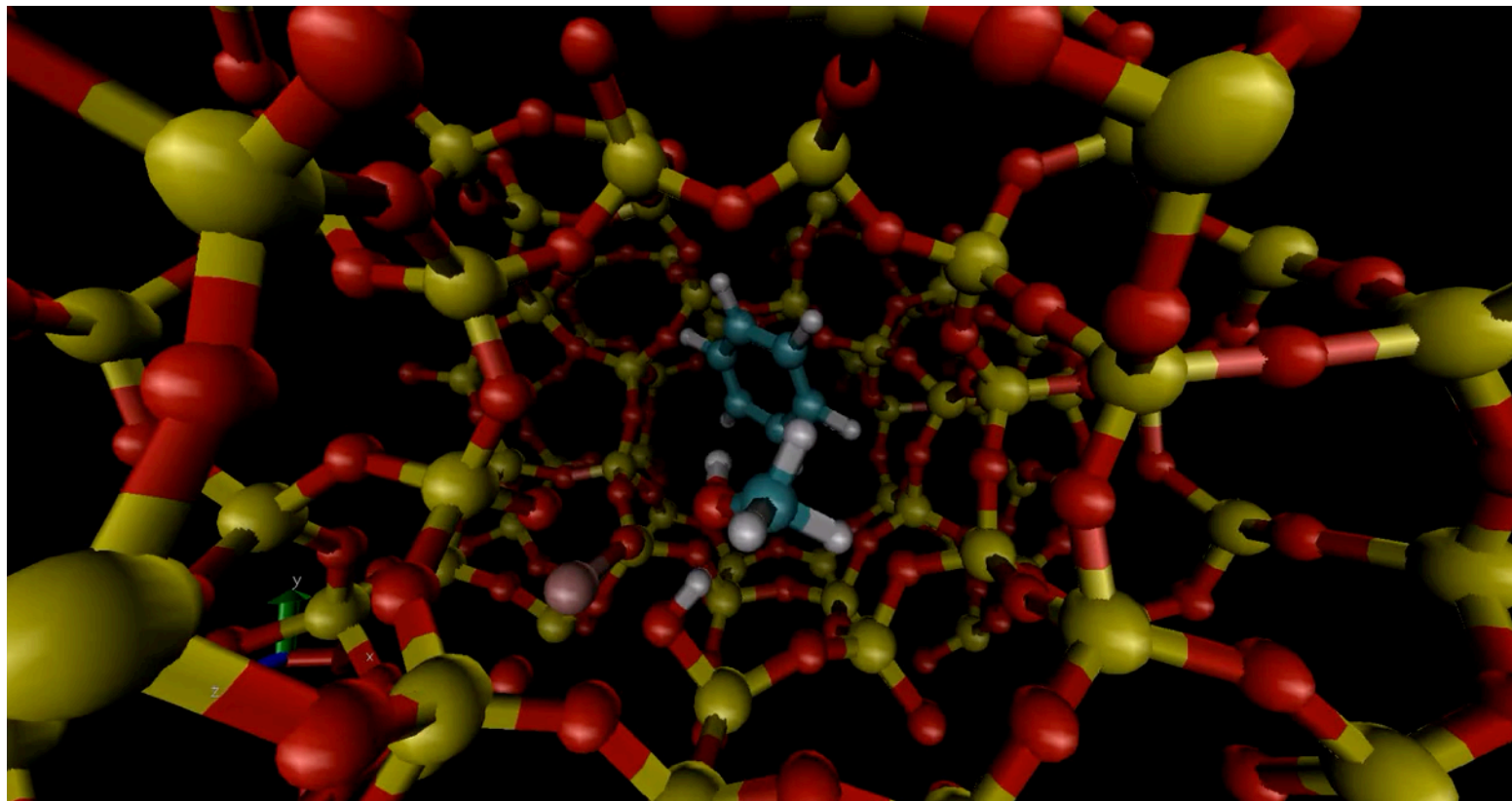
Showing hard drives that store the data collected to make the first image of a black hole (5PBs = 5000 TBs)



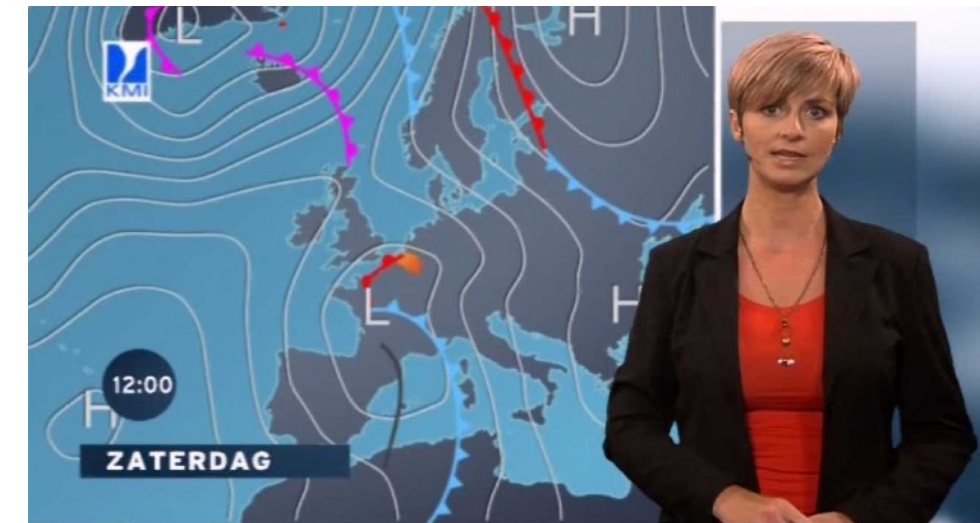
HIGH-PERFORMANCE COMPUTING (HPC)

- Running **large scale** calculations (on a supercomputer)
- Example: running a physics simulation on 100,000 cores
- Typically heavily relying on fast interconnect, shared storage, ...
- Related:
 - High-Throughput Computing (HTC): running **a lot** of (small & short) calculations
 - Scientific computing: simulations, data analysis, machine learning, ...
 - Training LLMs (AI): “just another HPC workload”

APPLICATIONS ACROSS ALL SCIENTIFIC DOMAINS



Material research ([CMM](#) @ UGent)



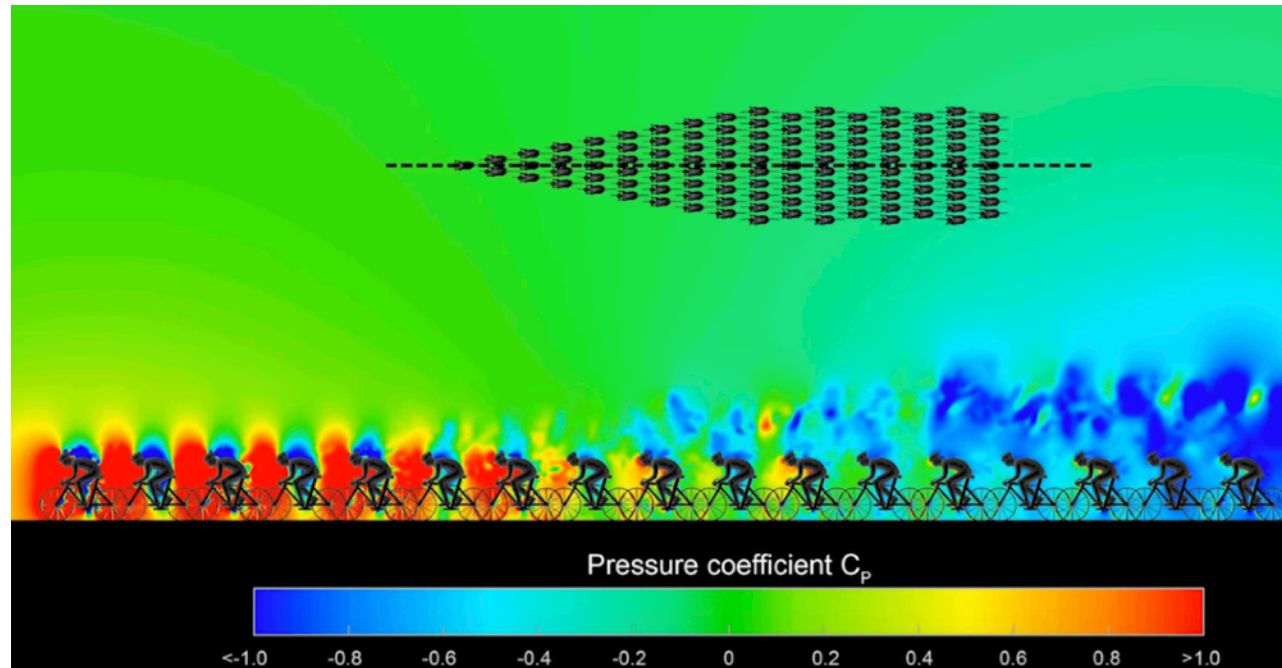
Weather predictions & climate simulations ([RMI](#))



DNA analysis, for example
in cancer research ([CRIG](#) @ UGent)

COMPUTATIONAL FLUID DYNAMICS

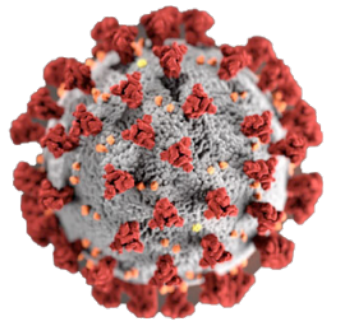
- Example: determine reduce of drag on riders well embedded in peloton (only ~5-10% compared to drag of isolated rider!)
- "Largest sports simulation to date" (July 2018)



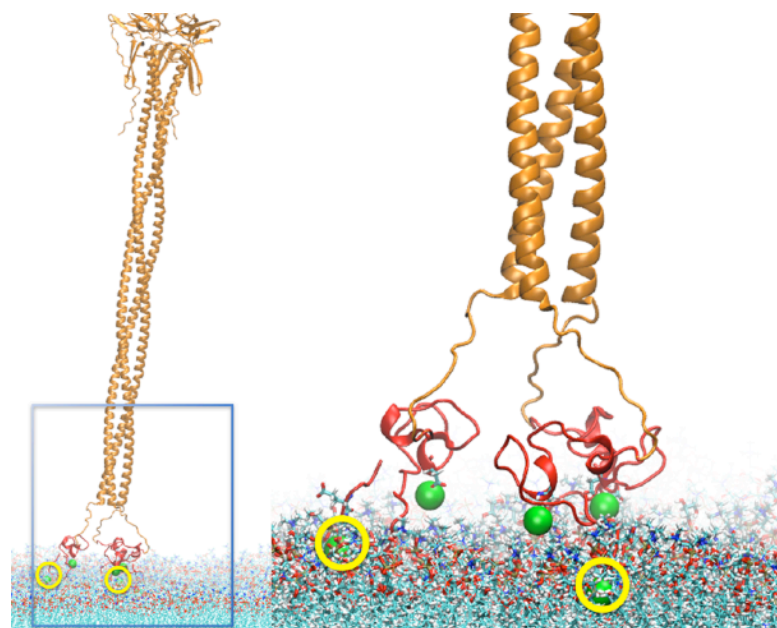
PELOTON POWER

Pervasive Simulation in Sport

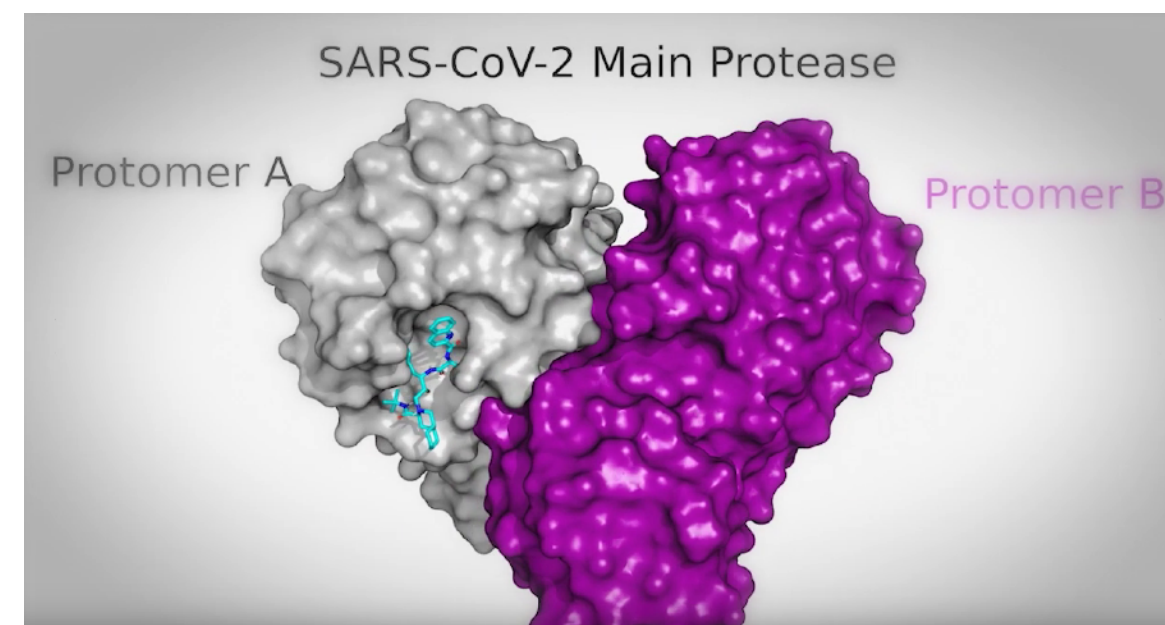
FIGHTING COVID-19 WITH HPC



- **COVID-19 High Performance Computing Consortium** (industry + academia)
- Free access for researchers to some of the largest supercomputers in the world
- ~600 PFLOPS of available compute resources in total
- CFD simulations of airflow in rooms & airplanes to evaluate distribution of particles
- Analysis of virus RNA to trace back how virus was spread, identify variants, ...
- This had massive positive impact on time-to-market for effective vaccines, drugs, etc.



<https://covid19-hpc-consortium.org>



https://www.youtube.com/watch?v=6GQs9M_O95A

TRAINING LARGE LANGUAGE MODELS

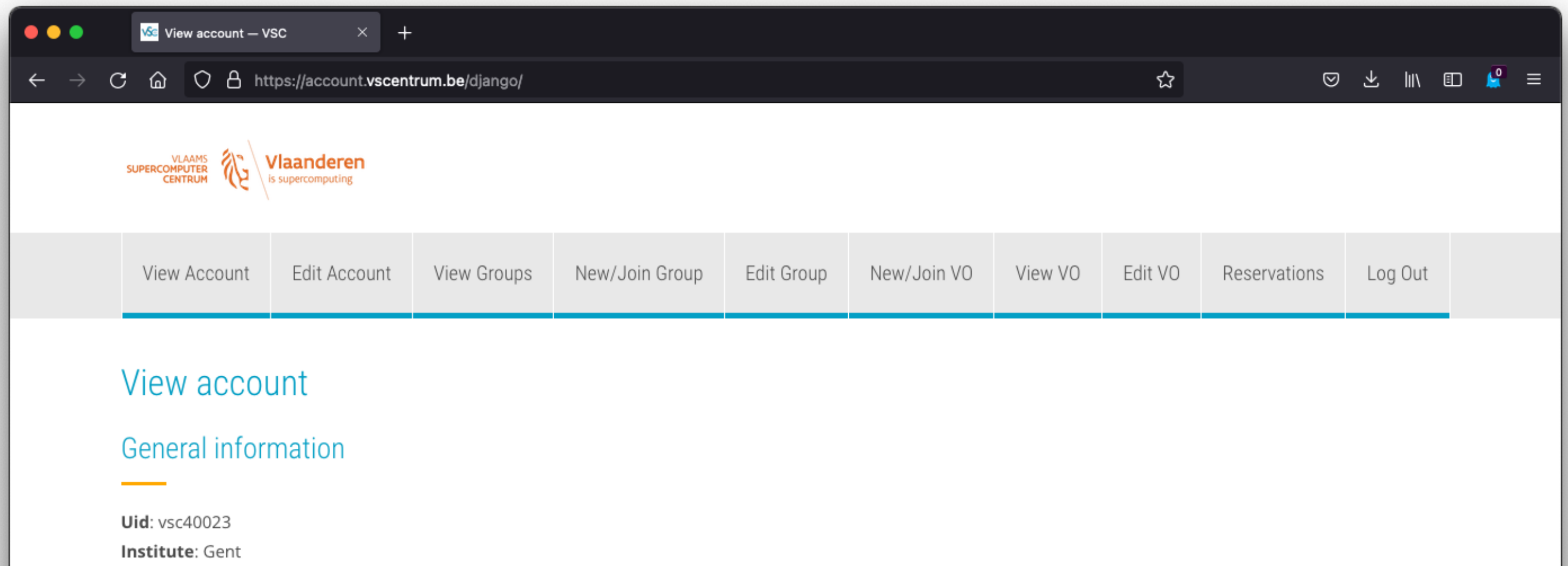
- Example HPC workload: training Large Language Models

For GPT-4o (one of the ChatGPT models):

- 25,000 NVIDIA A100 GPUs
 - Took 90-100 days
 - Estimated cost: \$100 million
 - Energy consumption: ~50 GWh
- Training LLMs requires lots of GPUs *in a single system* (fast interconnect)
- Dedicated “AI Factories” are currently being set up at various EuroHPC sites
 - Supercomputers with specific features (lots of GPUs) + dedicated support for AI workloads
 - https://www.eurohpc-ju.europa.eu/ai-factories_en

VSC ACCOUNT

- You need a VSC account to access VSC infrastructure (UGent, VUB, KUL, UA)
- See <https://ugent.be/hpc/en/access/faq/access>
- To manage your VSC account, use <https://account.vscentrum.be>



GETTING ACCESS TO HPC-UGENT TIER-2 CLUSTERS

Logging in



<https://login.hpc.ugent.be>

Web portal (no SSH key required)

OnDemand provides an integrated, single access point for all of your HPC resources.

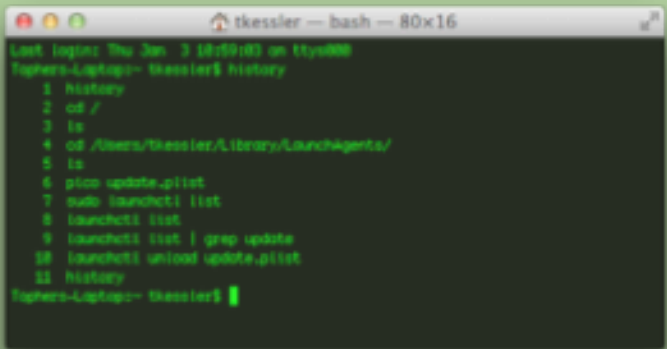
Status

STEVIN HPC-UGent infrastructure status on Thu, 08 Jun 2023 07:55:01

cluster	full nodes	free nodes	partial free nodes	total nodes	running jobs	queued jobs
swirot	102	0	10	116	214	728
slimy	6	16	45	67	111	29
victini	3	48	33	84	422	20907
julisk	5	0	4	9	29	11
doduo	29	0	72	101	1182	35515
acridige	8	0	1	9	26	16
doughard	9	7	16	32	7	9
gallade	11	0	4	15	17	183

`ssh login.hpc.ugent.be`

Terminal (requires SSH key)



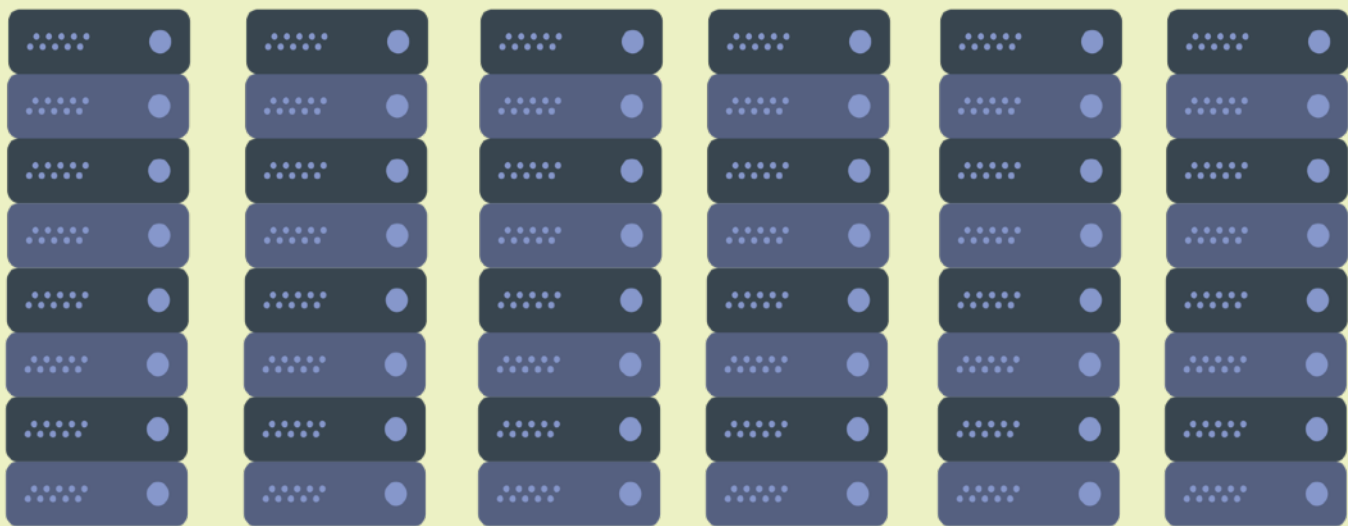
or

Login nodes



gligar09 or gligar10

Cluster (worker) nodes



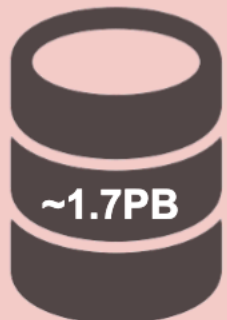
Shared filesystems



Home



Data



Scratch



Arcanine Scratch
(SSD)

HANDS-ON: LOGGING IN + UP/DOWNLOADING DATA

- VSC accountpage: <https://account.vscentrum.be>
- Login via HPC-UGent web portal: <https://login.hpc.ugent.be>
- Login via SSH: `ssh vsc40000@login.hpc.ugent.be`
- Uploading/downloading data via web portal
- Uploading/downloading data via WinSCP, scp, rsync, wget, cURL
- Editing files directly on the system (editor in terminal, or editor in web portal)

HANDS-ON: INTERACTIVE JOB

- Start an interactive job via `qsub -I # dash capital I`
- Make sure you are targetting the debug/interactive cluster!
First run: `module swap cluster/donphan`
- Try running a small Python script (Hello World) in the interactive job
- What do you notice in terms of environment in which you're working?
- To stop the session, run the `exit` command

- **Scientific software is made available via *environment modules***
- An environment module prepares the environment for using a particular software application
- Module naming scheme: `<name>/<version>-<toolchain>[-<suffix>]`
- Interacting with module files is done via the `module` command (Lmod)
- Load a module to prepare the session or job environment for using the software:

```
module load SciPy-bundle/2023.07-gfbf-2023a
```
- Modules that are required as dependencies will be loaded automatically
- To see list of currently loaded modules, run `module list` (or `ml`)

- To get an overview of *all* available modules, run `module avail` (or `ml av`)
- To see available versions for specific software, run `module avail soft_name/`
- To unload all currently loaded modules, run `module purge`
- Modules are installed using a particular toolchain (`foss, intel, ...`), which includes C/C++/Fortran compilers, MPI library, BLAS/LAPACK/FFT libraries
- **You should only combine modules that were installed with the same toolchain,** or a subtoolchain thereof (for example `foss/2023a + GCC/12.3.0`)
- See also HPC-UGent documentation: docs.hpc.ugent.be/running_batch_jobs/#modules
docs.hpc.ugent.be/available_software

CENTRAL SOFTWARE STACK VIA MODULES (EXAMPLE)

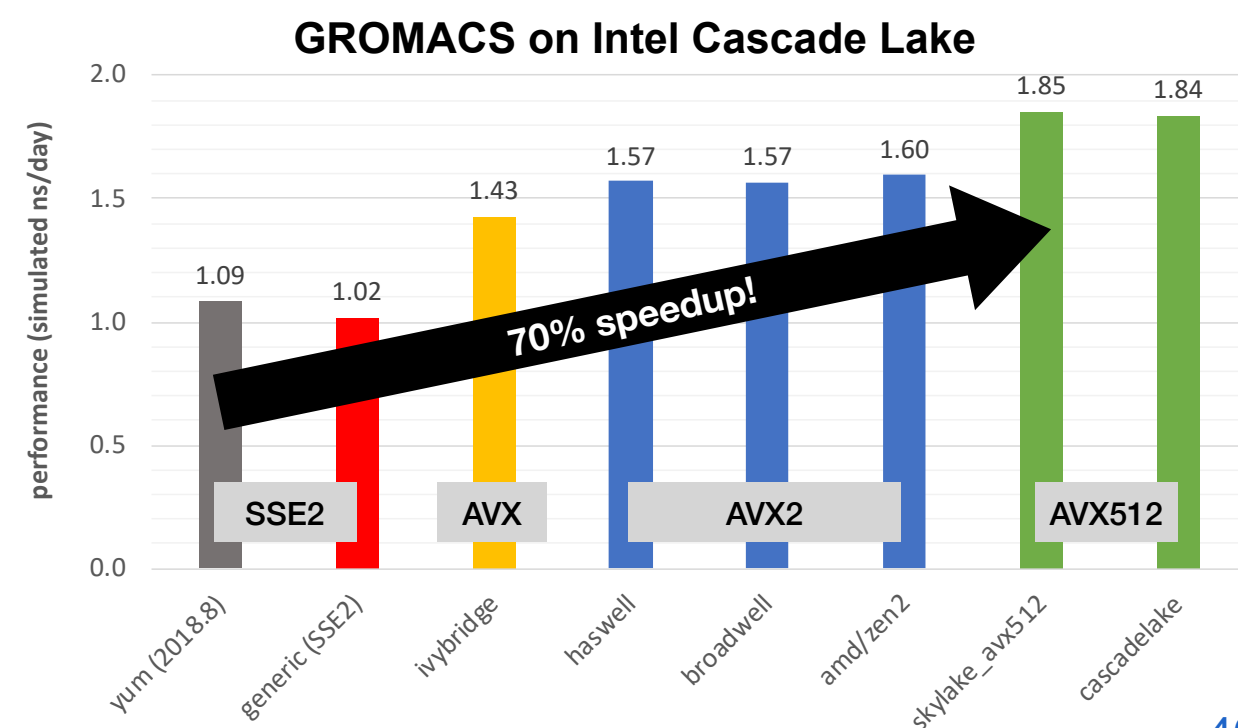
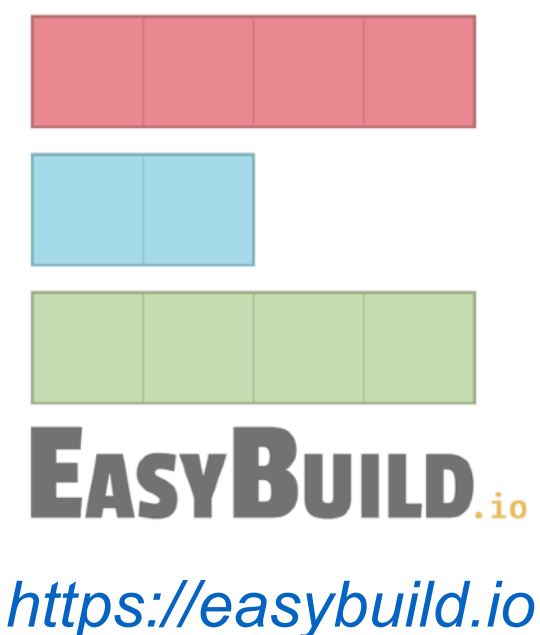
```
$ python -V; which python  
Python 3.6.8  
/usr/bin/python
```

```
$ python -c 'import numpy; print(numpy.__version__)'  
Traceback (most recent call last):  
  File "<string>", line 1, in <module>  
ModuleNotFoundError: No module named 'numpy'
```

```
$ module load SciPy-bundle/2023.07-gfbf-2023a  
$ python -V; which python  
Python 3.11.3  
/apps/gent/RHEL8/zen2-ib/software/Python/3.11.3-GCCcore-12.3.0/bin/python  
$ python -c 'import numpy; print(numpy.__version__)'  
1.25.1
```

GETTING SCIENTIFIC SOFTWARE INSTALLED

- Scientific software can be difficult to install (scientists are often not trained software developers...)
- Ideally built from source code (results in better performance)
- Used to be a very time-consuming, tedious and manual task...
- EasyBuild was created by the HPC-UGent team to **automate** this
- Open source software: github.com/easybuilders
- Implemented in Python
- Now used by hundreds of HPC sites worldwide!



SCIENTIFIC SOFTWARE MADE EESSI

- Getting scientific software installed can be a challenge...
- More and more software is used by researchers
- Diversity in hardware is increasing (Intel/AMD/Arm/RISC-V CPUs, ...)
- “Mobility of compute”: from laptop to cloud to HPC systems
- **European Environment for Scientific Software Installations (EESSI)**
 - Stack of *optimized* installations of scientific software for broad range of hardware
 - Works on any Linux system (incl. WSL in Windows, Linux VM on macOS, ...)
 - “On-Demand Streaming”: only what is needed is downloaded (like Netflix, Spotify)
 - Community effort
 - Broad set of software already available in EESSI, see https://eessi.io/docs/available_software



<https://eessi.io>

USING EESSI IS... EASY



- Step 0: Make sure that EESSI is available (if not, install it, see <https://eessi.io/docs>)
`ls /cvmfs/software.eessi.io/`
- Step 1: Initialize your shell environment to use EESSI (pick a version)
`source /cvmfs/software.eessi.io/versions/2025.06/init/lmod/bash`
- Step 2: Activate the software you want to use by loading the corresponding module(s)
`module load GROMACS/2025.2-foss-2025a`
- Step 3: Run the software!
`gmx . . .`

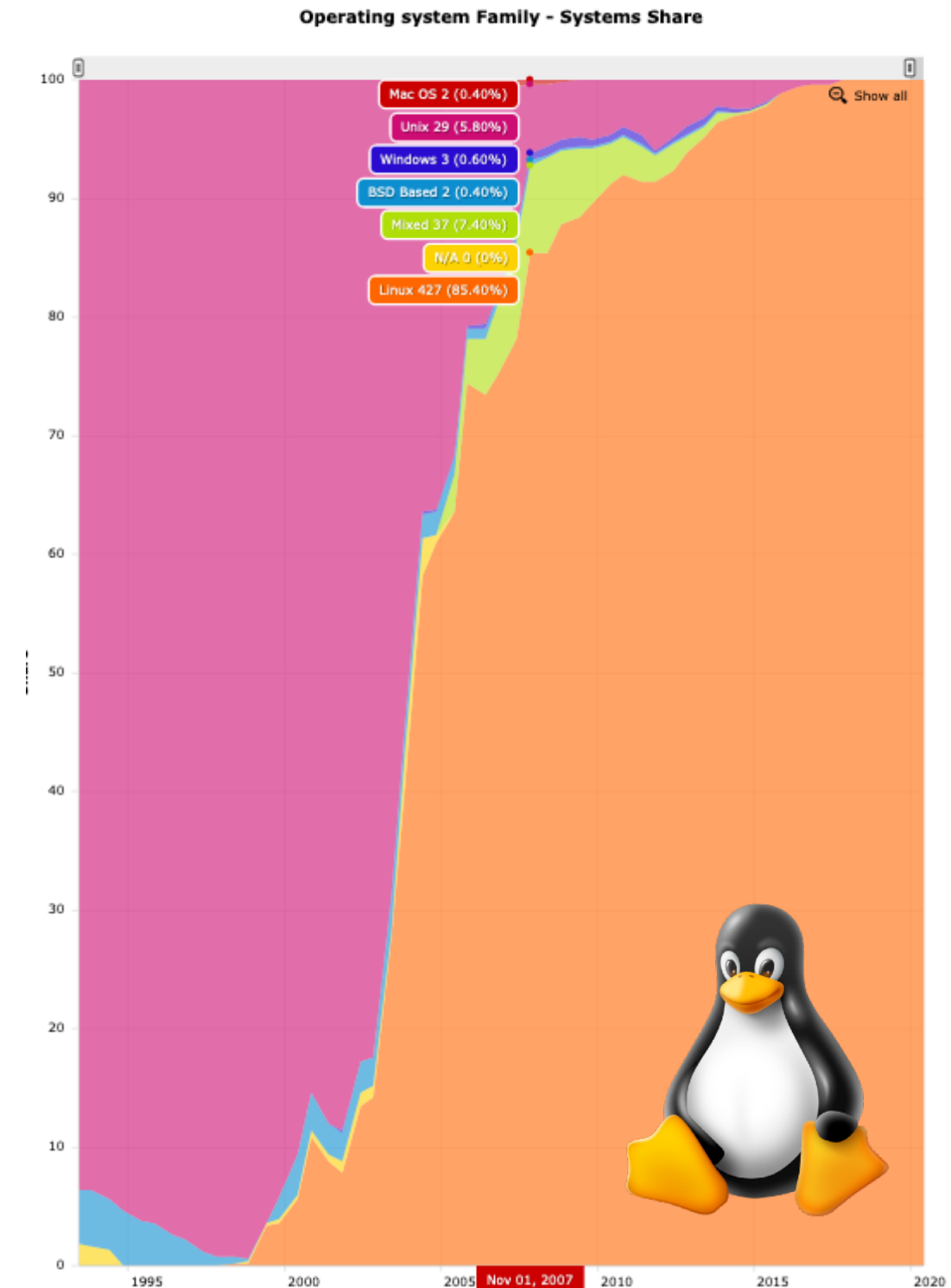
The workflow is **exactly** the same no matter which system you use...

HANDS-ON: EXPLORE AVAILABLE SOFTWARE

- Use the `module` command to explore the available software
- How many software packages are installed centrally?
- Which bioinformatics tools can you find?
- Can you find which module to load for the pandas Python package?
- Try loading a R-bundle-Bioconductor module, check what has changed.
(tip: try `module list`, try starting R and loading a Bioconductor package)

SYSTEM SOFTWARE ON HPC SYSTEMS

- **100% of supercomputers in Top 500 list runs Linux!**
- Mix of commercial and open source software on top
- Specific tools & services
 - Shared filesystems: IBM Storage Scale (GPFS), Lustre, ..
 - Job scheduling: Slurm, flux, ...
 - Environment modules: Tcl implementation, or Lmod (Lua)
 - Testing: ReFrame, JUBE
 - Custom scripting: Perl, Python, ...
 - Monitoring of systems and services (nagios, Kibana, ...)



SOFTWARE ENVIRONMENT ON TIER-2 + TIER-1 HORTENSE

- HPC-UGent Tier-2 + Tier-1 clusters all run **Red Hat Enterprise Linux 9** (RHEL9)
- Shared filesystems accessible from all login nodes + worker nodes:
 - On Tier-2: IBM Storage Scale (a.k.a. GPFS) for home/data/scratch/apps
 - On Tier-1: Lustre + NFS export of Tier-2 home/data
- Slurm as resource manager (with Torque/PBS frontend)
- Scientific software via environment modules system (via Lmod modules tool)
- Apptainer to run container images

(WORKER) NODES

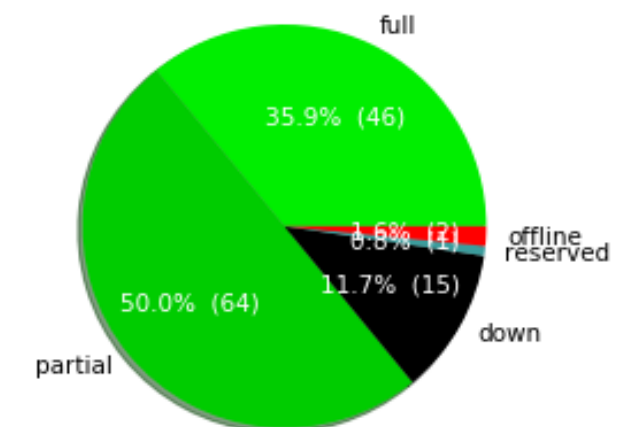
- Example cluster from the HPC-UGent

Tier-2 infrastructure: doduo (default cluster)

- 128 **nodes**, also referred to as “servers”
- 1 node is the equivalent of 1 computer (but with more cores, memory, ...)
- Also: compute nodes, worker nodes, ...

doduo

3501	3502	3503	3504	3505	3506	3507	3508
3509	3510	3511	3512	3513	3514	3515	3516
3517	3518	3519	3520	3521	3522	3523	3524
3525	3526	3527	3528	3529	3530	3531	3532
3533	3534	3535	3536	3537	3538	3539	3540
3541	3542	3543	3544	3545	3546	3547	3548
3549	3550	3551	3552	3553	3554	3555	3556
3557	3558	3559	3560	3561	3562	3563	3564
3565	3566	3567	3568	3569	3570	3571	3572
3573	3574	3575	3576	3577	3578	3579	3580
3581	3582	3583	3584	3585	3586	3587	3588
3589	3590	3591	3592	3593	3594	3595	3596
3597	3598	3599	3600	3601	3602	3603	3604
3605	3606	3607	3608	3609	3610	3611	3612
3613	3614	3615	3616	3617	3618	3619	3620
3621	3622	3623	3624	3625	3626	3627	3628



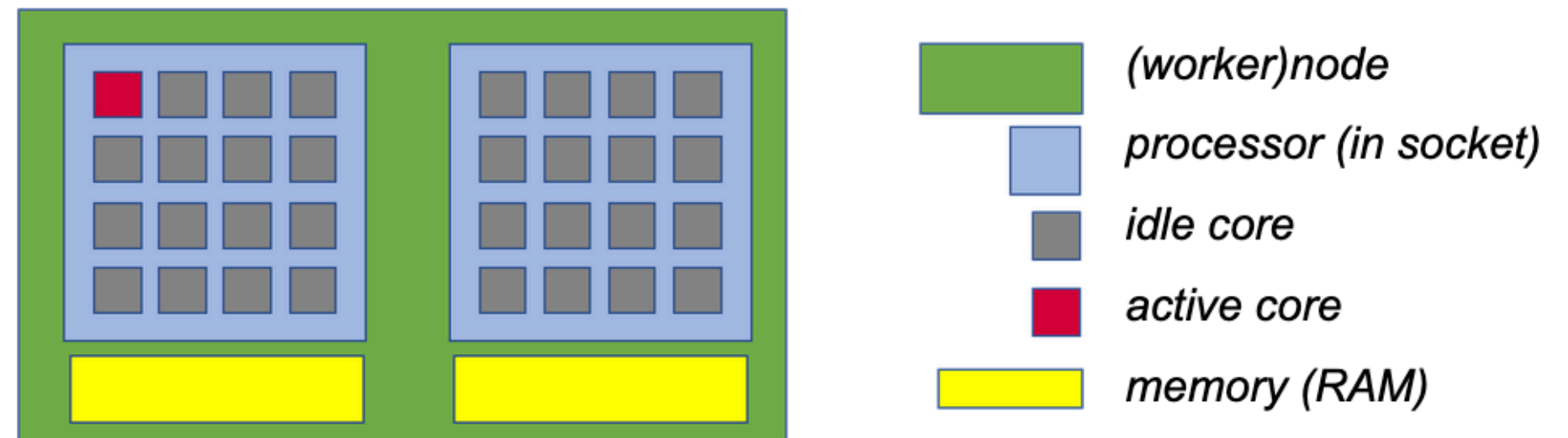
CORES, CPUS, PROCESSORS, SOCKETS

Modern servers, also referred to as **(worker)nodes** in the context of HPC, include one or more *sockets*, each housing a **multi-core processor** (next to memory, disk(s), network cards, GPUs, ...).

A modern (micro)processor consists of **multiple cores** that are used to perform calculations.

Example:

a single workernode
with two 16-core processors
running a single core job

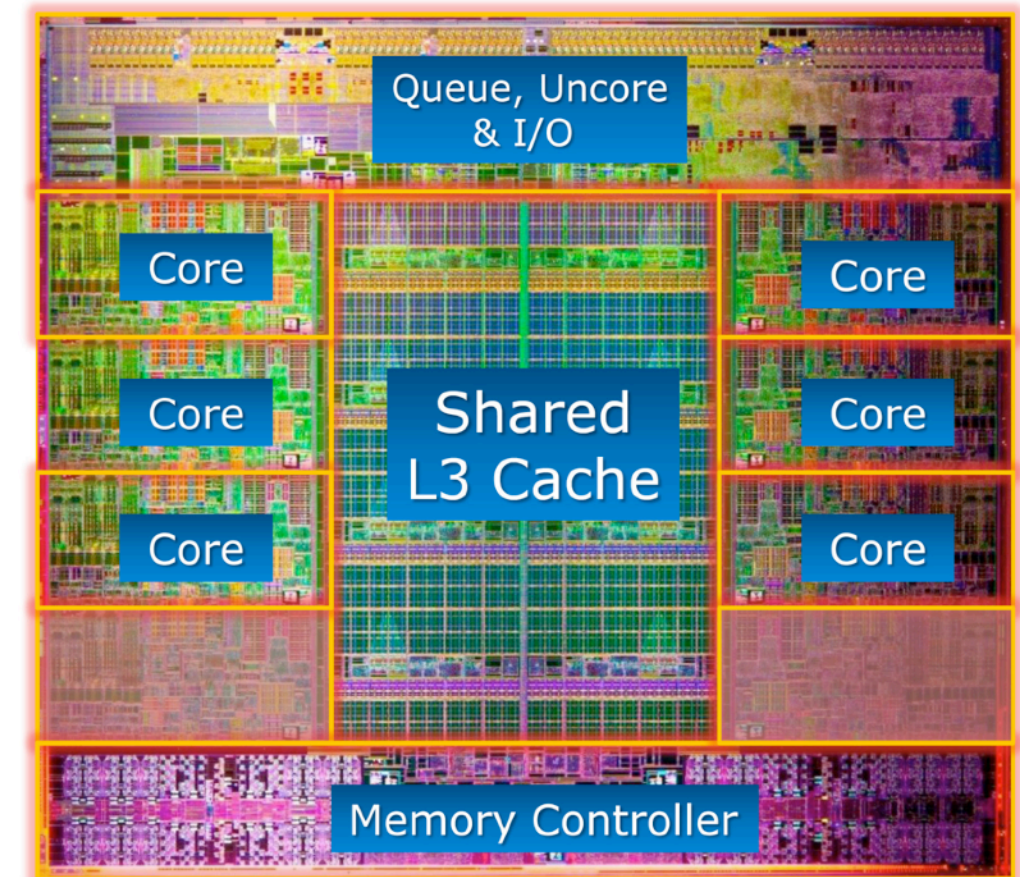
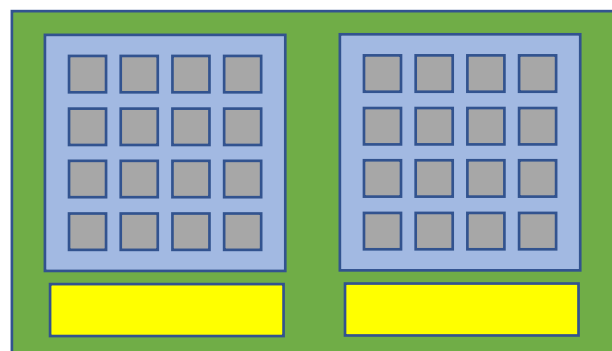


Not shown here: local disk, network cards, GPUs, ...

MODERN MULTI-CORE PROCESSORS

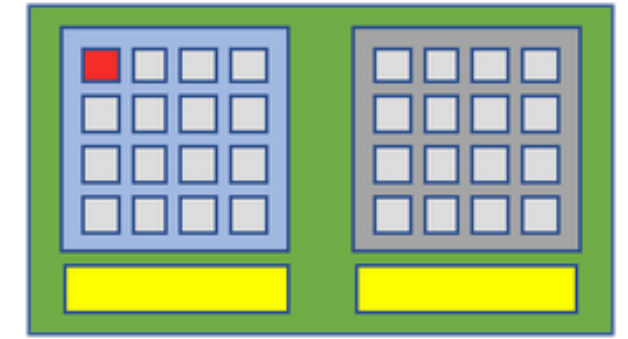


- CPU (central processing unit) a.k.a. (micro)processor
- **Multiple cores** to run multiple computations in parallel
- Hierarchy of small (~MBs) but (very) fast on-chip *cache* memory
- Channels to access memory & co
- Typically 2 multi-core processors per node (each housed in a *socket*)
- Simplified view (2x 16 cores):



SEQUENTIAL SOFTWARE

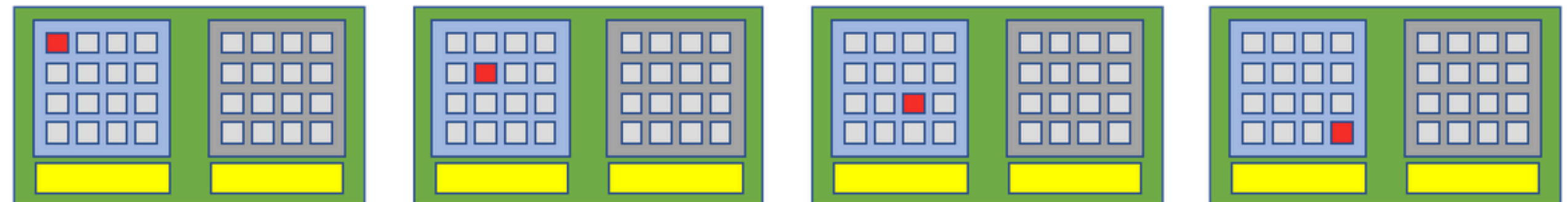
Sequential (a.k.a. serial) software does not do calculations in parallel, it only uses one **single core** of a single workernode.



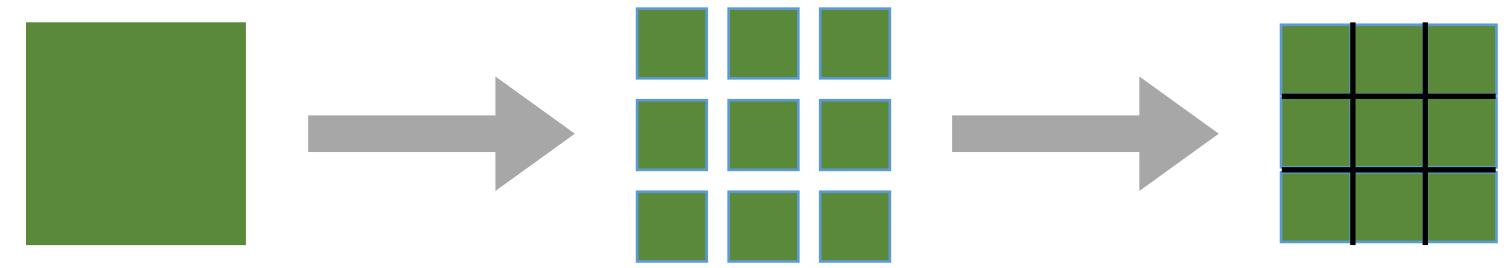
This type of software does not run faster by just throwing cores (or nodes) at it...

But, you can run multiple instances of the same program at the same time!

Example: running a Python script 100 times, each on 1 core, to quickly analyse 100 datasets



PARALLEL COMPUTING



- Long-running calculations can be done faster through **parallelization**
- Split up large problem into smaller problems, divide and conquer
- Challenge is to **keep all cores busy** (doing useful work)
 - Avoid *bottlenecks*: slow memory/disk, network latency, limited bandwidth
 - Avoid *load imbalance*: waiting for longest running part to be done
 - Avoid *overhead* (extra work) in breaking up problem + composing final result

TERMINOLOGY: PARALLEL SOFTWARE

In **parallel** software, many calculations are carried out simultaneously.

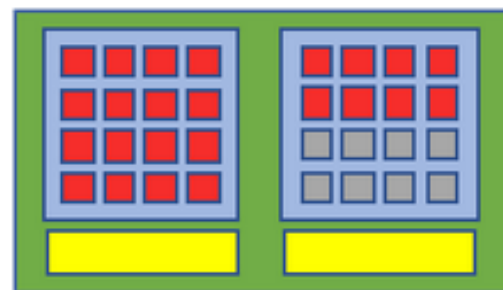
This is based on the principle that large problems can often be divided into smaller tasks, which are then solved concurrently (“in parallel”).

Example: OpenFOAM can easily use 160 cores at the same time to solve a CFD problem.

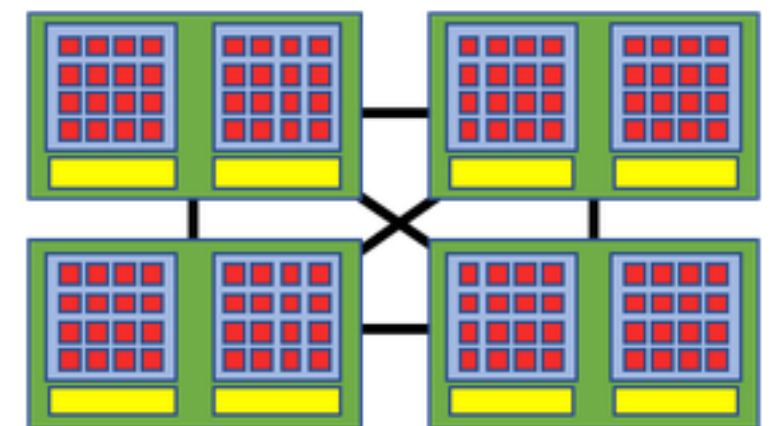
There are two common parallel programming paradigms (among others):

- **OpenMP** for shared memory systems (multi-threading) → using cores of a *single* node
- **MPI** for distributed memory systems (multi-processing) → using cores of multiple nodes

OpenMP software can use **multiple cores** in a **single node** by running **threads** that share memory



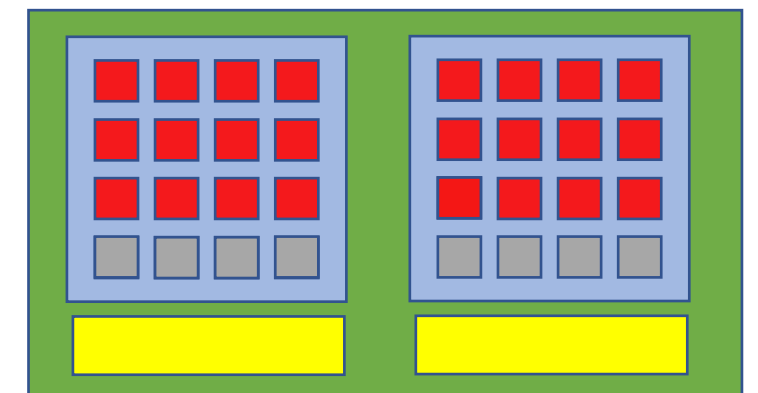
MPI software can use (all) cores in **multiple nodes** by running **processes** that communicate



SHARED MEMORY PROGRAMMING

- **Intra-node** parallel computing: using multiple *threads* on a **single node**
- Each thread runs calculations on 1 core
- Communication between threads via shared data structures in memory
- Limited to resources available in a single node (cores, memory, ...)
- Care must be taken when writing to shared data structures (use of locks)
- Typical programming paradigm: OpenMP

```
#pragma omp parallel for  
for(int x=0; x < width; x++){  
    ...  
}
```



OPENMP HELLO WORLD

```
#include <stdio.h>
#include <omp.h>

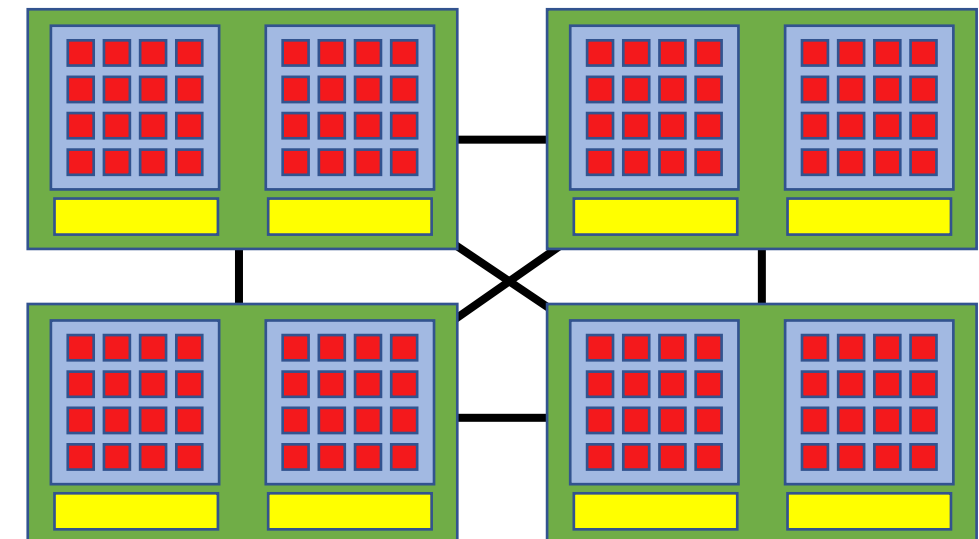
int main() {
    // Parallel region
    #pragma omp parallel
    {
        int thread_id = omp_get_thread_num(); // Get the thread ID
        int n_threads = omp_get_num_threads(); // Get total # threads
        printf("Hello, World! from thread %d of %d\n", thread_id, n_threads);
    }
    return 0;
}
```

Compile with:

```
gcc example_openmp.c -fopenmp -o example_openmp
```

DISTRIBUTED MEMORY PROGRAMMING

- **Inter-node** parallel programming: using cores in **multiple nodes**
- Multiple instances of same software are started (*processes, ranks*)
- Coordination by exchanging messages (who does what, partial results, ...)
- Fast interconnect is *crucial* if a lot of *communication* is needed to coordinate
- Typical programming paradigm: Message Passing Interface (MPI)
`MPI_Send(...)`
`MPI_Recv(...)`
- *Hybrid* parallel programming (OpenMPI + MPI):
multiple processes, each with multiple threads



MPI HELLO WORLD

```
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv); // Initialize MPI

    int world_rank, world_size;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank); // Get process rank
    MPI_Comm_size(MPI_COMM_WORLD, &world_size); // Get # processes

    printf("Hello, World! from process %d of %d\n", world_rank, world_size);
    MPI_Finalize(); // Finalize MPI
    return 0;
}
```

Compile with: `mpicc example_mpi.c -o example_mpi`

Run with: `mpirun -np 4 ./example_mpi` # or use `mympirun`

AMDAHL'S LAW

$$speedup(c) = \frac{1}{(1 - p) + \frac{p}{c}}$$

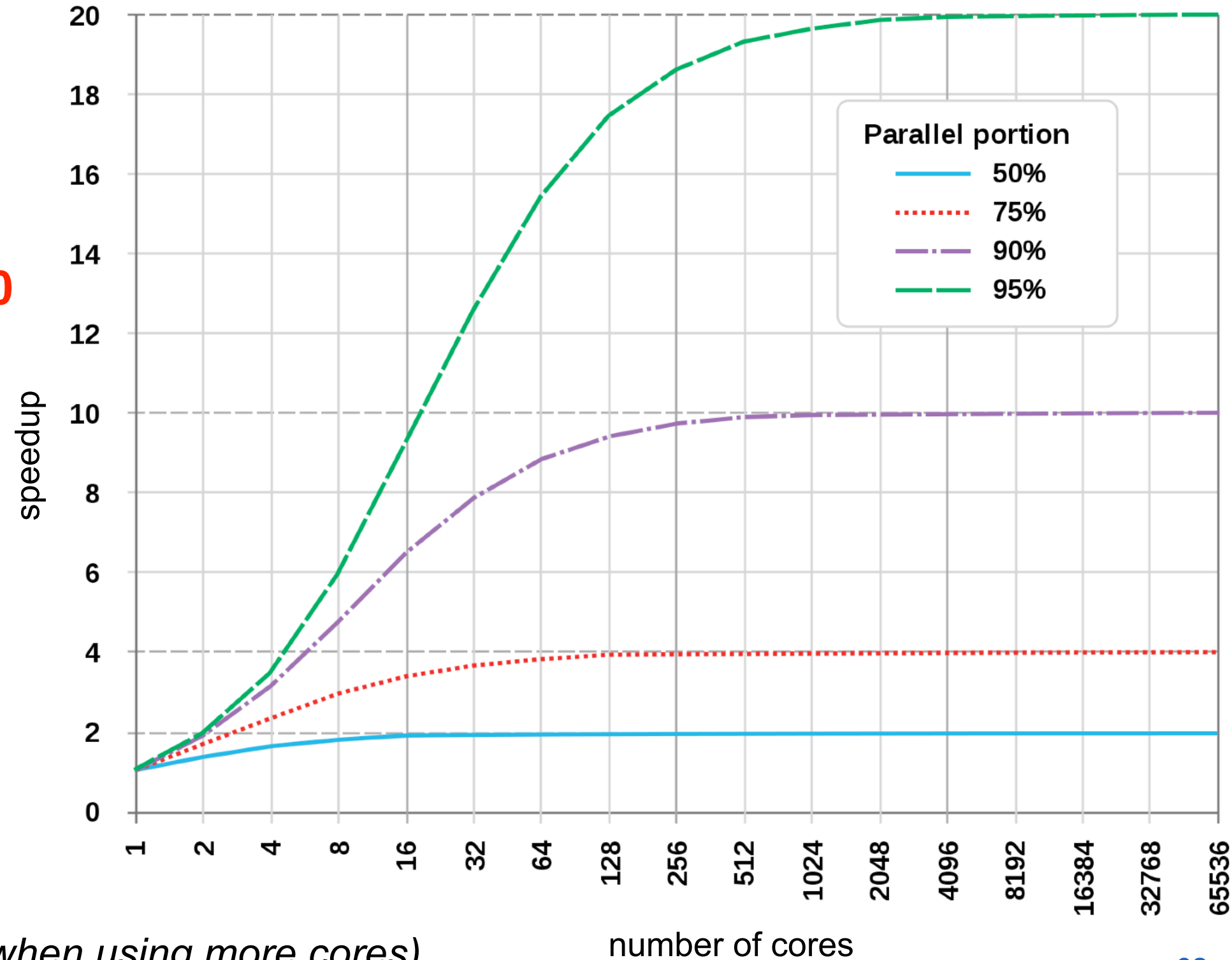
c : number of cores
 p : parallel portion ($[0, 1]$)

Examples:

- $p = 0.5$, 2 cores $\rightarrow$ speedup = 1.33
- $p = 0.5$, 64 cores $\rightarrow$ speedup = 2
- **$p = 0.99$, 1 million cores $\rightarrow$ speedup = 100**

Assumptions:

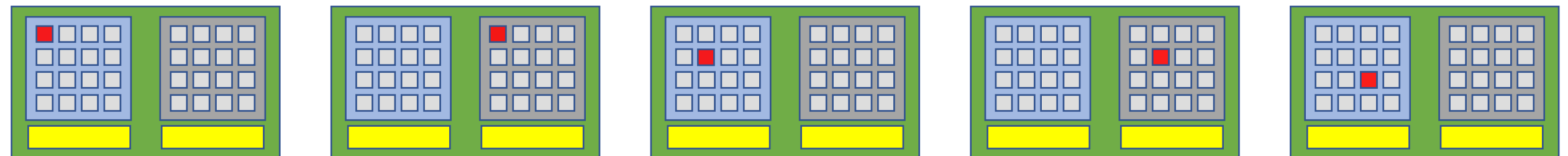
- Fixed workload (strong scaling*)
- Parallel portion is *perfectly* scalable
- No bottlenecks, overhead, etc.



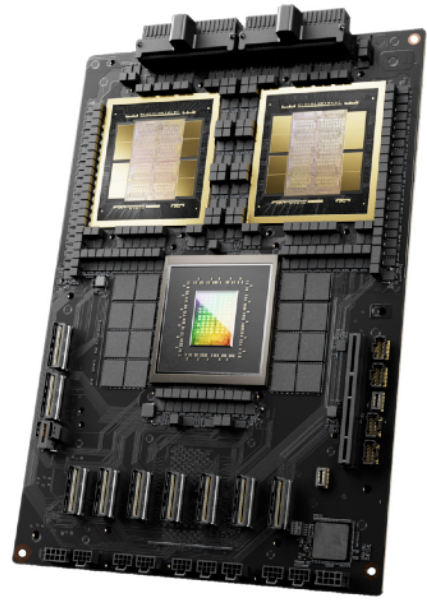
(* weak scaling: increase workload when using more cores)

EMBARASSINGLY PARALLEL TASKS

- What if you need to run a lot of very small (single-core) calculations?
- a.k.a. High-Throughput Computing (HTC)
- Only uses part of available resources (no need for fast interconnect or shared storage)
- Same software can be started multiple times within a single job
- Each instance is running independent of the others (no communication between them)
- Tools: worker, GNU parallel, dask, ...



ACCELERATED COMPUTING WITH GPUS



- Some type of calculations can benefit from special-purpose hardware
- Prime example of these *accelerators* are GPUs
- Originally created for detailed video game rendering (shaders, ray tracing, ...)
- Now also GPGPU: General Purpose Graphics Processing Unit
- Programming paradigm: CUDA, OpenCL, OpenACC, ...
- Difficult to use efficiently: separate GPU memory, lots of tiny cores, ...
- Potentially spectacular speedup: ~10-100x (but not for everything...)
- Example workloads: training LLMs + fast inference, molecular dynamics (GROMACS), ...

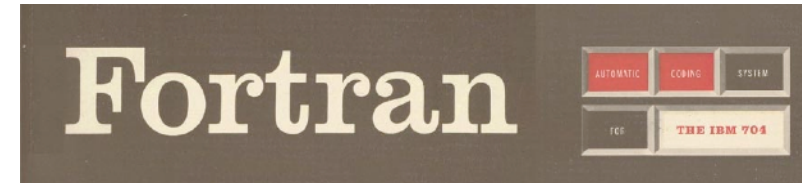
POP QUIZ

Most popular programming languages in scientific computing?

- Assembly
- C
- C++
- C#
- COBOL
- Delphi
- FORTRAN
- Go
- Java
- Javascript
- MATLAB
- Perl
- PHP
- Python
- R
- SQL
- Swift
- Ruby
- Visual Basic
- ...



PROGRAMMING LANGUAGES IN SCIENTIFIC COMPUTING



(back in [top 20 of programming languages](#), currently #12)



SUBMITTING AND MANAGING JOBS

- HPC-UGent clusters run **Slurm** as resource manager + job scheduler
- **Torque (PBS) frontend is (still) available and recommended** (via *jobcli* project)
 - `qsub` command to submit jobs, `qdel` command to delete jobs
 - `qstat` command to list queued + running jobs
 - `qalter` command to change jobs (before they start running)
 - `qhold` command to put jobs on hold, `qr1s` to release them again
- Use `--help` option to get list of available options for each command
- Use `--debug` option to get more information about what's going on behind the scenes
- Use `--dryrun` option to inspect what will be done (without actually doing it)

WHAT IS A JOB SCRIPT?

```
#!/bin/bash  
echo "I am a minimal job script"
```

A job script is (bash) shell script, a text file that includes shell commands, that specifies:

- The **resources** that are required by the calculation
(number of nodes & cores, amount of memory, how much time is required, ...)
- The **software** that is used for the calculation (usually via `module load` commands)
- The steps that should be done to execute the calculation (starting from home dir.),
specified as **shell commands**, typically:
 - 1) Staging in of input files
 - 2) Running the calculation
 - 3) Staging out of results

REQUIRED RESOURCES VIA #PBS DIRECTIVES

```
#!/bin/bash
#PBS -N solving_42           # job name
#PBS -l nodes=1:ppn=4        # single-node job, 4 cores
#PBS -l walltime=10:00:00    # max. 10h of wall time
#PBS -l vmem=50gb            # 50GB of (virtual) memory required
# rest of job script goes here ...
```

- Required resources can be specified via #PBS lines in job script
- Or via options to job submission command (`qsub -l ...`)
- **Maximum walltime of jobs on HPC-UGent clusters: 72 hours (3 days)**
- For longer calculations: break it up in shorter jobs, use a different (faster) cluster, use more cores (if software scales), use some form of “checkpointing”, ...

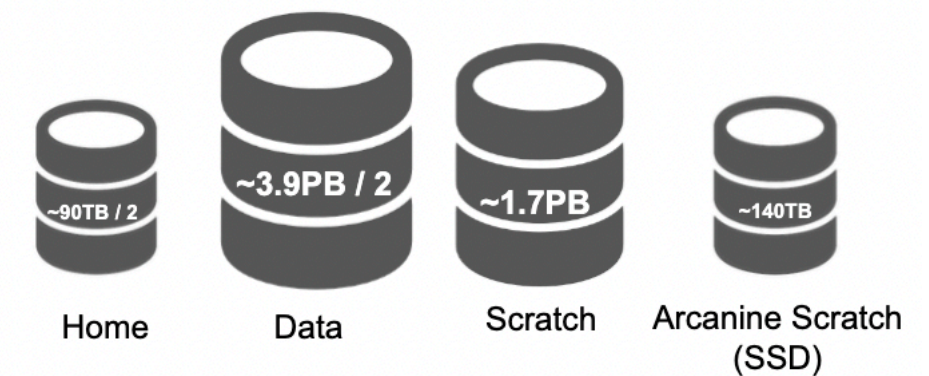
USEFUL ENVIRONMENT VARIABLES FOR JOB SCRIPTS

(most of these are only defined in the context of a running job!)

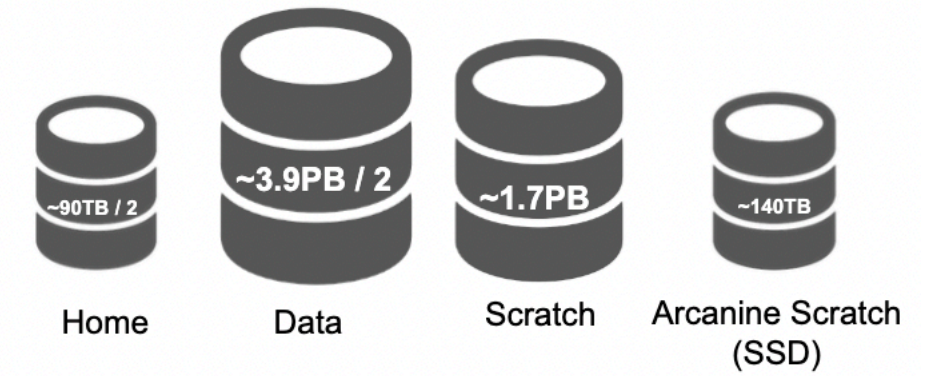
- **\$PBS_JOBID**: job id of running job
- **\$PBS_O_WORKDIR**: directory from which job was submitted on login node
 - It is common to use `cd $PBS_O_WORKDIR` at beginning of a job script
- **\$PBS_ARRAYID**: array id of running job
 - Only relevant when submitting array jobs (`qsub -t`)
- **\$TMPDIR**: unique *local* directory specific to running job
 - Cleaned up automatically when job is done, so make sure to copy result files!
- **\$EBROOTXYZ**, **\$EBVERSIONXYZ**: root directory/version for software package XYZ
 - Only available when module for XYZ is loaded

INPUT/OUTPUT DATA AND SHARED FILESYSTEMS

- See HPC-UGent docs: docs.hpc.ugent.be/running_jobs_with_input_output_data
- Think about input/output:
 - How and where will you *stage in* your data and input files?
 - How and where will you *stage out* your output and result files?
- Manually (on login nodes) vs automatically (as a part of job script)
- **Home filesystem** (\$VSC_HOME): only for limited number of small files & scripts
- **Data filesystem** (\$VSC_DATA*): 'long-term' storage, large files
- **Scratch filesystems** (\$VSC_SCRATCH*): for 'live' input/output data in jobs



STORAGE QUOTA (DISK SPACE)



- Home directory (`$VSC_HOME`): 3GB (fixed!)
- Personal data directory (`$VSC_DATA`): 25GB (fixed!)
- Personal scratch directory (`$VSC_SCRATCH`): 25GB (fixed!)
- Current quota usage can be consulted on [VSC accountpage](#)
- **More storage quota (100s of GBs, even TBs) available for *virtual organisations (VOs)***
see [dedicated section on VOs in HPC-UGent documentation](#)
- Additional quota can be requested via [VSC accountpage \(“Edit” tab\)](#)
- Shared directories with VO members: `$VSC_DATA_VO`, `$VSC_SCRATCH_VO`
- Personal VO subdirectories: `$VSC_DATA_VO_USER`, `$VSC_SCRATCH_VO_USER`

FULL EXAMPLE JOB SCRIPT (SINGLE-CORE JOB)

```
#!/bin/bash
#PBS -N count_example      # job name
#PBS -l nodes=1:ppn=1      # single-node job, single core
#PBS -l walltime=2:00:00   # max. 2h of wall time

# load Python 3.13, with batteries included (extra PyPI packages)
module load Python-bundle-PyPI/2025.07-GCCcore-14.3.0
# copy input data from location where job was submitted from
cp $PBS_O_WORKDIR/input.txt $TMPDIR
# go to temporary working directory (on local disk) & run Python code
cd $TMPDIR
python -c "print(len(open('input.txt').read()))" > output.txt
# copy back output data, ensure unique filename using $PBS_JOBID
cp output.txt $VSC_DATA/output_${PBS_JOBID}.txt
```

FULL EXAMPLE JOB SCRIPT (MULTI-NODE MPI JOB)

```
#!/bin/bash
#PBS -N mpi_hello           # job name
#PBS -l nodes=2:ppn=4       # 2 nodes, 4 cores per node
#PBS -l walltime=2:00:00    # max. 2h of wall time

module load OpenMPI/5.0.8-GCC-14.3.0
module load vsc-mypirun

# go to working directory, compile and run MPI hello world program
cd $PBS_O_WORKDIR
# C code for MPI Hello: https://mpitutorial.com/tutorials/mpi-hello-world
mpicc mpi_hello.c -o mpi_hello
mympirun ./mpi_hello
```


JOB OUTPUT FILES

- **Your job script may produce informative, warning, and/or error messages.**
 - Two output files are created for each job: stdout (* .o*) + stderr (* .e*)
 - Located in directory where job was submitted from (by default)
 - Messages produced by a particular command in the job script can be "caught" and redirected to a particular file instead:

```
example > out.log 2> err.log
```


(see [*dedicated section of our Linux tutorial*](#) for more details)
- In addition, the software used for the calculation may have generated additional output or result files (which is very software-specific).

JOB SUBMISSION AND MANAGEMENT

- Submit job scripts from a login node to a cluster for execution using `qsub` command:

```
$ module swap cluster/donphan
```

```
$ qsub example.sh
```

```
12345
```

- An overview of the active jobs is available via the `qstat` command:

```
$ qstat      # or qstat -ant
```

Job ID	Name	User	Time Use	S	Queue
12345	example	vsc40000	1:32:57	R	donphan

- To remove a job that is no longer necessary, use the `qdel` command: `qdel 12345`

JOB SCHEDULING

- All HPC-UGent clusters use a **fair-share scheduling** policy.
- No guarantees on when job will start (and impossible to predict), so **plan ahead!**
- Job priority is determined by various factors:
 - *Historical usage*
 - Aim is to balance usage over users
 - Infrequent/frequent users => higher/lower priority
 - *Requested resources* (# nodes/cores, walltime, memory, ...)
 - Larger resource request => lower priority
 - *Time waiting in queue*
 - Queued jobs get higher priority over time

HANDS-ON: SUBMITTING JOBS

- Single-core “Hello World” job
- Running a Python or R script in a job
- Multithreaded “Hello World” jobs (OpenMP)
- Multiprocess “Hello World” job (MPI)

Tip: use donphan cluster to avoid waiting in the queue

HANDS-ON: RUNNING BLAST

- Try running the `makeblastdb` and `blast` command from this BLAST mini-tutorial:
https://github.com/enormandeau/ncbi_blast_tutorial

- Start by cloning the GitHub repository in your VSC account
- Create a separate job script for running `makeblastdb` and `blast`
- Use the latest version of BLAST+ that is available on the HPC-UGent clusters
Consider using EESSI, so you can run the exact same script on your laptop!

- Submit the job scripts to the cluster, make sure they run correctly

Tip: use donphan cluster to avoid waiting in the queue

PYTHON VIRTUAL ENVIRONMENTS (VENV'S)



- Isolates project dependencies from the system-wide Python installation
- Makes local installation of Python packages (which are not available via modules) easier
- Recommendations & caveats on HPC clusters:
 - Prefer using **Python software installed as module** (if available): better performance
 - **Activate** the Python virtual environment with the **same modules loaded**
 - Virtual environment built on one cluster **will likely not work** on another cluster (diff. OS / CPU)
- Making a virtual environment (manually):

```
$ module load SciPy-bundle/2023.11-gfbf-2023b
$ python -m venv example-venv
$ source example-venv/bin/activate # activate virtual environment (similar to 'module load')
$ pip install example              # install package not available as module
$ python script.py                 # use virtual environment to run Python script
$ deactivate                       # deactivate virtual environment (like 'module unload')
```


VSC-VEENV: PYTHON VIRTUAL ENVIRONMENT WRAPPER SCRIPT



- Encapsulates the creation and management of Python virtual environments
- Facilitates working with Python virtual environments across HPC-UGent clusters
- It makes sure that:
 - The same modules are loaded on each activation
 - A new virtual environment is created for each cluster OS & CPU microarchitecture
- Arguments for the `vsc-venv` tool:
 - ***modules.txt*** (optional): file with list of names of environment modules to load
 - ***requirements.txt*** (required): file with list of Python packages to install

VSC-VENV: EXAMPLE + DEMO



modules.txt:

```
SciPy-bundle/2023.11-gfbf-2023b  
Pillow/10.2.0-GCCcore-13.2.0
```

requirements.txt:

```
beautifulsoup4==4.12.3
```

```
$ module load vsc-venv
```

```
$ source vsc-venv --activate --requirements requirements.txt --modules modules.txt
```

```
$ python --version # Python provided by dependency of SciPy-bundle module
```

```
Python 3.11.5
```

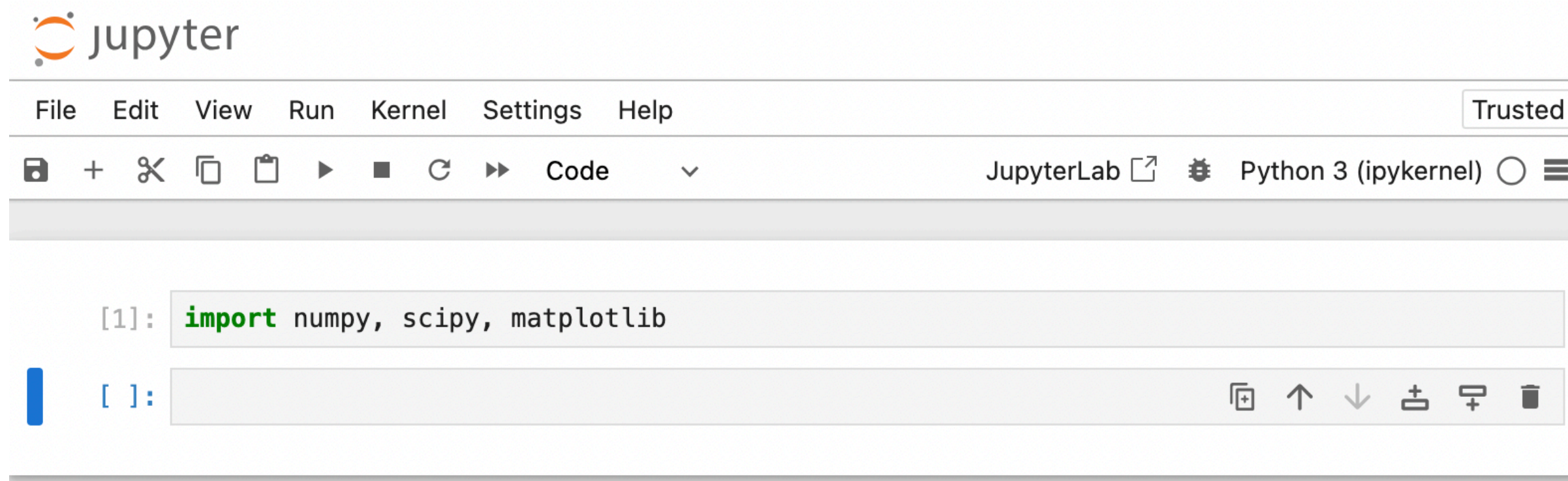
```
$ python -c "import PIL" # import PIL from Pillow module
```

```
$ python -c "import numpy" # import numpy from SciPy bundle
```

```
$ python -c "import bs4" # import bs4 from beautifulsoup4 package
```

HANDS-ON: JUPYTER NOTEBOOK

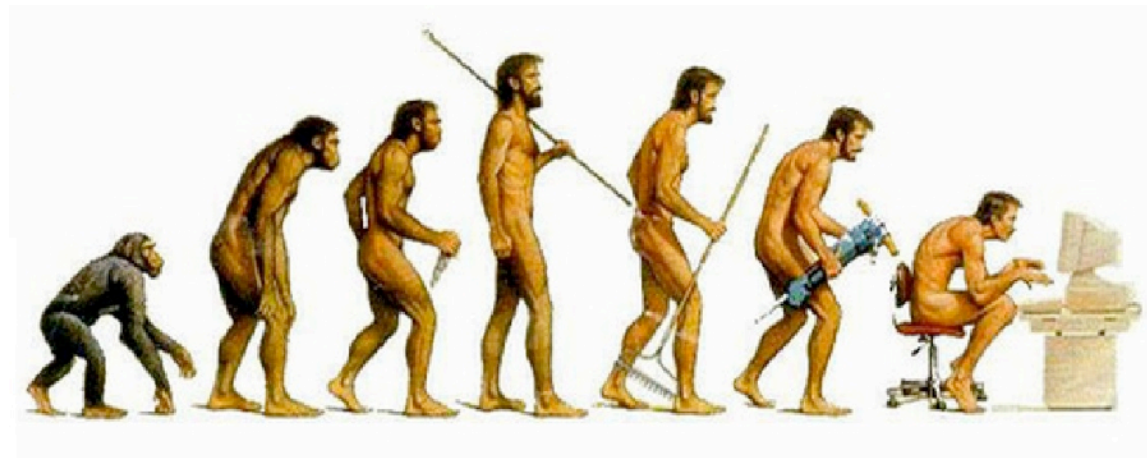
- Start a Jupyter notebook via “Interactive apps” on HPC-UGent web portal
- Use “donphan” cluster to avoid waiting in the queue
- **Make sure that numpy, scipy, matplotlib Python packages can be used**



POLL

What will be the next big (r)evolution in scientific computing?

- Zettascale supercomputers
- End of Intel/AMD processors as most prominent CPUs, switch to Arm
- High-performance RISC-V microprocessors
- HPC in the cloud
- Quantum computing
- Something else



Questions?

Kenneth Hoste

HPC user support

GHENT UNIVERSITY

INFORMATION AND COMMUNICATION TECHNOLOGY (DICT)

✉ kenneth.hoste@ugent.be - hpc@ugent.be

<https://www.ugent.be/hpc>



HANDS-ON: EXAMPLE SCRIPTS

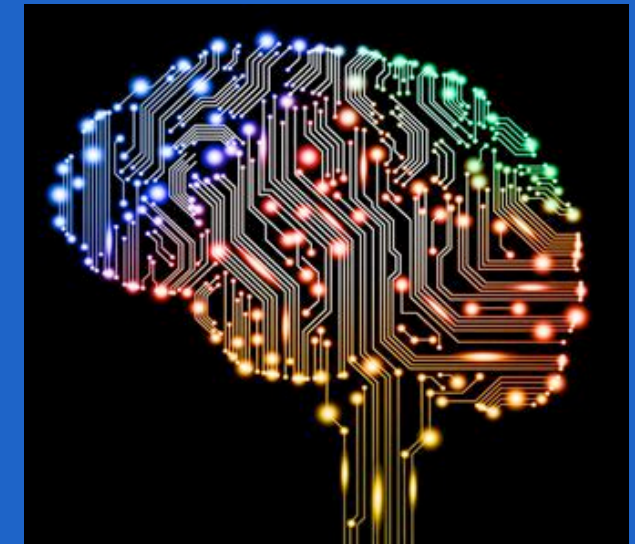
- Example scripts for hands-on are available on HPC-UGent login nodes in
`/apps/gent/tutorials/HOWEST-hands-on-20260306`
- You can copy them into your VSC account (so you can edit them):

```
cp -a /apps/gent/tutorials/HOWEST-hands-on-20260306 $HOME
```




✚ WOULD YOU LIKE TO KNOW MORE?

EXTRA MATERIAL



artificial intelligence



cloud computing

TRENDS IN HPC

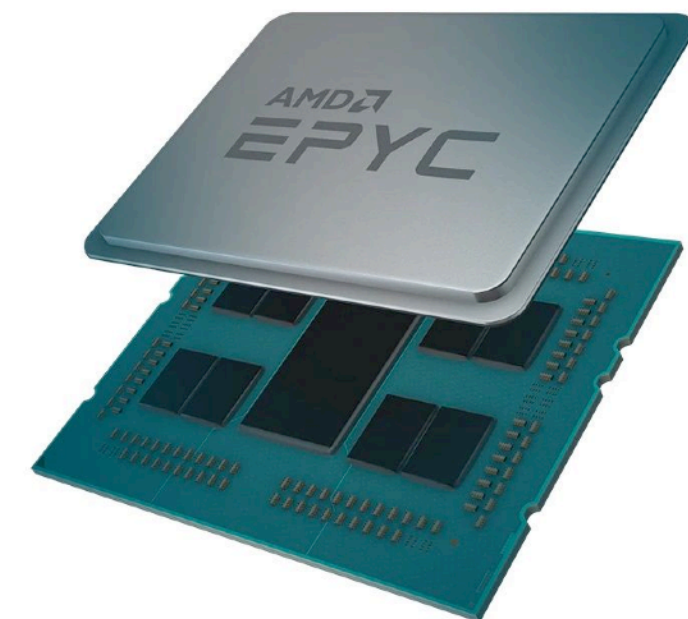
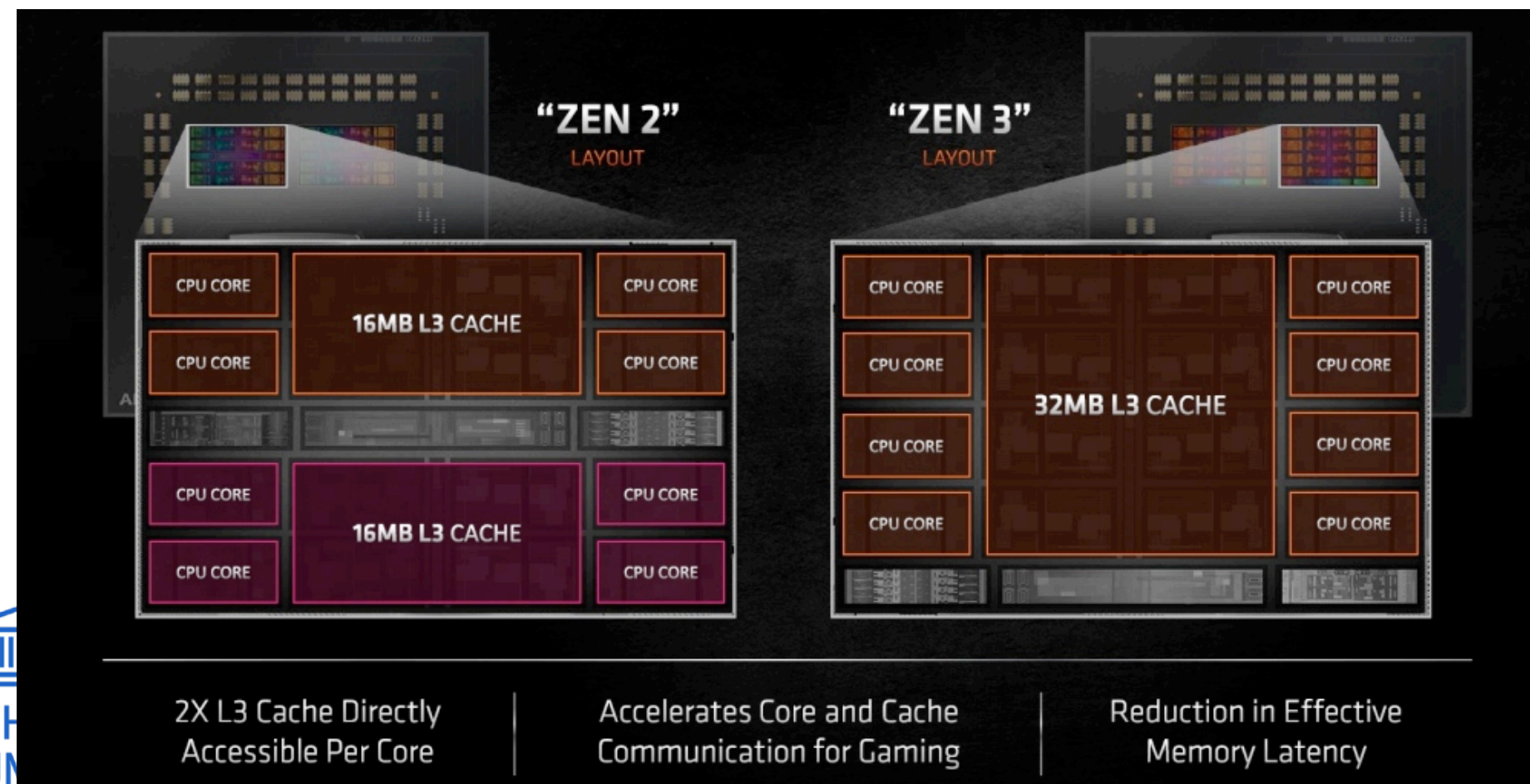
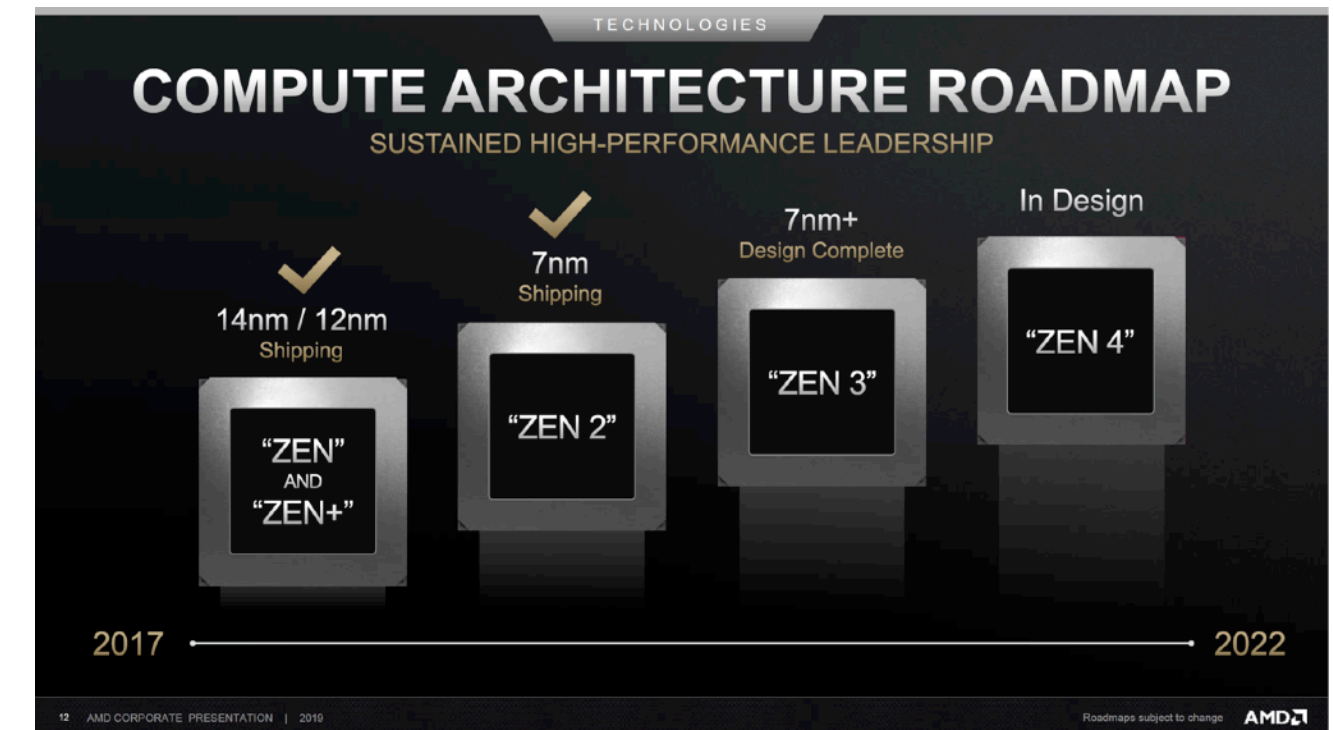
RETURN OF THE PROCESSOR WARS!

- In the last decade, Intel was king (w.r.t. microprocessors in servers)
- AMD is back in the game since ~2017 (AMD EPYC processor generation)
- Interest in ARM CPUs is booming (see Fugaku, Apple M1, NVIDIA Grace)
- IBM POWER is still around (sort of)
- In the (near?) future: "open source" high-performance RISC-V processors?



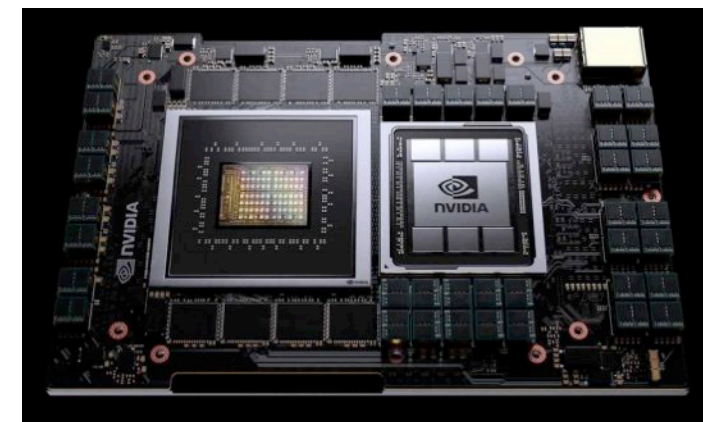
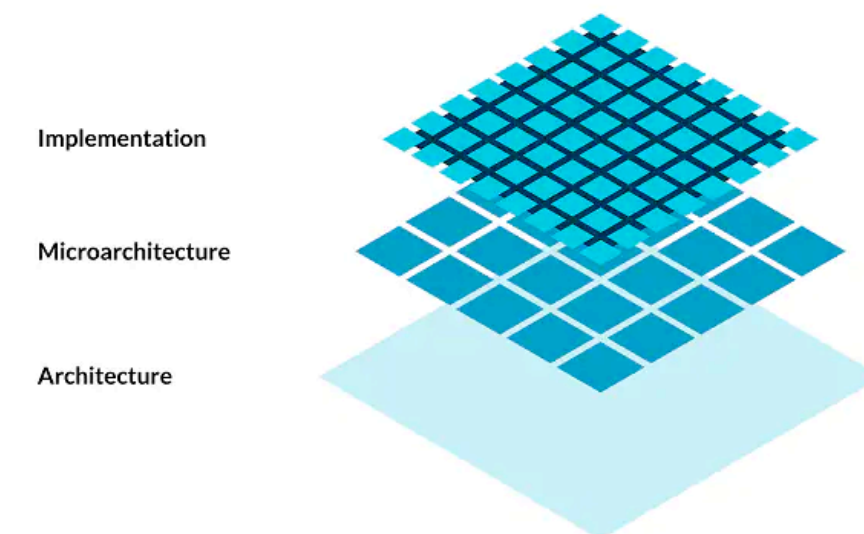
AMD EPYC PROCESSORS (ZEN* MICROARCHITECTURE)

- **AMD is competitive again with Intel**
- Different microarchitecture ("chiplet" design), same instruction set architecture (x86_64)



INTEREST IN ARM CPUS IS PICKING UP SPEED

- ARM is a different processor Instruction Set Architecture (ISA)
- ARM processors are the most common in the world
 - > 1 billion units per year, used in smartphones, tablets, wearables, ...
- License from Arm Ltd. is required to use ARM CPU designs
- ARM CPUs are now also used in recent large supercomputers:
Fugaku (Japan), Isambard (UK), Astra (US), Deucalion (Portugal), ...
- NVIDIA has announced its first ARM-based server CPU "Grace"
 - Is used in first European exascale supercomputer: JUPITER



RISC-V OPEN STANDARD ISA

- RISC-V is the 5th generation of RISC (Reduced Instruction Set Computer) ISA
- **Open source licensed:** free to use by anyone (no license fees)
- Several companies are designing RISC-V processors (some general purpose)
- Long term goal of [European Processor Initiative \(EPI\)](#) project is to design and produce a RISC-V processor for supercomputers & co, manufactured in Europe
- “Digital Autonomy with RISC-V in Europe” (DARE) project led by Barcelona Supercomputing Centre: 38 partners, 240 million Euro
- Will we see a (European?) RISC-V based supercomputer soon?
- Lots of work to be done on software side: libraries, applications, etc.

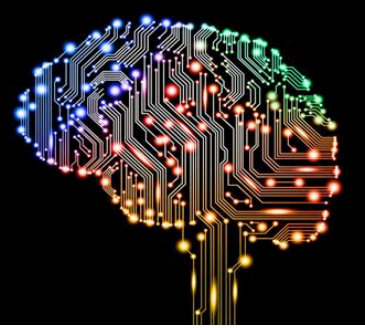


RISE OF CLOUD COMPUTING

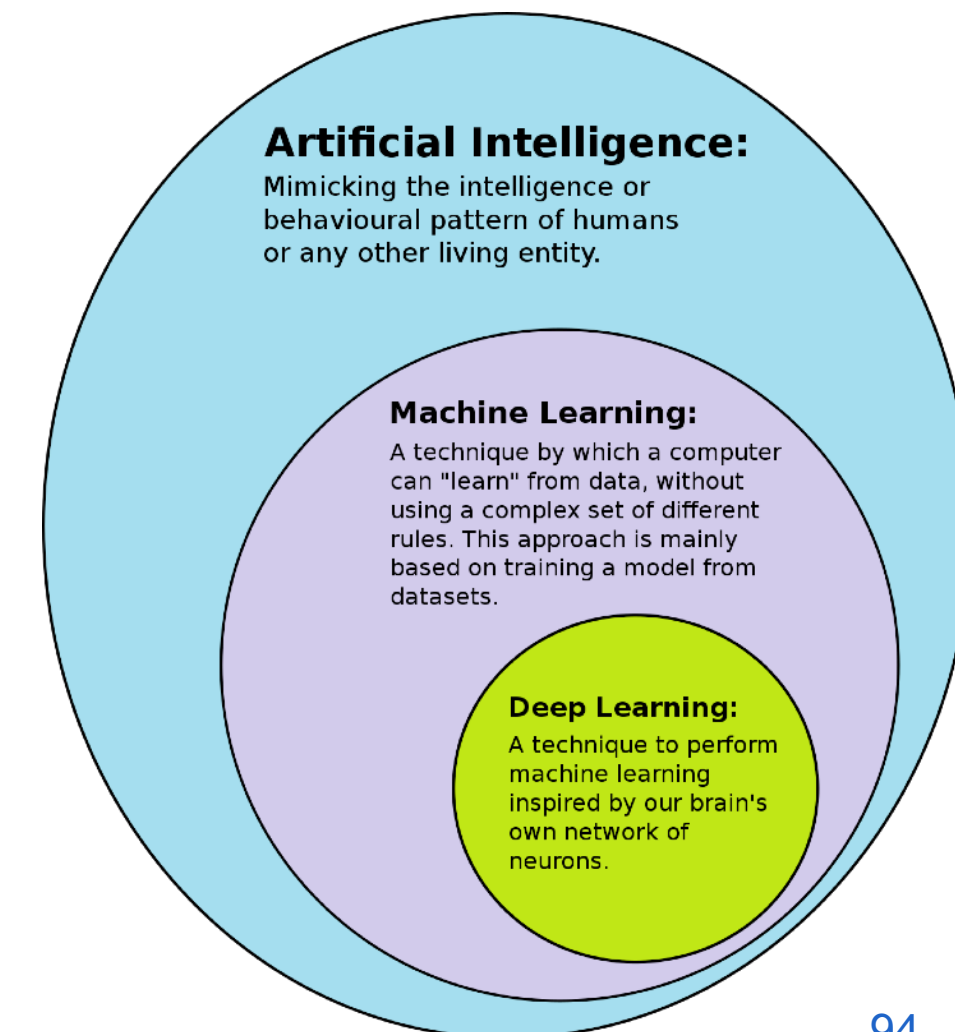
- Several companies provide "cloud computing" services
- "Rent" computing time & co, pay only for what you use
- Easy (?) to **create your own (throwaway) supercomputer!**
- Including shared storage, GPUs, fast interconnect, ...
- Very flexible, but also expensive (or is it?)
- "There is no cloud, just somebody else's computer."

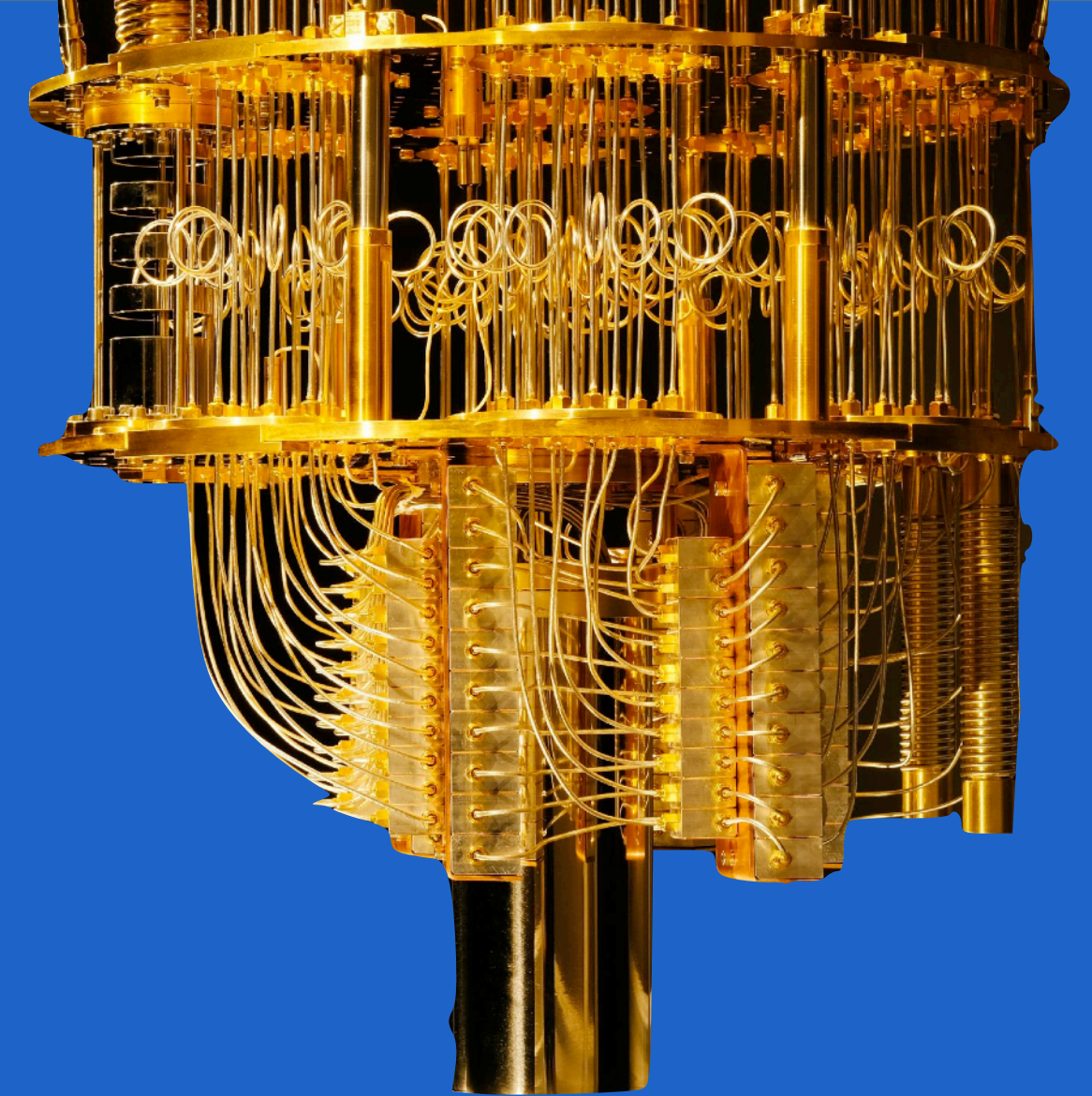


ARTIFICIAL INTELLIGENCE: A NEW HOPE



- Increased interest and adoption of machine learning techniques
- Fueled by capabilities of high-end computers (incl. supercomputers, GPUs)
- Typically requires large amounts of data ("big data")
- Lots of applications: digital imaging, self-driving cars, ...
- Deep learning "big bang" in 2009 fed by GPU boom
- **Large Language Models (LLMs) fueling the hype!**





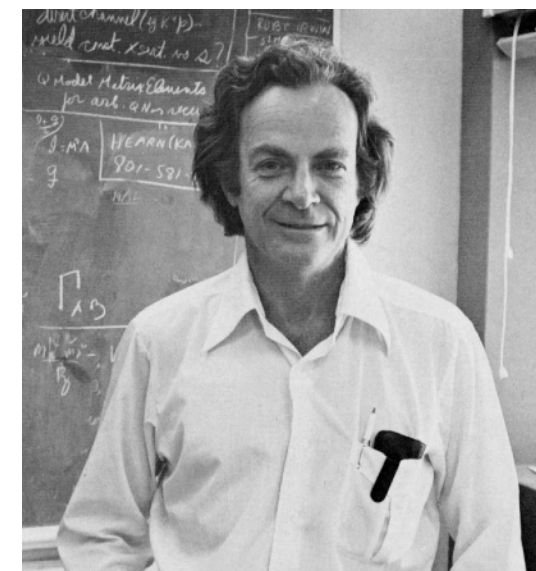
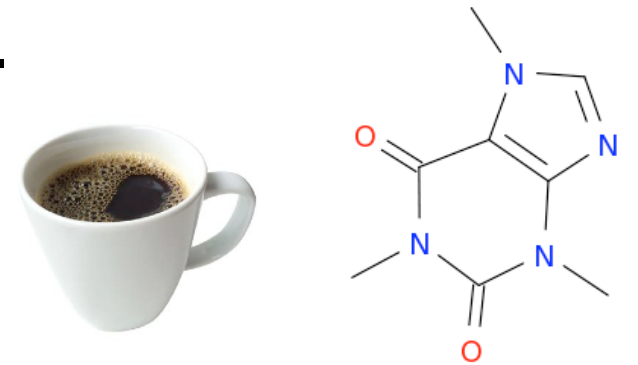
QUANTUM COMPUTING (IN A NUTSHELL)

DISCLAIMER

- I am *not* an expert in quantum computing.
- I have been trying to follow progress in quantum computing for a couple of years.
- Statements I make may be oversimplifying things too much (or may even be incorrect).
- I am *totally* skipping the math behind all this (complex numbers, probability, etc.)
- Some of this is likely outdated already.
- Wikipedia & Google ChatGPT was my friend when puzzling this together...
- Please don't invest money based on what I say (unless you don't need it).

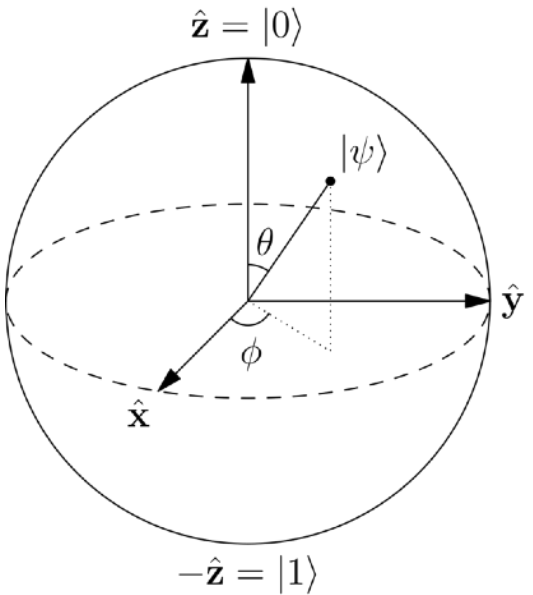
QUANTUM COMPUTING: HYPE OR REVOLUTION?

- Some problems are just **too big**, even for the largest (future) supercomputers!
 - Would take way too long to solve, require too much memory, etc.
- Example: exact simulation of a caffeine molecule
 - ~95 electrons, would require $\sim 10^{48}$ classical bits (1 terabyte = $\sim 2^{40} = \sim 10^{12}$ bytes)
- **Quantum computers** could be the answer...
- We're getting close to "*quantum supremacy*"
(or maybe we are there already...)



Richard Feynman (1918-1988)
"father" of quantum computing

CLASSICAL BITS VS QUBITS

- Classical bits are either 0 or 1 (like flipping a coin: heads or tails) 
- **Quantum bits ("qubits")** can be in a state somewhere in between 0 and 1...
 - Like a *spinning* coin, in some sense they can be "both 0 and 1 at the same time" 
- Measuring (observing) a qubit collapses the state to either 0 or 1 (with a certain probability)
- Geometrical representation of a single qubit: *Bloch sphere* $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$
 - North pole : $|0\rangle$ (equivalent with 0 in classical bit)
 - South pole: $|1\rangle$ (equivalent with 1 in classical bit)
 - Equator: 50% probability of observing 0 or 1 when measuring
 - Somewhere else on surface of the sphere: leaning towards either $|0\rangle$ or $|1\rangle$...

LEVERAGING QUANTUM PHYSICS

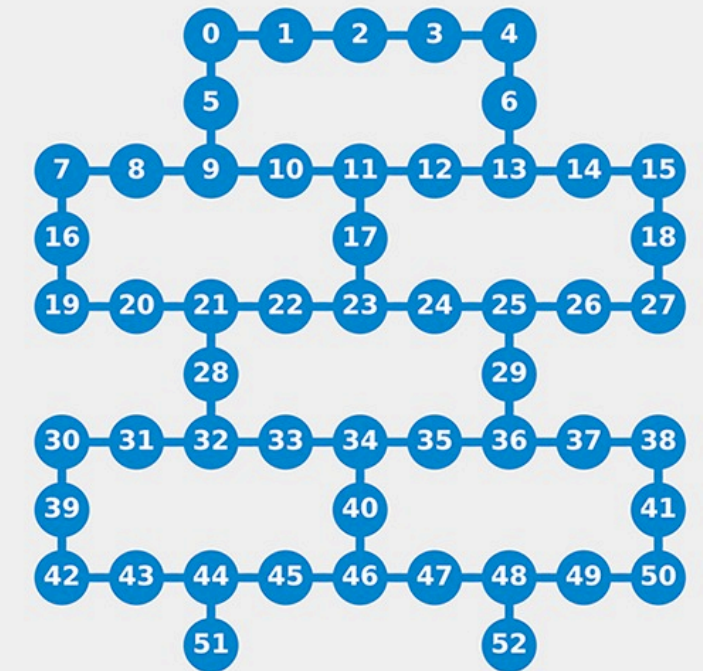


- **Superposition**
 - A qubit can be put in a state between $|0\rangle$ and $|1\rangle$
 - A collection of qubits can be manipulated to **favor** the "optimal" state
- **Entanglement**
 - Two qubits can be "entangled": measuring one impacts the other
 - The distance between the entangled qubits doesn't matter!
 - Can used for *teleportation* of information (faster-than-light communication!)
 - Einstein called this "spooky action at a distance"
- **Interference**
 - The state of a qubit can be changed via constructive or destructive interference

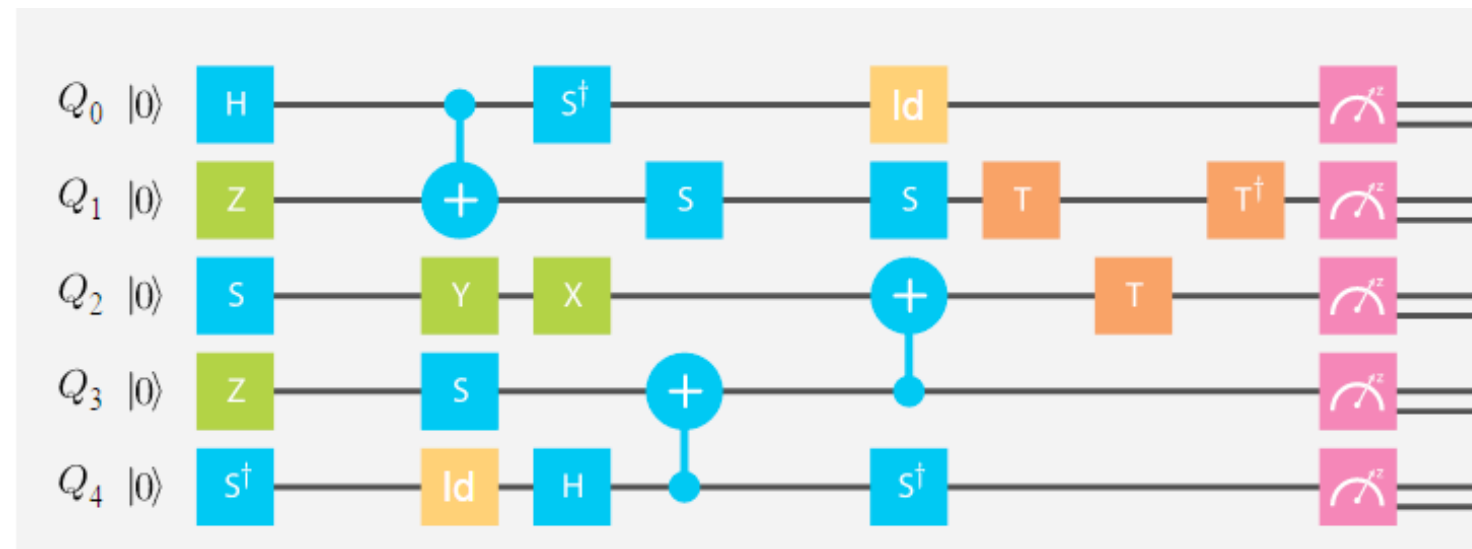
QUBITS AND QUANTUM CIRCUITS

- Combining multiple qubits makes things interesting...
- In some sense, every qubits *doubles* the compute power
 - Assuming each qubit is connected to all others (*which it won't be*)
- Quantum circuits* can be used to manipulate a set of qubits
 - Gate-model quantum computing: a circuit is a series of *gates*
 - Equivalent to a computer program in classical computers

53 Qubit Rochester Device



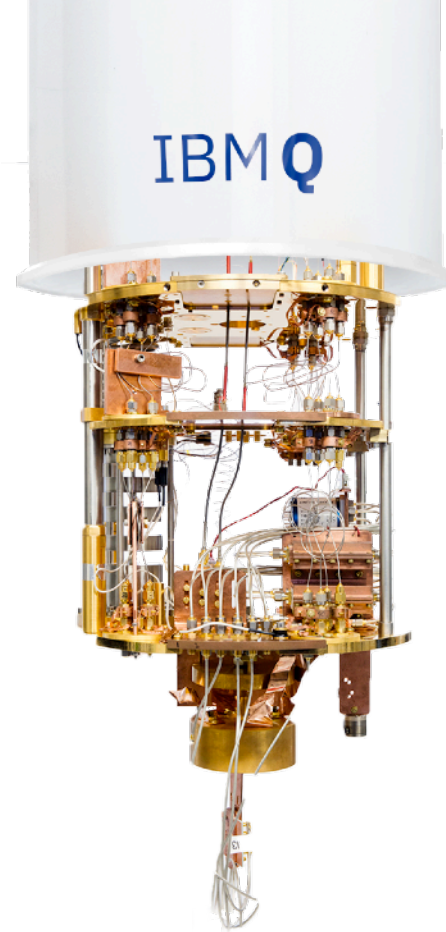
Operator	Gate(s)
Pauli-X (X)	
Pauli-Y (Y)	
Pauli-Z (Z)	
Hadamard (H)	
Controlled Not (CNOT, CX)	
Controlled Z (CZ)	



TYPES OF QUANTUM COMPUTERS

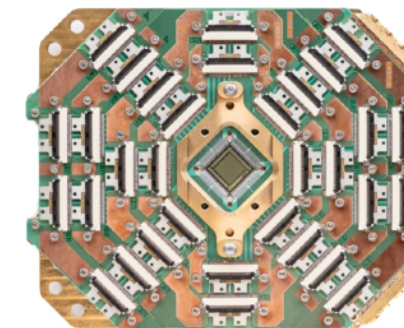
- **Circuit-based quantum processors**

- IBM, Google, Intel, Rigetti, Xanadu Quantum Technologies, ...
- Different technologies: superconducting, trapped ions, quantum dot, ...



- **Quantum annealers**

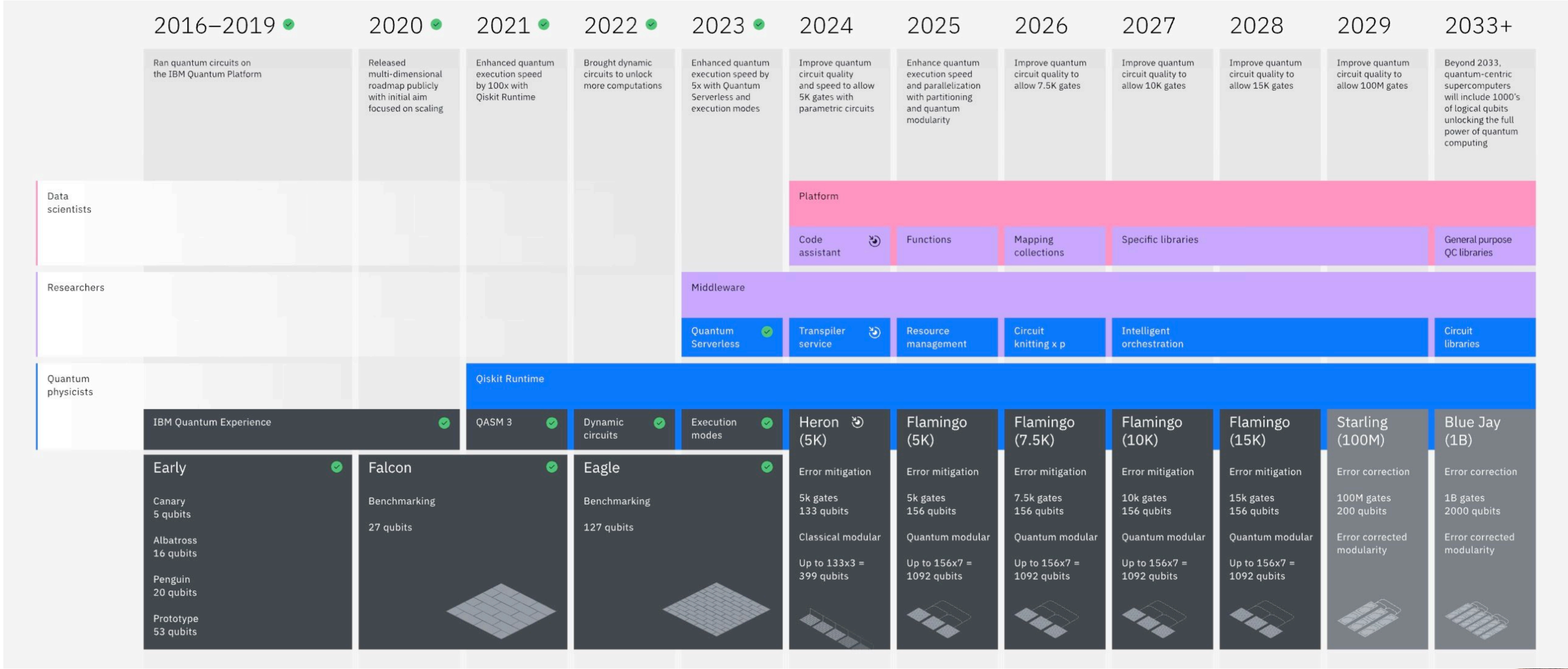
- Can only be used for certain type of problems (optimization)
- D-Wave One (2011): 128 qubits
- D-Wave 2000Q (2017): 2048 qubits
- D-Wave Advantage (2020): 5760 qubits



Recently purchased by Jülich Supercomputing Centre!



IBM QUANTUM ROADMAP



APPLICATIONS FOR QUANTUM COMPUTING



Cryptography

- Break popular encryption algorithms (RSA)
- Would be *very* disruptive (quantum computing as a "weapon")
- Post-quantum cryptography (who wins the race?)

Simulation of quantum physics

- Molecular simulations, drug development, etc.
- Very popular application in supercomputers today
- Some problems are impossible to solve today, even on supercomputers

Machine learning

- Quantum Machine learning algorithms already exist today
- Artificial Intelligence (AI) more popular than ever
- Could be the way to AGI / the "Singularity"

Search & optimization

- Searching for "best" item in unstructured space
- Can be proved Grover's is the *optimal* way of searching

QUANTUM ALGORITHMS

- **Grover's search algorithm**

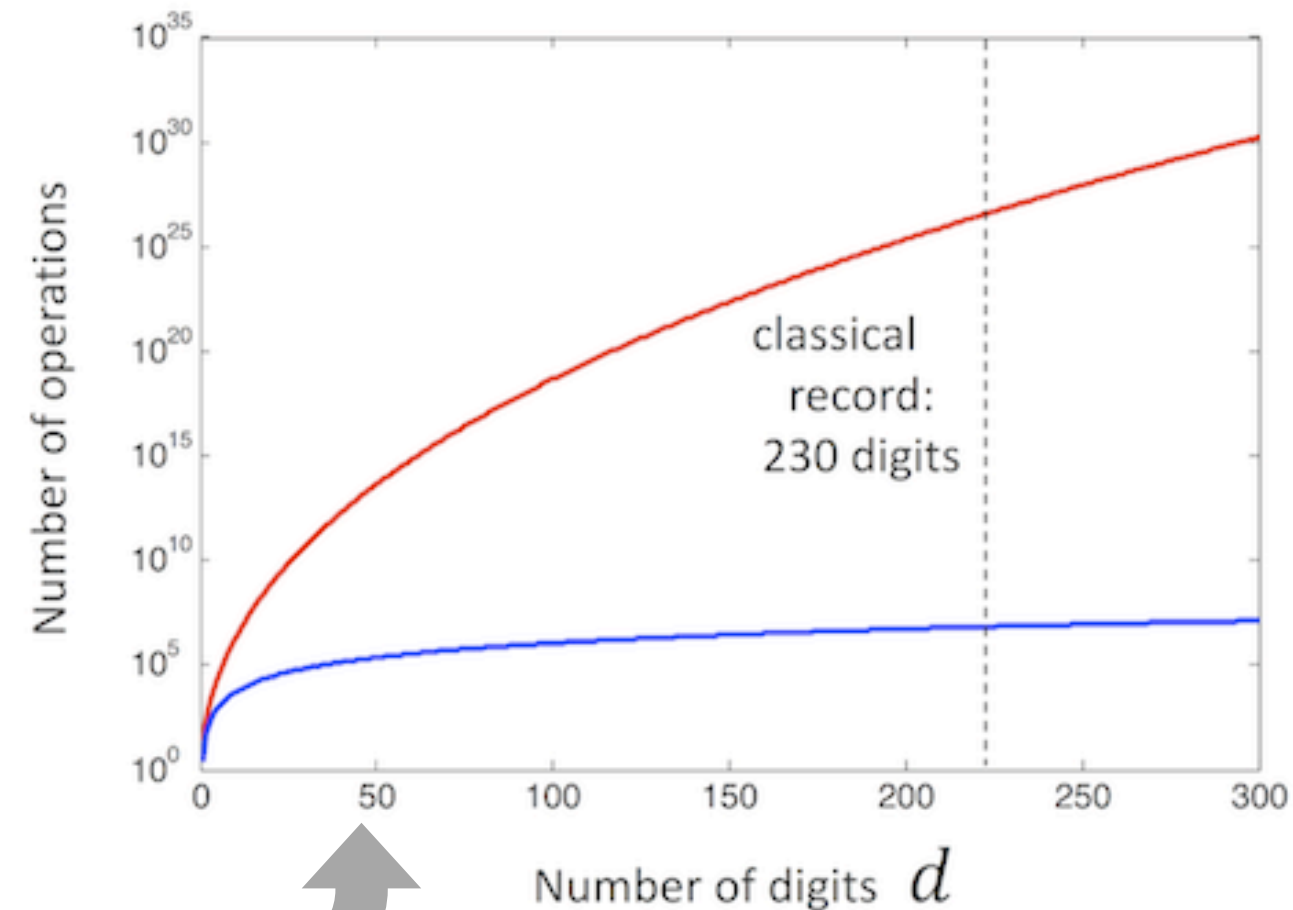
- Unstructured search problem: find best out of N
- On average: $\frac{N}{2}$ classically, $\sqrt{N}$ with Grover's

- **Shor's algorithm**

- Factoring integer numbers into primes in polynomial time
- Can be used to break RSA encryption (uses 2,048 bit integers, or larger)
- Shor requires $\sim 2N$ *logical* qubits (N: number of bits to represent integer to factor)

- **Quantum phase estimation**

- Central building block for many quantum algorithms

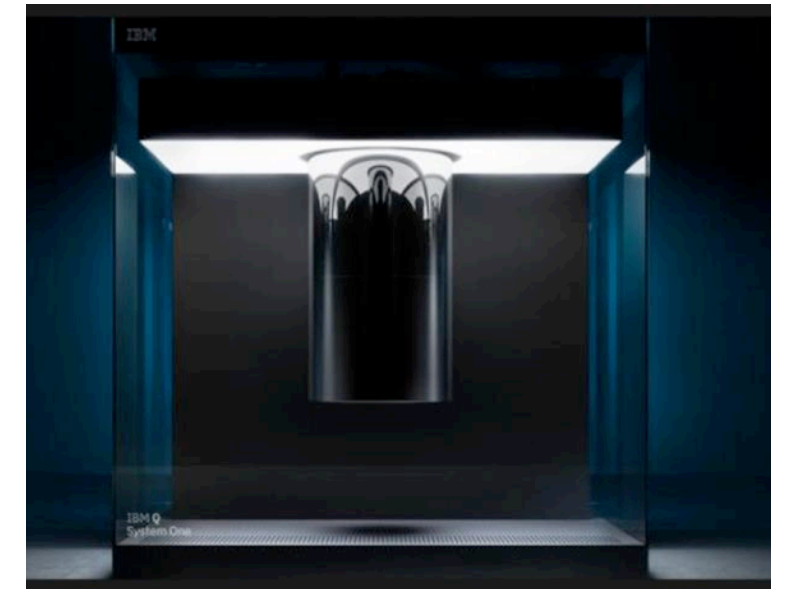


CHALLENGES IN QUANTUM COMPUTING

- **Physics**
 - Creating and controlling qubits: superconducting, photonics, ...
 - Keeping out the noise, temperatures close to 0K (colder than outer space!)
- **Decoherence**
 - Qubits "collapse" due to outside influence: noise, vibrations, radiation, ...
 - Quantum "programs" must finish before decoherence happens (< 1 ms currently)
- **Connectivity**
 - Multiple qubits must be as connected as possible so they can be entangled, etc.
 - Quantum "compilers" are used to map quantum circuits on physical qubit layout
- New generation of **programmers** needs to be trained to use quantum computers...

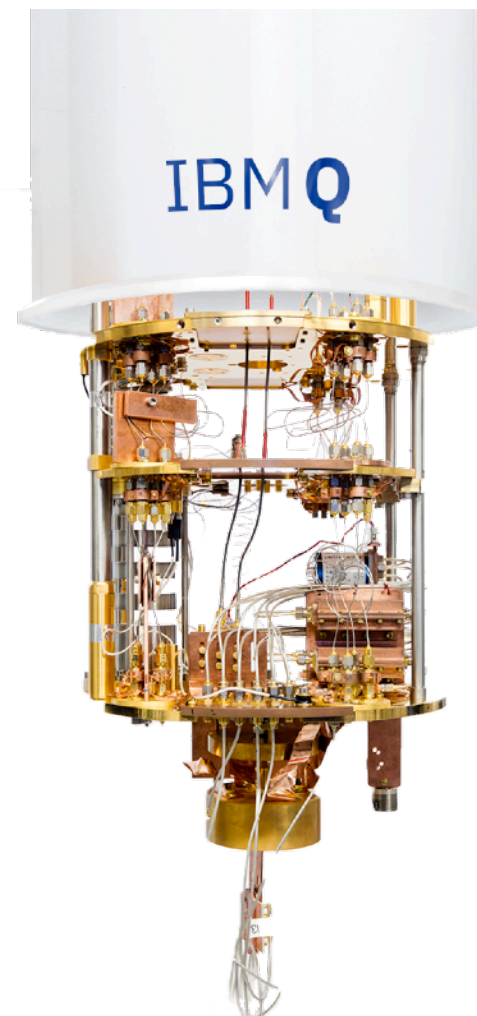
CURRENT QUANTUM COMPUTERS

- Currently in **Noisy Intermediate Scale Quantum** (NISQ) era
- Small number of qubits, very short coherence time (~ 0.1 ms)
- Examples: [Google Willow](#) (105 qubits), [IBM Heron](#) (156 qubits)
- Procurement of quantum computers @ EuroHPC sites ongoing...



IBM Q System One
Commercial quantum computer (20 qubits)

"Naked"
quantum
computer



IBM Q System Two (2023)
3x IBM Heron (each with 156 qubits)

- You can play with a **real quantum computer** yourself (or simulate one).
- In the cloud! For free! Using Python code!

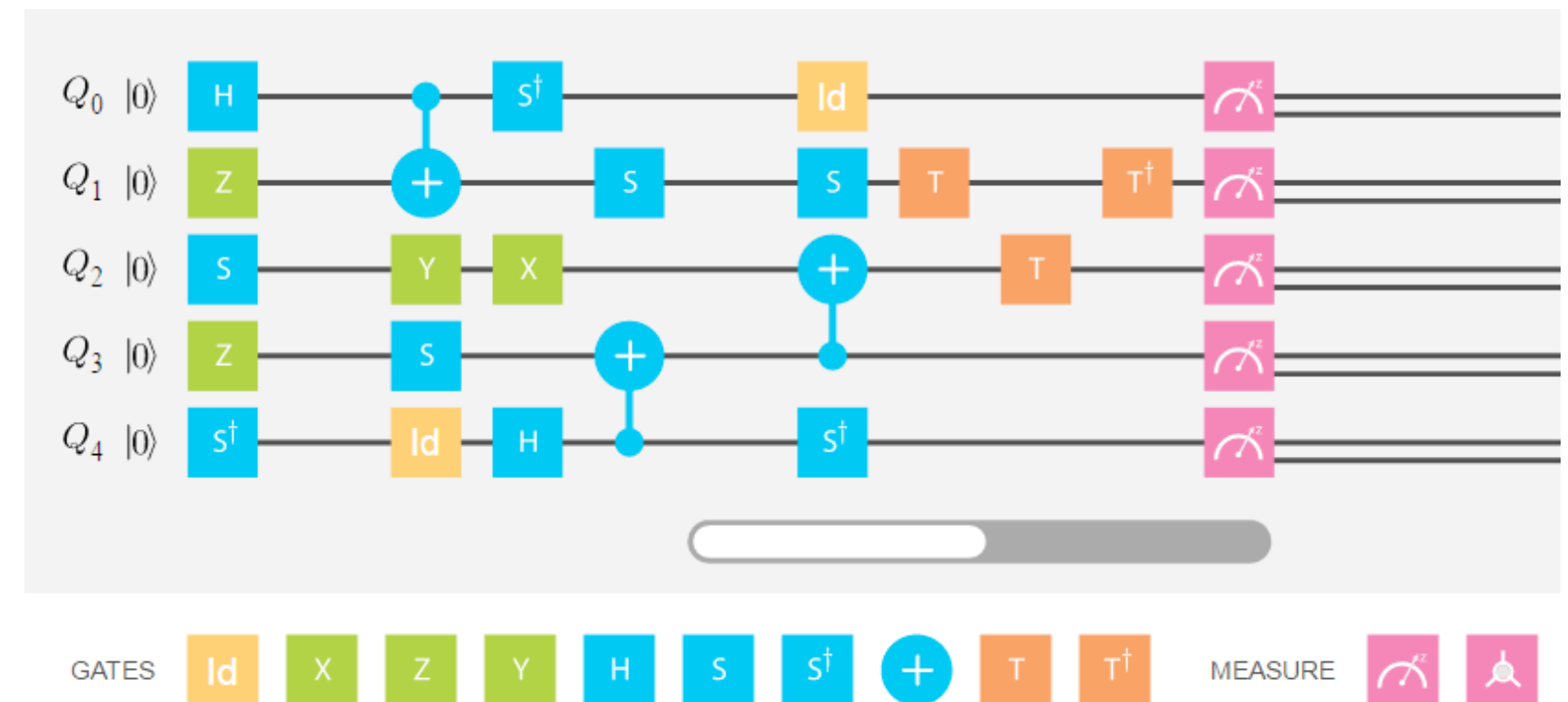
```
from qiskit import QuantumCircuit

# Create a Quantum Circuit (2 qubits, 2 classical bits)
circuit = QuantumCircuit(2, 2)

# Add a H gate on qubit 0
circuit.h(0)

# Add a CX (CNOT) gate on control qubit 0 and target qubit 1
circuit.cx(0, 1)

# Map the quantum measurement to the classical bits
circuit.measure([0,1], [0,1])
```



WOULD YOU LIKE TO KNOW EVEN MORE?

- Quantum Open Source Foundation: <https://www.qosf.org>
- <https://learning.quantum.ibm.com/catalog/tutorials>
- <https://fosdem.org/2025/schedule/track/quantum>
- https://www.youtube.com/playlist?list=PLqxhJj6bcnY92luCrMt78ThcBOJ5Nb5_O
- <https://www.thequantumdaily.com/2020/05/15/7-quantum-computing-books-for-the-uninitiated>

