HPC

High Performance Computing

Dr. Maxim Masterov
Han-sur-Lesse Winter School
24-28.11.2025

Contents

Overview

Day 1

- What is HPC?
- A lifecycle of an HPC application
- Parallelisation strategies
    - Threads
    - MPI
    - Hybrid

Day 2

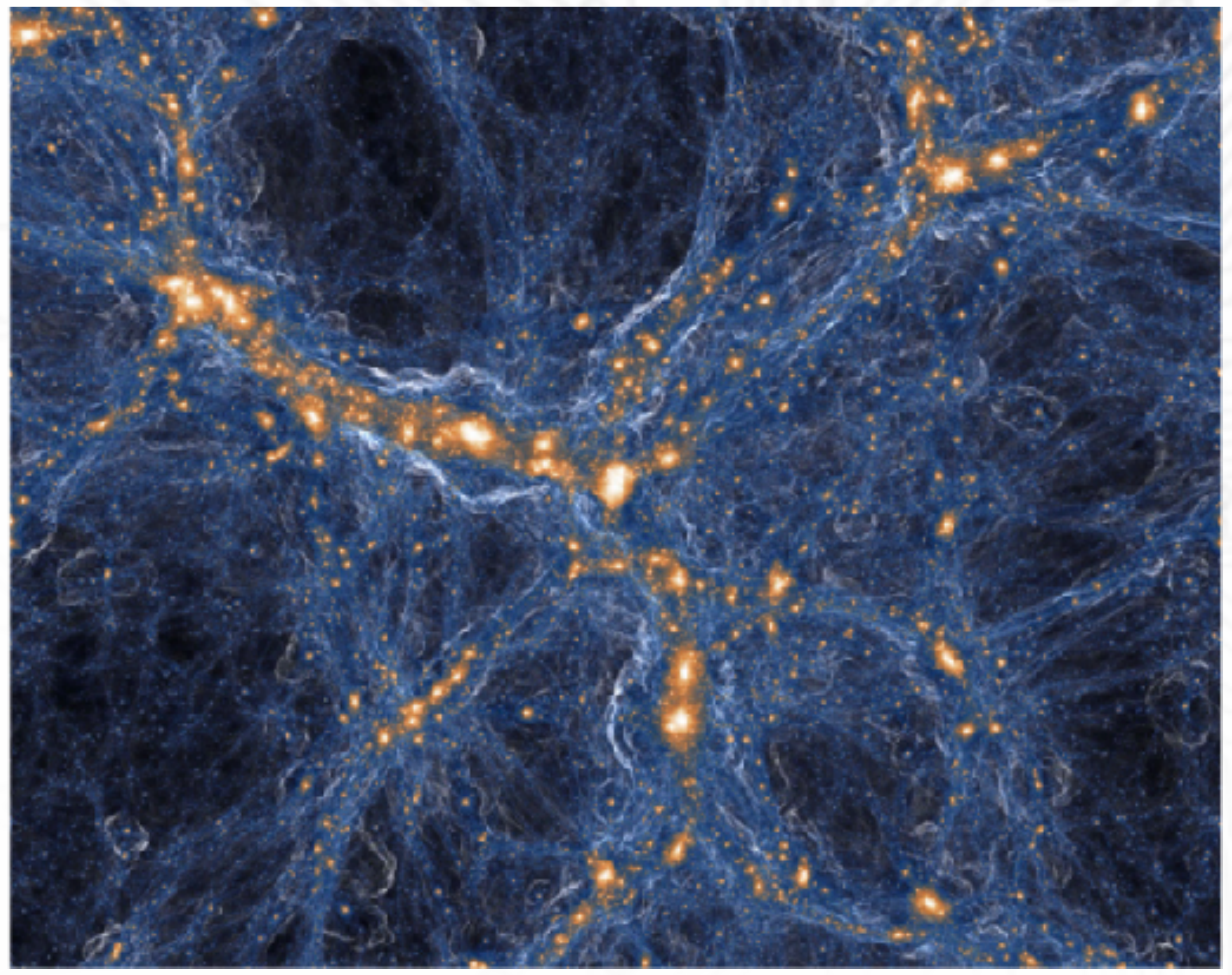
- Profiling
- Hackathon
    - Challenge
    - Work in groups
    - Presentations

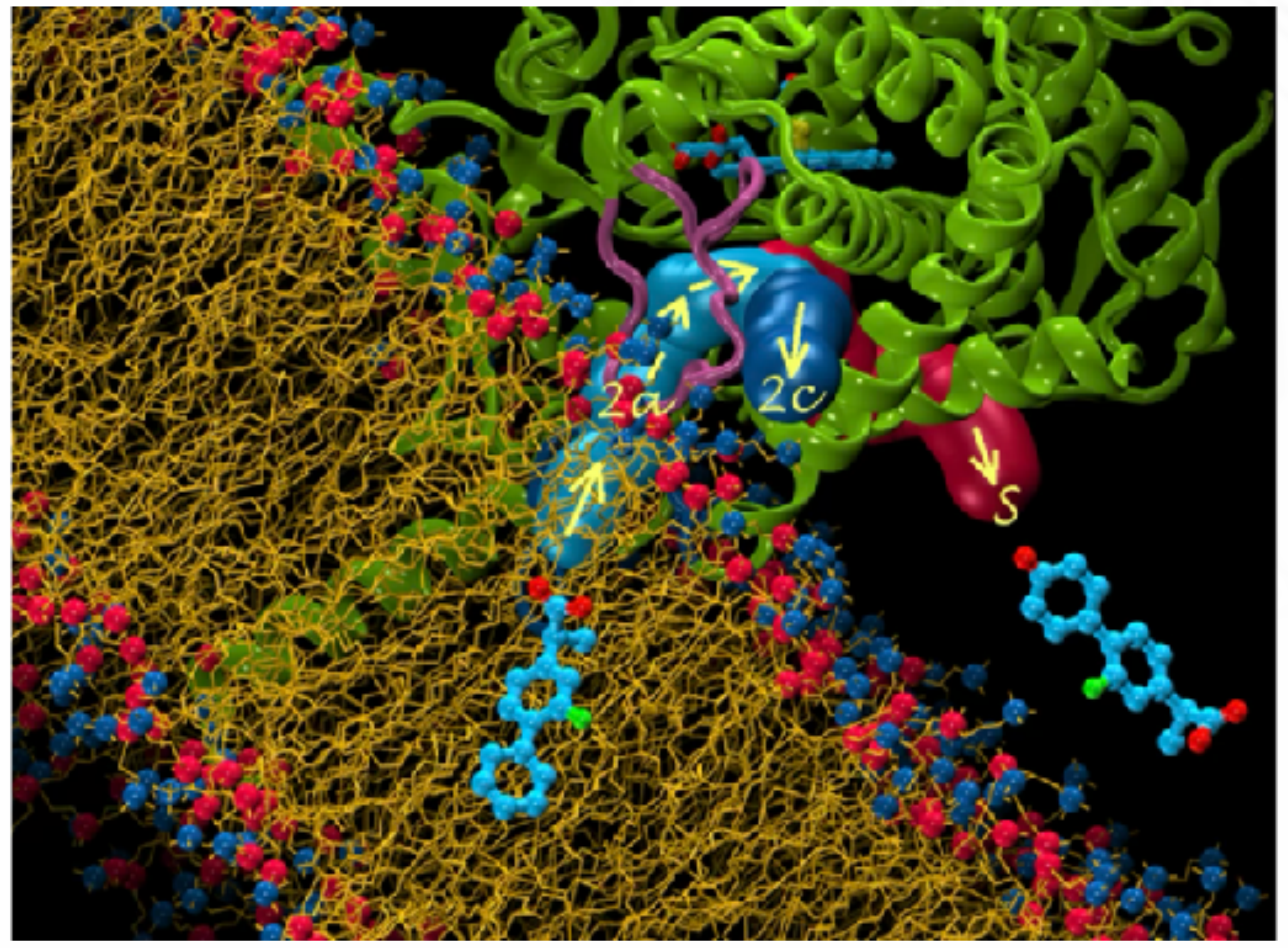
What is HPC?

What is HPC?

And why it matters?

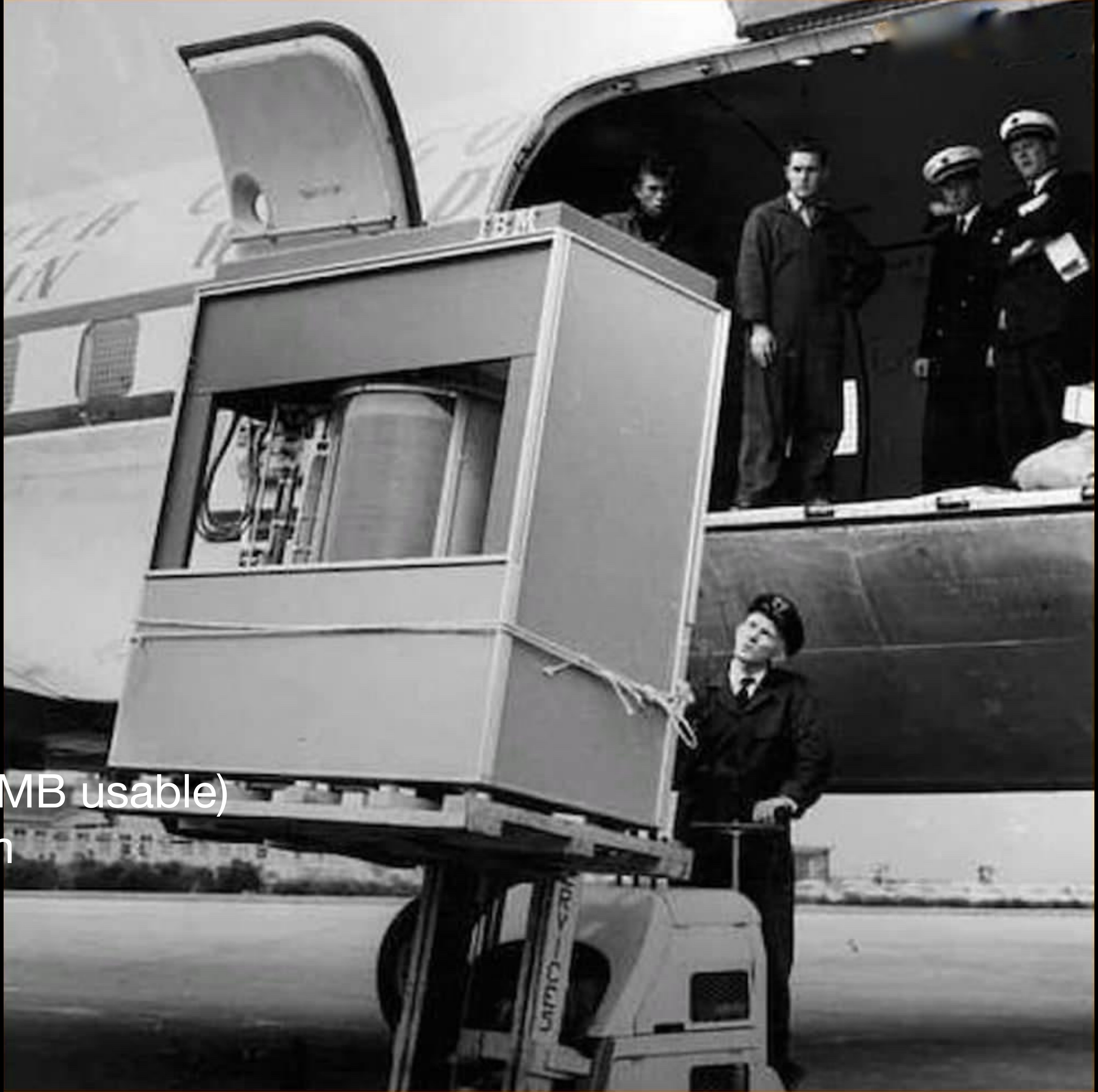
- **HPC — Hight Performance Computing** — a scalable and optimal compute coordination of the hardware and software
- **HPC applications** are software designed to exploit the power of supercomputers or high-performance clusters to solve complex problems
- **HPC applications** perform intensive computations, facilitate communications between parallel processes, and manage input/output operations on a high-performance file system





1. IllustrisTNG. Astrophysics model. https://phys.org/news/2018-02-astrophysicists-illustristng-advanced-universe-kind.html
2. VMD. Visualisation software. (2011) PLoS Computational Biology Issue Image | Vol. 7(8) August 2011. PLoS Comput Biol 7(8)

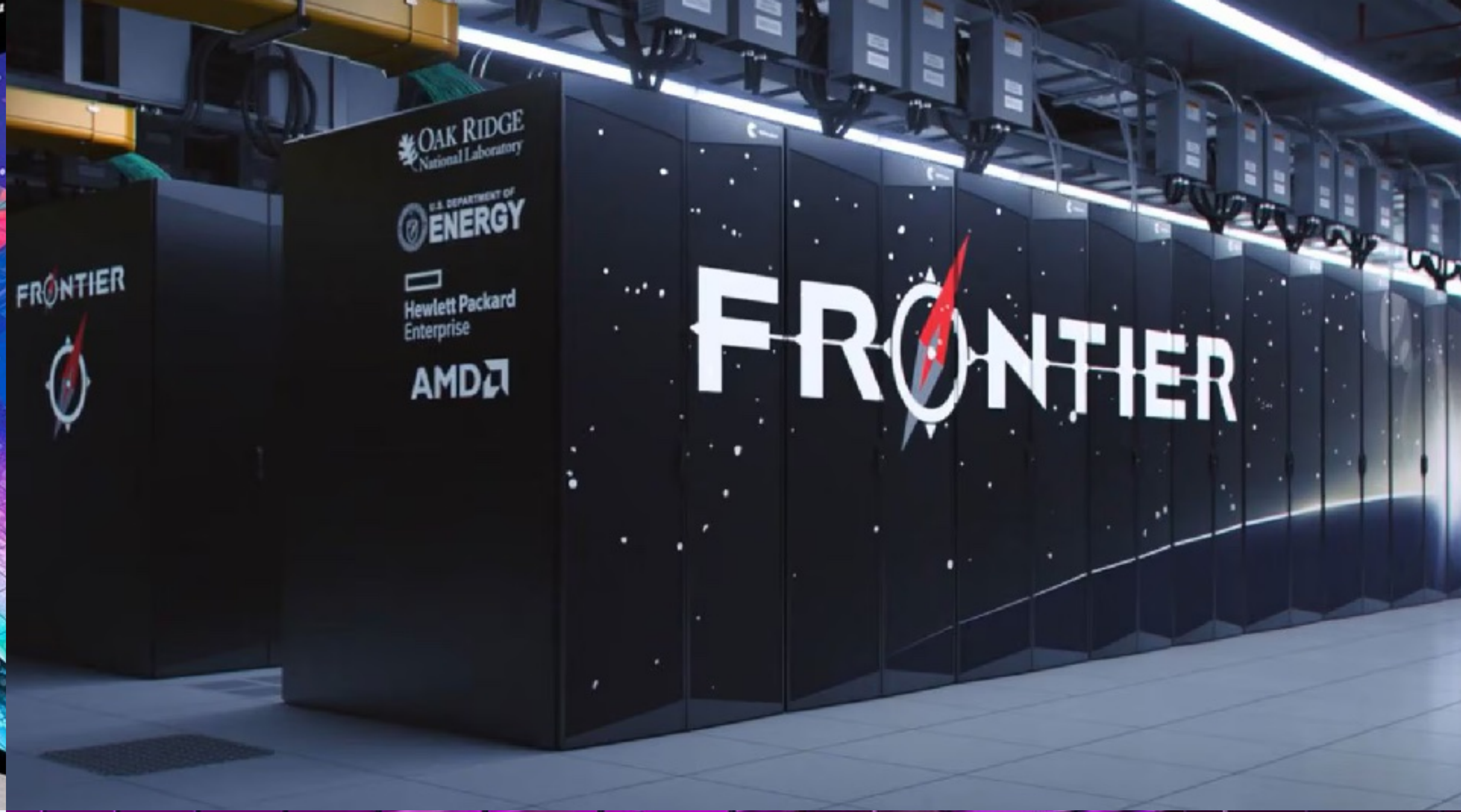
- ENIAC - Electronic Numerical Integrator and Computer
- 17,468 vacuum tubes
- Approx. 500 floating-point operations per second
- 167 m2
- 30 tonns
- 1945



- IBM 350
- 5MB (only 3.75MB usable)
- 150x170x74 cm
- ~1 tonn
- 52 x 24” disks
- 1956

- RPi-5 (2023): 25,000M floating-point operations per second
- iPhone-17 (2025): Up to 512GB of memory

Aurora


FRONTIER
OAK RIDGE National Laboratory
U.S. DEPARTMENT OF ENERGY
Hewlett Packard Enterprise
AMD
FRONTIER

LUMI
LUMI

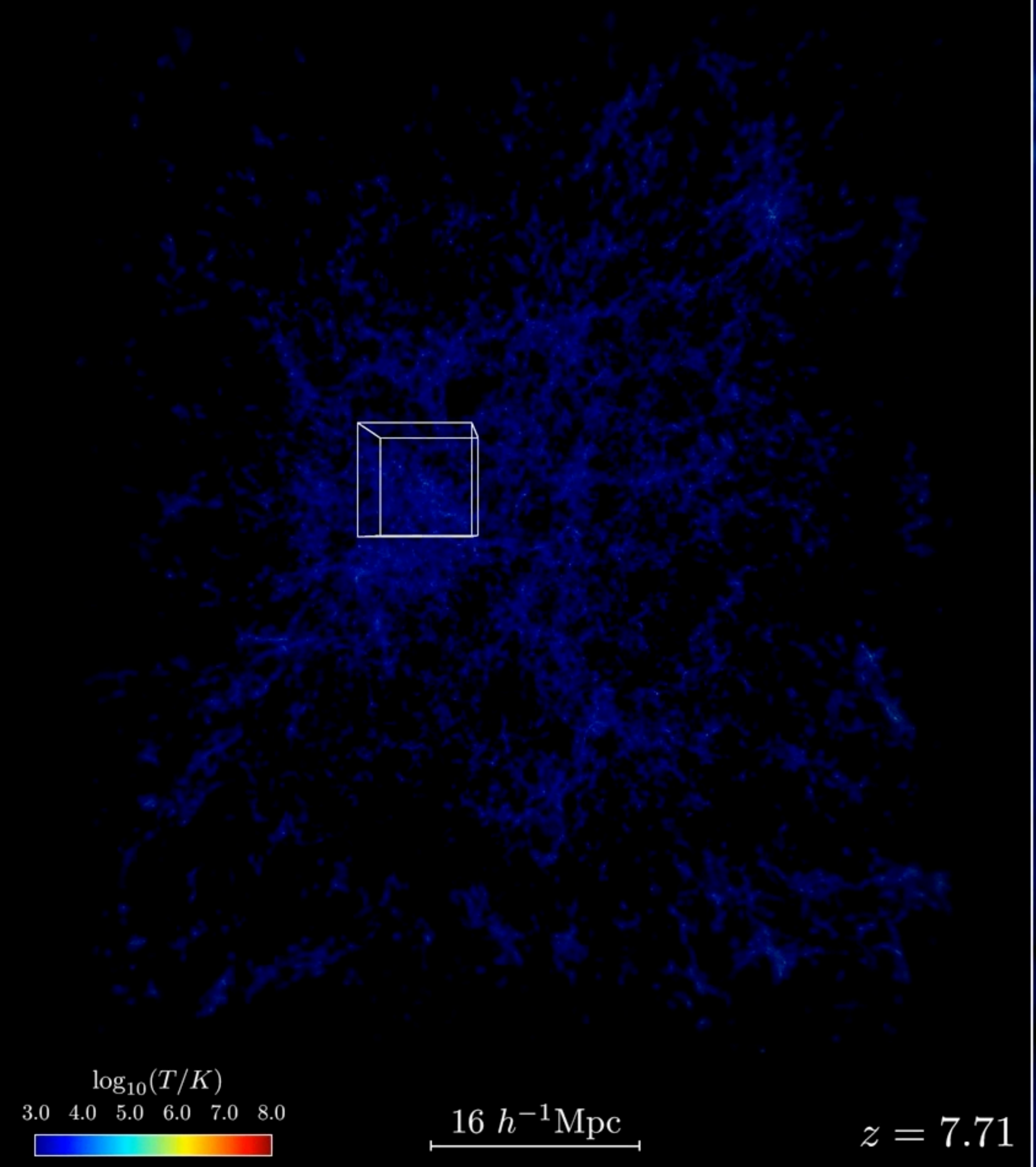
CSCS
ETH zürich
ALPS

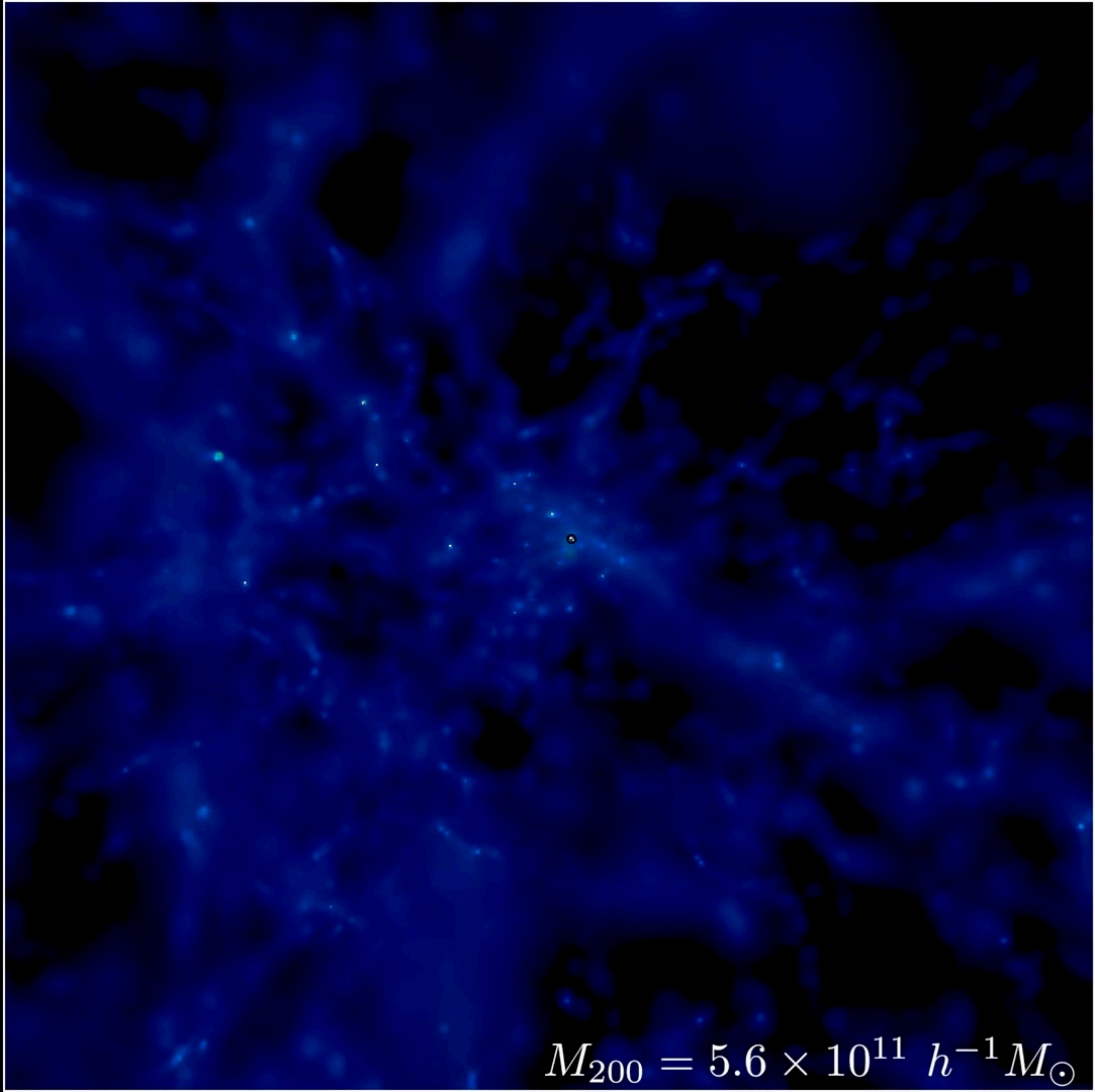
Most powerful supercomputers

November 2025

Rank	System	Cores	Rmax [PFLOP/s]	Rpeak [PFLOP/s]	Power [kW]
1	El Capitan, USA	11,340,000	1,809.00	2,821.10	29,685.00
2	Frontier, USA	9,066,176	1,353.00	2,055.72	24,607.00
3	Aurora, USA	9,264,128	1,012.00	1,980.01	38,698.00
4	JUPITER Booster, Germany	4,801,344	1,000.00	1,226.28	15,794.00
5	Eagle, USA	2,073,600	561.20	846.84	
6	HPC6, Italy	3,143,520	477.90	606.97	8,461.00
7	Supercomputer Fugaku, Japan	7,630,848	442.01	537.21	29,899.00
8	Alps, Switzerland	2,121,600	434.90	574.84	7,124.00
9	LUMI, Finland	2,752,704	379.70	531.51	7,107.00
10	Leonardo, Italy	1,824,768	241.20	306.31	7,494.00

Frontier


$\log_{10}(T/K)$
3.0 4.0 5.0 6.0 7.0 8.0
$16\ h^{-1}\mathrm{Mpc}$
$z = 7.71$


$M_{200} = 5.6 \times 10^{11}\ h^{-1} M_\odot$

LUMI

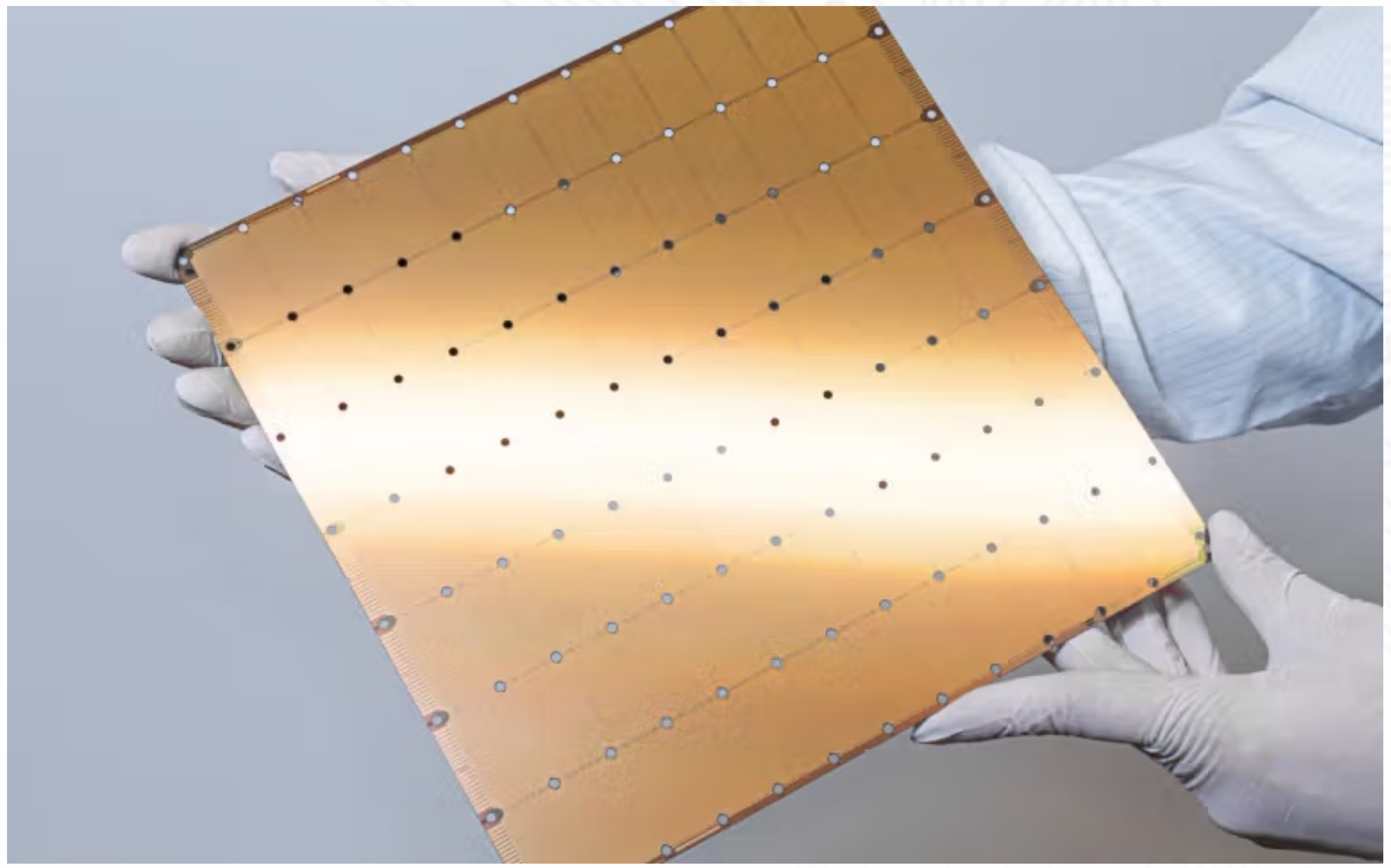
Funded by the European Union Destination Earth implemented by ECMWF esa EUMETSAT EuroHPC Joint Undertaking LUMI

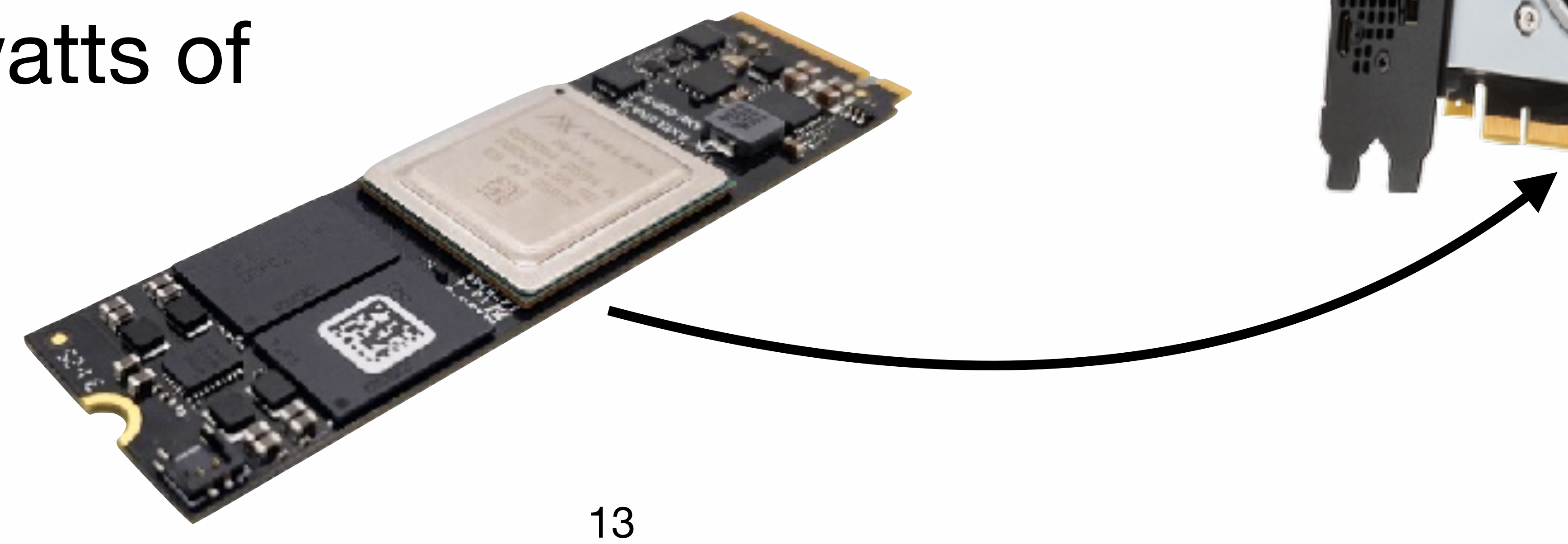
Supercomputer Fugaku

Not exactly HPC

AI

- Dedicated hardware (usually GPUs, or other accelerators)
- Different type of the workflow (no batching, constant work)
- Different type of IO (dozens of companies emerged in the past few years)
- Small-scale systems demand Megawatts of energy to run traditional projects
- Enterprise systems demand Gigawatts of energy to run competitive projects

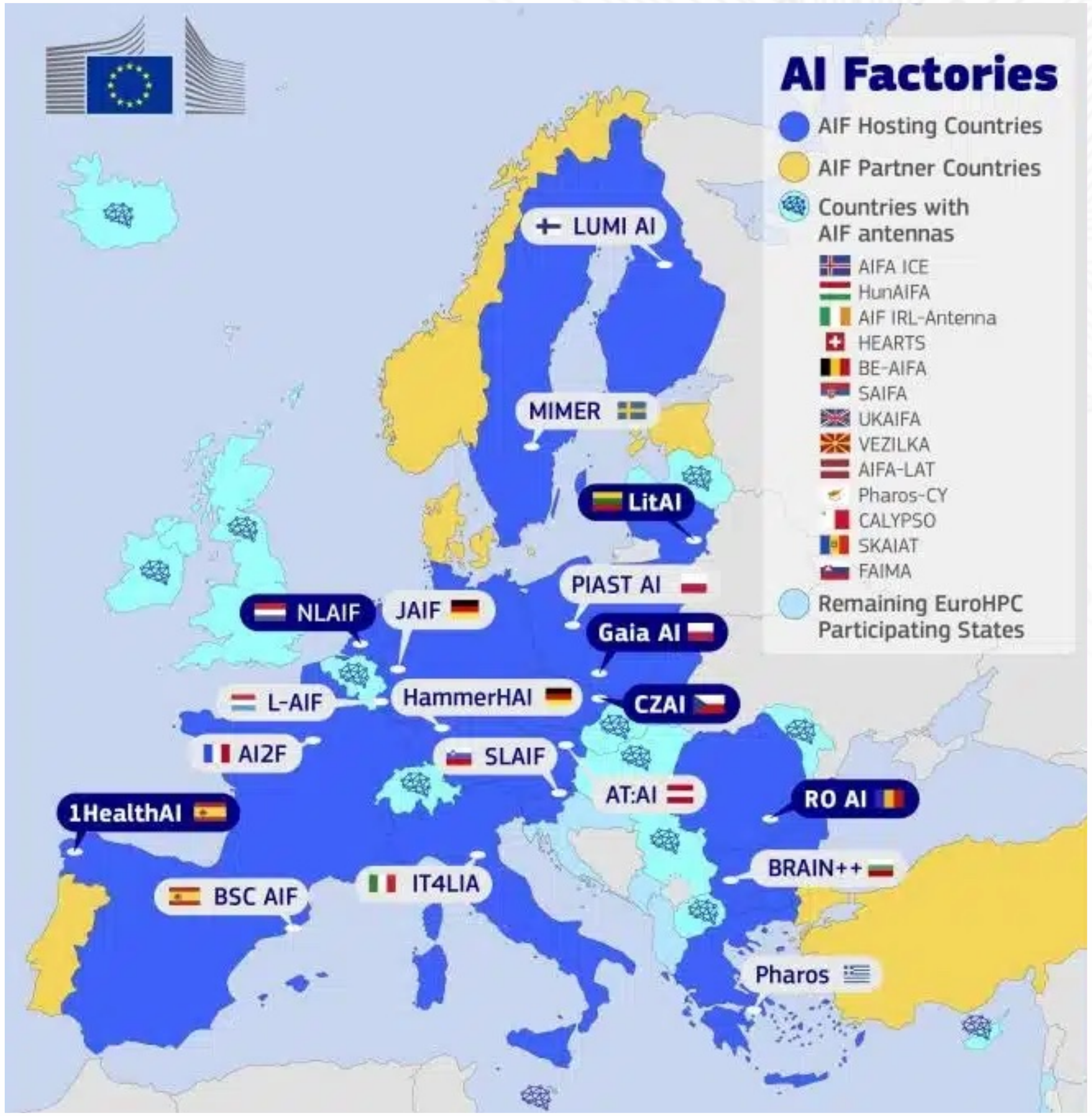




Not exactly HPC

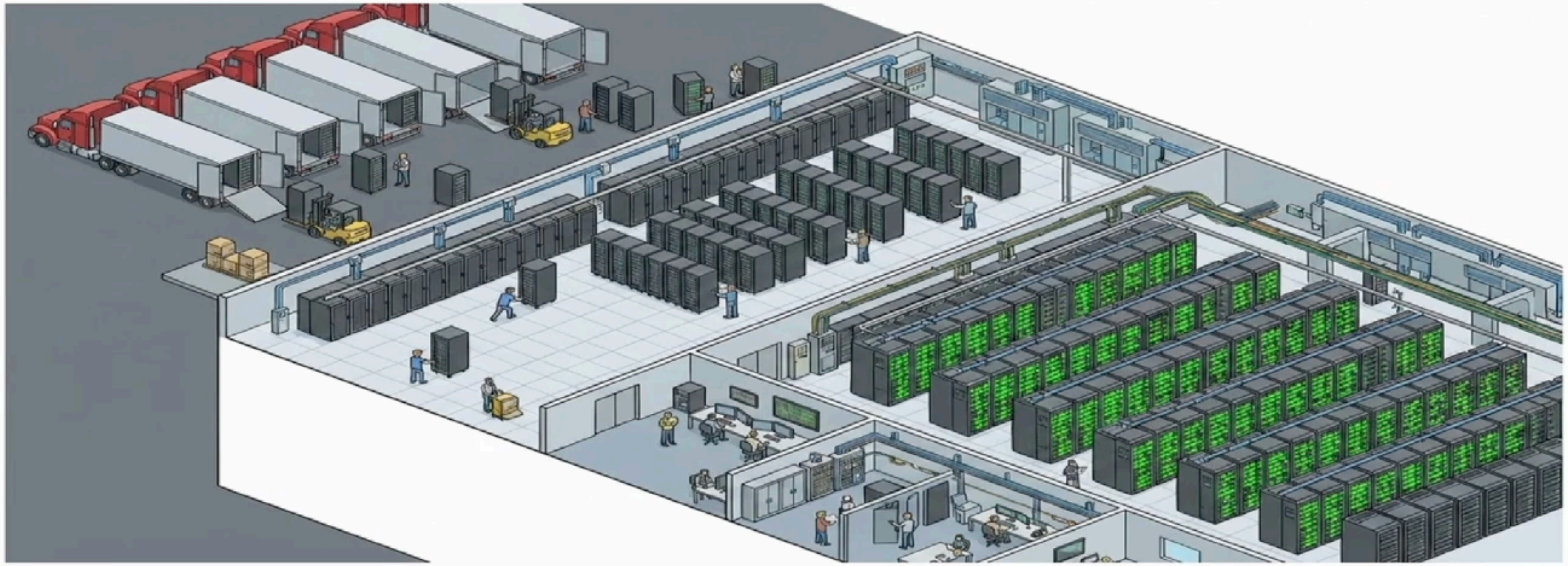
AI

- Dedicated hardware (usually GPUs, or other accelerators)
- Different type of the workflow (no batching, constant work)
- Different type of IO (dozens of companies emerged in the past few years)
- Small-scale systems demand Megawatts of energy to run traditional projects
- Enterprise systems demand Gigawatts of energy to run competitive projects



Gigawatt Scale Machine Learning: Deployment

Traditional: build a 100MW data center, fill it with servers over 12-36 months

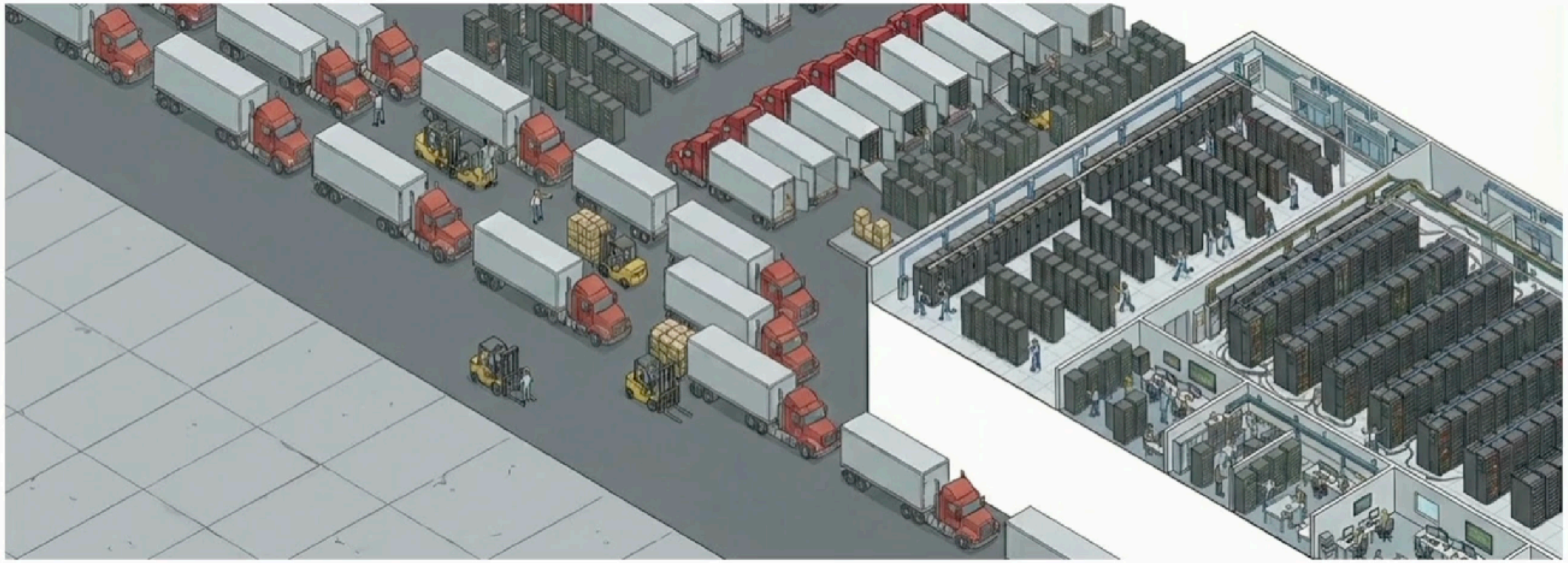


That's a max throughput of 12 *minutes* per rack over a 10-hour day.

SC25 St. Louis, MO | hpc ignites.

Gigawatt Scale Machine Learning: Deployment

Gigascale: build a 200MW data center, fill it with ML servers in 2 months.



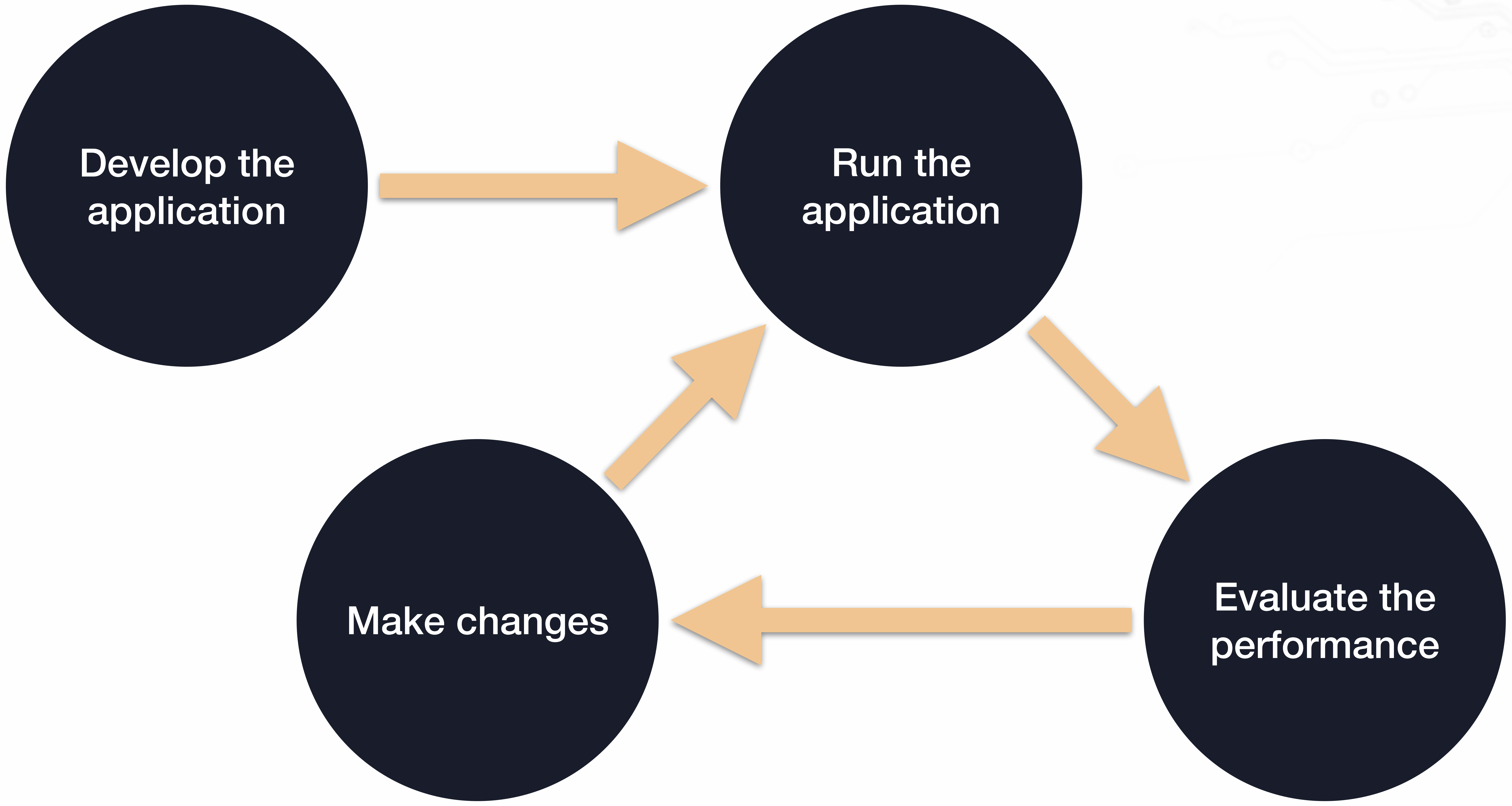
320 racks/day, or *1 rack every 2 minutes*. At five campuses. Simultaneously.

Gigawatt-Scale AI Infrastructure: Challenges, Opportunities, and Best Practices
Benjamin Treynor Sloss, Google LLC

A lifecycle of an HPC application

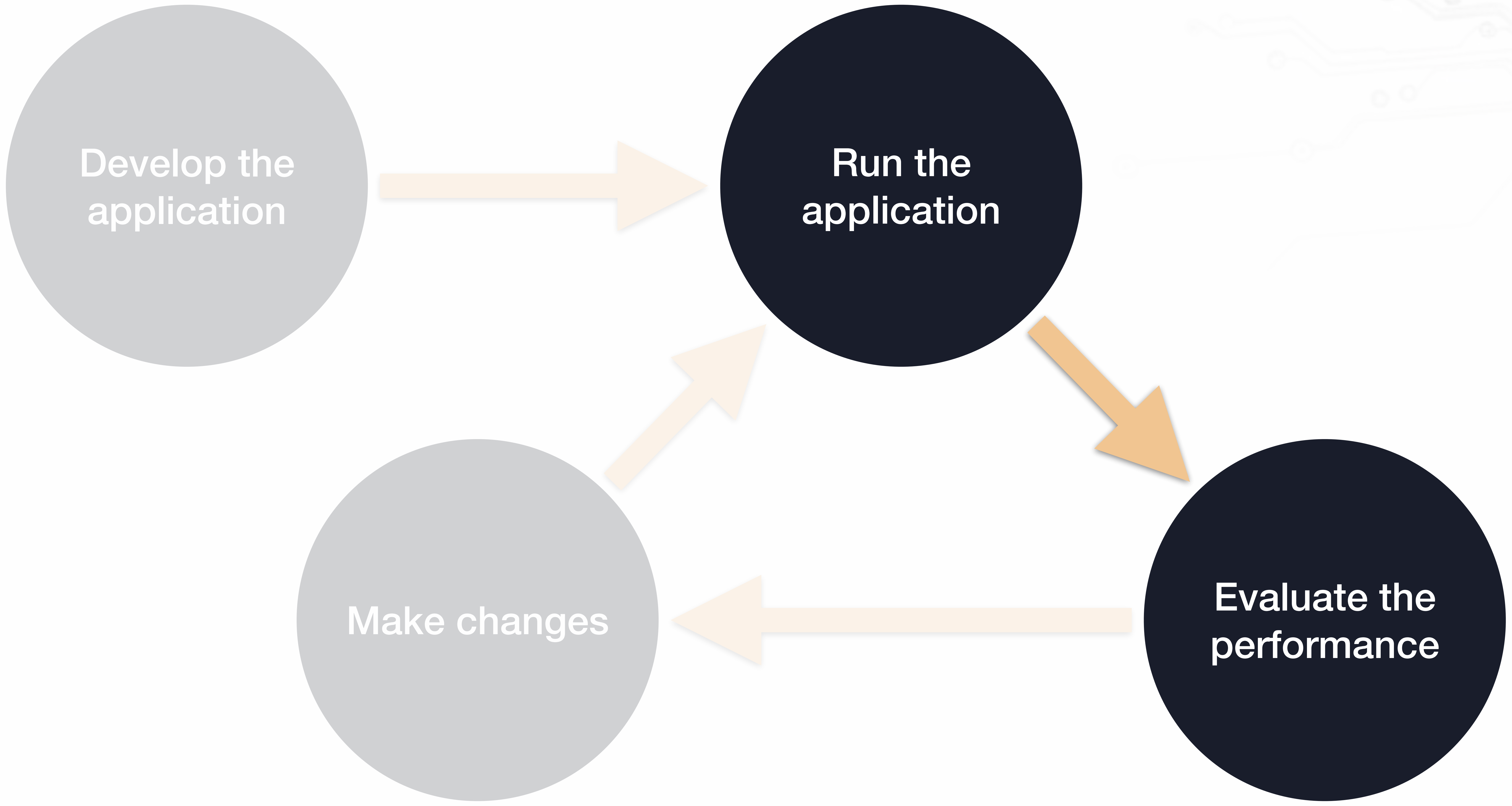
HPC applications

A typical lifecycle



HPC applications

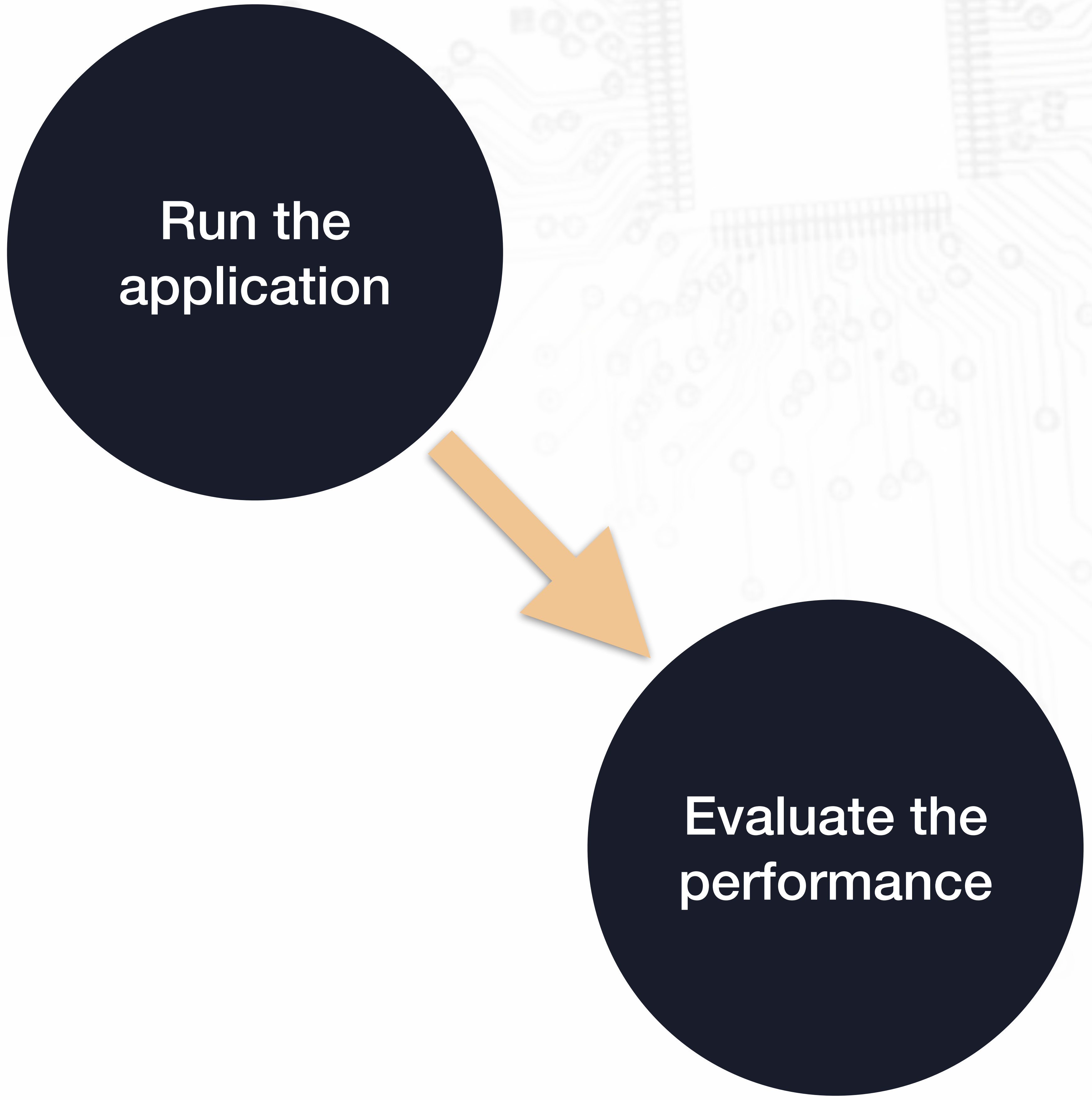
A typical lifecycle



HPC applications

How to evaluate the performance?

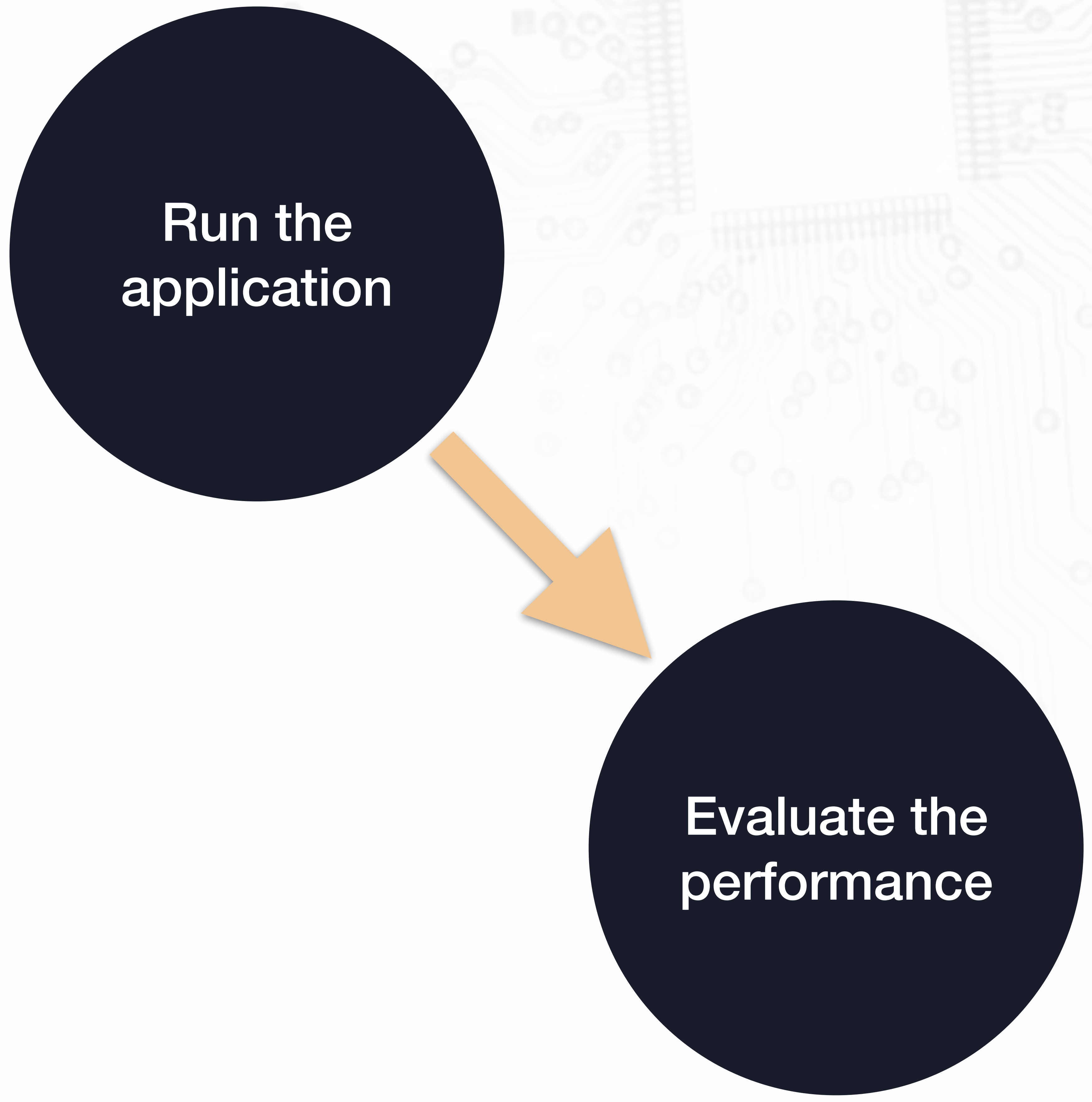
- Using tools, e.g. Vtune, Scalasca, nvprof, memory-profiler
- Measuring the execution time
- Measuring the memory footprint
- Measuring the network load
- Comparing the performance with similar applications/models
- Comparing the performance across different hardware
- Using the “guts feeling”



HPC applications

How to evaluate the performance?

- An application consists of multiple levels of complexity
- Each level may have certain effect on the performance and should be evaluated individually
  - **IO**
  - **Communications**
  - **Computations**
  - **Memory**



IO

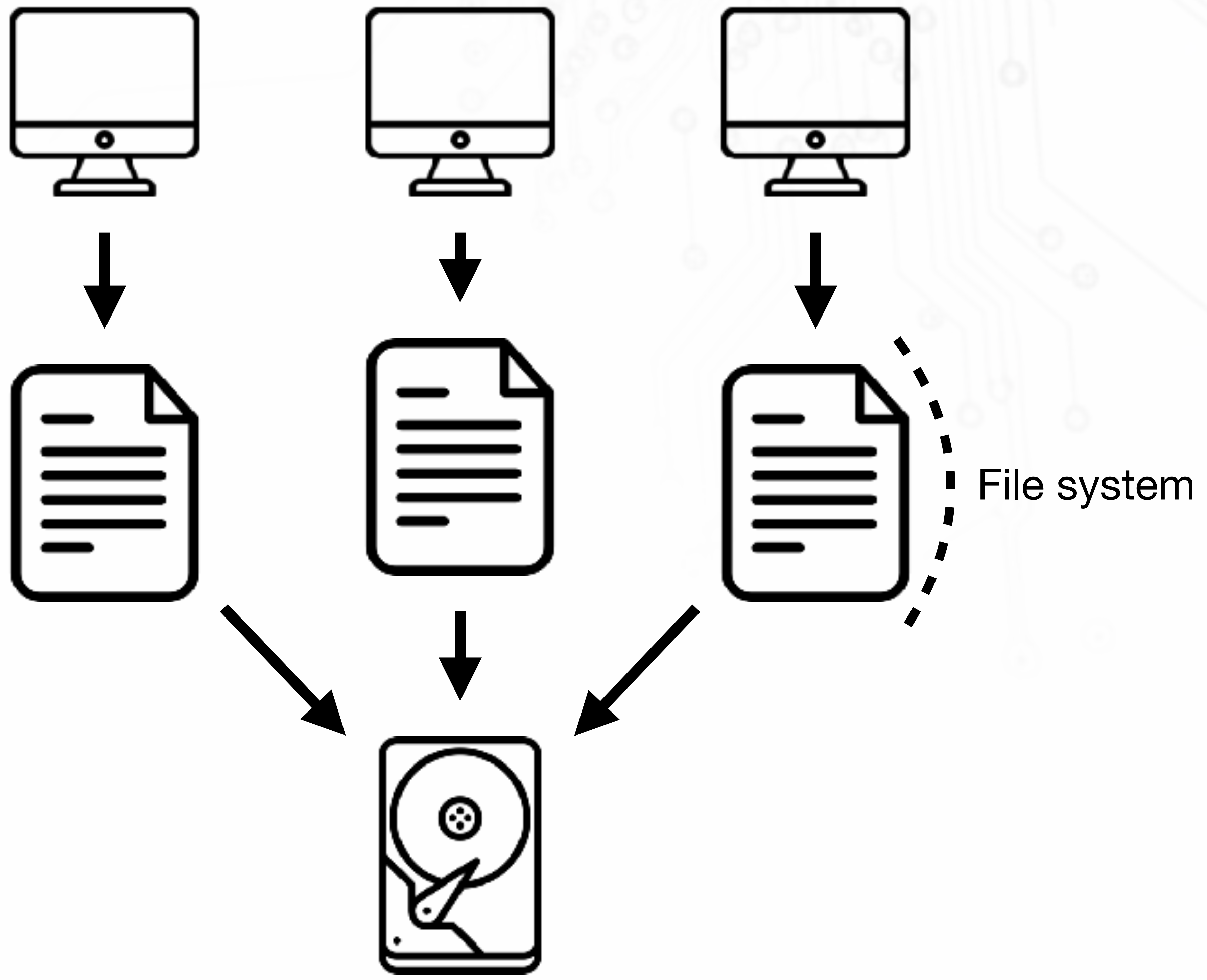
HPC applications

IO

- Parallel vs sequential filesystems
- We can measure read/write speed
- We can use ASCII/Binary
- We can use different file formats, as some are inherently sequential, some are parallel
- The easiest way to evaluate performance is to measure the elapsed time of IO operations

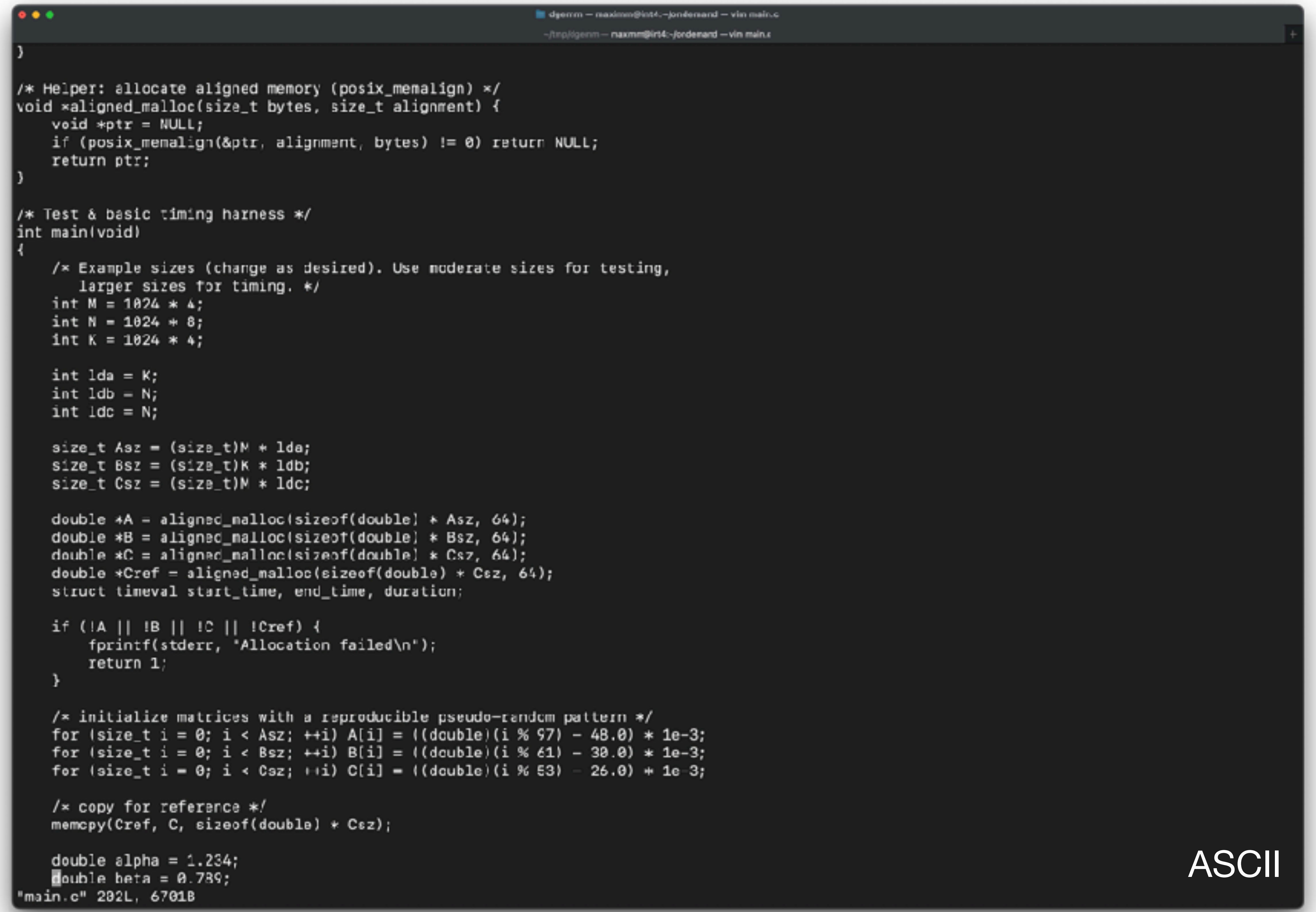
10M vector of 64-bit FP values on NVME SSD

Format	IO operation	Speed, [MB/s]
ASCII	Read	29.7
	Write	24.9
Binary	Read	2,666.4
	Write	1,586.5



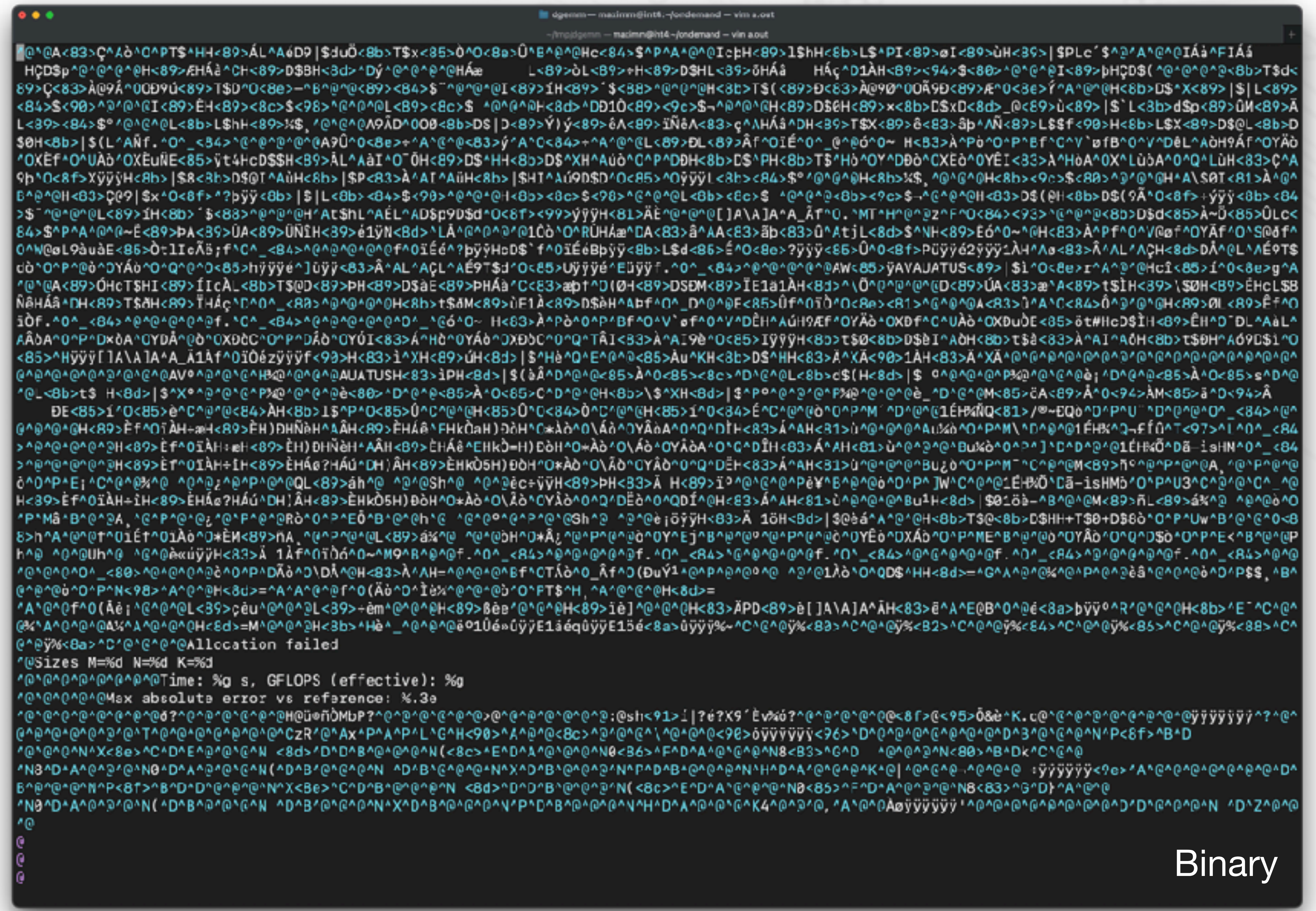
HPC applications

IO



```
}

/* Helper: allocate aligned memory (posix_memalign) */
void *aligned_malloc(size_t bytes, size_t alignment) {
    void *ptr = NULL;
    if (posix_memalign(&ptr, alignment, bytes) != 0) return NULL;
    return ptr;
}

/* Test & basic timing harness */
int main(void)
{
    /* Example sizes (change as desired). Use moderate sizes for testing,
       larger sizes for timing. */
    int M = 1024 * 4;
    int N = 1024 * 8;
    int K = 1024 * 4;

    int lda = K;
    int ldb = N;
    int ldc = N;

    size_t Asz = (size_t)M * lda;
    size_t Bsz = (size_t)K * ldb;
    size_t Csz = (size_t)M * ldc;

    double *A = aligned_malloc(sizeof(double) * Asz, 64);
    double *B = aligned_malloc(sizeof(double) * Bsz, 64);
    double *C = aligned_malloc(sizeof(double) * Csz, 64);
    double *Cref = aligned_malloc(sizeof(double) * Csz, 64);
    struct timeval start_time, end_time, duration;

    if (!A || !B || !C || !Cref) {
        fprintf(stderr, "Allocation failed\n");
        return 1;
    }

    /* initialize matrices with a reproducible pseudo-random pattern */
    for (size_t i = 0; i < Asz; ++i) A[i] = ((double)(i % 97) - 48.0) * 1e-3;
    for (size_t i = 0; i < Bsz; ++i) B[i] = ((double)(i % 61) - 30.0) * 1e-3;
    for (size_t i = 0; i < Csz; ++i) C[i] = ((double)(i % 53) - 26.0) * 1e-3;

    /* copy for reference */
    memcpy(Cref, C, sizeof(double) * Csz);

    double alpha = 1.234;
    double beta = 0.789;
"main.c" 202L, 6701B
```

ASCII



```
[illegible]
@^@ÿ%<8a>^C^@^@^@^@Allocation failed
^@Sizes M=%d N=%d K=%d
^@^@^@^@^@^@^@^@Time: %g s, GFLOPS (effective): %g
^@^@^@^@^@Max absolute error vs reference: %.3e
[illegible]
```

Binary

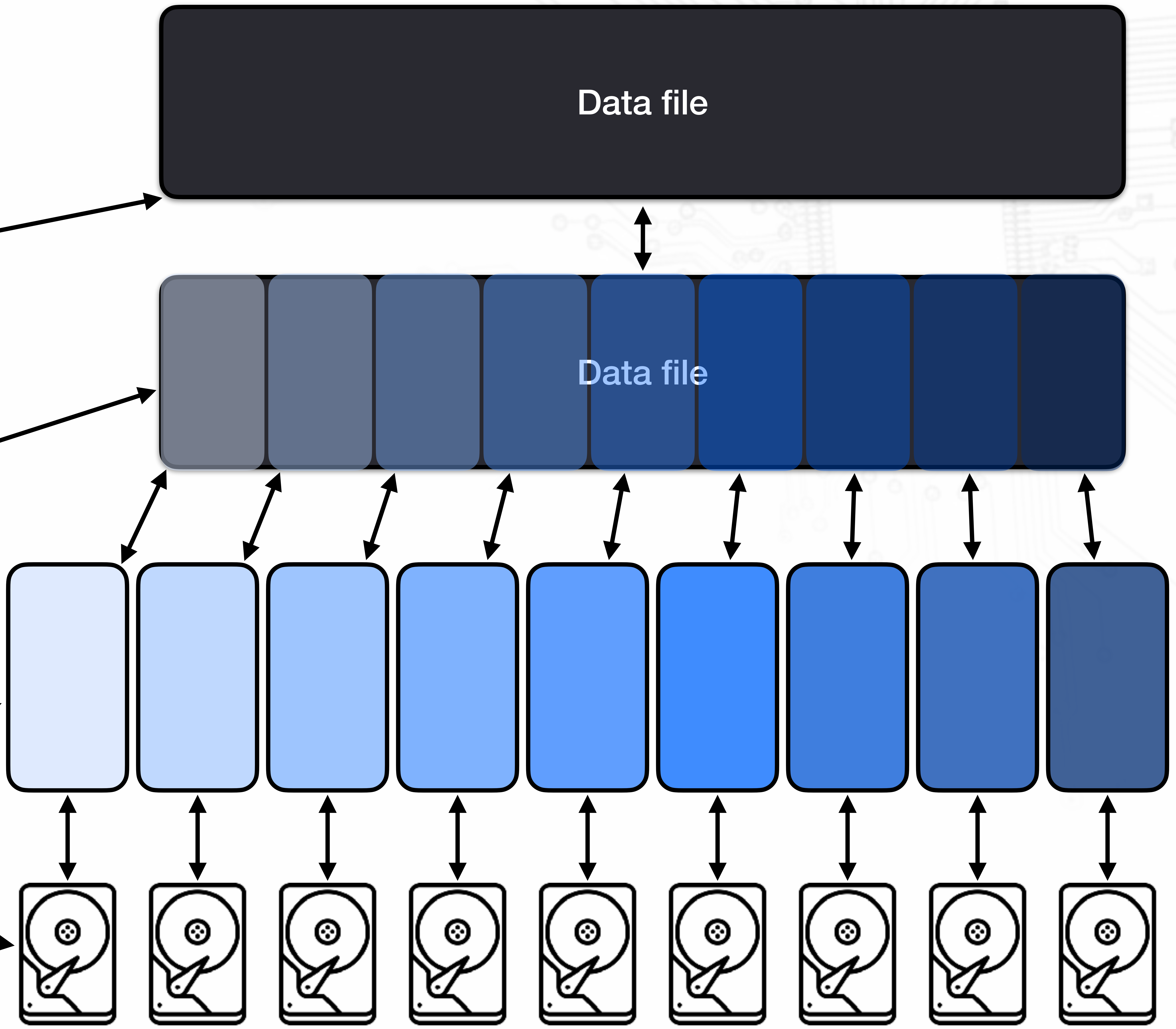
HPC applications

IO

One single file

FS automatically divides this file into stripes

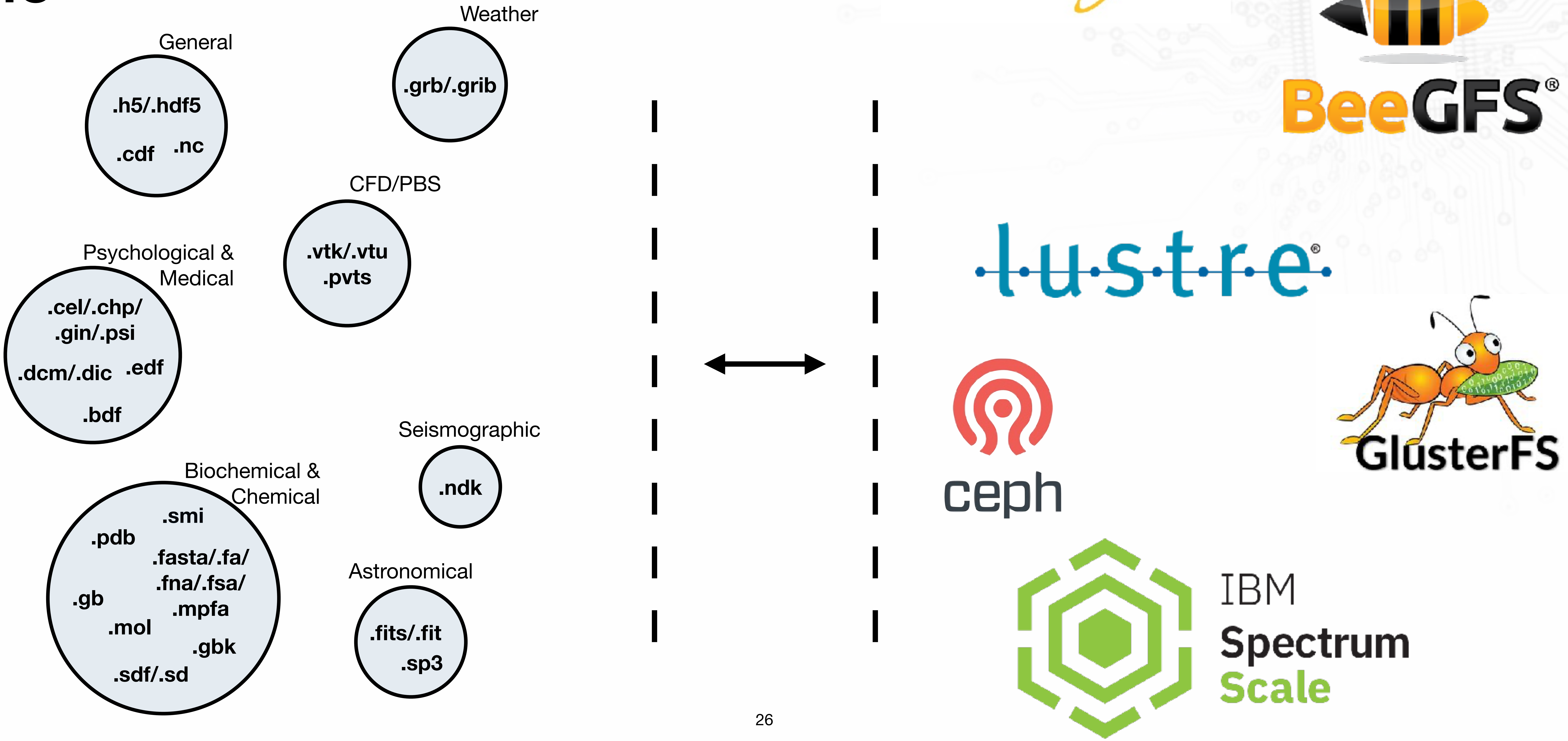
IO operation is split across multiple drives



1. https://www.flaticon.com/free-icon/hard-disk-drive_2004609

HPC applications

IO



HPC applications

IO

Hot

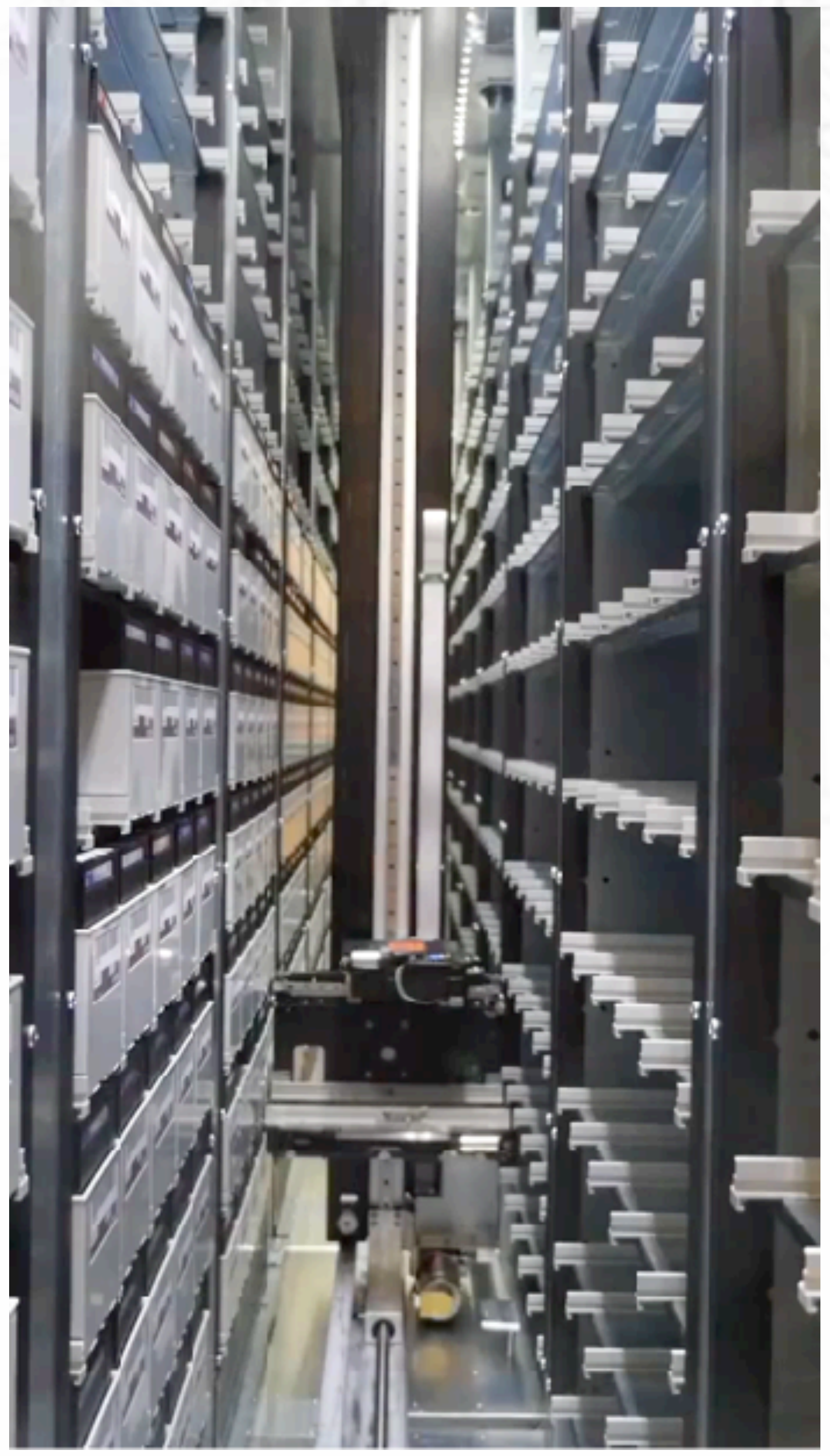
- Lowest latency (very fast)
- Highest cost
- Stored on high-performance SSDs, in-memory databases, or real-time systems

Warm

- Moderate speed
- Medium cost
- Often stored on slightly slower SSDs or lower-tier cloud storage

Cold

- High latency (access may take minutes or hours)
- Very low cost
- Stored on low-cost HDDs, tape drives, or deep archive cloud storage

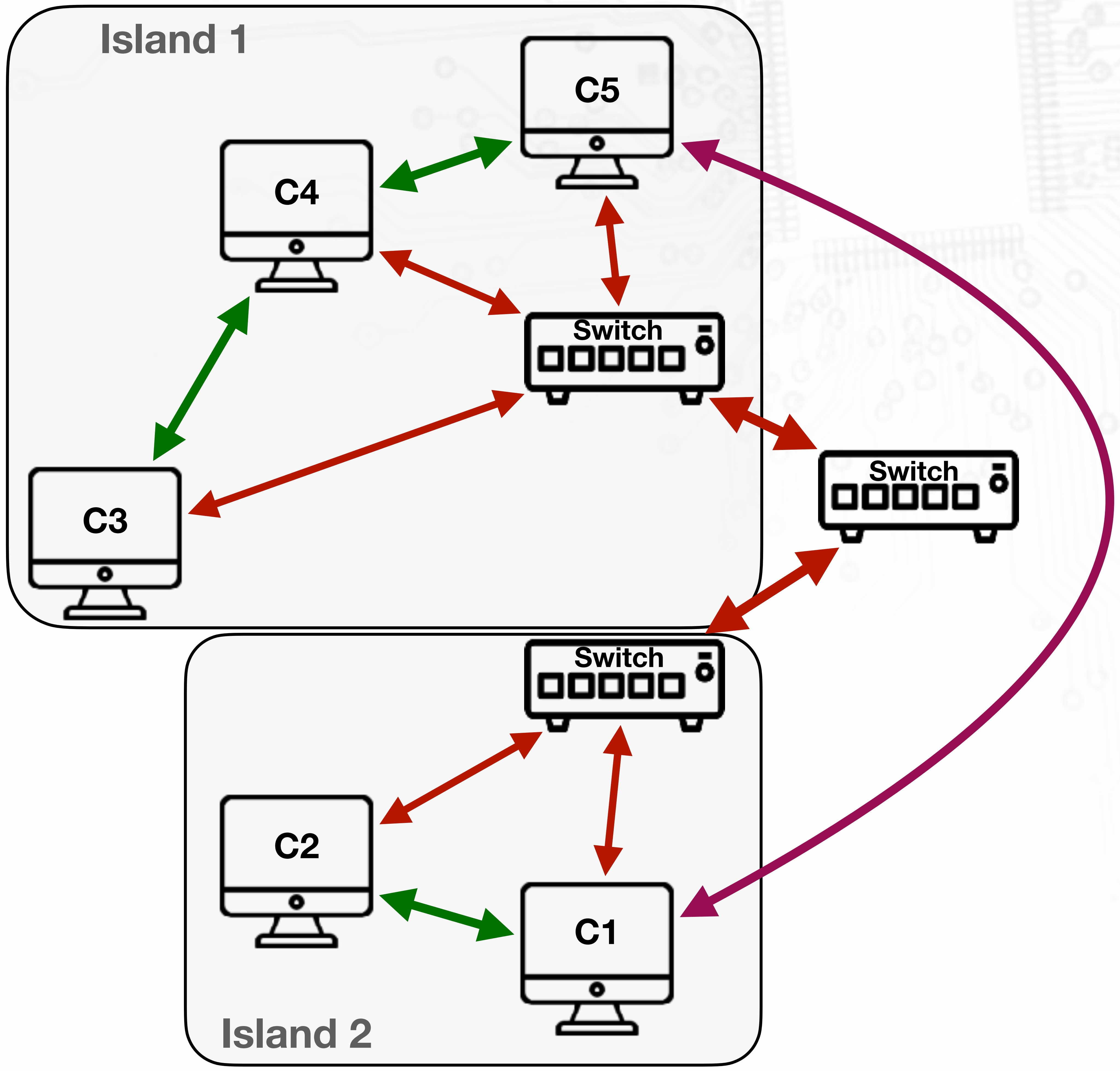


Communications

HPC applications

Communications

- Distribution of the workload across multiple processes
- Main influence from:
  - the network
  - the communication frequency
  - the message size
  - the file system

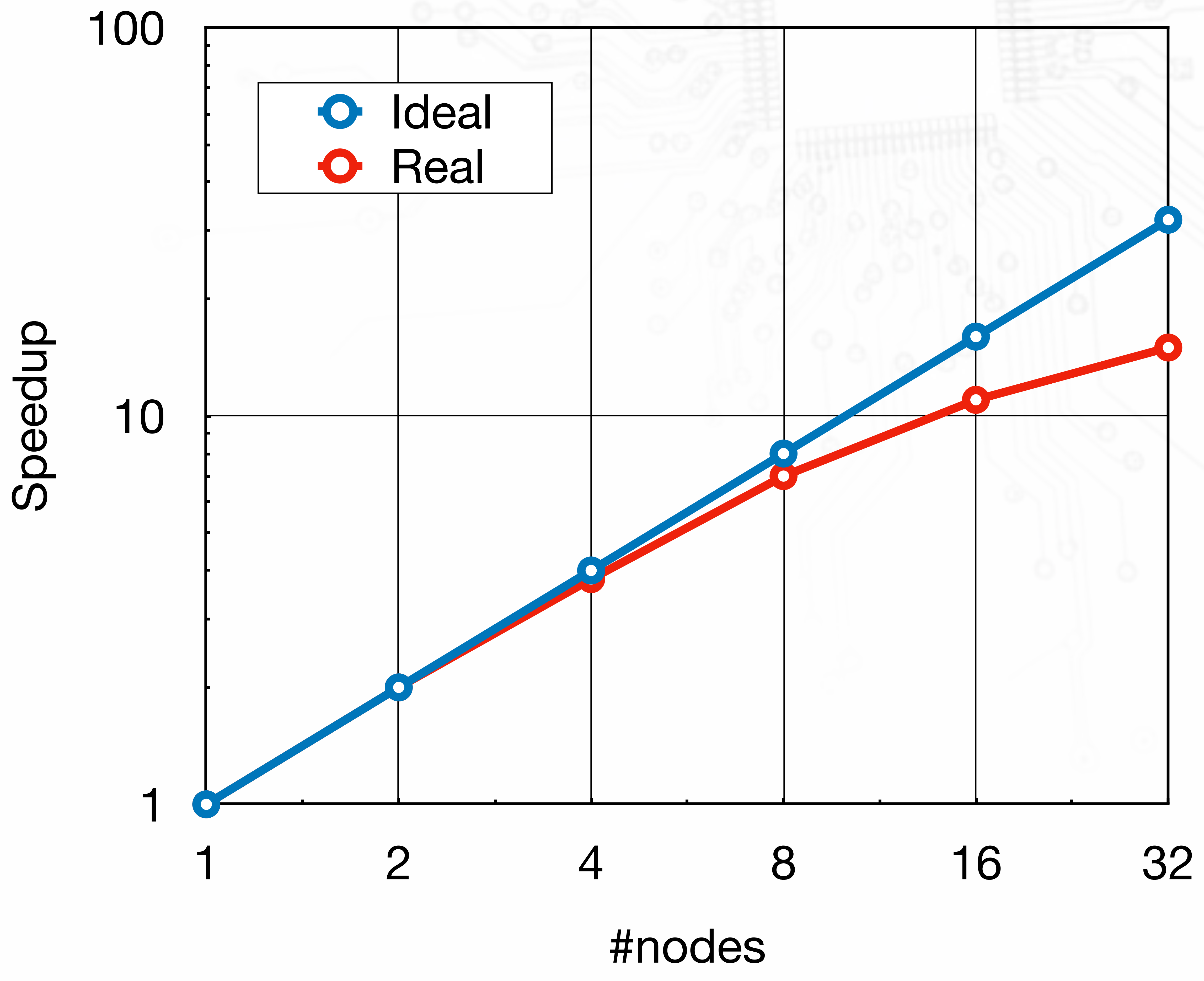


1. https://www.flaticon.com/free-icon/computer_2344269
2. https://www.flaticon.com/free-icon/switch_1089006

HPC applications

Communications

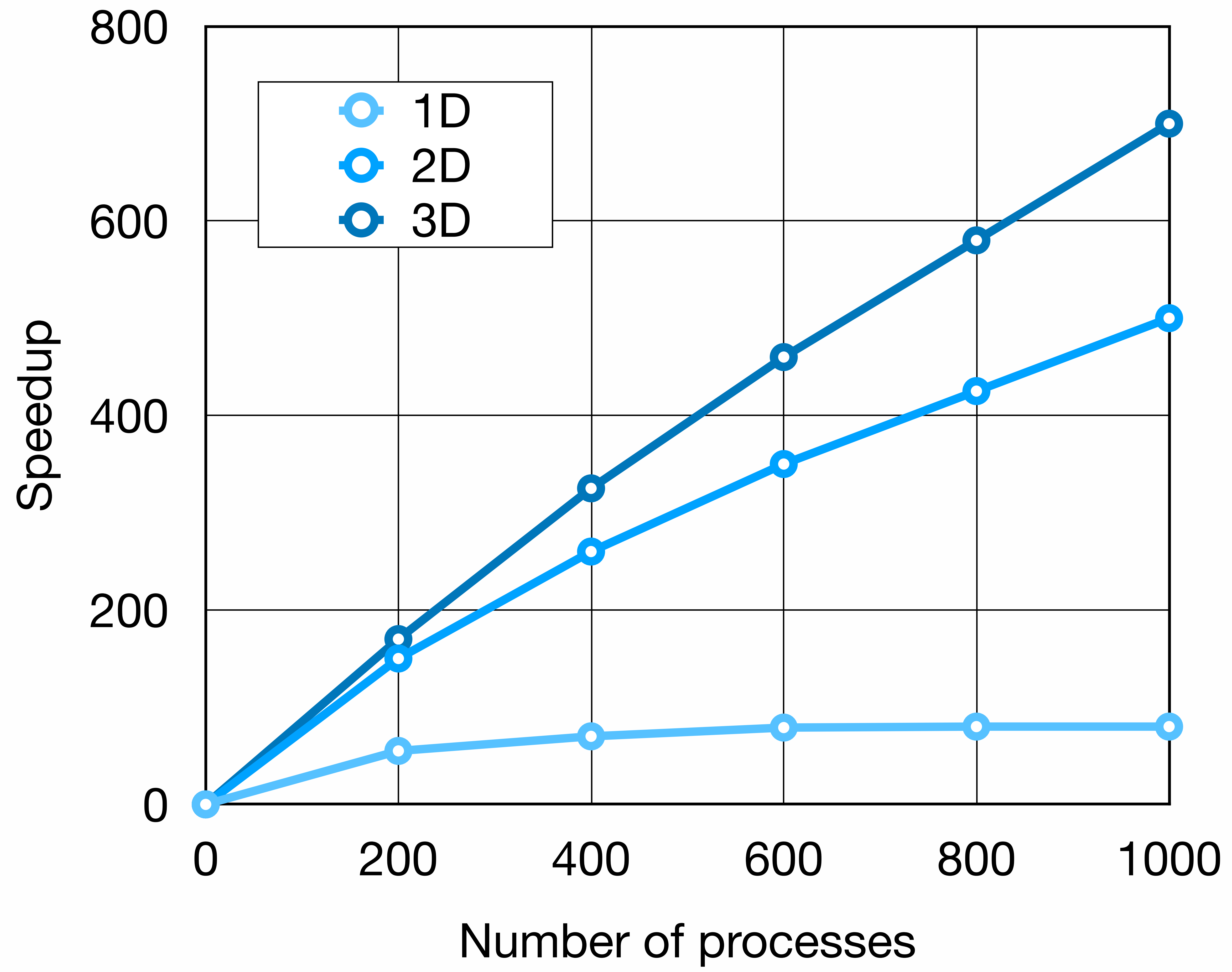
- Leads to a decrease in performance as the number of active processes increases
- Optimisation may require knowledge of the network structure
- Some frameworks allow for run-time tuning of the communication algorithms (e.g. with MPI tune)
- Scales with O(√P) for *P* processes



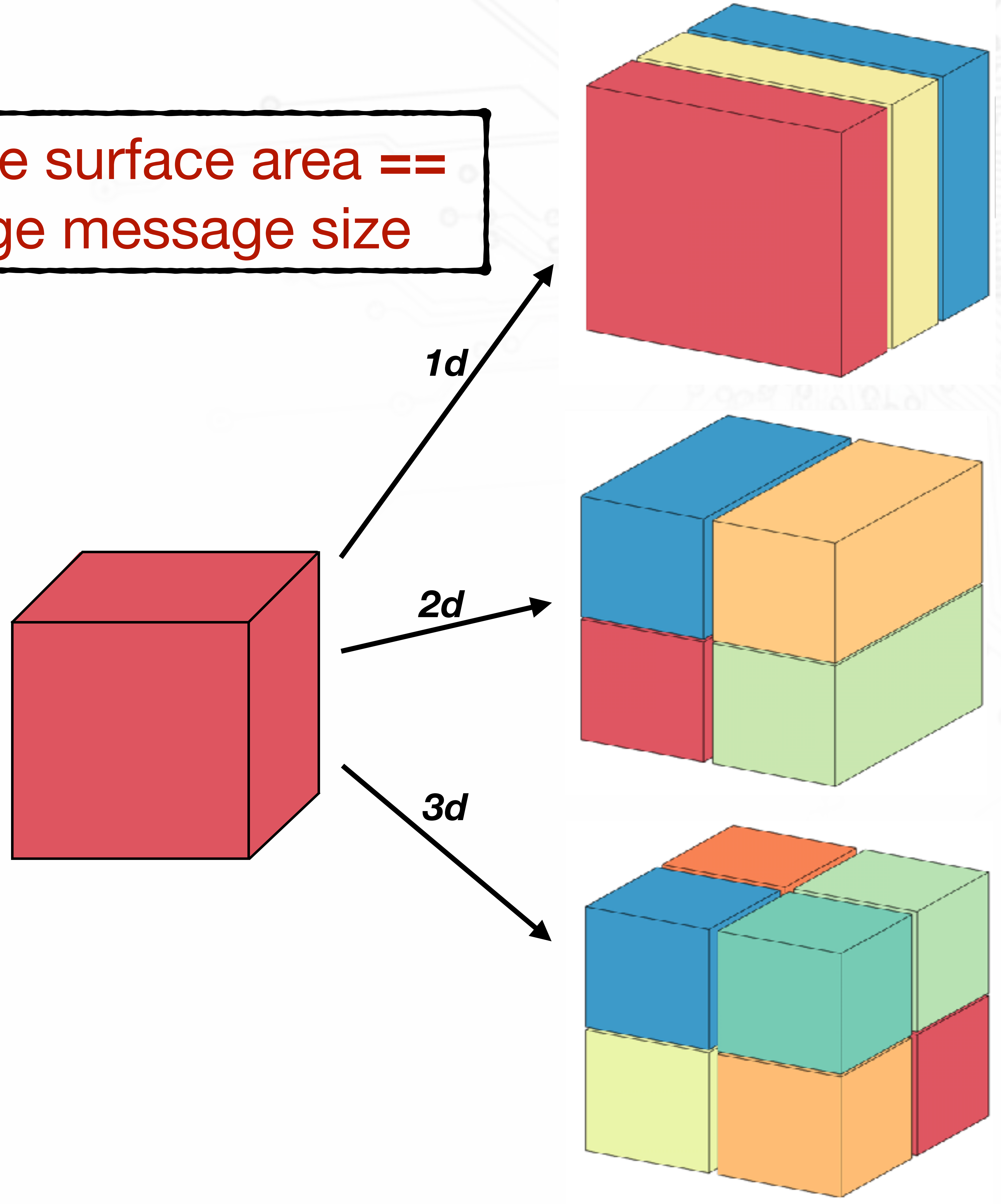
HPC applications

Communications

Theoretical speedup



Large surface area == large message size



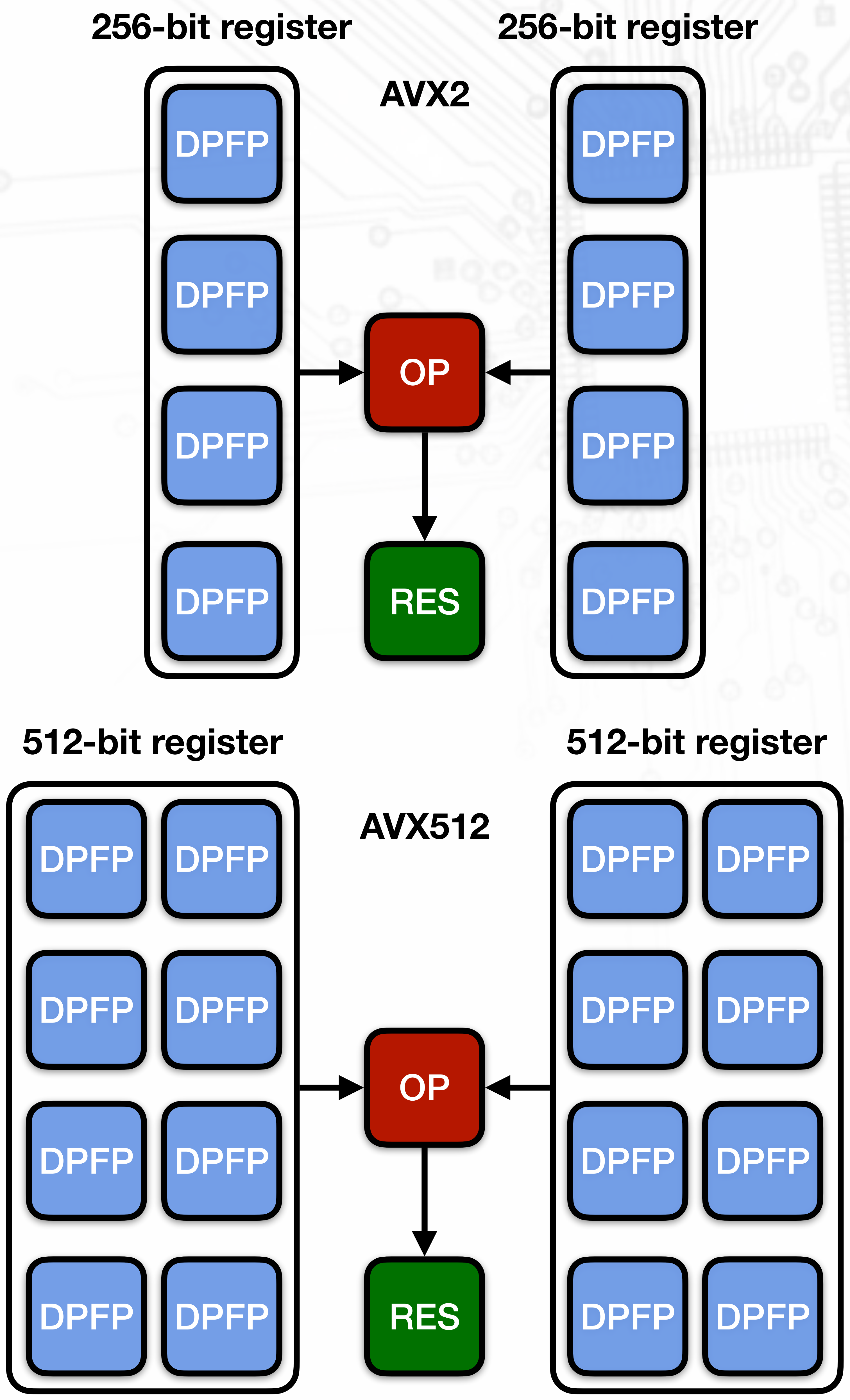
Computations

HPC applications

Computations

- **SIMD directive**: multiple iterations of the loop can be executed concurrently using SIMD instructions
- For successful vectorisation, the **data should be aligned (not just contiguous)**
- CPUs have registers of different sizes, choose the data type wisely
- Some APIs (e.g., OpenMP) and libraries (e.g., numpy) provide embedded vectorisation

```
#pragma omp simd
for (int n = 0; n < N; ++n) {
    compute(…);
}
```



HPC applications

Computations

```
# Dot product
import time
import numpy
import array

# 8 bytes size int
a = array.array('q')
b = array.array('q')
...

# classic dot product of vectors implementation
tic = time.process_time()
dot = 0.0;
for i in range(len(a)):
    dot += a[i] * b[i]
toc = time.process_time()

print(“..."');

n_tic = time.process_time()
n_dot_product = numpy.dot(a, b)
n_toc = time.process_time()
print(“...");
```

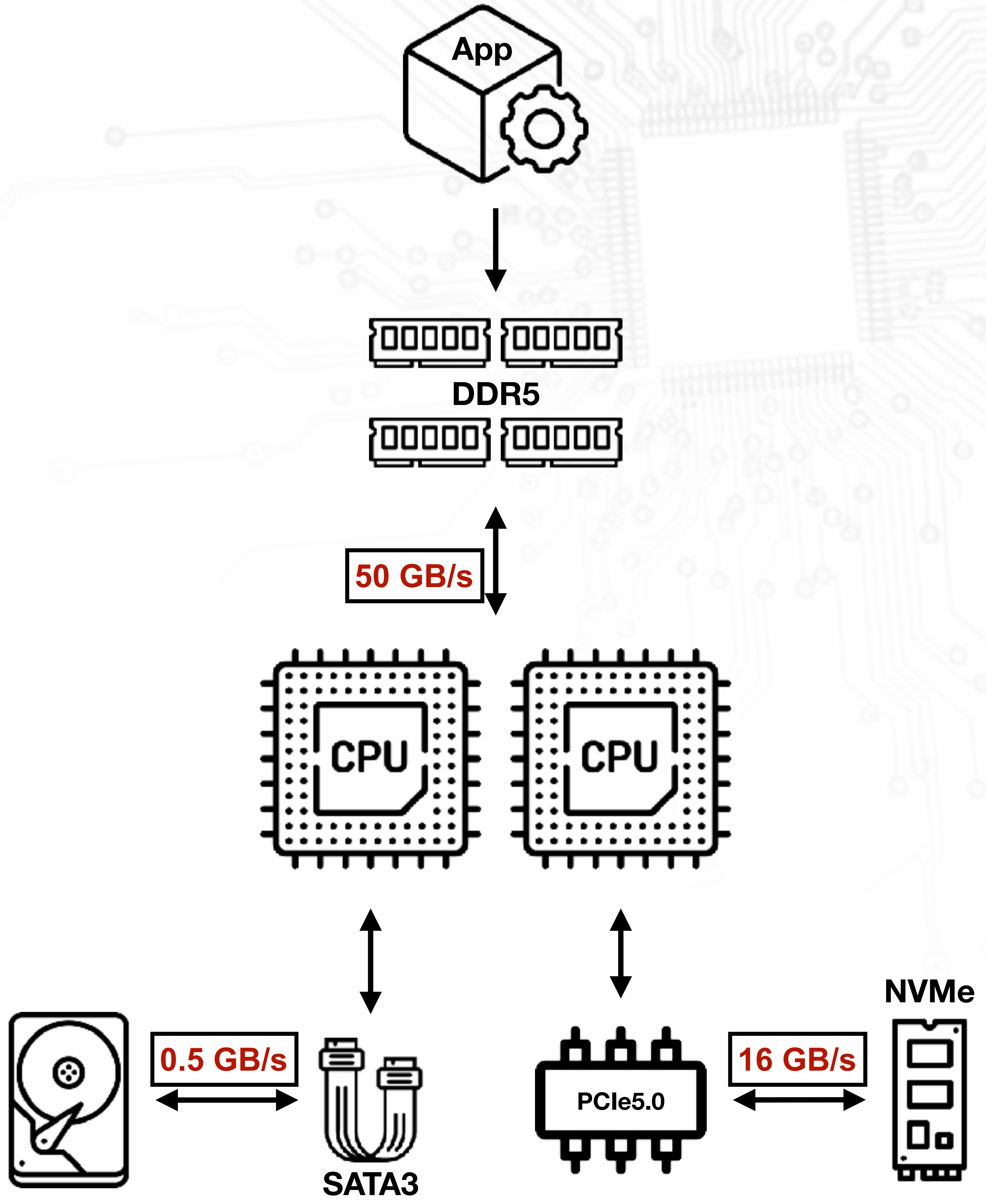
```
dot_product = 833323333350000.0
Computation time = 24.998999999999995ms

n_dot_product = 833323333350000
Computation time = 0.09100000000000774ms
```

Memory

HPC applications

Memory footprint

- Reducing the memory footprint decreases the frequency of accessing slower storage layers
- Minimising cache misses helps ensure that CPU caches (L1/L2/L3) remain available for essential data and critical operations
- A large memory footprint increases the likelihood of the OS triggering paging, which severely impacts performance
- Lower memory usage enables more efficient data transfer across the network by reducing payload size and overhead
- Reclaiming and cleaning up unused memory is vital to maintaining optimal system resource utilisation



HPC applications

Data structures

sizeof(Foo) = 32B

```
struct Foo {
    double eps;
    char key1;
    int id;
    double time;
    char key2;
};
```

sizeof(Foo) = 24B

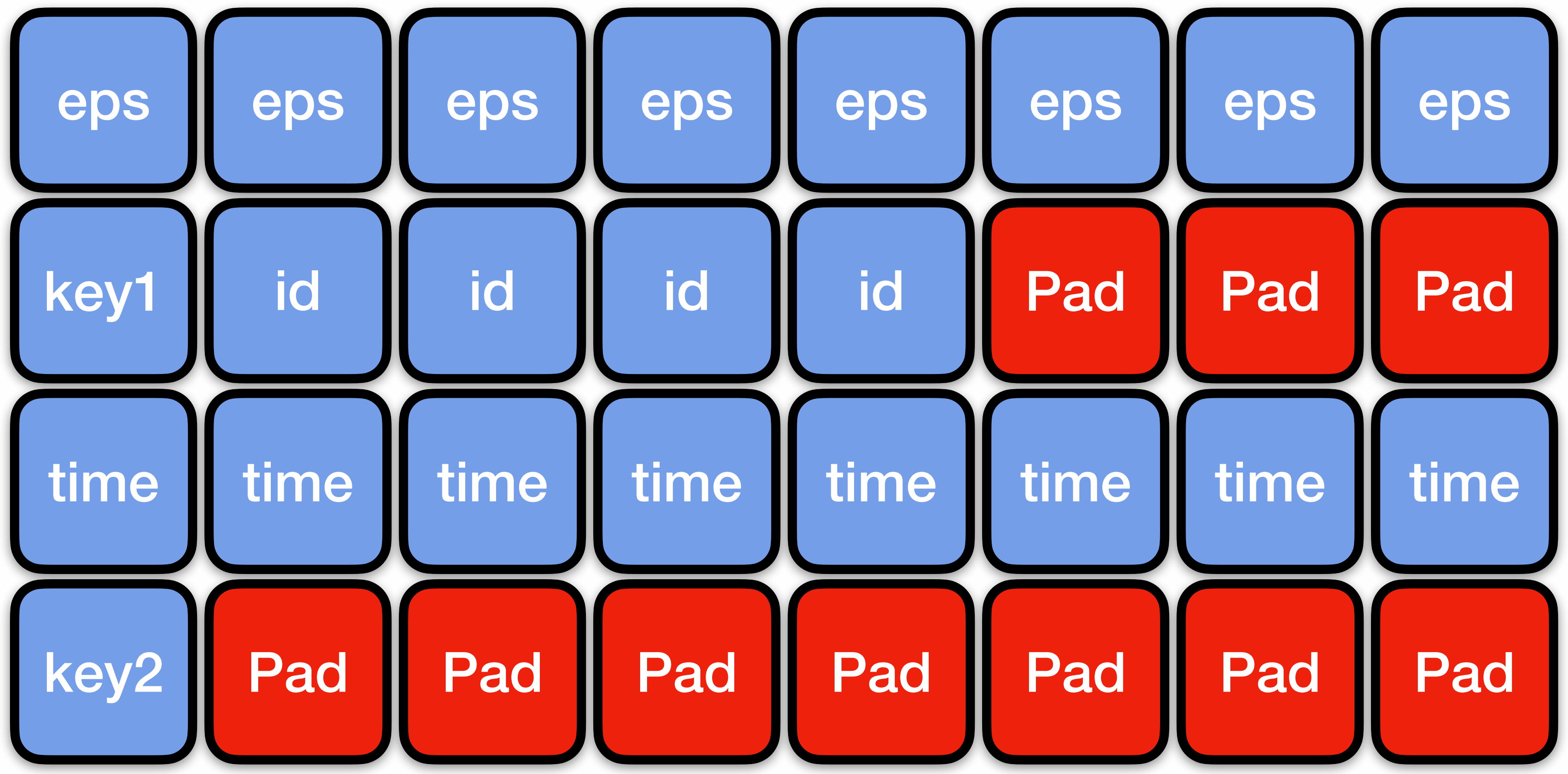
```
struct Foo {
    double eps;
    double time;
    char key1;
    char key2;
    int id;
};
```

HPC applications

Data structures

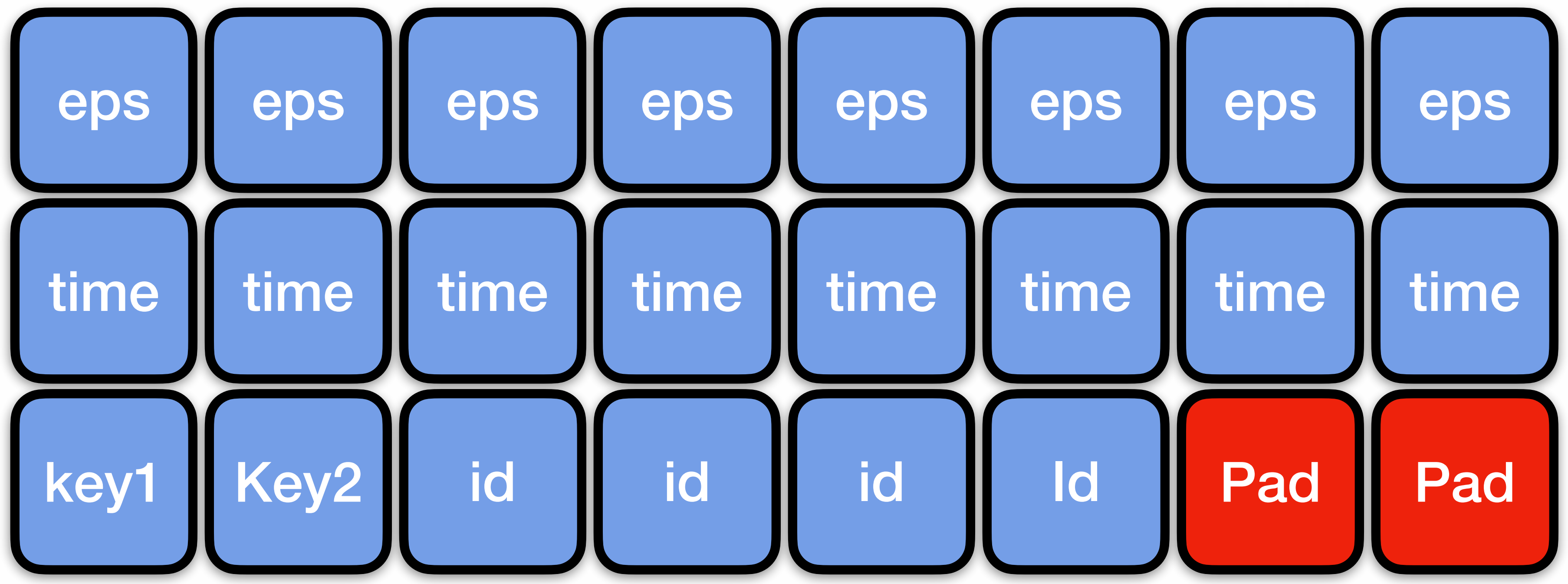
sizeof(Foo) = 32B

```
struct Foo {
    double eps;       // 8B
    char key1;        // 1B
    int id;           // 4B + padding 3B
    double time;      // 8B
    char key2;        // 1B + padding 7B
};
```


eps eps eps eps eps eps eps eps
key1 id id id id Pad Pad Pad
time time time time time time time time
key2 Pad Pad Pad Pad Pad Pad Pad

sizeof(Foo) = 24B

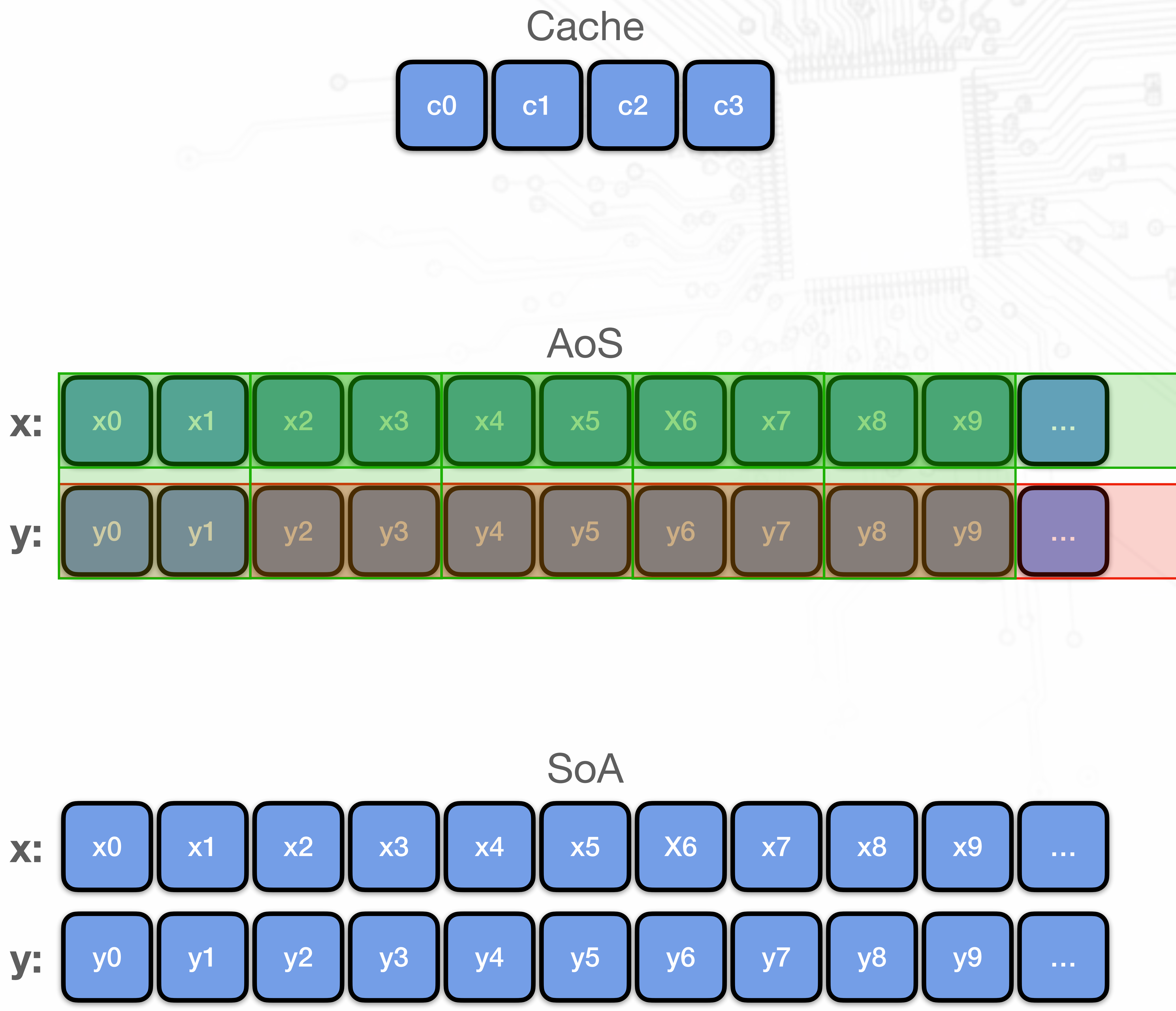
```
struct Foo {
    double eps;       // 8B
    double time;      // 8B
    char key1;        // 1B
    char key2;        // 1B
    int id;           // 4B + padding 2B
};
```


eps eps eps eps eps eps eps eps
time time time time time time time time
key1 Key2 id id id Id Pad Pad

HPC applications

AoS vs SoA

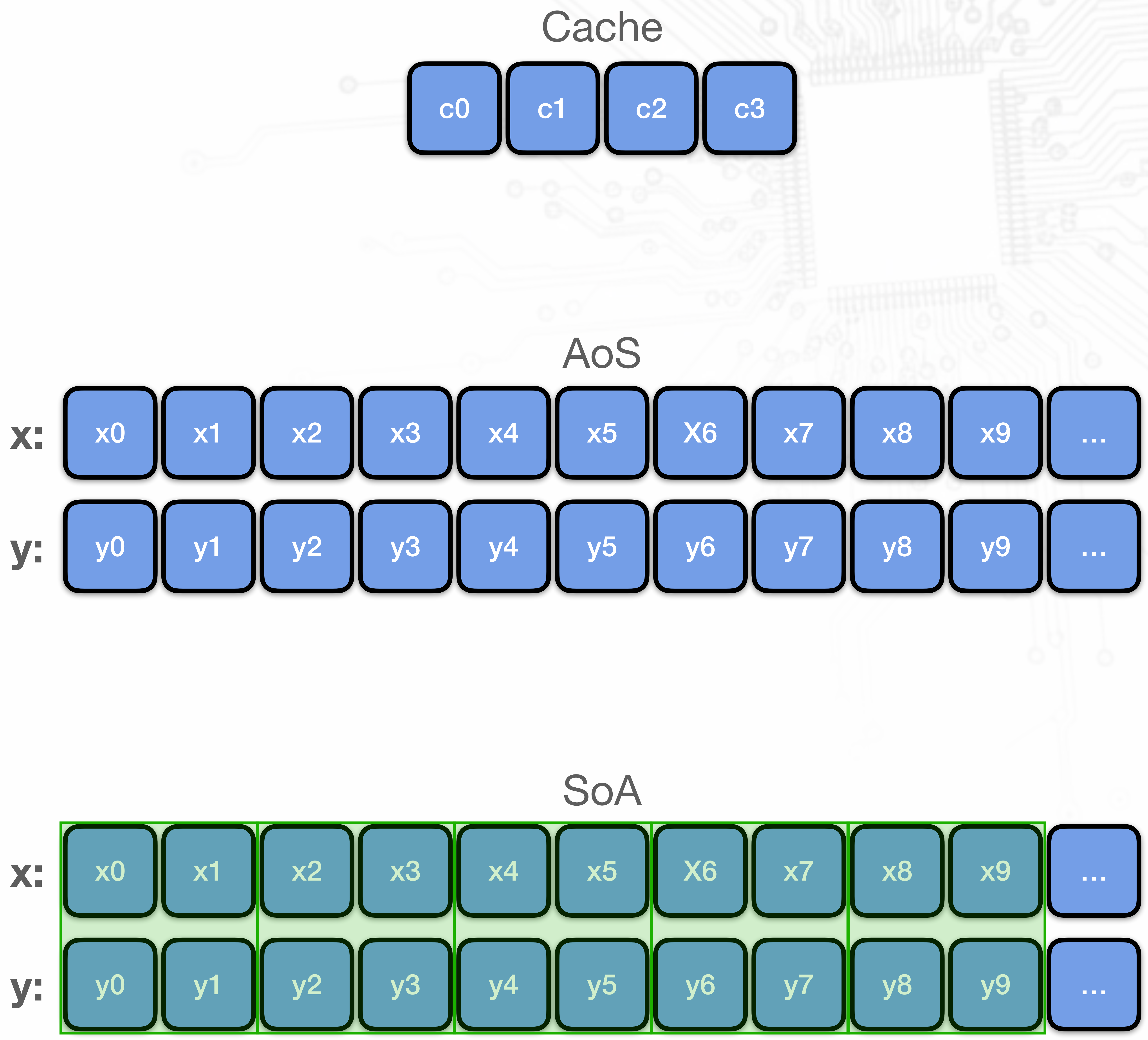
```
struct Foo {
    double x;
    double y;
};

struct Bar {
    double x[N];
    double y[N];
};

void main() {
    Foo foo[N];
    Bar bar;
    ...
    for (int n = 0; n < N; ++n) {
        foo[n].x = ...;
        foo[n].y = ...;
    }
    ...
    for (int n = 0; n < N; ++n) {
        bar.x[n] = ...;
        bar.y[n] = ...;
    }
    ...
}
```



HPC applications

AoS vs SoA

```
struct Foo {
    double x;
    double y;
};

struct Bar {
    double x[N];
    double y[N];
};

void main() {
    Foo foo[N];
    Bar bar;
    ...
    for (int n = 0; n < N; ++n) {
        foo[n].x = ...;
        foo[n].y = ...;
    }
    ...
    for (int n = 0; n < N; ++n) {
        bar.x[n] = ...;
        bar.y[n] = ...;
    }
    ...
}
```



Cache
c0
c1
c2
c3
AoS
x:
x0 x1 x2 x3 x4 x5 X6 x7 x8 x9 ...
y:
y0 y1 y2 y3 y4 y5 y6 y7 y8 y9 ...
SoA
x:
x0 x1 x2 x3 x4 x5 X6 x7 x8 x9 ...
y:
y0 y1 y2 y3 y4 y5 y6 y7 y8 y9 ...

Overview

HPC applications

Standard techniques

- Optimise **IO** operations
- Optimise **data transfers** (cache, network)
- Optimise **memory footprint**
- Reduce the **precision** (e.g. from 64-bit FP to 32-bit FP)
- Optimise **data structures** (e.g. from AOS to SOA)
- Make application **hardware-aware**
- Use **different** hardware (CPUs, GPUs, FPGAs)
- Change the **model/method** to a faster one
- Change the **backend** (e.g. linear algebra) to a faster one
- Use different **programming language**

Profiling And Optimisation

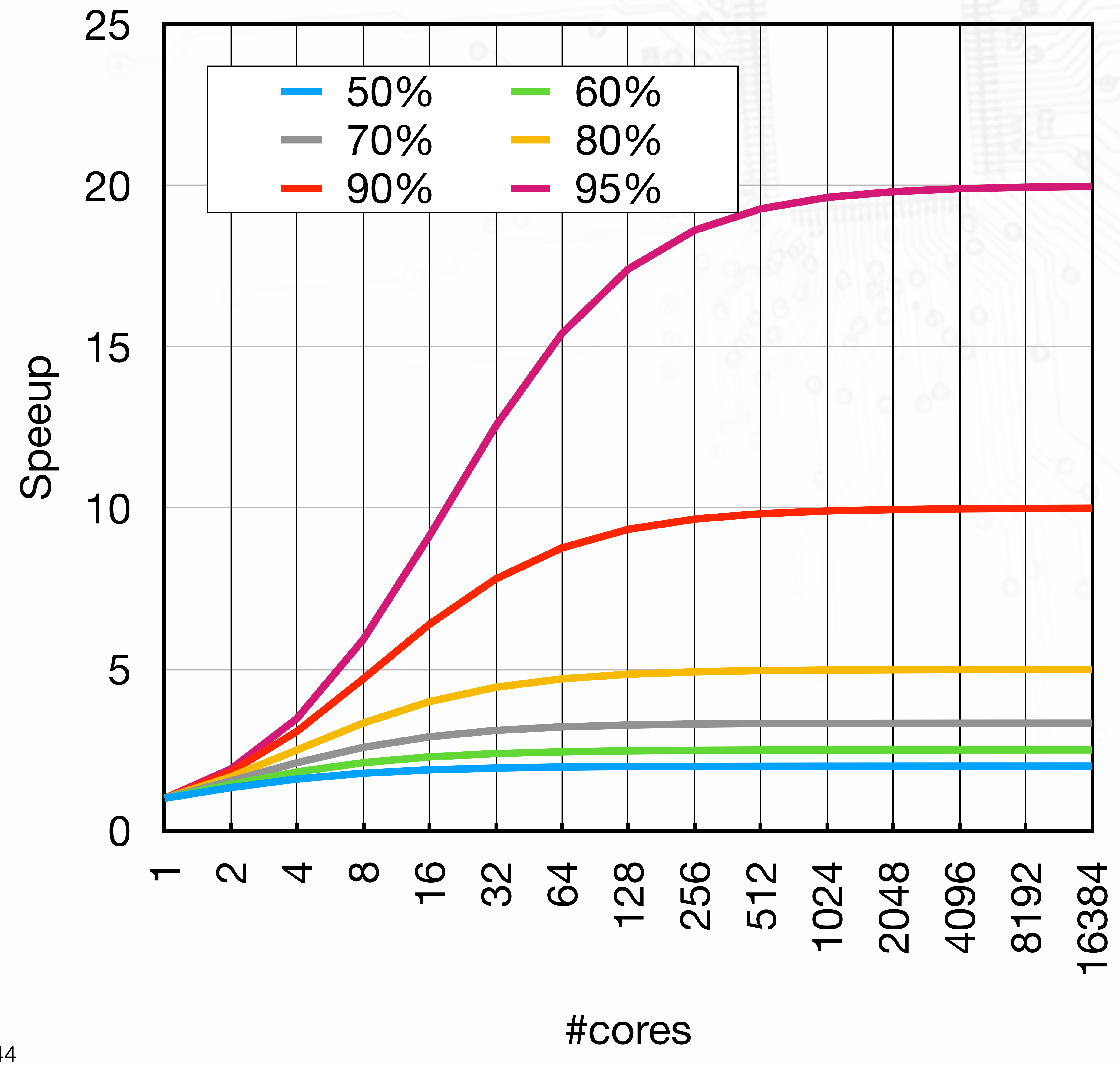
Scaling

Strong

- **Strong scaling** – the problem size is **fixed**, the number of processors is **increased**
- **Amdahl's law** - the speedup is limited by the fraction of the serial part of the software that is not amenable to parallelization

$$speedup = \frac{1}{s + p/N}$$

- s – proportion of the execution time spent on the serial part
- p – proportion of the execution time spent on the parallel part
- N – number of processes



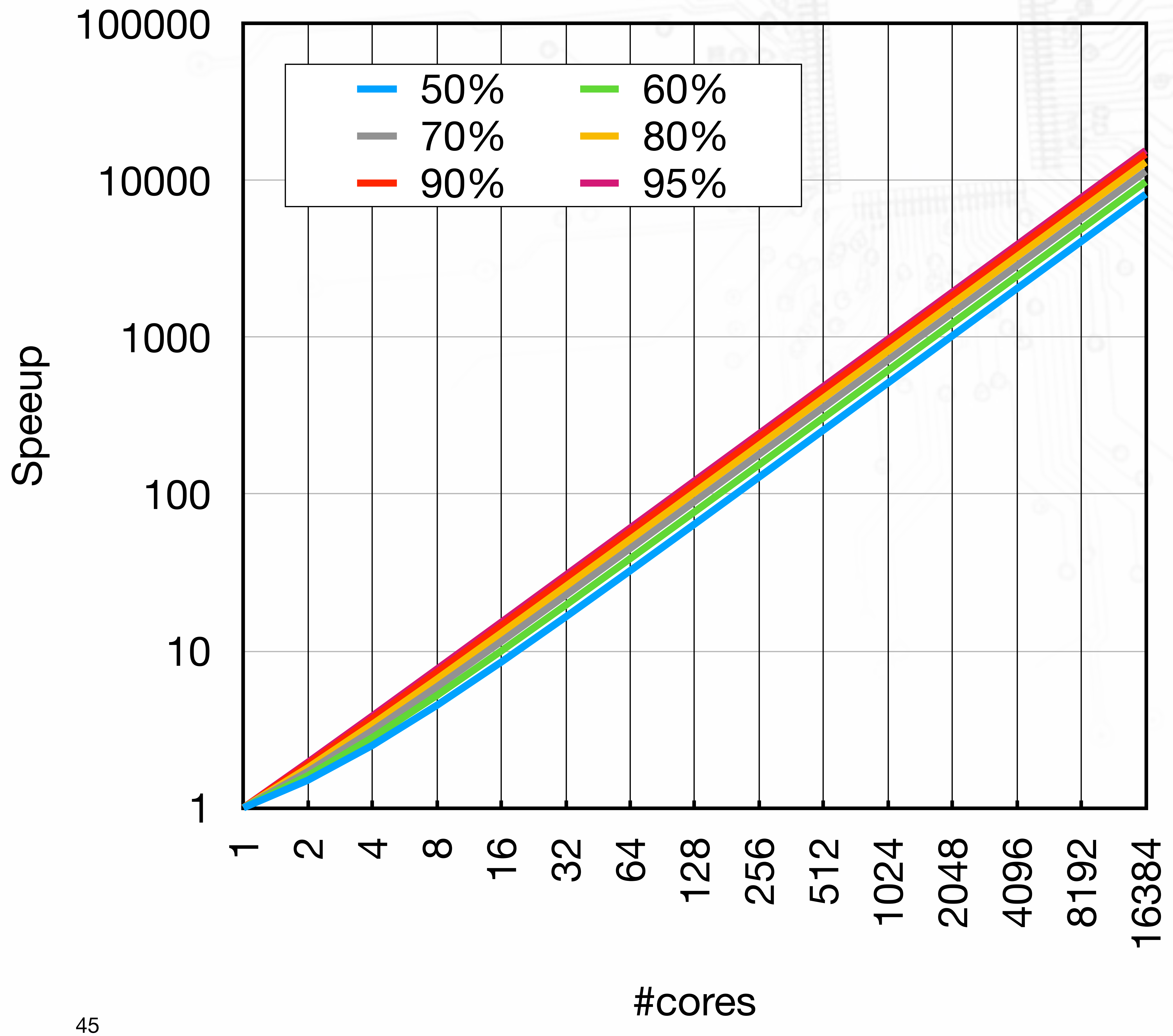
Scaling

Weak

- **Weak scaling** – the problem size **increases** with the number of processes
- **Gustafson's law** - the parallel part scales linearly with the amount of resources, and the serial part does not increase with respect to the size of the problem

$$speedup = s + p \cdot N$$

- s – proportion of the execution time spent on the serial part
- p – proportion of the execution time spent on the parallel part
- N – number of processes



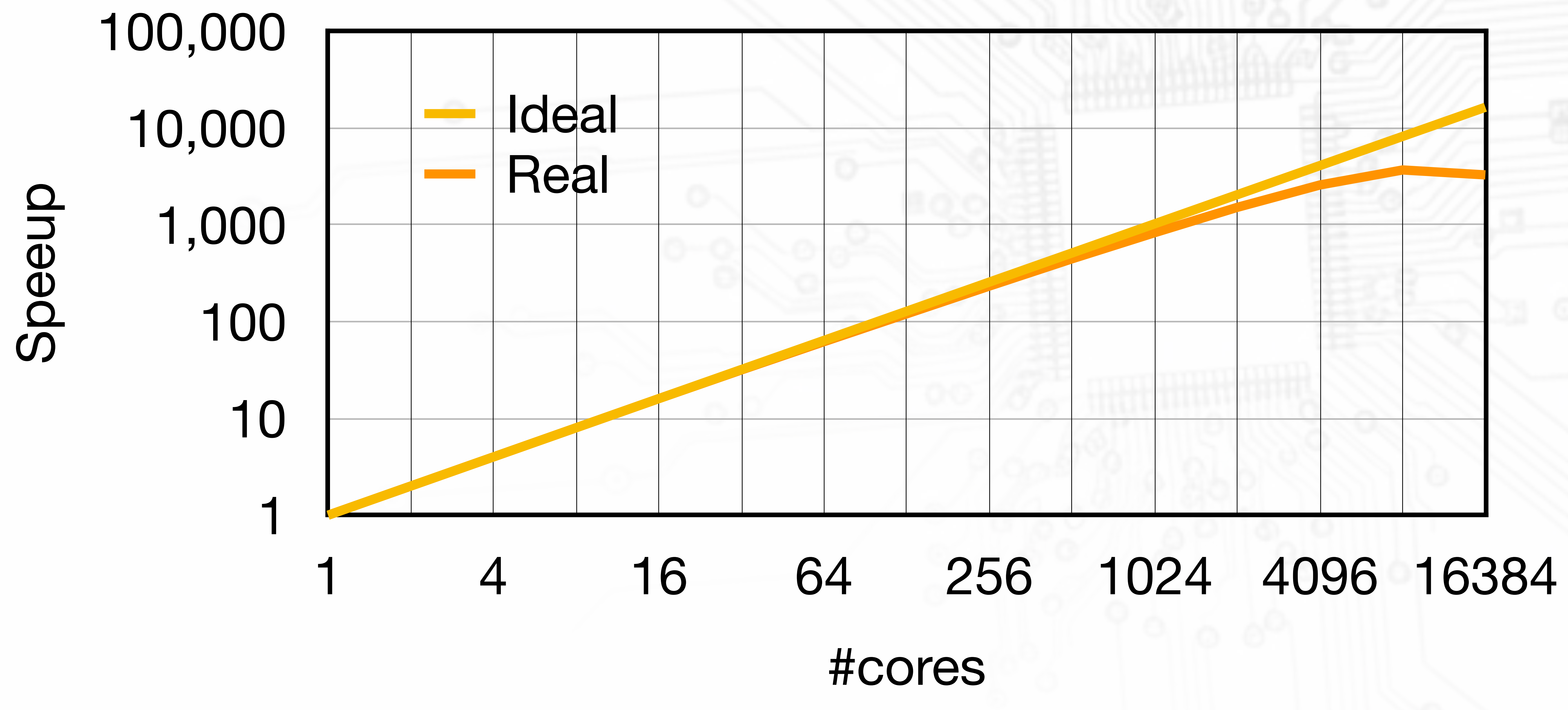
Scaling

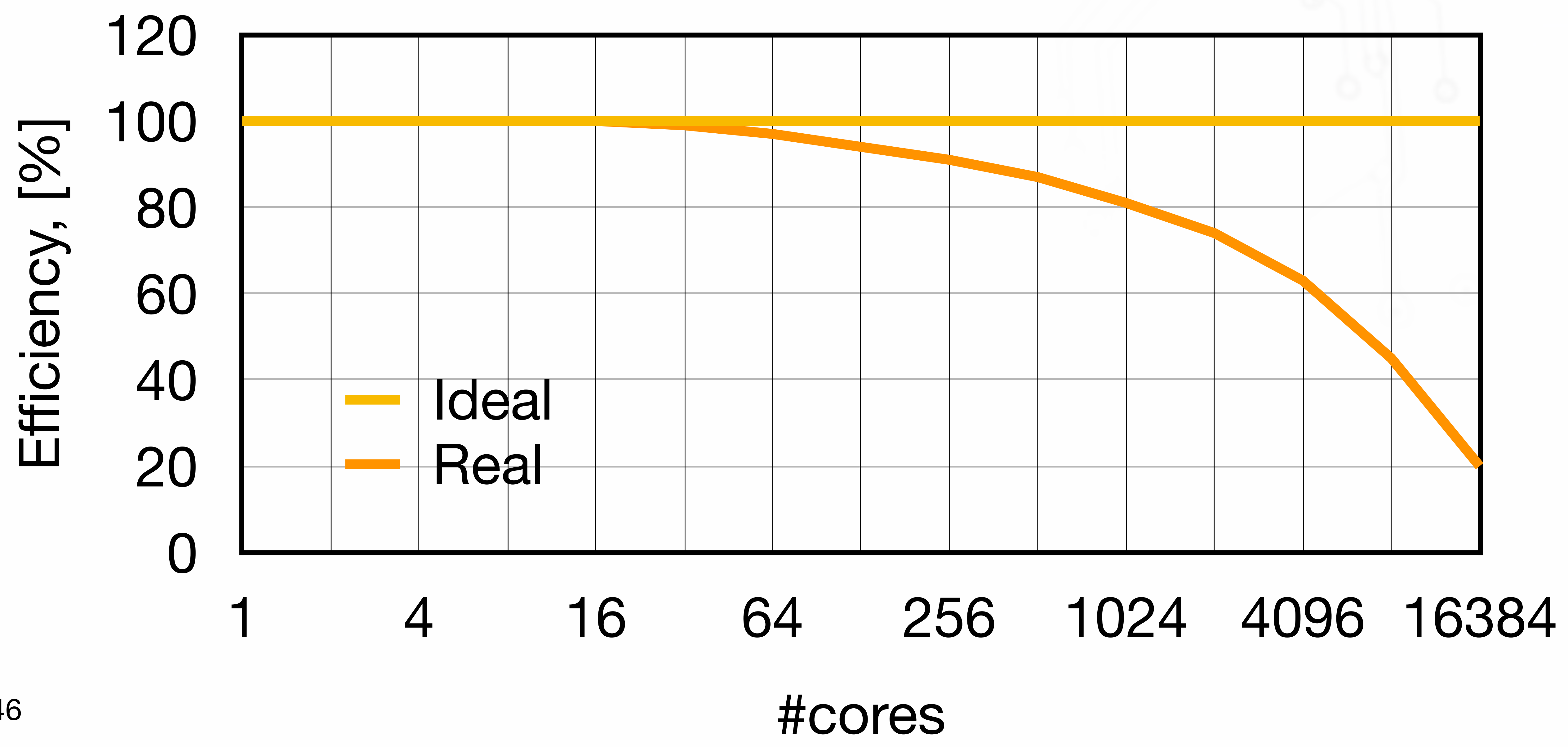
Efficiency

- Speedup and efficiency:

$$speedup = \frac{T_s}{T_N} \qquad efficiency = \frac{1}{N}\frac{T_s}{T_N}$$

- T_S – the amount of time needed to complete a serial task
- T_N – the amount of time needed to complete a parallel task on N processes
- N – number of processes

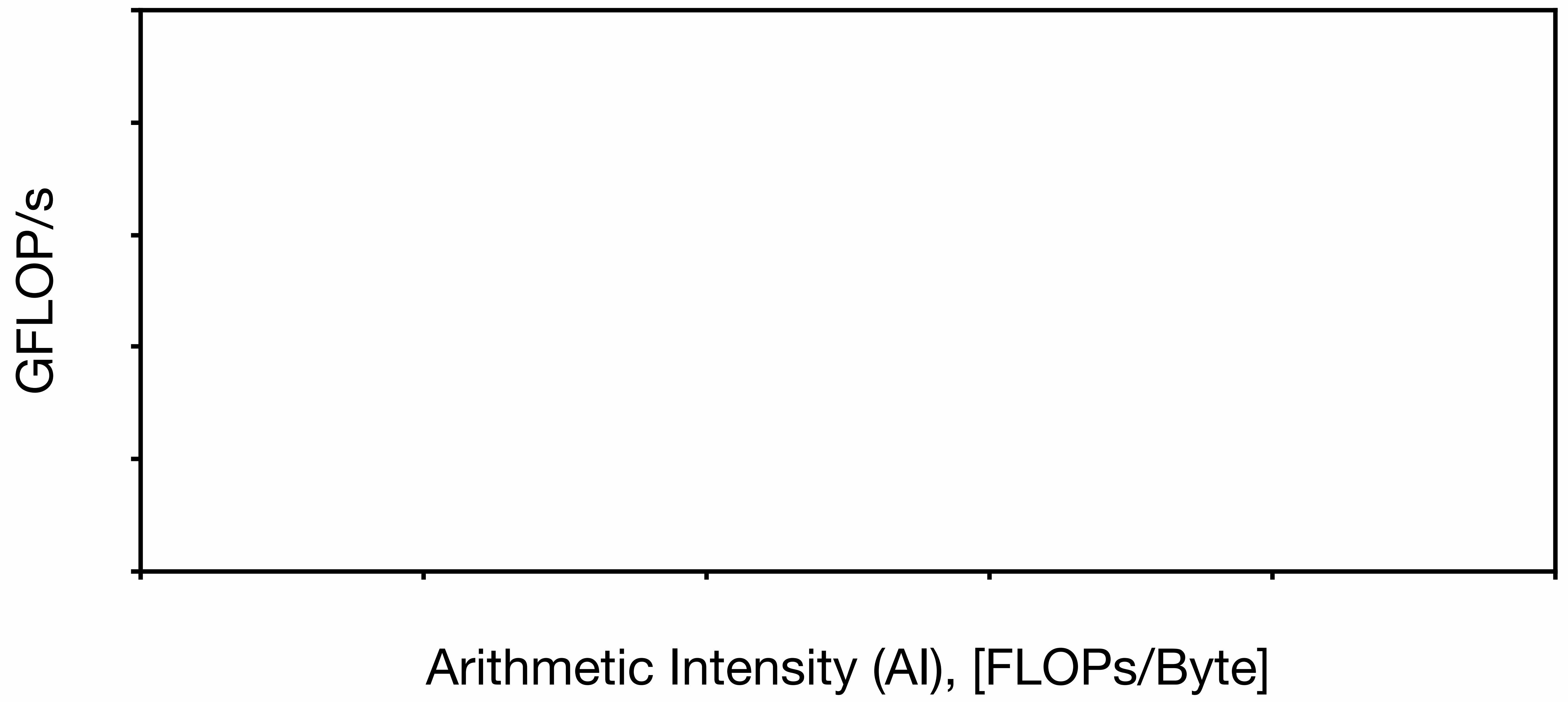




Performance analysis

Roofline model

- A visual model that demonstrates the performance of an application or its kernel with respect to the hardware limitations.
- Demonstrates potential benefits and priority of optimisations.

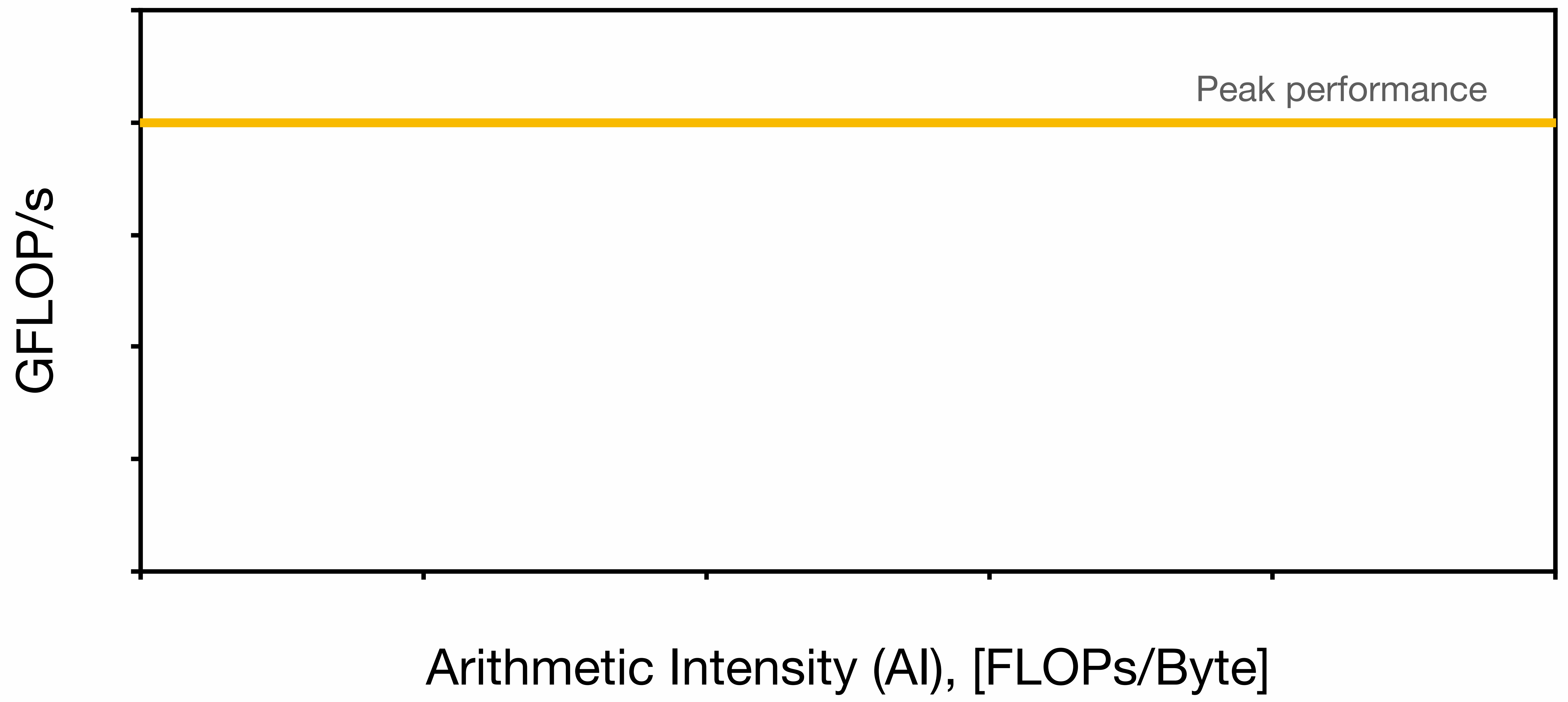


Performance analysis

Roofline model

- A visual model that demonstrates the performance of an application or its kernel with respect to the hardware limitations.

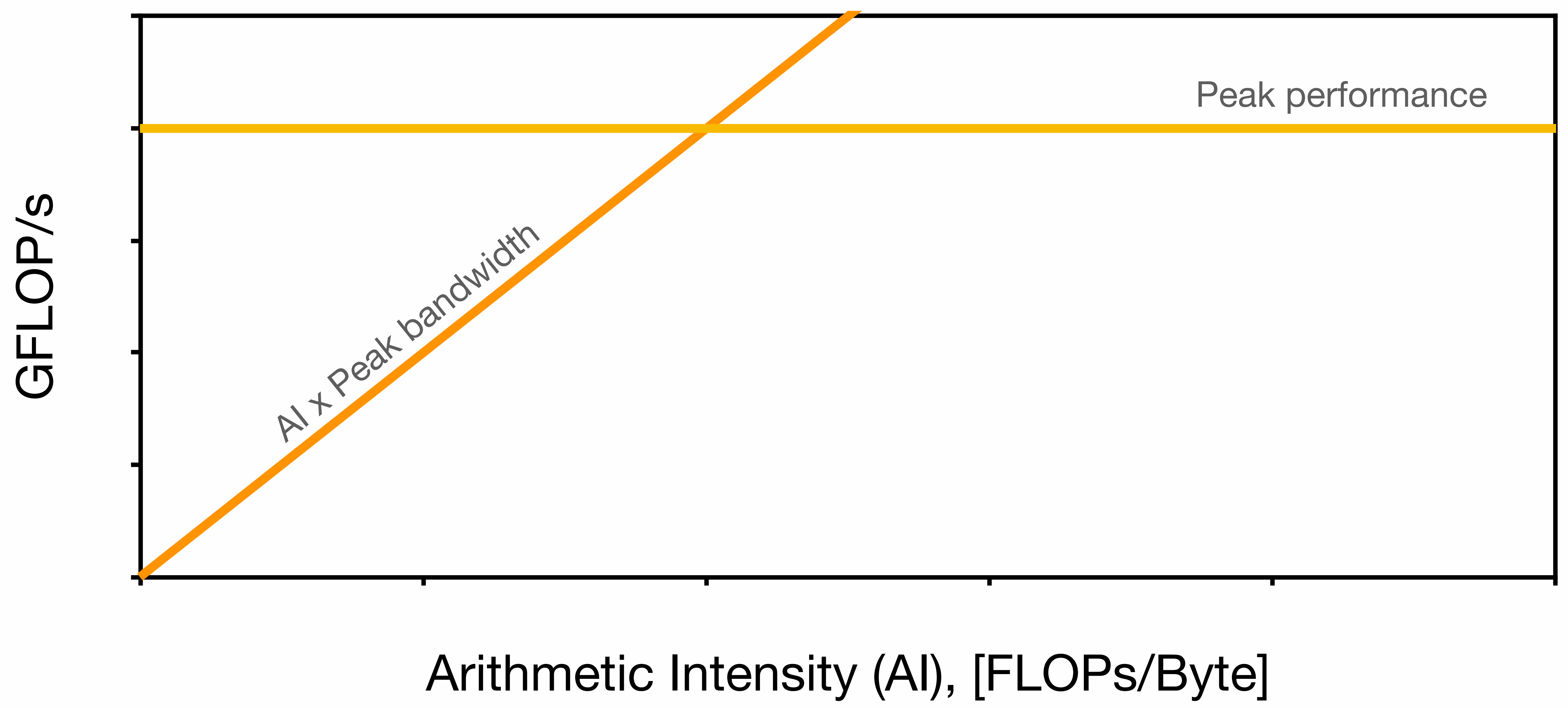
- Demonstrates potential benefits and priority of optimisations.



Performance analysis

Roofline model

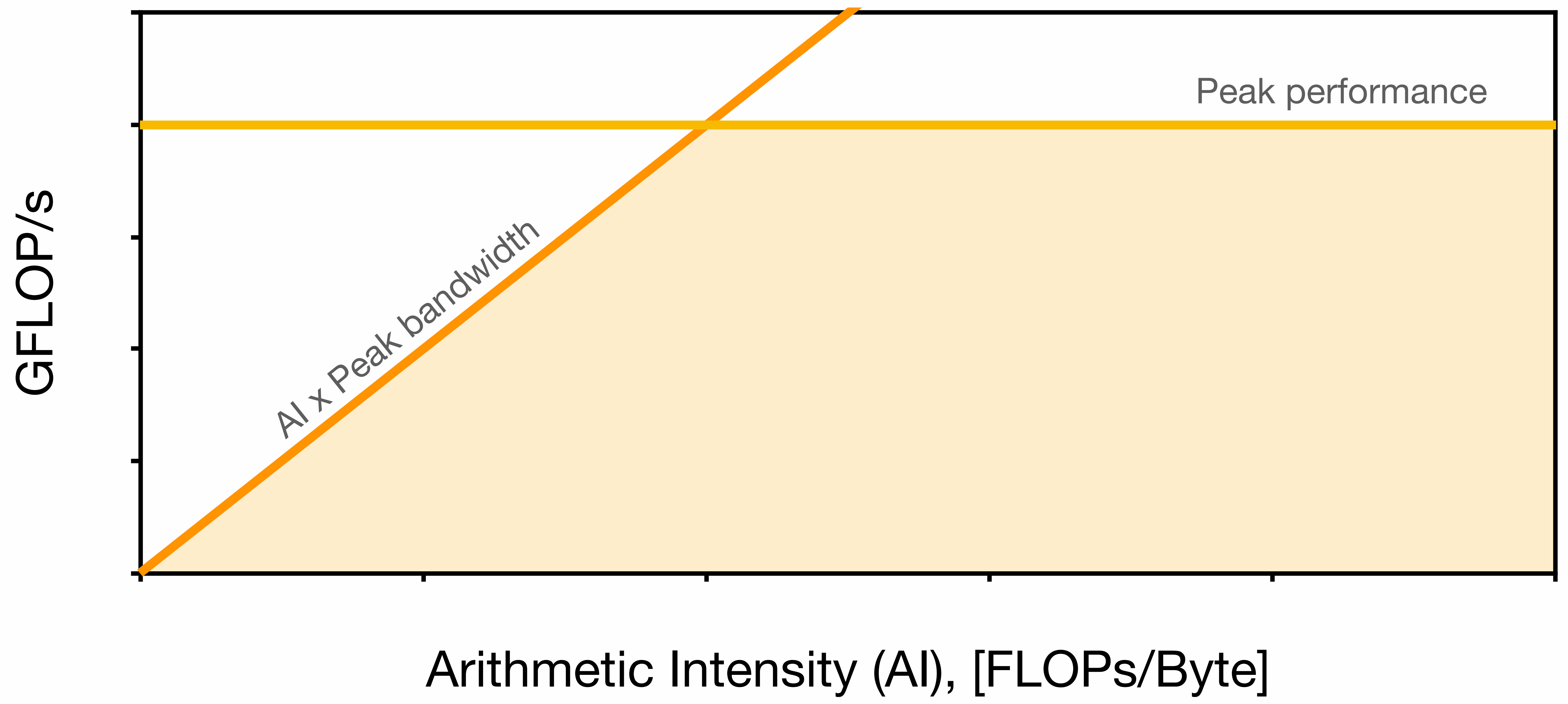
- A visual model that demonstrates the performance of an application or its kernel with respect to the hardware limitations.
- Demonstrates potential benefits and priority of optimisations.



Performance analysis

Roofline model

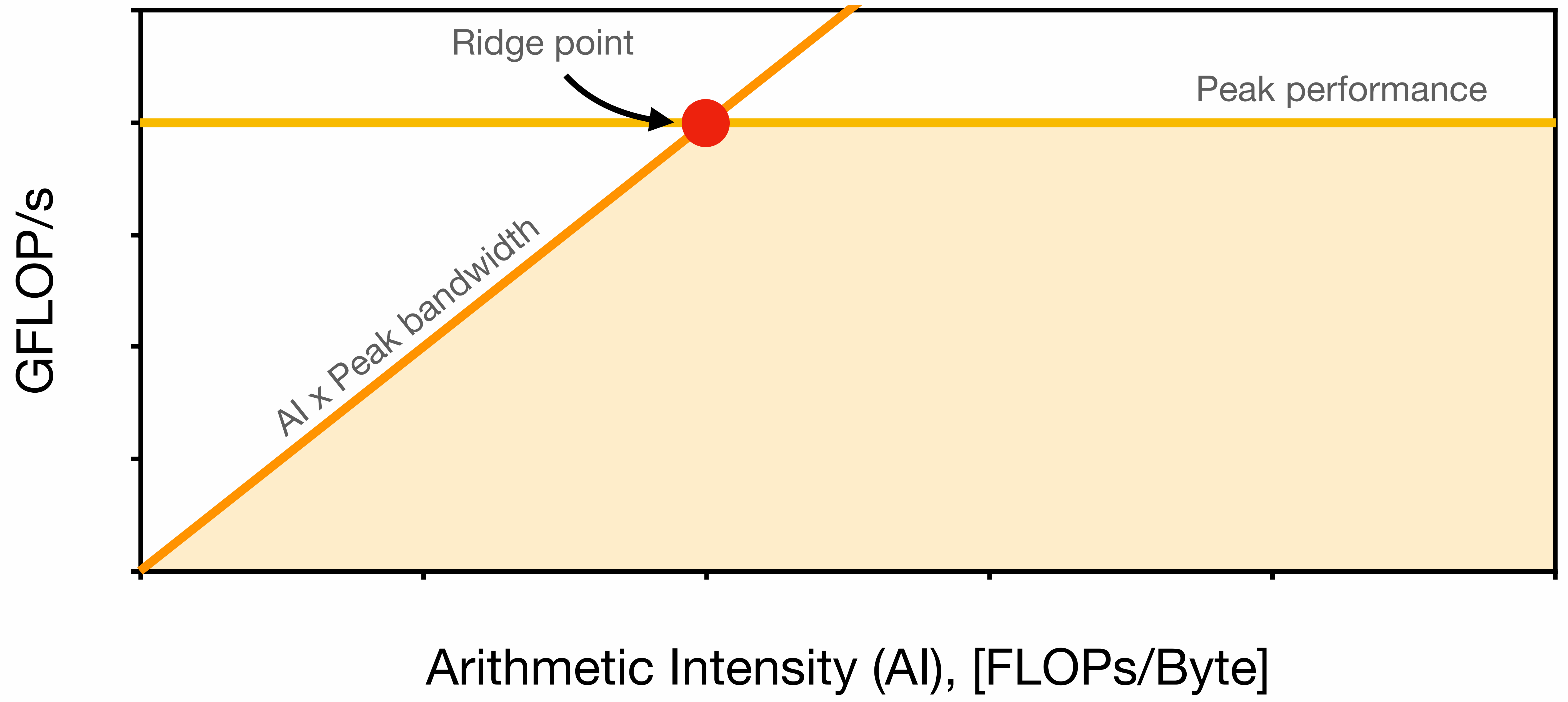
- A visual model that demonstrates the performance of an application or its kernel with respect to the hardware limitations.
- Demonstrates potential benefits and priority of optimisations.



Performance analysis

Roofline model

- A visual model that demonstrates the performance of an application or its kernel with respect to the hardware limitations.
- Demonstrates potential benefits and priority of optimisations.

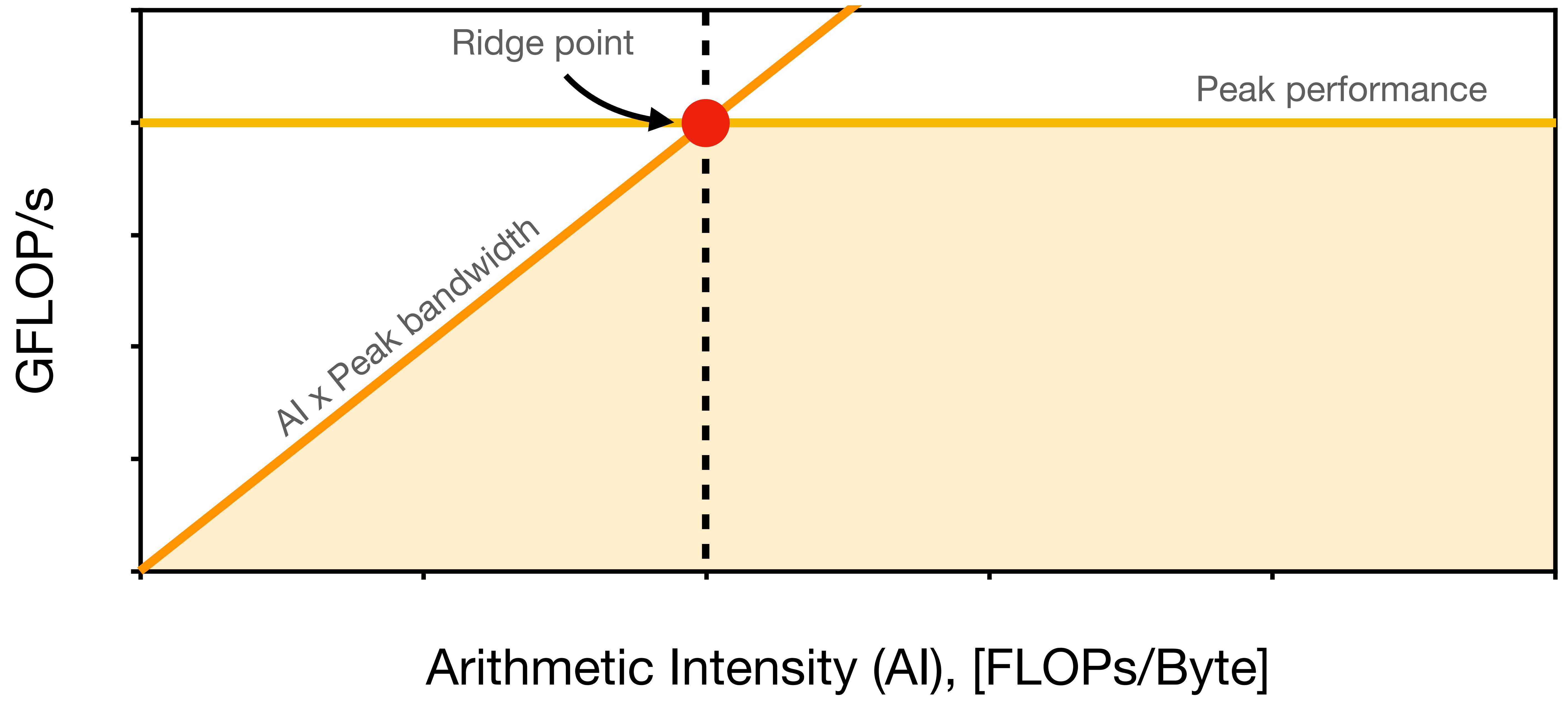


Performance analysis

Roofline model

- A visual model that demonstrates the performance of an application or its kernel with respect to the hardware limitations.

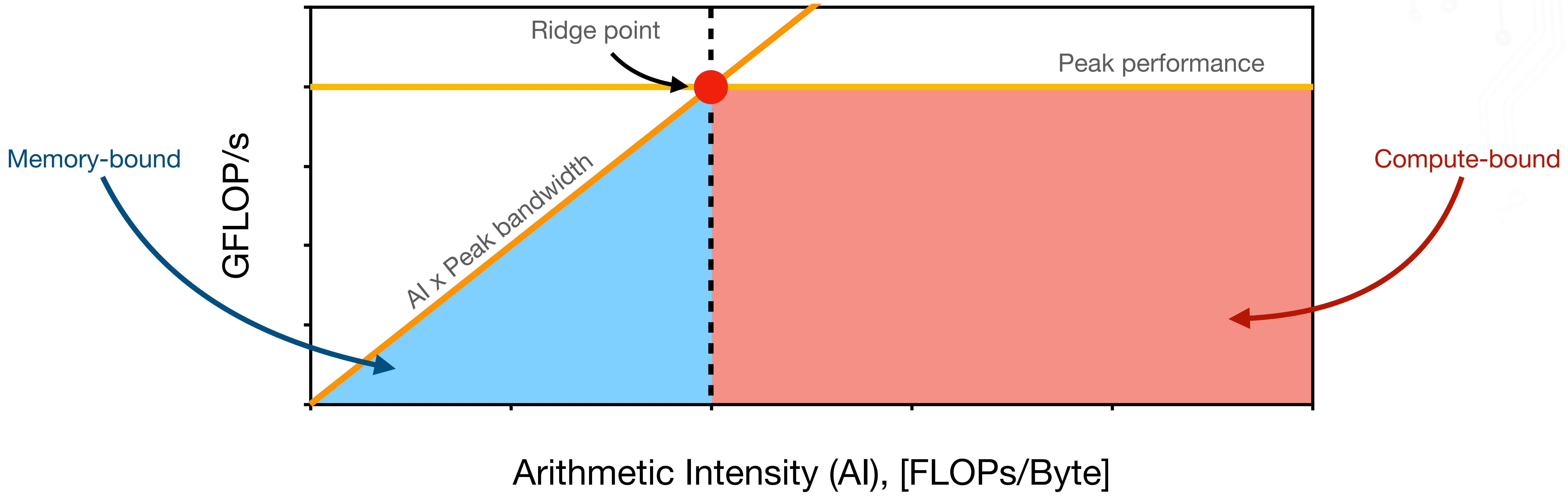
- Demonstrates potential benefits and priority of optimisations.



Performance analysis

Roofline model

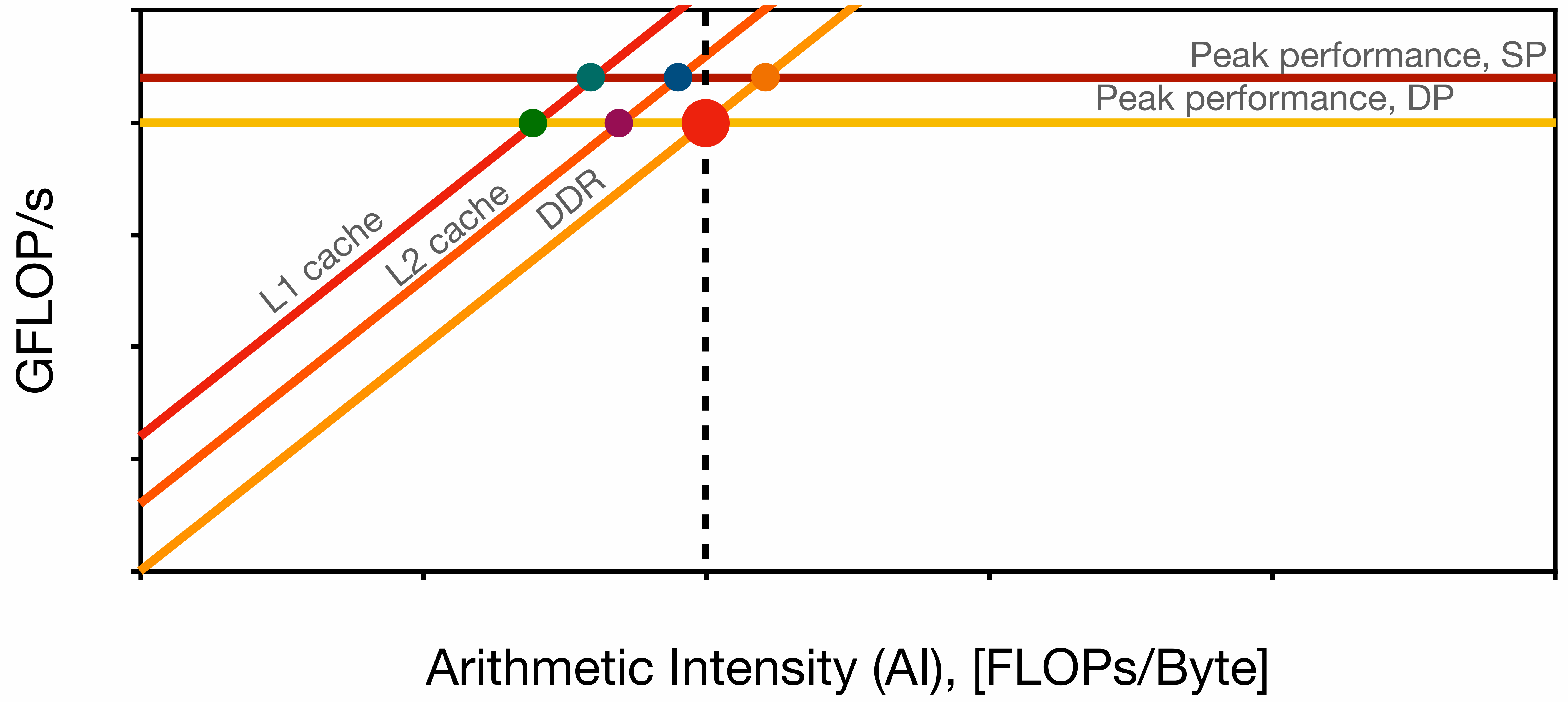
- A visual model that demonstrates the performance of an application or its kernel with respect to the hardware limitations.
- Demonstrates potential benefits and priority of optimisations.



Performance analysis

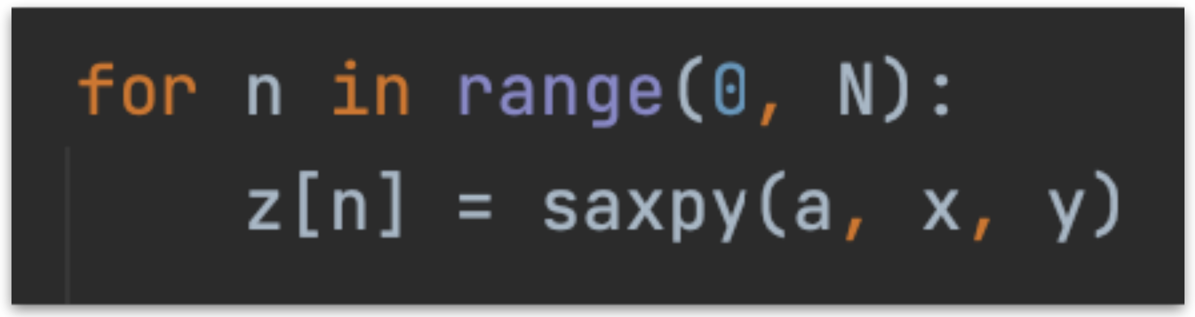
Roofline model

- A visual model that demonstrates the performance of an application or its kernel with respect to the hardware limitations.
- Demonstrates potential benefits and priority of optimisations.

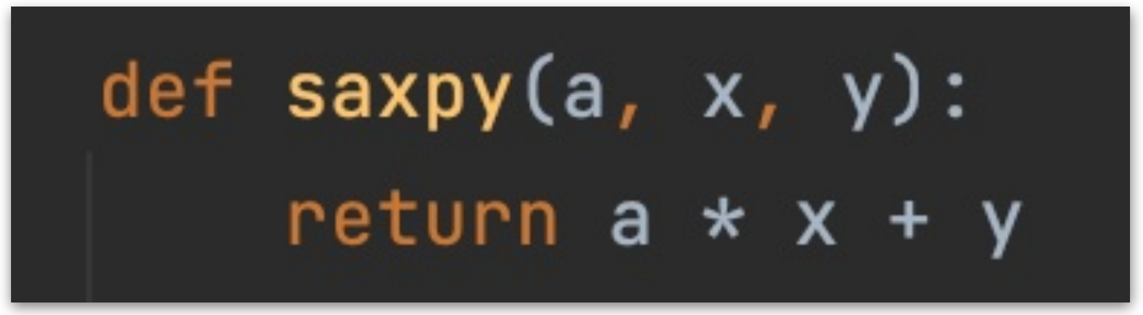


How to estimate FLOPs?

Manually



```
for n in range(0, N):
    z[n] = saxpy(a, x, y)
```



```
def saxpy(a, x, y):
    return a * x + y
```

$\rightarrow$

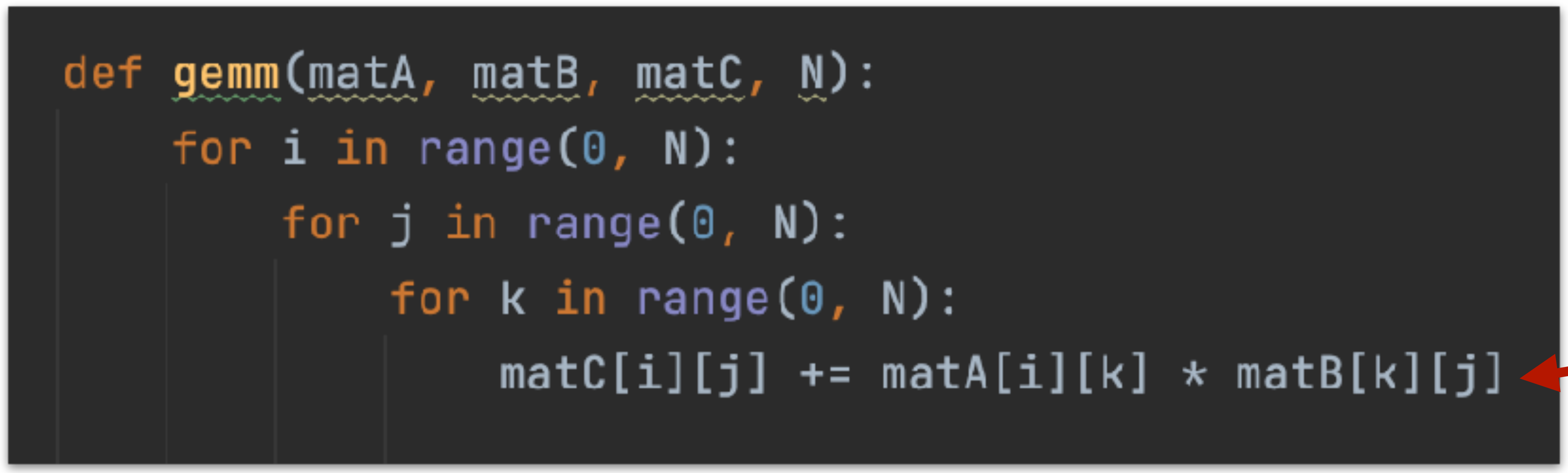
Addition: 1 Flop
Multiplication: 1 Flop

* N = **2N FLOPs**

How to estimate FLOPs?

Using algorithm complexity

- For instance:
  - Generic dense matrix-matrix multiplication: $\approx O(N^3)$ FLOPs
  - FFT algorithm: $\approx O(Nlog_2(N))$ FLOPs
- This approach often does not consider the total number of FLOPs in the algorithm

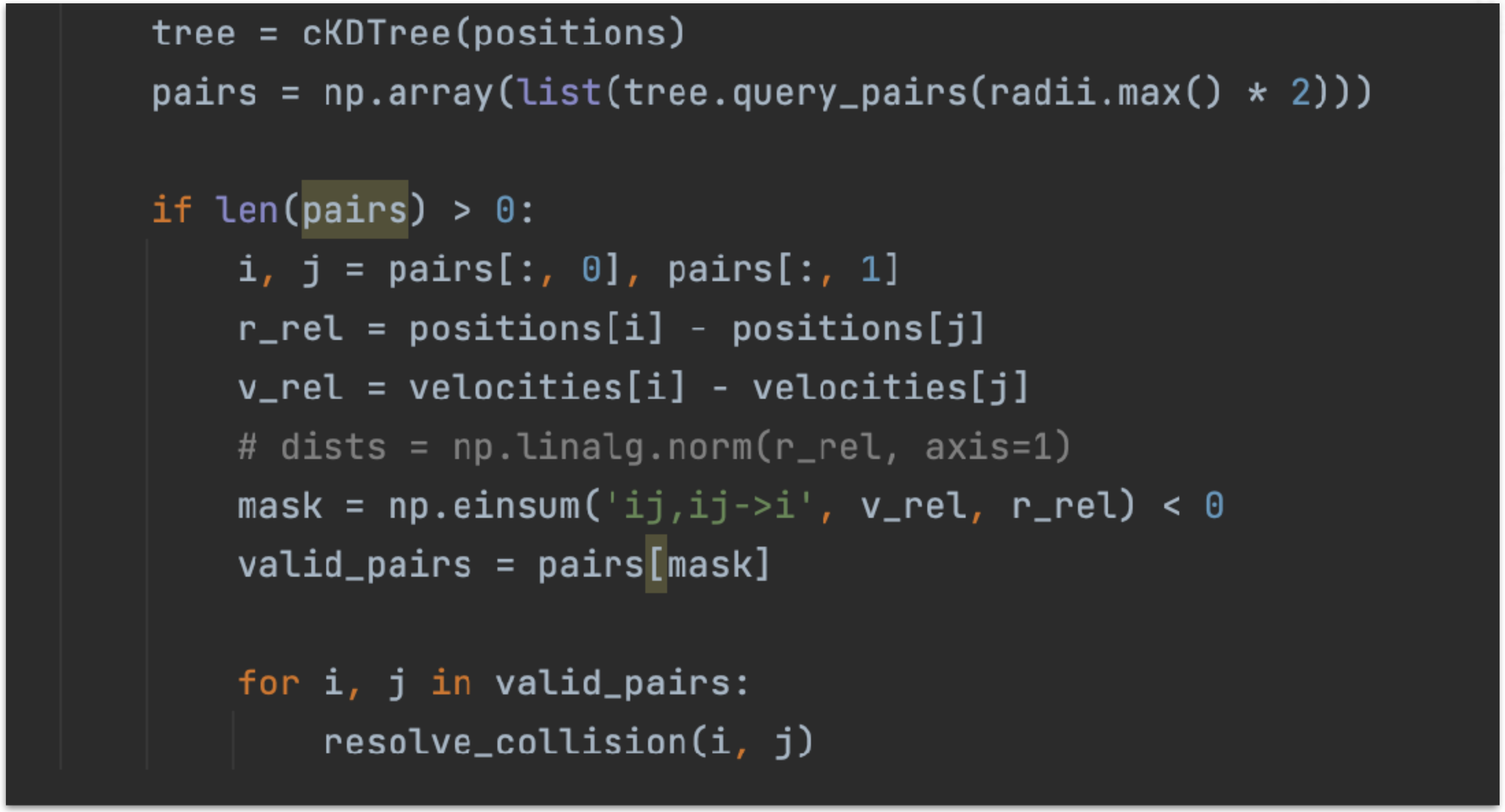


```python
def gemm(matA, matB, matC, N):
    for i in range(0, N):
        for j in range(0, N):
            for k in range(0, N):
                matC[i][j] += matA[i][k] * matB[k][j]
```

$2N^3$ **FLOPs**

How to estimate FLOPs?

What to do with a complex implementation?



```
tree = cKDTree(positions)
pairs = np.array(list(tree.query_pairs(radii.max() * 2)))

if len(pairs) > 0:
    i, j = pairs[:, 0], pairs[:, 1]
    r_rel = positions[i] - positions[j]
    v_rel = velocities[i] - velocities[j]
    # dists = np.linalg.norm(r_rel, axis=1)
    mask = np.einsum('ij,ij->i', v_rel, r_rel) < 0
    valid_pairs = pairs[mask]

    for i, j in valid_pairs:
        resolve_collision(i, j)
```

Tools

Overview: HPC tools

Tool name	Costs	Description
ARM DDT	Non-free	Full featured graphical, parallel debugger
HPCToolkit	Free	Integrated suite of tools for parallel program performance analysis
Intel One API	Free under certain conditions	Stack of different performance analysis and debuggingtools (MPI/OpenMP/SIMD)
Valgrind	Free	Memory errors debugging tool
TotalView	Non-free	Full featured graphical, parallel debugger
Vampir	Non-free	Full featured trace visualizer for parallel program OTF trace files
memP	Free	Lightweight memory profiling tool
mpiP	Free	Lightweight MPI profiling tool
MUST	Free	MPI runtime error detection tool
PAPI	Free	A standardized and portable API for accessing performance counter hardware
likwid	Free	A **tool** to measure hardware performance counters
TAU	Free	Full featured parallel program performance analyses toolkit
Extrae	Free	MPI/OpenMP profiler
Scalasca	Free	performance analysis tool for MPI+OpenMP
Darshan	Free	IO profiler
nvprof	Free	Thread profiler (inc. GPU) from NVIDIA
gdb	Free	Standard GNU debugger
ARM MAP	Non-free	performance analysis tool for MPI+OpenMP
uProf	Free	performance analysis tool for MPI+OpenMP
gprof	Free	Standard unix/linux profiling utility

Different support for:

- Hardware
- Parallelisation strategies
- Compilers
- Interface

Other important aspects:

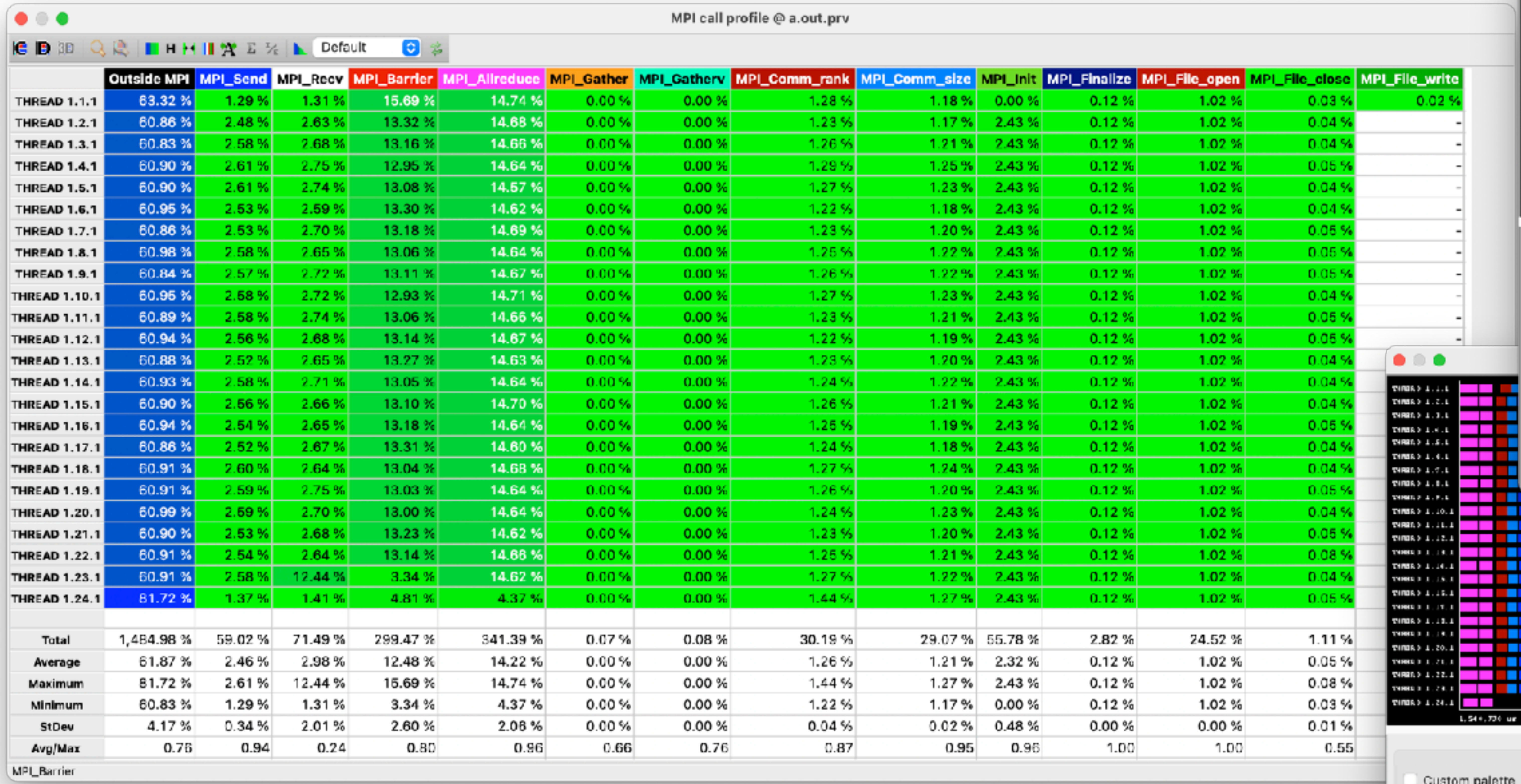
- Learning curve
- Completeness of the reports
- Costs and licenses
- Community support
- Documentation

Good overview:

https://hpc.llnl.gov/software/development-environment-software

Tools

Overview: Extrae + Paraver

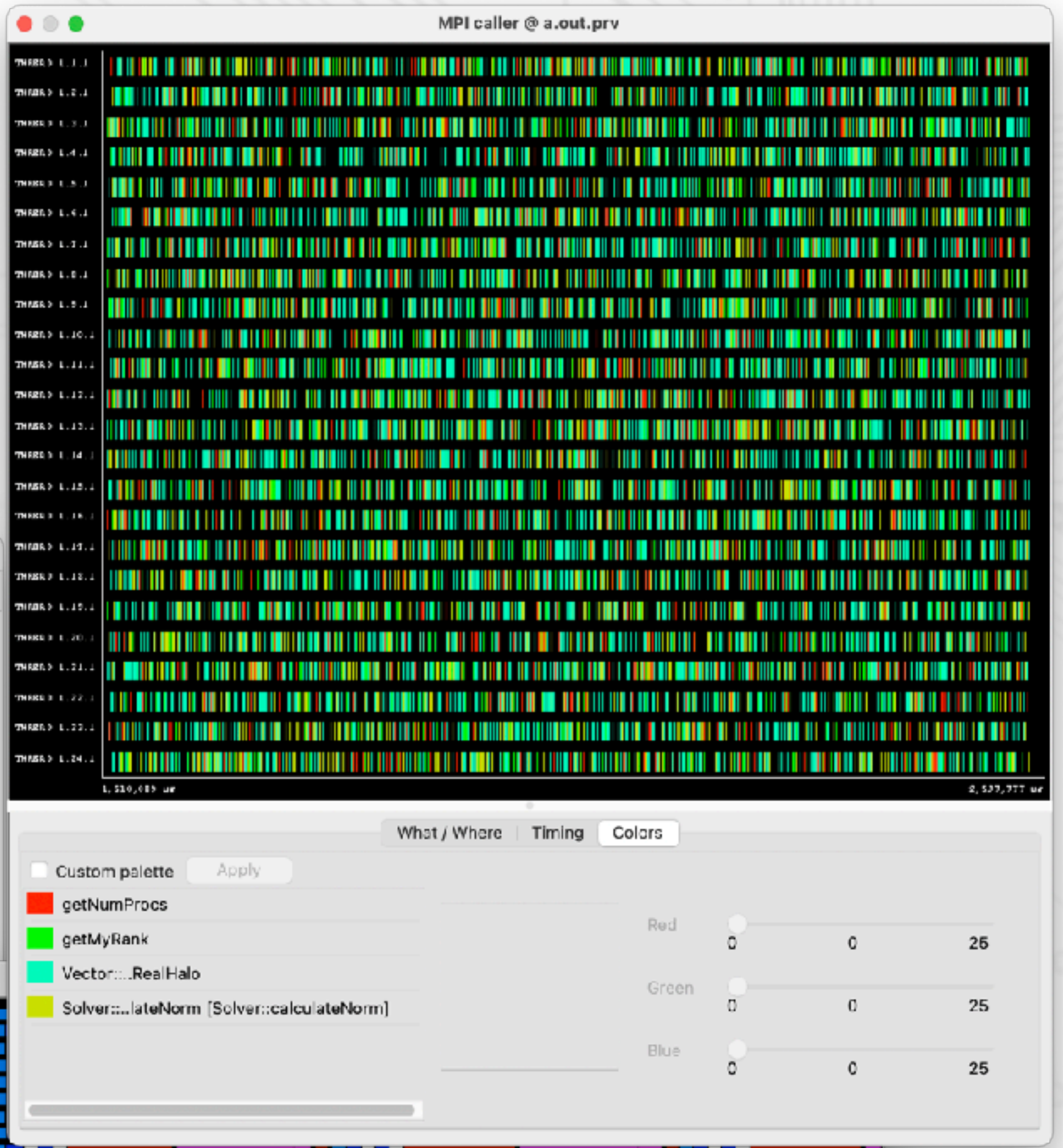


MPI call profile @ a.out.prv

Default

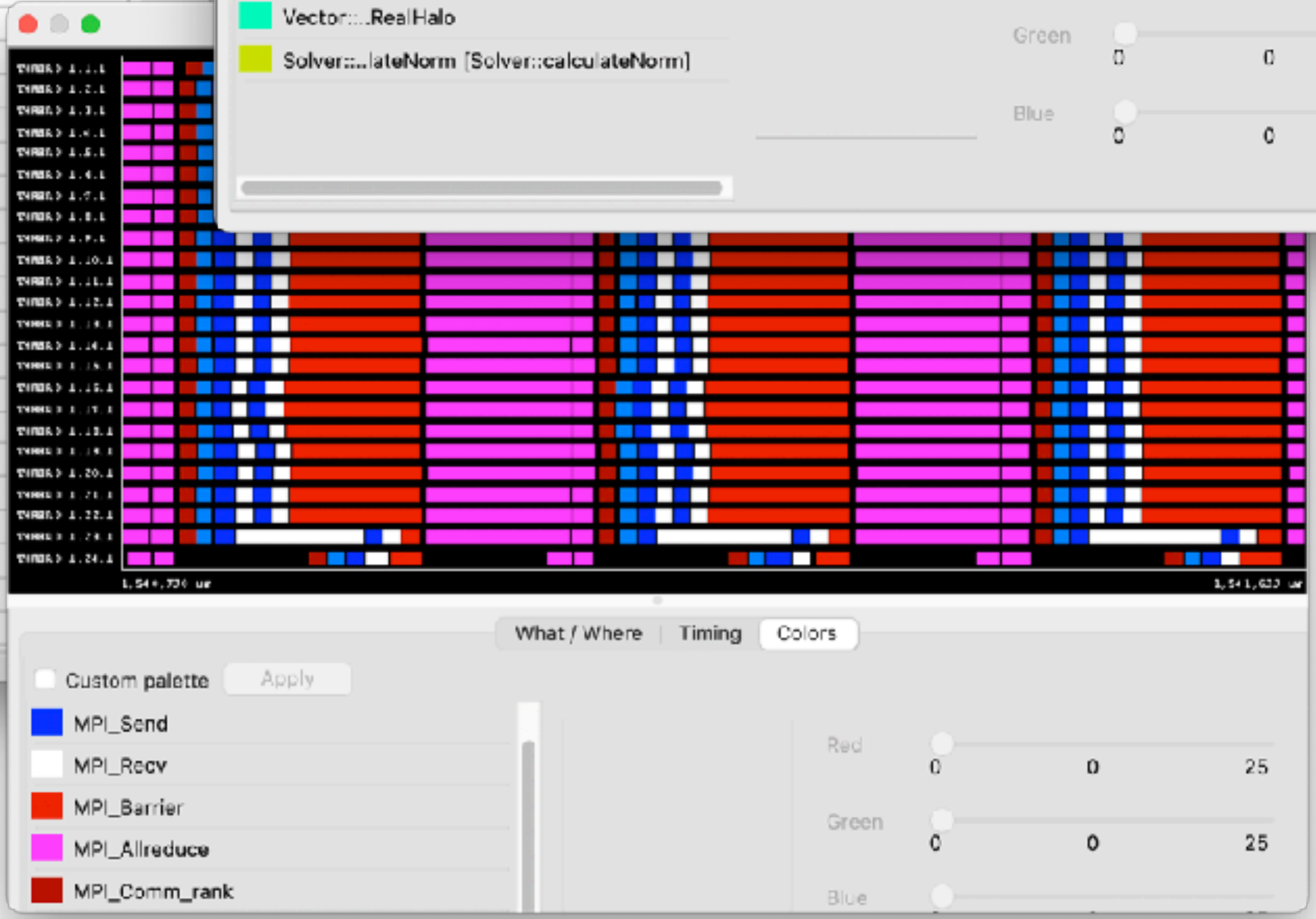
	Outside MPI	MPI_Send	MPI_Recv	MPI_Barrier	MPI_Allreduce	MPI_Gather	MPI_Gatherv	MPI_Comm_rank	MPI_Comm_size	MPI_Init	MPI_Finalize	MPI_File_open	MPI_File_close	MPI_File_write
THREAD 1.1.1	63.32 %	1.29 %	1.31 %	15.69 %	14.74 %	0.00 %	0.00 %	1.28 %	1.18 %	0.00 %	0.12 %	1.02 %	0.03 %	0.02 %
THREAD 1.2.1	60.86 %	2.48 %	2.63 %	13.32 %	14.68 %	0.00 %	0.00 %	1.23 %	1.17 %	2.43 %	0.12 %	1.02 %	0.04 %	-
THREAD 1.3.1	60.83 %	2.58 %	2.68 %	13.16 %	14.66 %	0.00 %	0.00 %	1.26 %	1.21 %	2.43 %	0.12 %	1.02 %	0.04 %	-
THREAD 1.4.1	60.90 %	2.61 %	2.75 %	12.95 %	14.64 %	0.00 %	0.00 %	1.29 %	1.25 %	2.43 %	0.12 %	1.02 %	0.05 %	-
THREAD 1.5.1	60.90 %	2.61 %	2.74 %	13.08 %	14.57 %	0.00 %	0.00 %	1.27 %	1.23 %	2.43 %	0.12 %	1.02 %	0.04 %	-
THREAD 1.6.1	60.95 %	2.53 %	2.59 %	13.30 %	14.62 %	0.00 %	0.00 %	1.22 %	1.18 %	2.43 %	0.12 %	1.02 %	0.04 %	-
THREAD 1.7.1	60.86 %	2.53 %	2.70 %	13.18 %	14.69 %	0.00 %	0.00 %	1.23 %	1.20 %	2.43 %	0.12 %	1.02 %	0.05 %	-
THREAD 1.8.1	60.98 %	2.58 %	2.65 %	13.06 %	14.64 %	0.00 %	0.00 %	1.25 %	1.22 %	2.43 %	0.12 %	1.02 %	0.05 %	-
THREAD 1.9.1	60.84 %	2.57 %	2.72 %	13.11 %	14.67 %	0.00 %	0.00 %	1.26 %	1.22 %	2.43 %	0.12 %	1.02 %	0.05 %	-
THREAD 1.10.1	60.95 %	2.58 %	2.72 %	12.93 %	14.71 %	0.00 %	0.00 %	1.27 %	1.23 %	2.43 %	0.12 %	1.02 %	0.04 %	-
THREAD 1.11.1	60.89 %	2.58 %	2.74 %	13.06 %	14.66 %	0.00 %	0.00 %	1.23 %	1.21 %	2.43 %	0.12 %	1.02 %	0.05 %	-
THREAD 1.12.1	60.94 %	2.56 %	2.68 %	13.14 %	14.67 %	0.00 %	0.00 %	1.22 %	1.19 %	2.43 %	0.12 %	1.02 %	0.05 %	-
THREAD 1.13.1	60.88 %	2.52 %	2.65 %	13.27 %	14.63 %	0.00 %	0.00 %	1.23 %	1.20 %	2.43 %	0.12 %	1.02 %	0.04 %	
THREAD 1.14.1	60.93 %	2.58 %	2.71 %	13.05 %	14.64 %	0.00 %	0.00 %	1.24 %	1.22 %	2.43 %	0.12 %	1.02 %	0.04 %	
THREAD 1.15.1	60.90 %	2.56 %	2.66 %	13.10 %	14.70 %	0.00 %	0.00 %	1.26 %	1.21 %	2.43 %	0.12 %	1.02 %	0.04 %	
THREAD 1.16.1	60.94 %	2.54 %	2.65 %	13.18 %	14.64 %	0.00 %	0.00 %	1.25 %	1.19 %	2.43 %	0.12 %	1.02 %	0.05 %	
THREAD 1.17.1	60.86 %	2.52 %	2.67 %	13.31 %	14.60 %	0.00 %	0.00 %	1.24 %	1.18 %	2.43 %	0.12 %	1.02 %	0.04 %	
THREAD 1.18.1	60.91 %	2.60 %	2.64 %	13.04 %	14.68 %	0.00 %	0.00 %	1.27 %	1.24 %	2.43 %	0.12 %	1.02 %	0.04 %	
THREAD 1.19.1	60.91 %	2.59 %	2.75 %	13.03 %	14.64 %	0.00 %	0.00 %	1.26 %	1.20 %	2.43 %	0.12 %	1.02 %	0.05 %	
THREAD 1.20.1	60.99 %	2.59 %	2.70 %	13.00 %	14.64 %	0.00 %	0.00 %	1.24 %	1.23 %	2.43 %	0.12 %	1.02 %	0.04 %	
THREAD 1.21.1	60.90 %	2.53 %	2.68 %	13.23 %	14.62 %	0.00 %	0.00 %	1.23 %	1.20 %	2.43 %	0.12 %	1.02 %	0.05 %	
THREAD 1.22.1	60.91 %	2.54 %	2.64 %	13.14 %	14.66 %	0.00 %	0.00 %	1.25 %	1.21 %	2.43 %	0.12 %	1.02 %	0.08 %	
THREAD 1.23.1	60.91 %	2.58 %	12.44 %	3.34 %	14.62 %	0.00 %	0.00 %	1.27 %	1.22 %	2.43 %	0.12 %	1.02 %	0.04 %	
THREAD 1.24.1	61.72 %	1.37 %	1.41 %	4.81 %	4.37 %	0.00 %	0.00 %	1.44 %	1.27 %	2.43 %	0.12 %	1.02 %	0.05 %	
Total	1,464.98 %	59.02 %	71.49 %	299.47 %	341.39 %	0.07 %	0.08 %	30.19 %	29.07 %	55.78 %	2.82 %	24.52 %	1.11 %	
Average	61.87 %	2.46 %	2.98 %	12.48 %	14.22 %	0.00 %	0.00 %	1.26 %	1.21 %	2.32 %	0.12 %	1.02 %	0.05 %	
Maximum	61.72 %	2.61 %	12.44 %	15.69 %	14.74 %	0.00 %	0.00 %	1.44 %	1.27 %	2.43 %	0.12 %	1.02 %	0.08 %	
Minimum	60.83 %	1.29 %	1.31 %	3.34 %	4.37 %	0.00 %	0.00 %	1.22 %	1.17 %	0.00 %	0.12 %	1.02 %	0.03 %	
StDev	4.17 %	0.34 %	2.01 %	2.60 %	2.06 %	0.00 %	0.00 %	0.04 %	0.02 %	0.48 %	0.00 %	0.00 %	0.01 %	
Avg/Max	0.76	0.94	0.24	0.80	0.96	0.66	0.76	0.87	0.95	0.96	1.00	1.00	0.55	

MPI_Barrier



MPI caller @ a.out.prv

- 🟥 getNumProcs
- 🟩 getMyRank
- 🟦 Vector::...RealHalo
- 🟨 Solver::...lateNorm [Solver::calculateNorm]



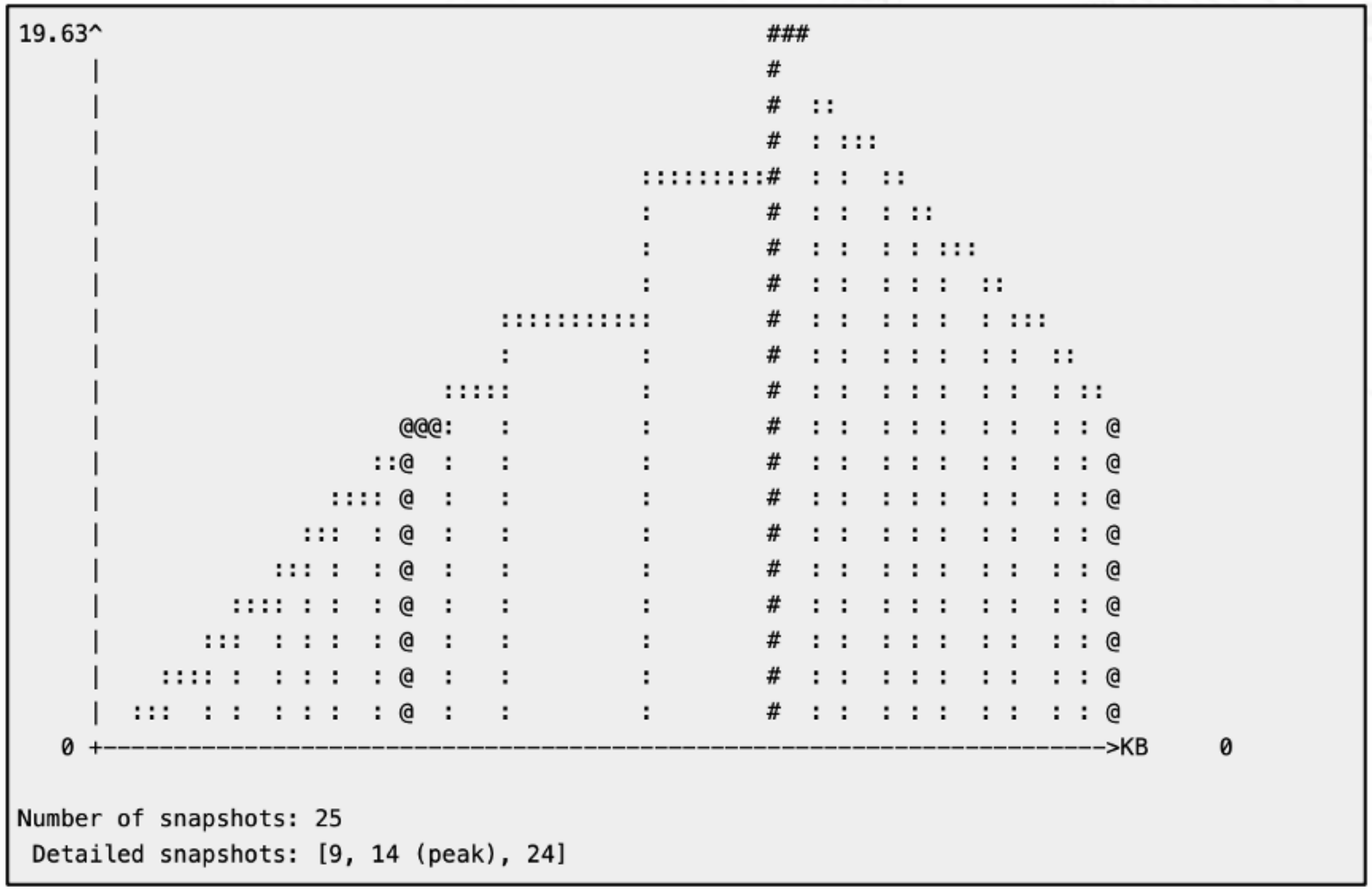
- 🟦 MPI_Send
- ⬜ MPI_Recv
- 🟥 MPI_Barrier
- 🟪 MPI_Allreduce
- 🟥 MPI_Comm_rank

Performance analysis

Memory

- Memory footprint – how much RAM does an application use
- May affect cache misses and, therefore, performance
- May dictate the scale at which application should be executed

```
$ valgrind --tool=massif <prog>
$ ms_print massif.out.XXX
```



```
19.63^                                               ###
     |                                               #
     |                                               #  ::
     |                                               #  : :::
     |                                      :::::::::#  : :  ::
     |                                      :        #  : :  : ::
     |                                      :        #  : :  : : :::
     |                                      :        #  : :  : : :  ::
     |                            :::::::::::        #  : :  : : :  : :::
     |                            :         :        #  : :  : : :  : :  ::
     |                        :::::         :        #  : :  : : :  : :  : ::
     |                     @@@:   :         :        #  : :  : : :  : :  : : @
     |                   ::@  :   :         :        #  : :  : : :  : :  : : @
     |                :::: @  :   :         :        #  : :  : : :  : :  : : @
     |              :::  : @  :   :         :        #  : :  : : :  : :  : : @
     |            ::: :  : @  :   :         :        #  : :  : : :  : :  : : @
     |         :::: : :  : @  :   :         :        #  : :  : : :  : :  : : @
     |       :::  : : :  : @  :   :         :        #  : :  : : :  : :  : : @
     |    :::: :  : : :  : @  :   :         :        #  : :  : : :  : :  : : @
     |  :::  : :  : : :  : @  :   :         :        #  : :  : : :  : :  : : @
   0 +----------------------------------------------------------------------->KB     0

Number of snapshots: 25
 Detailed snapshots: [9, 14 (peak), 24]
```

Performance analysis

Python tools

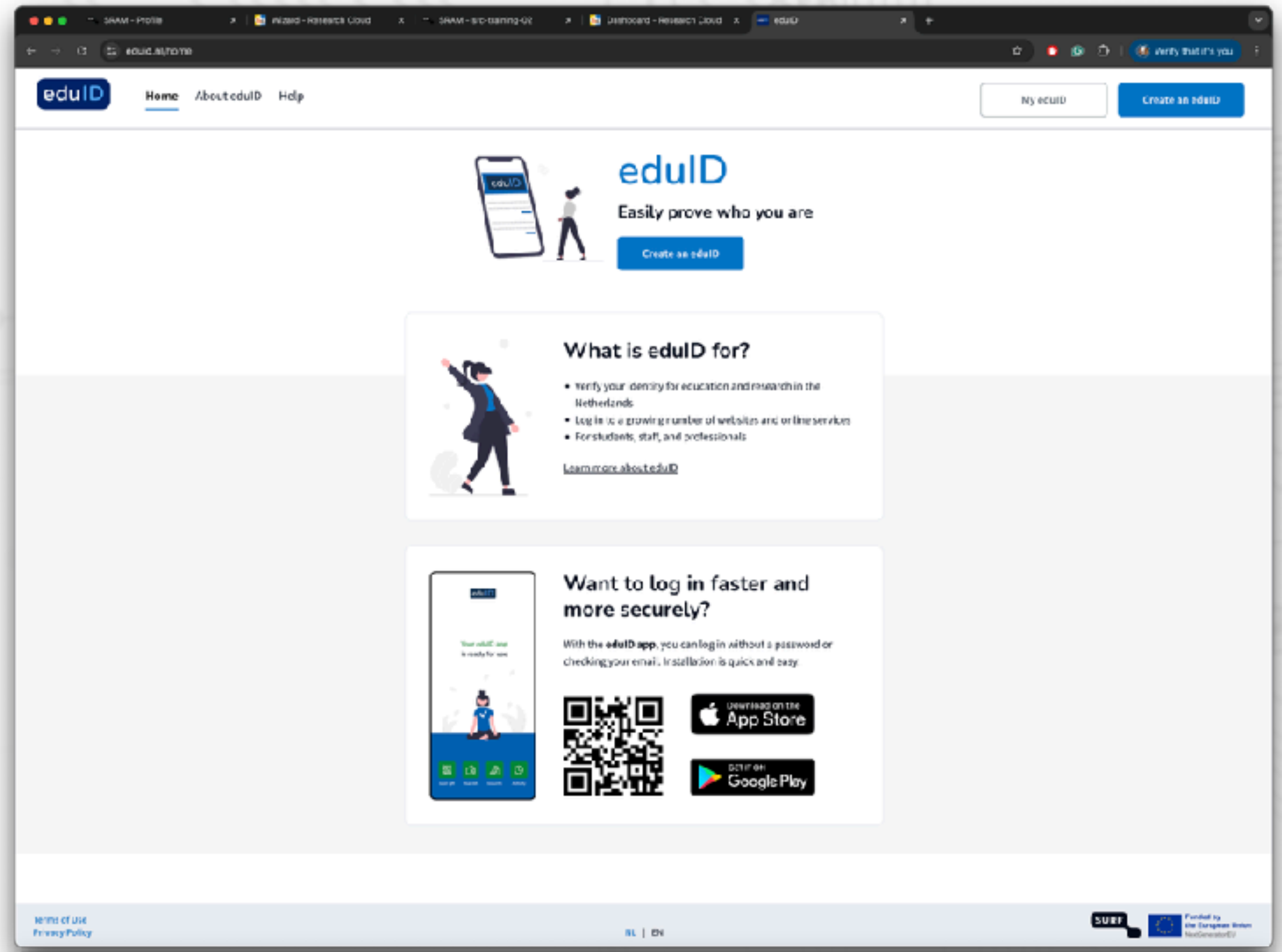
Profiler	Notes
cProfile	Deterministic CPU profiler, a bit slow
pyinstrument	Statistical profiler, report the call stack and elapsed times
yappi	Deterministic profiler, allows to profile multi-threaded applications
memory_profiler	Monitors memory consumption of a process
line_profiler	Profile the time individual lines of code take to execute

Connecting to SRC

SURF Research Cloud

Creating eduID: eduid.nl

- If not affiliated with Dutch institutions, you need to set up **eduID**
  - Go to eduid.nl
  - Register a new account with your email
  - Verify registration
  - Download eduID App
  - Verify App



eduID

Easily prove who you are

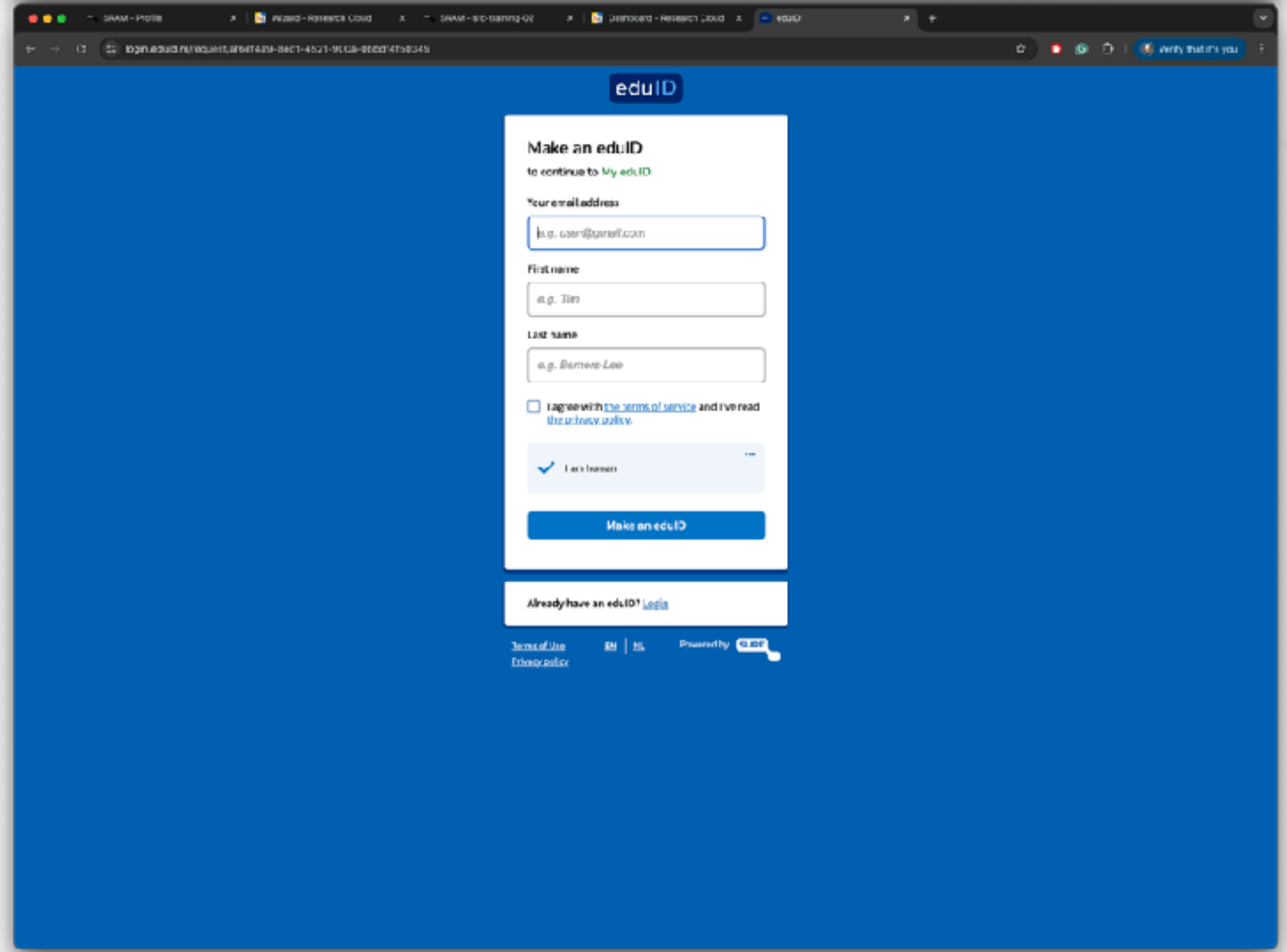
What is eduID for?

- Verify your identity for education and research in the Netherlands
- Log in to a growing number of websites and online services
- For students, staff, and professionals

Learn more about eduID

Want to log in faster and more securely?

With the **eduID app**, you can log in without a password or checking your email. Installation is quick and easy.



Make an eduID

to continue to My eduID

Your email address: e.g. user@gmail.com

First name: e.g. Tim

Last name: e.g. Berners-Lee

☐ I agree with the terms of service and I've read the privacy policy.

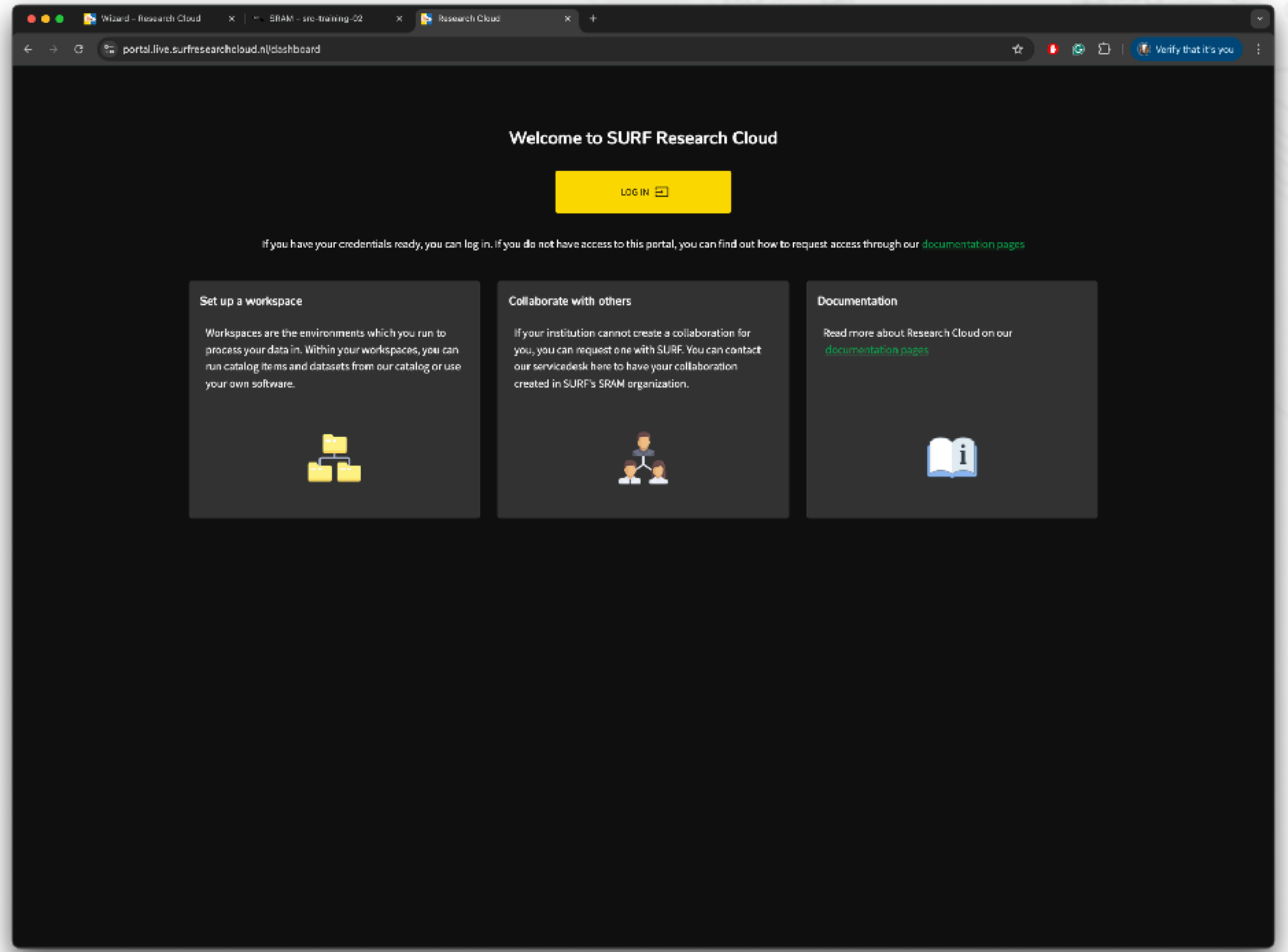
✓ I am human

Already have an eduID? Login

SURF Research Cloud

Connecting to SRC

- Check your email from **SURF Research Access Management**
- Join the collaboration (click on the **button** or copy the **link**)
- Log in using you **eduID** (or other Dutch institution) credentials



Welcome to SURF Research Cloud

If you have your credentials ready, you can log in. If you do not have access to this portal, you can find out how to request access through our documentation pages

Set up a workspace

Workspaces are the environments which you run to process your data in. Within your workspaces, you can run catalog items and datasets from our catalog or use your own software.

Collaborate with others

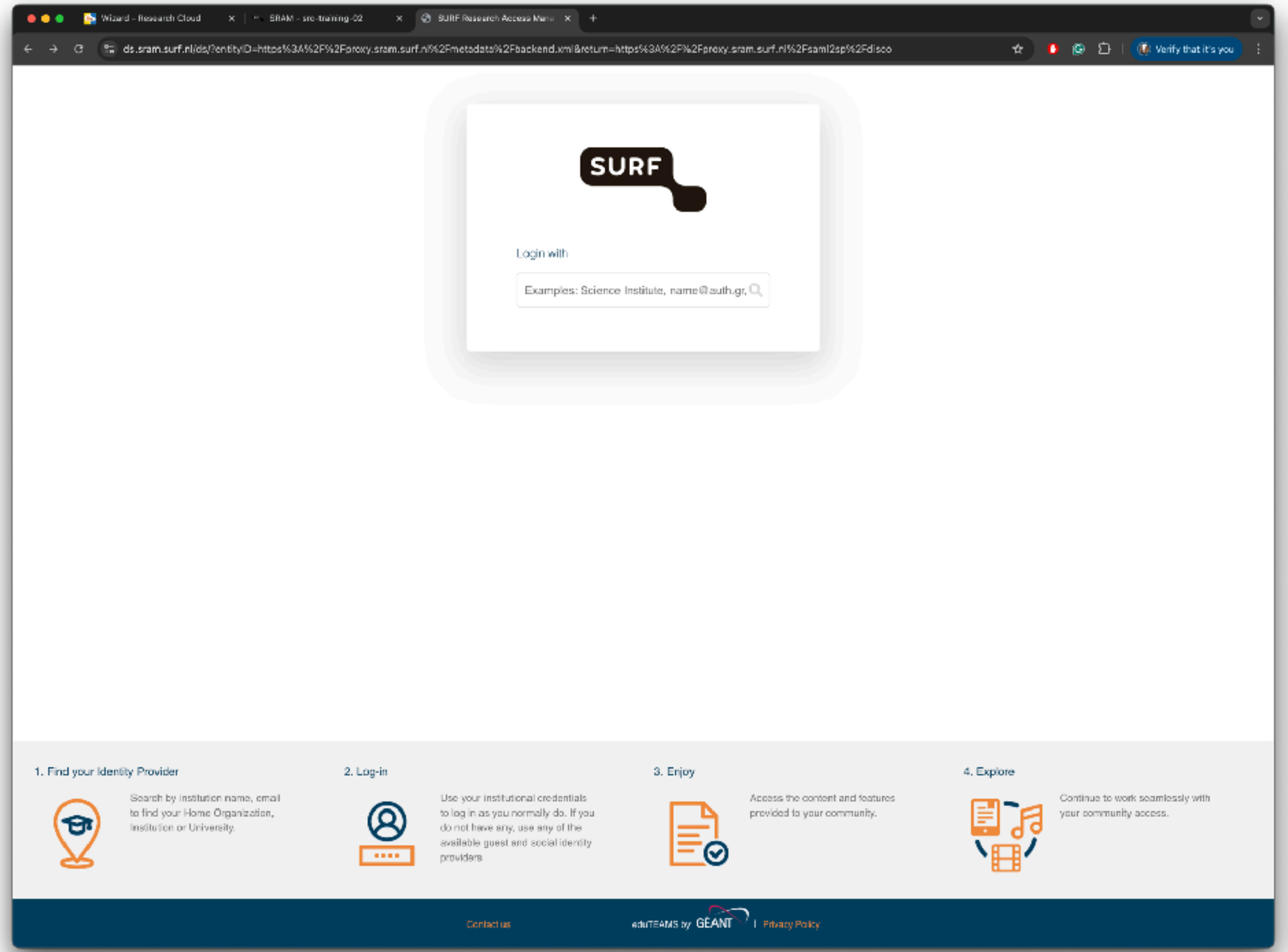
If your institution cannot create a collaboration for you, you can request one with SURF. You can contact our servicedesk here to have your collaboration created in SURF's SRAM organization.

Documentation

Read more about Research Cloud on our documentation pages

SURF Research Cloud

Connecting to: sram.surf.nl



ds.sram.surf.nl/ds/?entityID=https%3A%2F%2Fproxy.sram.surf.nl%2Fmetadata%2Fbackend.xml&return=https%3A%2F%2Fproxy.sram.surf.nl%2Fsaml2sp%2Fdisco

Login with

Examples: Science Institute, name@auth.gr,

1. Find your Identity Provider

Search by institution name, email to find your Home Organization, Institution or University.

2. Log-in

Use your institutional credentials to log in as you normally do. If you do not have any, use any of the available guest and social identity providers.

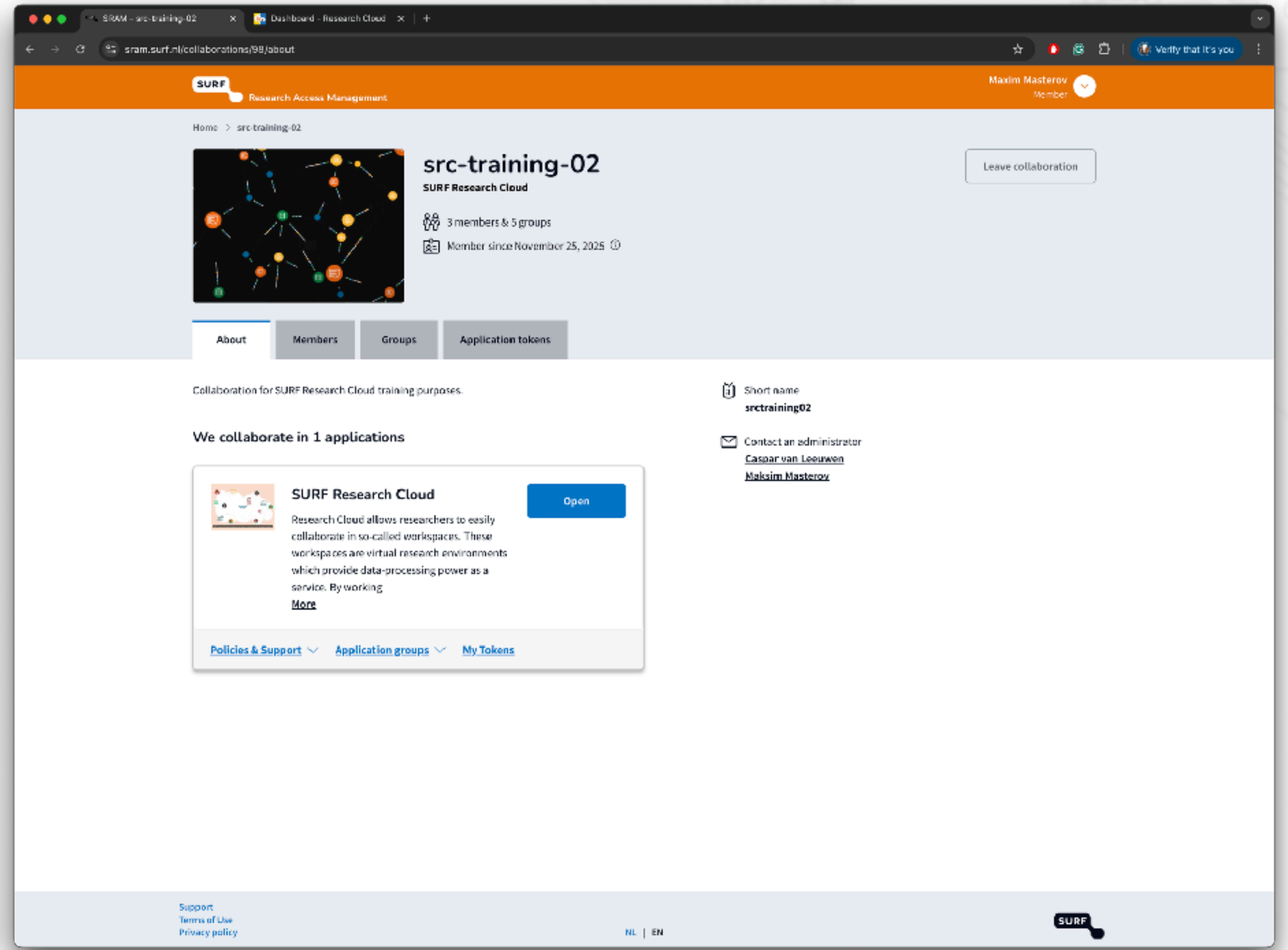
3. Enjoy

Access the content and features provided to your community.

4. Explore

Continue to work seamlessly with your community access.

Contact us | eduTEAMS by GÉANT | Privacy Policy



sram.surf.nl/collaborations/98/about

Home > src-training-02

src-training-02

SURF Research Cloud

3 members & 5 groups

Member since November 25, 2025

Collaboration for SURF Research Cloud training purposes.

Short name
srctraining02

Contact an administrator
Caspar van Leeuwen
Maksim Masterov

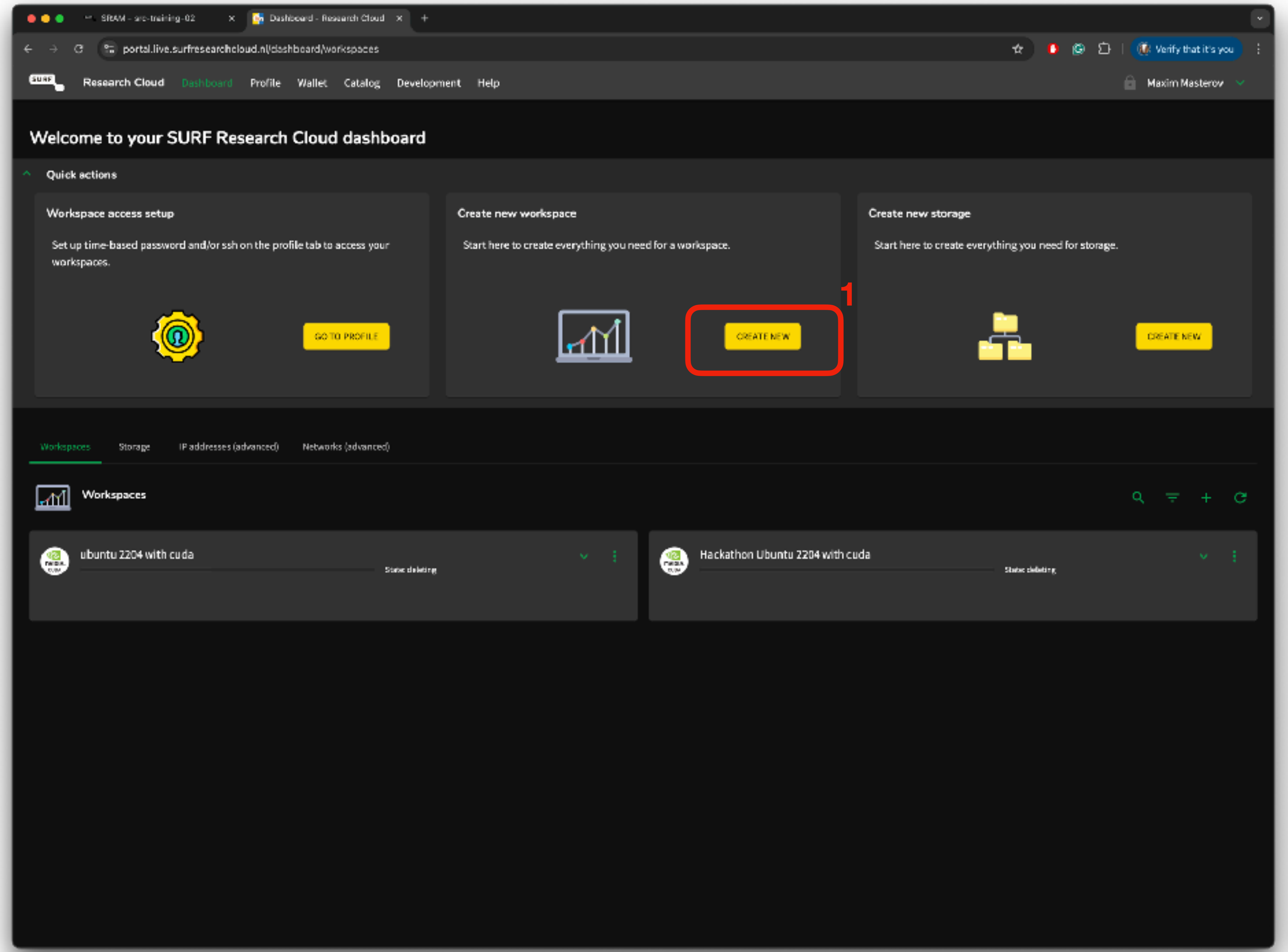
We collaborate in 1 applications

SURF Research Cloud

Research Cloud allows researchers to easily collaborate in so-called workspaces. These workspaces are virtual research environments which provide data-processing power as a service. By working
More

SURF Research Cloud

Setting up VM: portal.live.surfresearchcloud.nl



Welcome to your SURF Research Cloud dashboard

Quick actions

Workspace access setup

Set up time-based password and/or ssh on the profile tab to access your workspaces.

Create new workspace

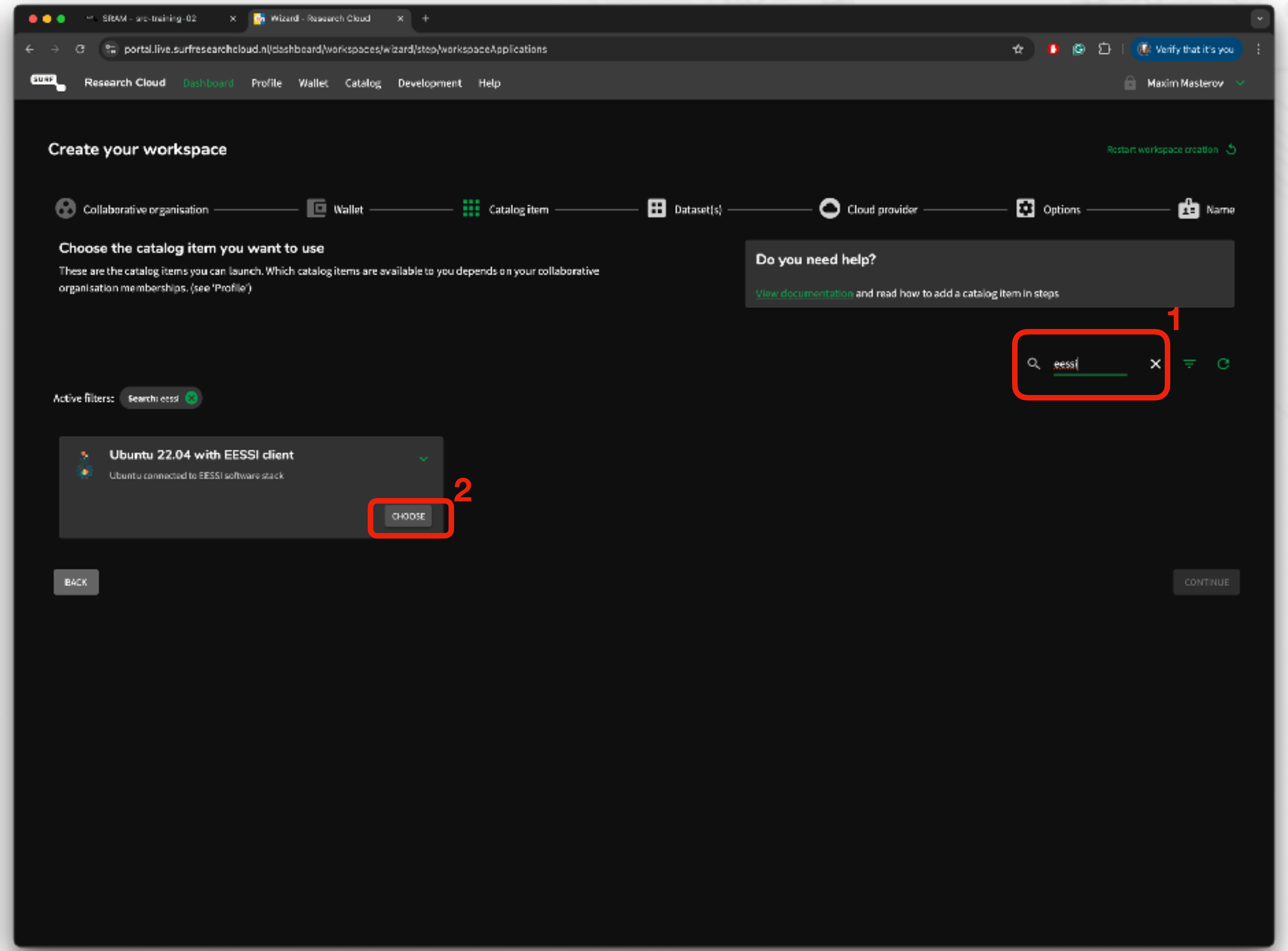
Start here to create everything you need for a workspace.

Create new storage

Start here to create everything you need for storage.

ubuntu 2204 with cuda — Status: deleting

Hackathon Ubuntu 2204 with cuda — Status: deleting



Create your workspace

Choose the catalog item you want to use

These are the catalog items you can launch. Which catalog items are available to you depends on your collaborative organisation memberships. (see 'Profile')

Do you need help?

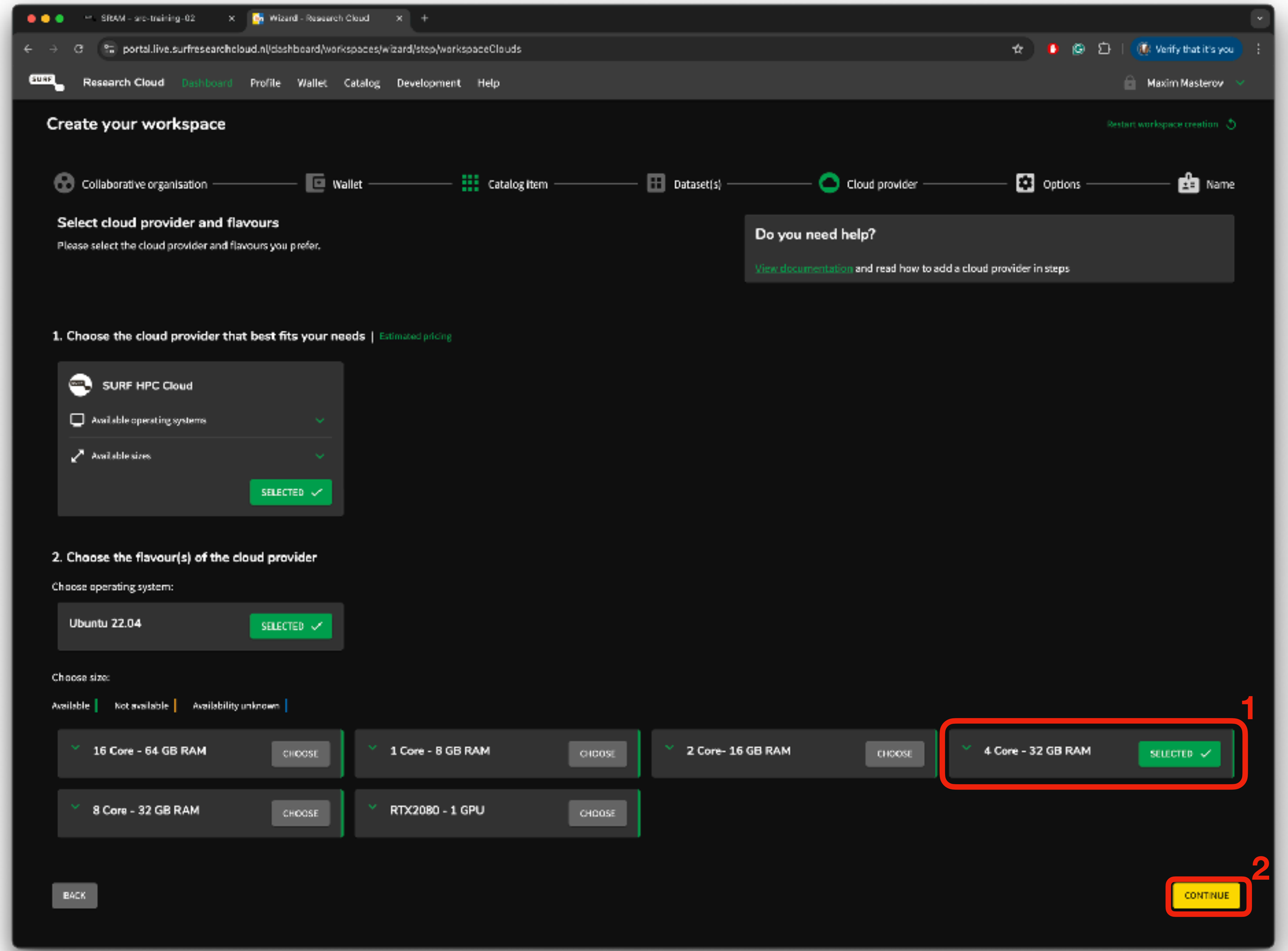
View documentation and read how to add a catalog item in steps

Ubuntu 22.04 with EESSI client

Ubuntu connected to EESSI software stack

SURF Research Cloud

Setting up VM: portal.live.surfresearchcloud.nl



portal.live.surfresearchcloud.nl/dashboard/workspaces/wizard/step/workspaceClouds

Create your workspace

Collaborative organisation — Wallet — Catalog item — Dataset(s) — Cloud provider — Options — Name

Select cloud provider and flavours

Please select the cloud provider and flavours you prefer.

Do you need help?

View documentation and read how to add a cloud provider in steps

1. Choose the cloud provider that best fits your needs | Estimated pricing

SURF HPC Cloud — SELECTED ✓

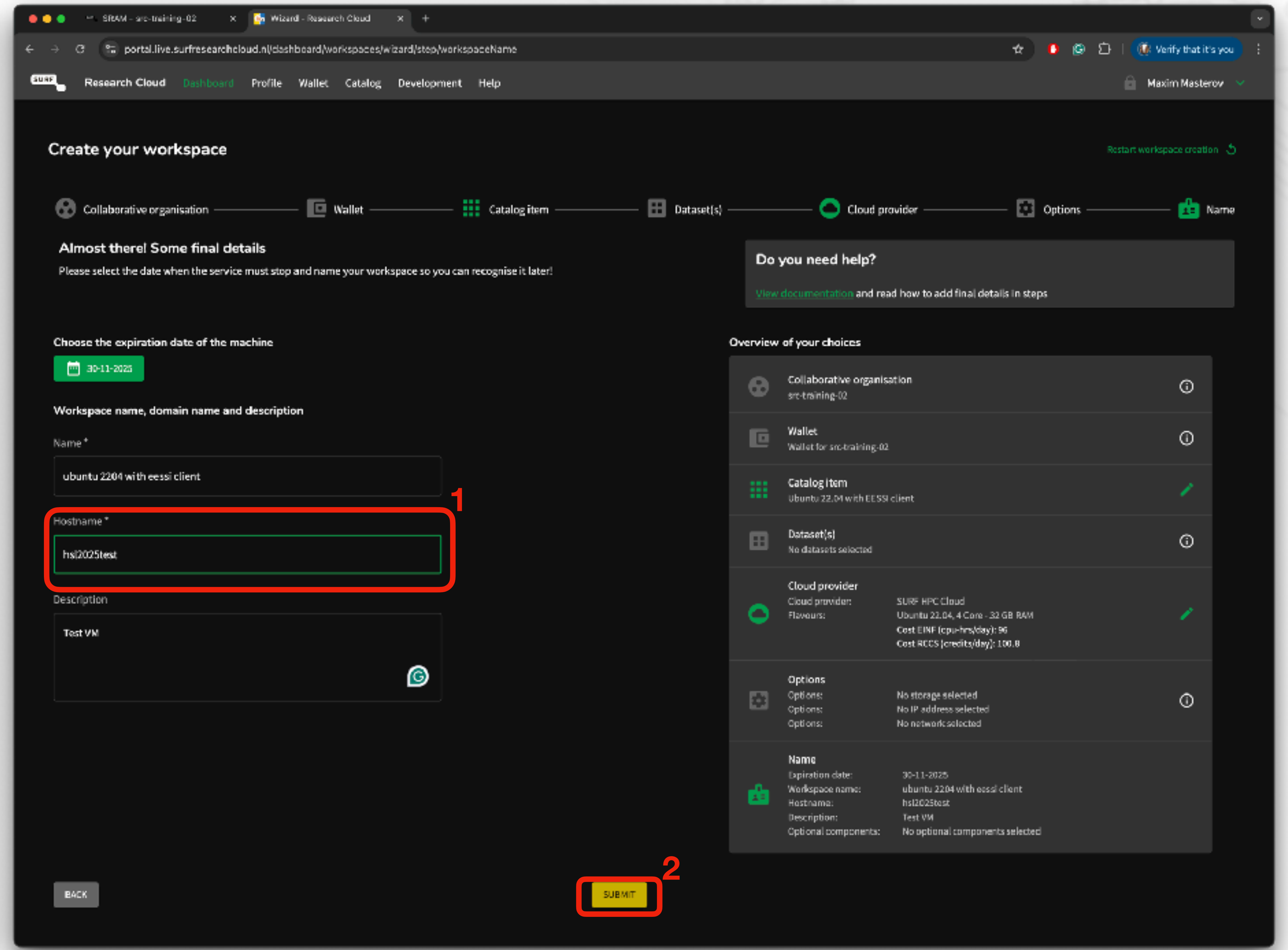
2. Choose the flavour(s) of the cloud provider

Choose operating system:

Ubuntu 22.04 — SELECTED ✓

Choose size:

16 Core - 64 GB RAM	CHOOSE
1 Core - 8 GB RAM	CHOOSE
2 Core- 16 GB RAM	CHOOSE
4 Core - 32 GB RAM	SELECTED ✓
8 Core - 32 GB RAM	CHOOSE
RTX2080 - 1 GPU	CHOOSE



portal.live.surfresearchcloud.nl/dashboard/workspaces/wizard/step/workspaceName

Create your workspace

Collaborative organisation — Wallet — Catalog item — Dataset(s) — Cloud provider — Options — Name

Almost there! Some final details

Please select the date when the service must stop and name your workspace so you can recognise it later!

Do you need help?

View documentation and read how to add final details in steps

Choose the expiration date of the machine

30-11-2025

Workspace name, domain name and description

Name *: ubuntu 2204 with eessi client

Hostname *: hsl2025test

Description: Test VM

Overview of your choices

Collaborative organisation: src-training-02

Wallet: Wallet for src-training-02

Catalog item: Ubuntu 22.04 with EESSI client

Dataset(s): No datasets selected

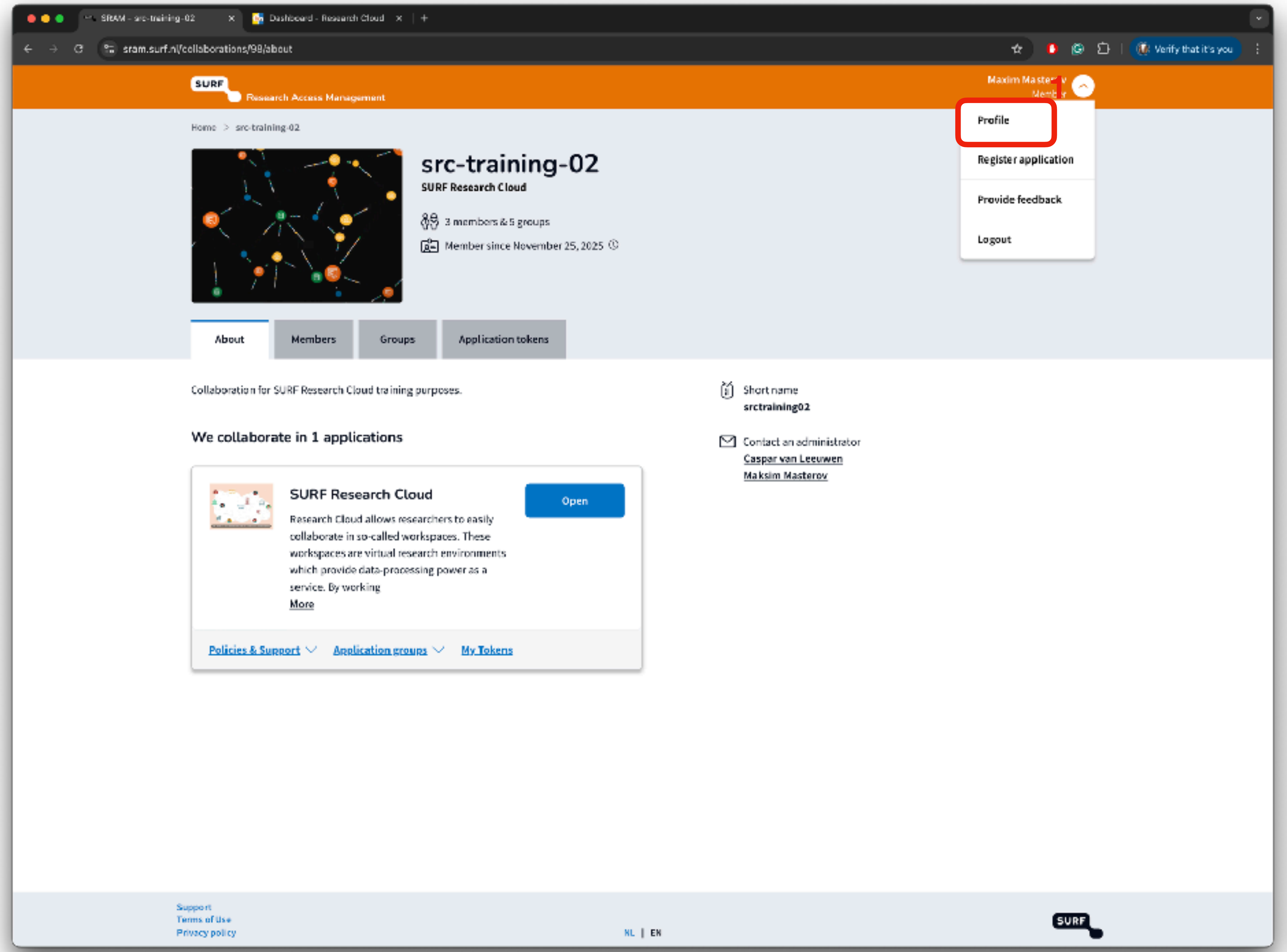
Cloud provider
Cloud provider:	SURF HPC Cloud
Flavours:	Ubuntu 22.04, 4 Core - 32 GB RAM
	Cost EINF (cpu-hrs/day): 96
	Cost RCCS (credits/day): 100.8

Options
Options:	No storage selected
Options:	No IP address selected
Options:	No network selected

Name
Expiration date:	30-11-2025
Workspace name:	ubuntu 2204 with eessi client
Hostname:	hsl2025test
Description:	Test VM
Optional components:	No optional components selected

SURF Research Cloud

Uploading SSH key: sram.surf.nl



src-training-02
SURF Research Cloud
3 members & 5 groups
Member since November 25, 2025

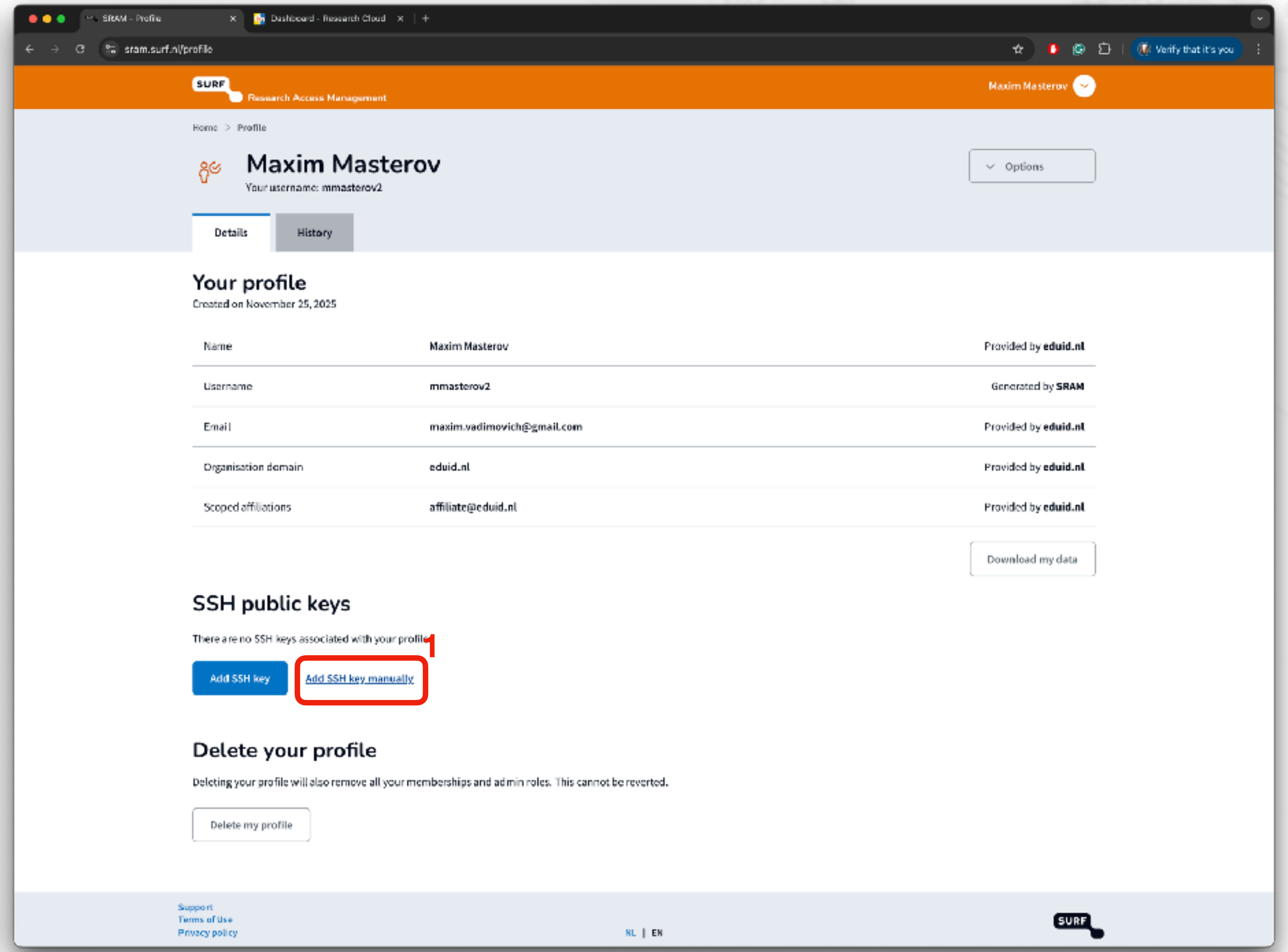
Collaboration for SURF Research Cloud training purposes.

We collaborate in 1 applications

SURF Research Cloud
Research Cloud allows researchers to easily collaborate in so-called workspaces. These workspaces are virtual research environments which provide data-processing power as a service. By working

Short name
srctraining02

Contact an administrator
Caspar van Leeuwen
Maksim Masterov



Maxim Masterov
Your username: mmasterov2

Your profile
Created on November 25, 2025

Name	Maxim Masterov	Provided by **eduid.nl**
Username	mmasterov2	Generated by **SRAM**
Email	maxim.vadimovich@gmail.com	Provided by **eduid.nl**
Organisation domain	eduid.nl	Provided by **eduid.nl**
Scoped affiliations	affiliate@eduid.nl	Provided by **eduid.nl**

SSH public keys
There are no SSH keys associated with your profile

Delete your profile
Deleting your profile will also remove all your memberships and admin roles. This cannot be reverted.

SURF Research Cloud

Uploading SSH key: sram.surf.nl

- The VM with “EESSI client” will contain the module environment standard for all HPC systems

- To activate the environment, source the initialisation script from the CVMFS mounting point

- Ones EESSI is activated, you can load the Python-3.13.1 module

- Use “pip install --user” to install Python packages locally

```
$ ssh <username>@<ip_address>
...
$ source /cvmfs/software.eessi.io/versions/2025.06/
init/bash
...
$ module load Python/3.13.1-GCCcore-14.2.0
$ pip install --user <package>
```

Hackathon

Hackathon

Particle Collision Performance Challenge

$$\mathbf{v}_i' = \mathbf{v}_i - \frac{2m_j}{m_i + m_j} \frac{\langle \mathbf{v}_i - \mathbf{v}_j \,|\, \mathbf{x}_i - \mathbf{x}_j \rangle}{||\mathbf{x}_i - \mathbf{x}_j||^2} (\mathbf{x}_i - \mathbf{x}_j)$$

$$\mathbf{v}_j' = \mathbf{v}_j - \frac{2m_i}{m_i + m_j} \frac{\langle \mathbf{v}_j - \mathbf{v}_i \,|\, \mathbf{x}_j - \mathbf{x}_i \rangle}{||\mathbf{x}_j - \mathbf{x}_i||^2} (\mathbf{x}_j - \mathbf{x}_i)$$