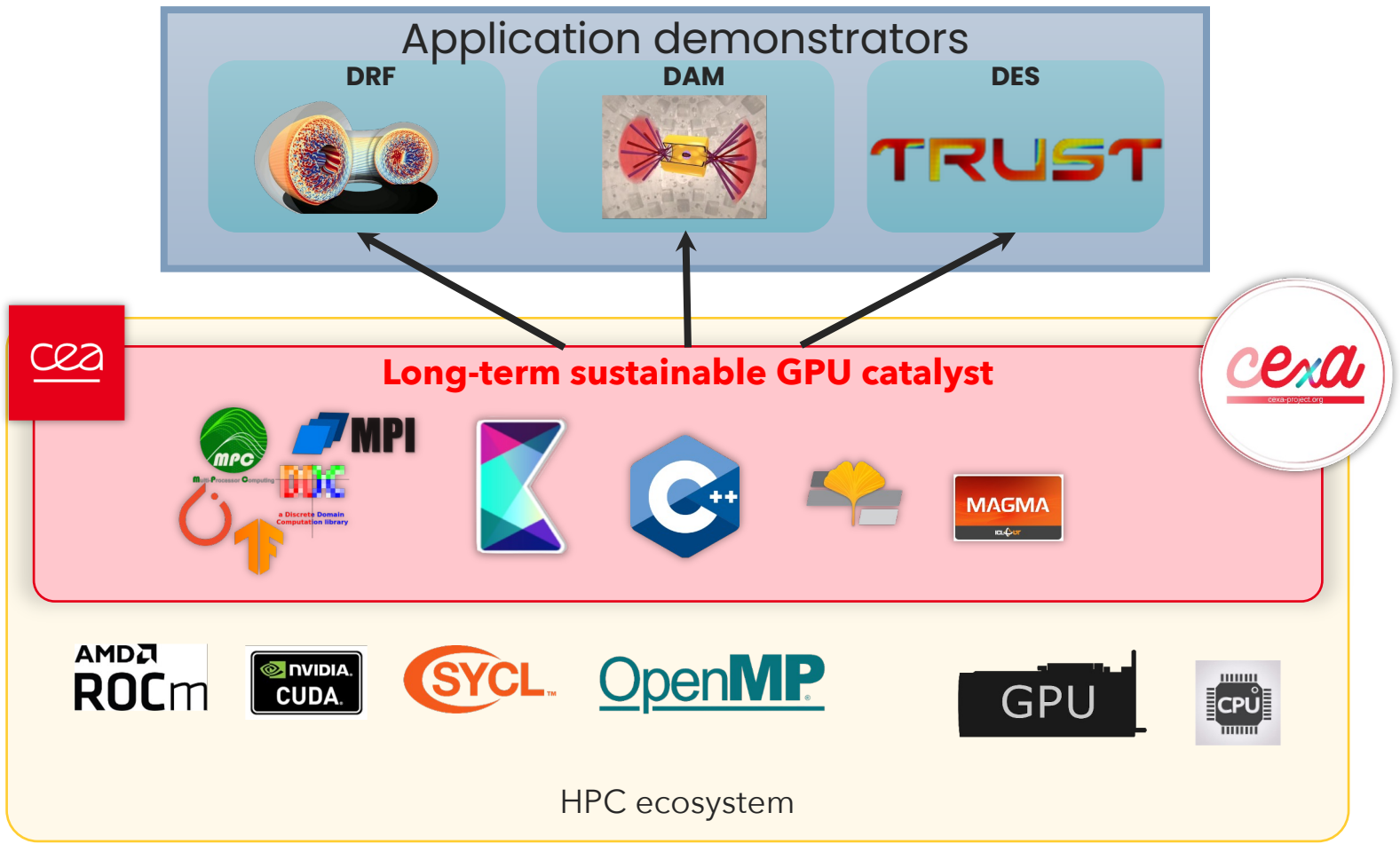
Welcome to the CExA kokkos Workshop 2024

CExA team, Damien Lebrun-Grandie (ORNL) and Luc Berger-Vergiat (SNL) in partnership with EoCoE

June 17 2024

1. The CExA project

CExA – an initiative to gather CEA experts and create a sustainable software catalyst for Exascale

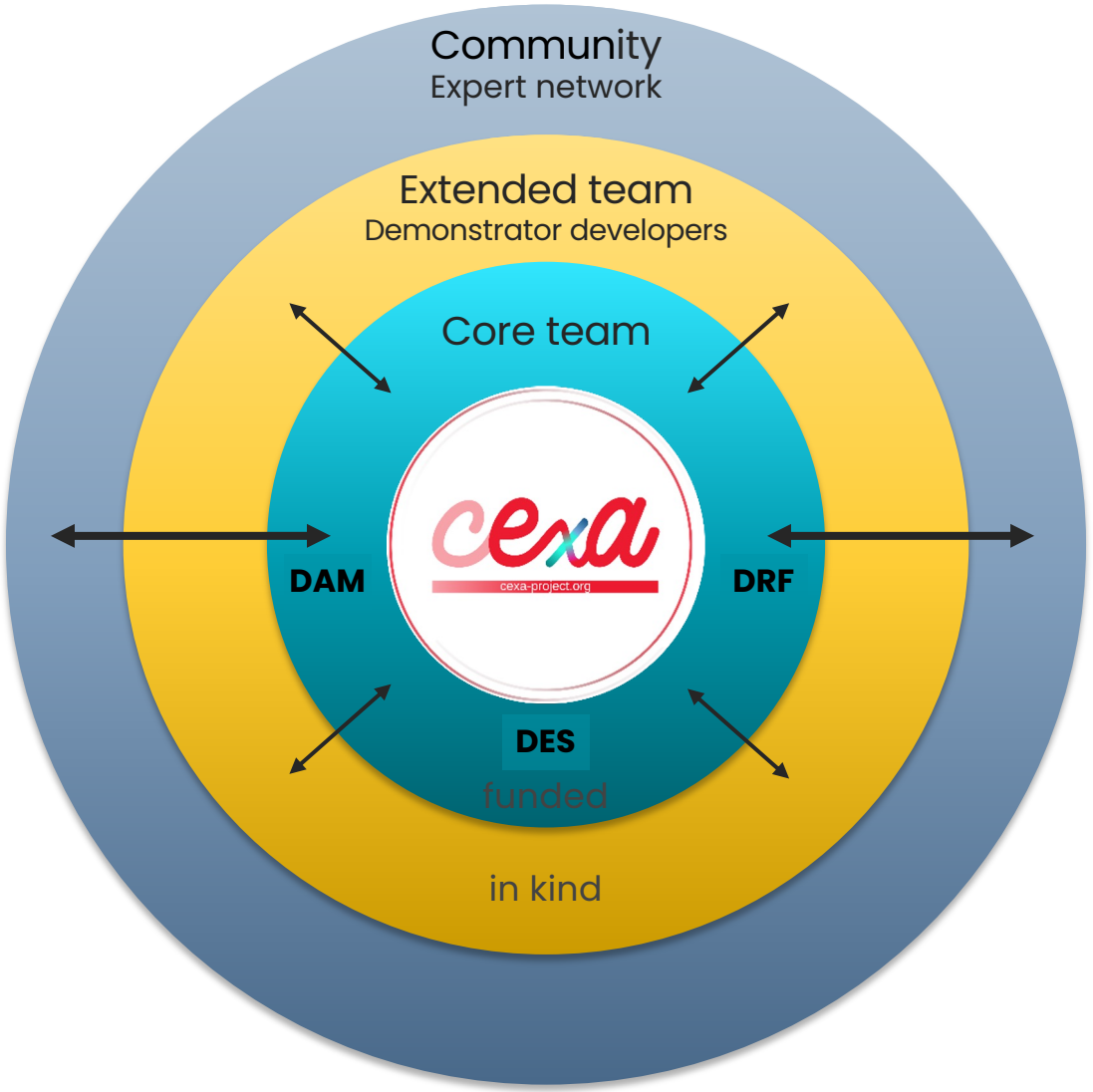


Disseminate and offer training in CEA and at large

Adapt application demonstrators

Provide a long-term sustainable software catalyst for GPU computing

Project organization



- **Core team**
  - Management, implementation and dissemination
  - 8 permanent researchers from all over CEA
  - 3 recrutements done, 3 more candidates selected
    - 1 as a permanent researcher !
  - Funding for 2 or 3 more hire expected next year
- **Extended team**
  - Demonstrator developers
    - Not funded
    - Find their interest in the participation
  - 3 new demonstrators to be selected next year
- **Community**
  - Federation of an expert network
  - Co-design of C**ExA**:
    - Identification of needs
    - Usage of C**ExA** in applications
  - Priority target for dissemination
  - Sustainability of the work

CEA is a member of the High-Performance Linux Fundation

HIGH PERFORMANCE SOFTWARE FOUNDATION

Projects

Spack

kokkos

APPTAINER

Viskores

HPCToolkit

E4S

Members

Premier

aws

Hewlett Packard Enterprise

Lawrence Livermore National Laboratory

Sandia National Laboratories

General

AMD

intel

NVIDIA

Argonne NATIONAL LABORATORY

OAK RIDGE National Laboratory

Los Alamos NATIONAL LABORATORY

kitware

cea

Associate

CDAC

UNIVERSITY OF OREGON

UNIVERSITY OF MARYLAND

2. Workshop organization

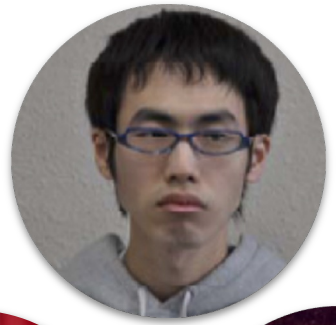
Your trainers during the next 3 days

Luc Berger-Vergiat

Kokkos Kernel project leader and developer (Sandia National Labs)

Yuuichi Asahi

Senior developer



Ansar Calloo

Group leader

Cedric Chevalier

Group leader

Mathieu Lobet

Group leader

Damlen Lebrun-Grandie

Kokkos project leader and developer (Oak Ridge National Lab)

Thierry Antoun

Developer

Rémi Baron

Senior developer

Thomas Padioleau

Senior developer

Paul Zenher

Developer

Workshop agenda

https://indico.math.cnrs.fr/event/12037/timetable

Day 1	Day 2	Day 3
9h15 – 12h Lecture (module 1 and 2)	9h15 – 12h10 Hands-on (module 3)	9h15 – 12h10 Hands-on (Kokkos kernels)
Lunch break	Lunch break	Lunch break
14h00 – 16h50 Lecture (module 2 and 3)	13h40 – 16h50 Hands-on (module 4)	13h40 – 16h00 Hands-on (Kokkos Tools)
		16h00 - 16h50 Kokkos Tea Time (Kokkos comm)

Workshop resources

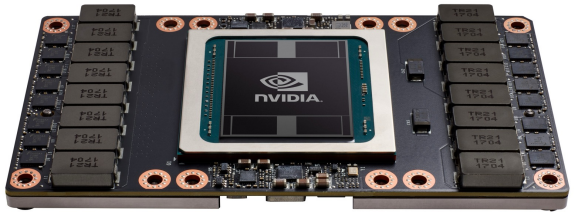
- GitHub page for Kokkos tutorials: https://github.com/kokkos/kokkos-tutorials
  - All slides (pdf)
  - All exercises (with solutions)
- Videos: https://kokkos.org/kokkos-core-wiki/videolectures.html
- Kokkos source code: https://github.com/kokkos/kokkos
- Kokkos core wiki: https://kokkos.org/kokkos-core-wiki/
- Cheat sheet: https://github.com/CExA-project/cheat-sheet-for-kokkos
- Our slides will be put on the Workshop indico: https://indico.math.cnrs.fr/event/12037/

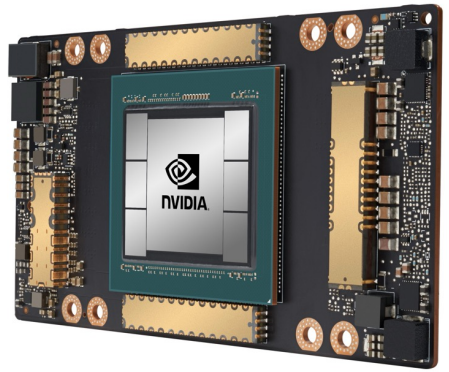
Computer resources

- You can use:
  - Your laptop
  - Our training desktop
  - Our training cluster (CPU and GPU)

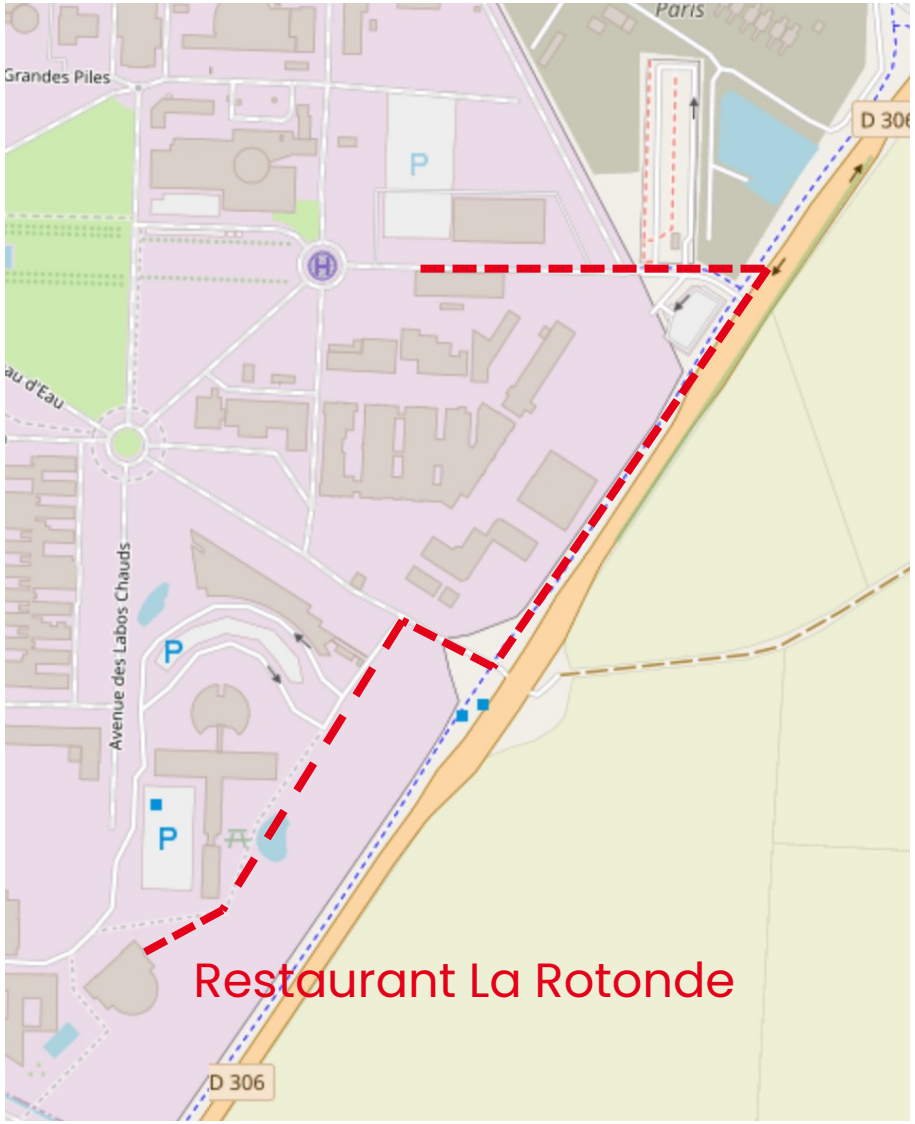
- Mésocentre Paris-Saclay – Ruche cluser
  - Doc: https://mesocentre.pages.centralesupelec.fr/user_doc/
  - CPU partition: bi-socket Intel Xeon Gold 6230 20C @ 2.1GHz
  - V100 GPU partition
  - A100 GPU partition
  - Only short runs

- Workshop GitHub page to help you: https://github.com/CExA-project/kokkos-workshop





We have lunch at La Rotonde Today

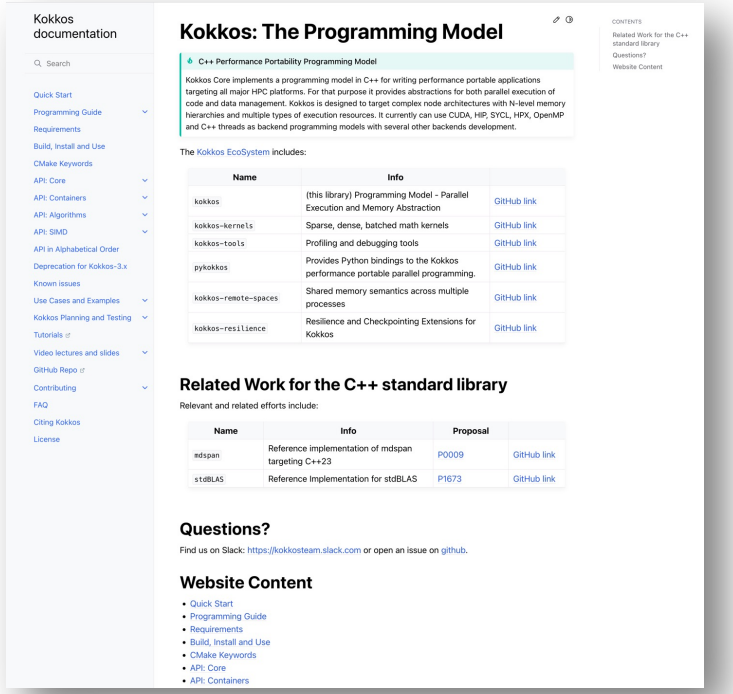


Approximatly 10min walk

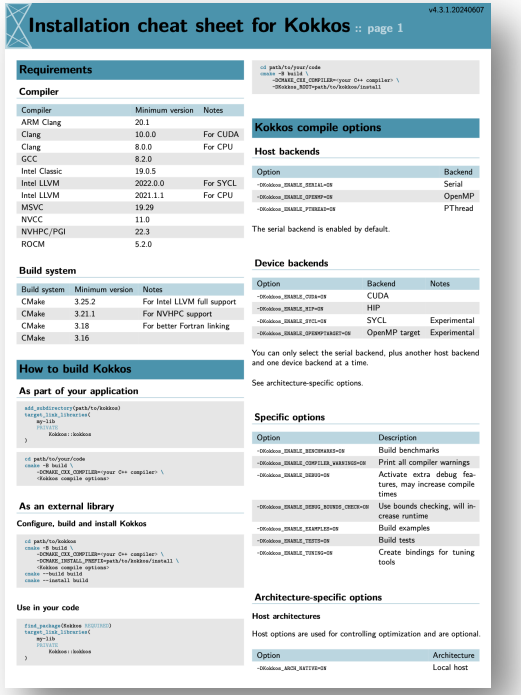
How we participate: help with documentation

https://github.com/CExA-project/cheat-sheet-for-kokkos

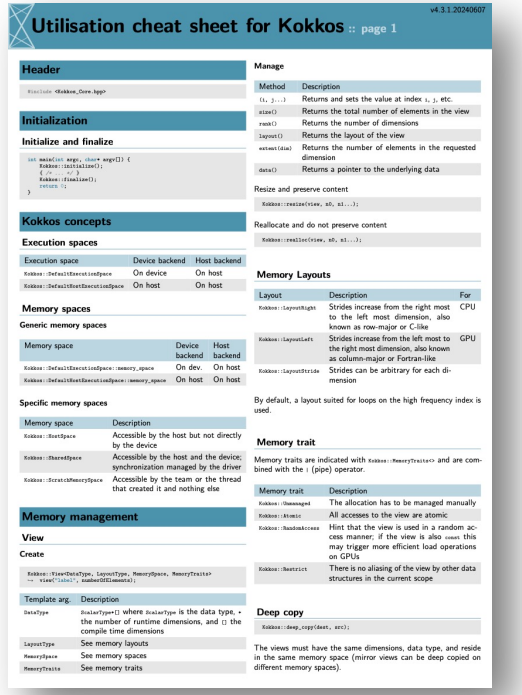
https://kokkos.org/kokkos-core-wiki/



Kokkos Wiki



Installation Cheat Sheet

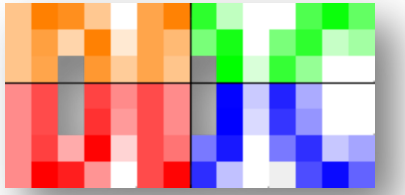


Utilization Cheat Sheet

How we participate: contribute to the Kokkos Ecosystem

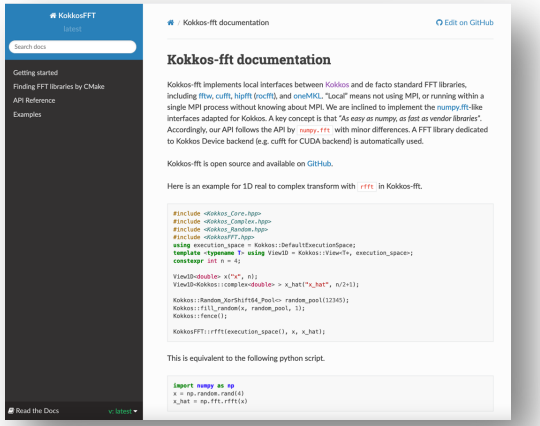
- **DDC** (Discrete data & computation) - https://ddc.mdls.fr/

  A C++-17 library that aims to offer to the C++/MPI world an equivalent to the xarray.DataArray/dask.Array python environment.



- **Kokkos-FFT** - https://github.com/kokkos/kokkos-fft/

  Kokkos-fft implements local interfaces between Kokkos and de facto standard FFT libraries, including fftw, cufft, hipfft (rocfft), and oneMKL.



- **Kokkos-Kernels** - https://github.com/kokkos/kokkos-kernels

  Implements local computational kernels for linear algebra and graph operations, using the Kokkos shared-memory parallel programming model.

- **Kokkos-Core** - https://github.com/kokkos/kokkos

  Kokkos Core implements a programming model in C++ for writing performance portable applications targeting all major HPC platforms.

- **Kokkos-Comm** - https://github.com/kokkos/kokkos-comm

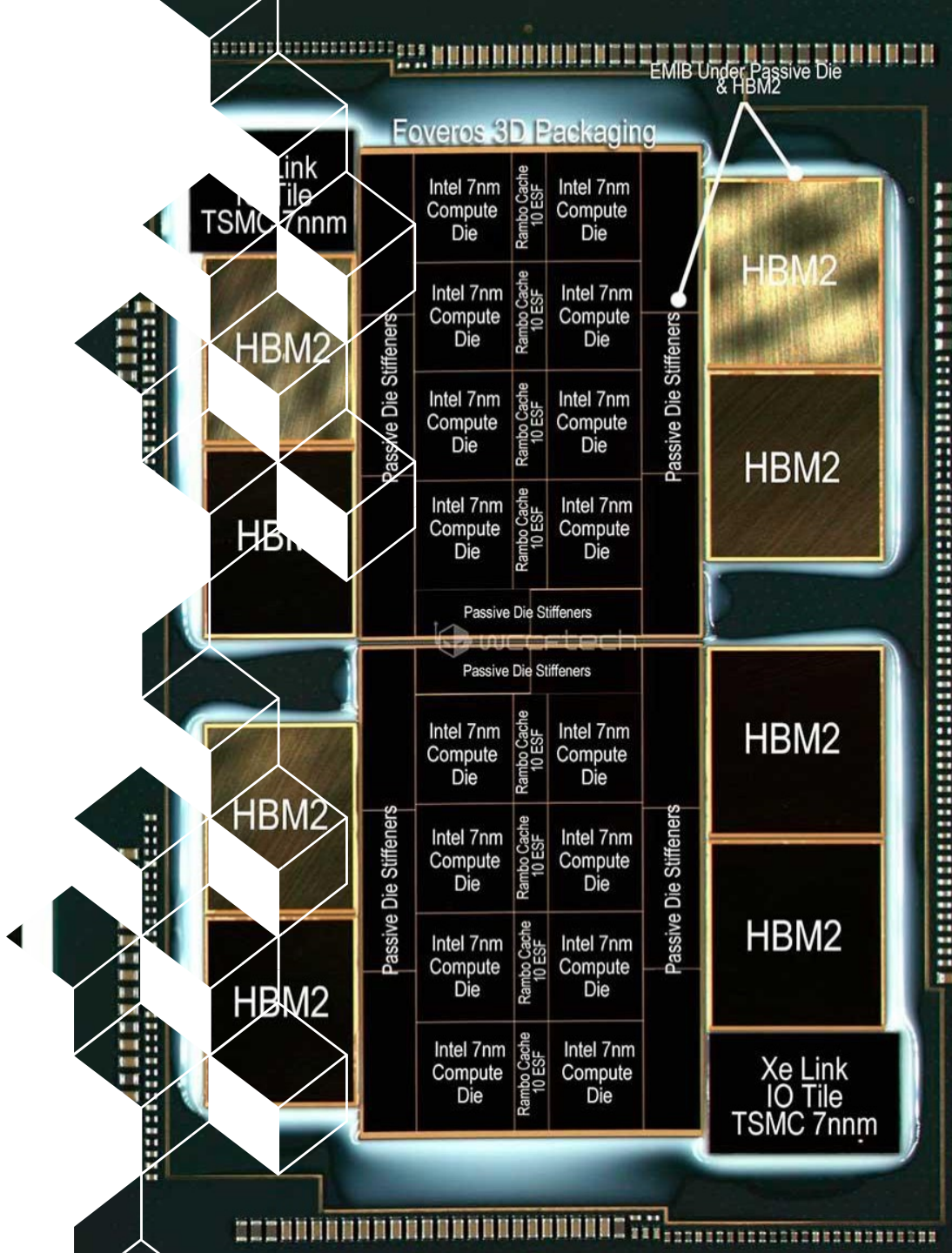
  Abtracted distributed communication interface integrated in the Kokkos ecosystem

And more

- Improve software quality
  - Setup GPU CI for CEA libraries
  - Maintaining Kokkos Spack recipes
  - Huge refactor & redesign of `create_mirror[_view][_and_copy]`

- Test hardware & improve kokkos for it
  - Intel Ponte Vecchio backend improvement
  - NVidia Grace Hopper memory management handling

- Support our applications
  - Test UVM viability & performance



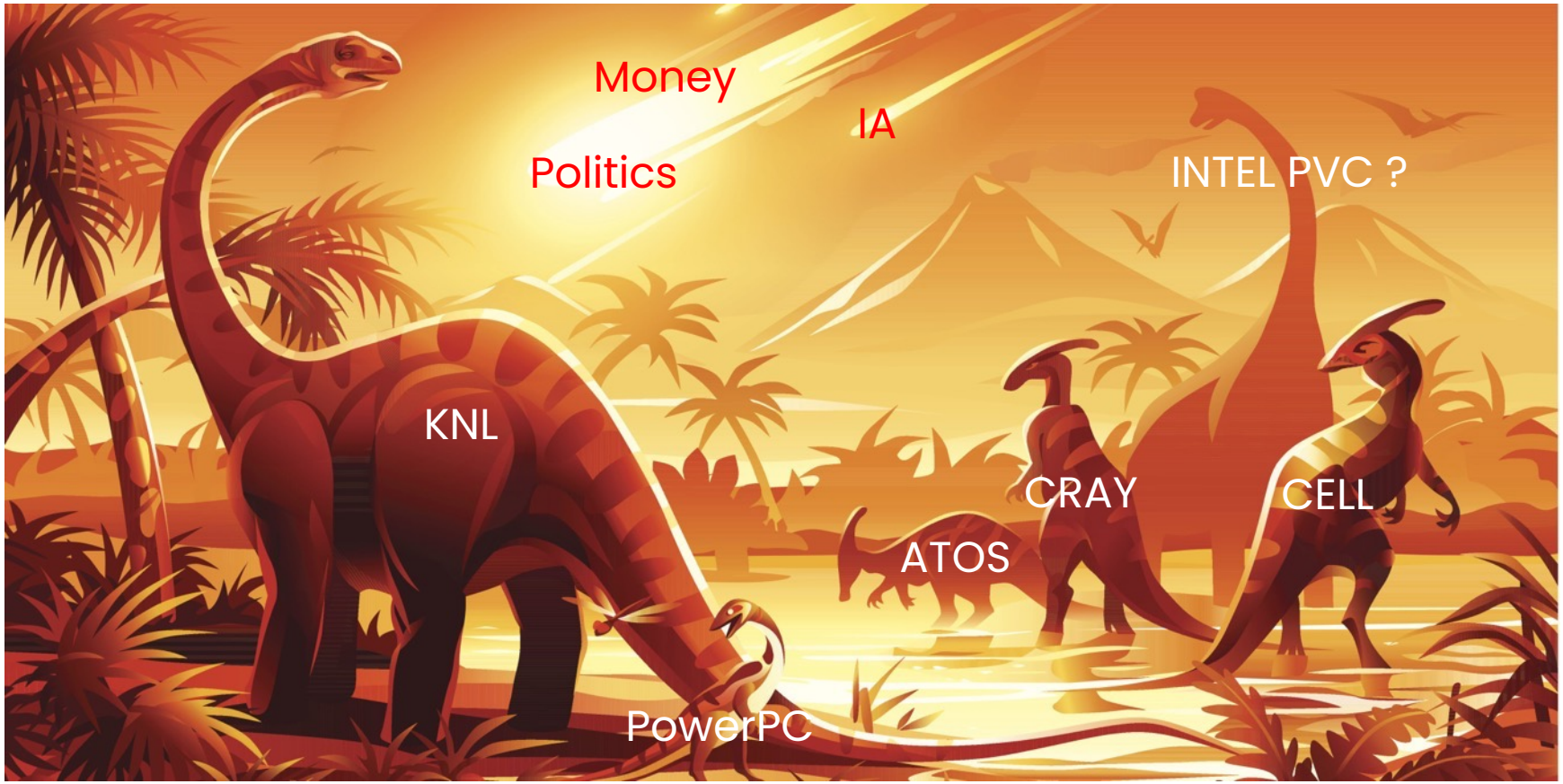
CExA project website

- https://cexa-project.org
- See our position offers
- Subscribe to our mailing list: https://lists.cexa-project.org/sympa/subscribe/network
- Regular remote seminars focusing on the Kokkos community (Kokkos coffees and tea-time)
- French channel on the Kokkos slack: https://kokkosteam.slack.com/

3. Why do we need a performance portable model?

Life in our HPC world

Evolution in HPC : exotic and disruptive hardware are permanently appearing (and disappearing) in super-computers history (does not mean they were bad hardware)



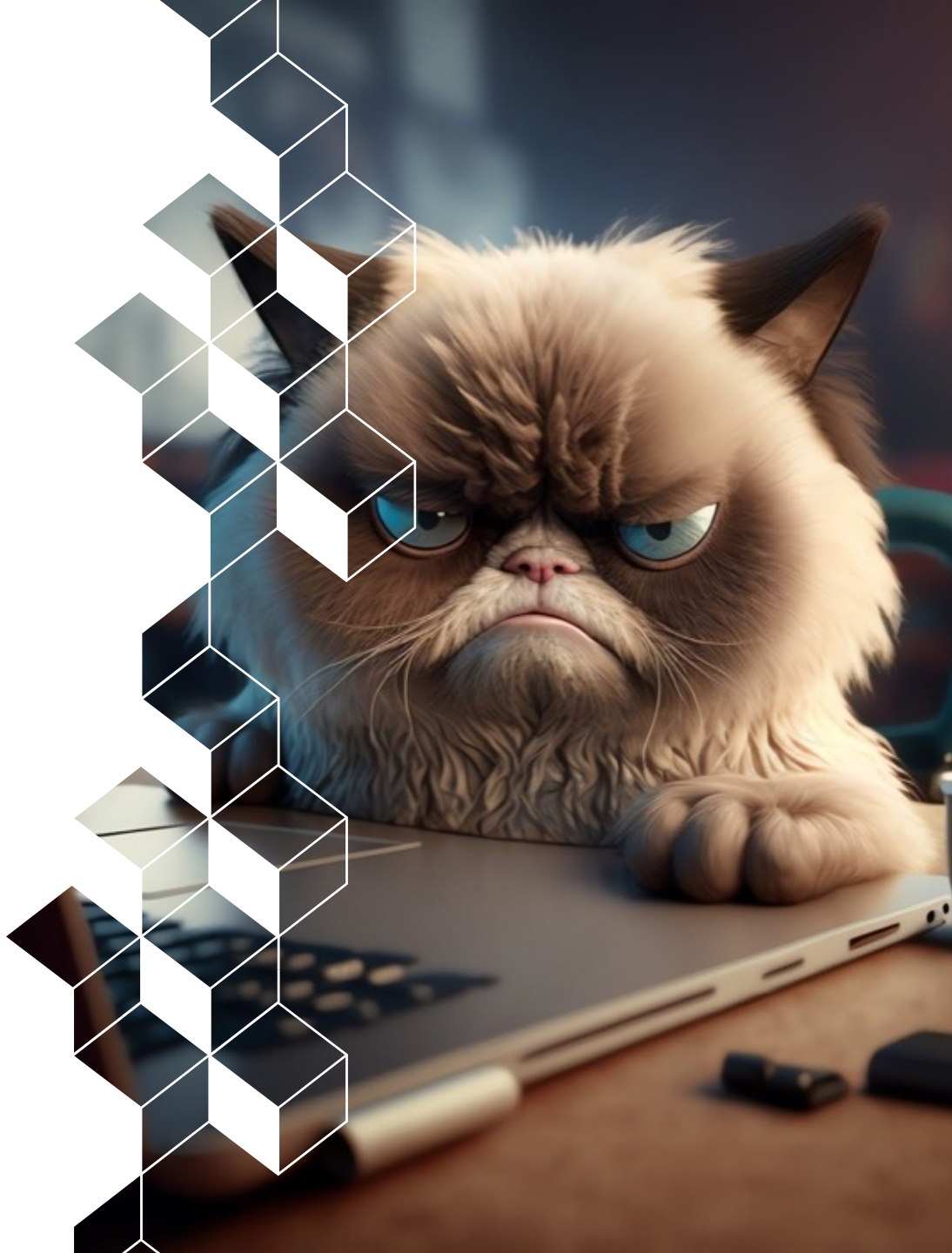
Exciting for research, not for application developers

New technologies are exciting for researchers but may be stressful for application developers and users :

- May require vendor specific programming models
- May require new programming paradigms
- Algorithms may have to be rewritten

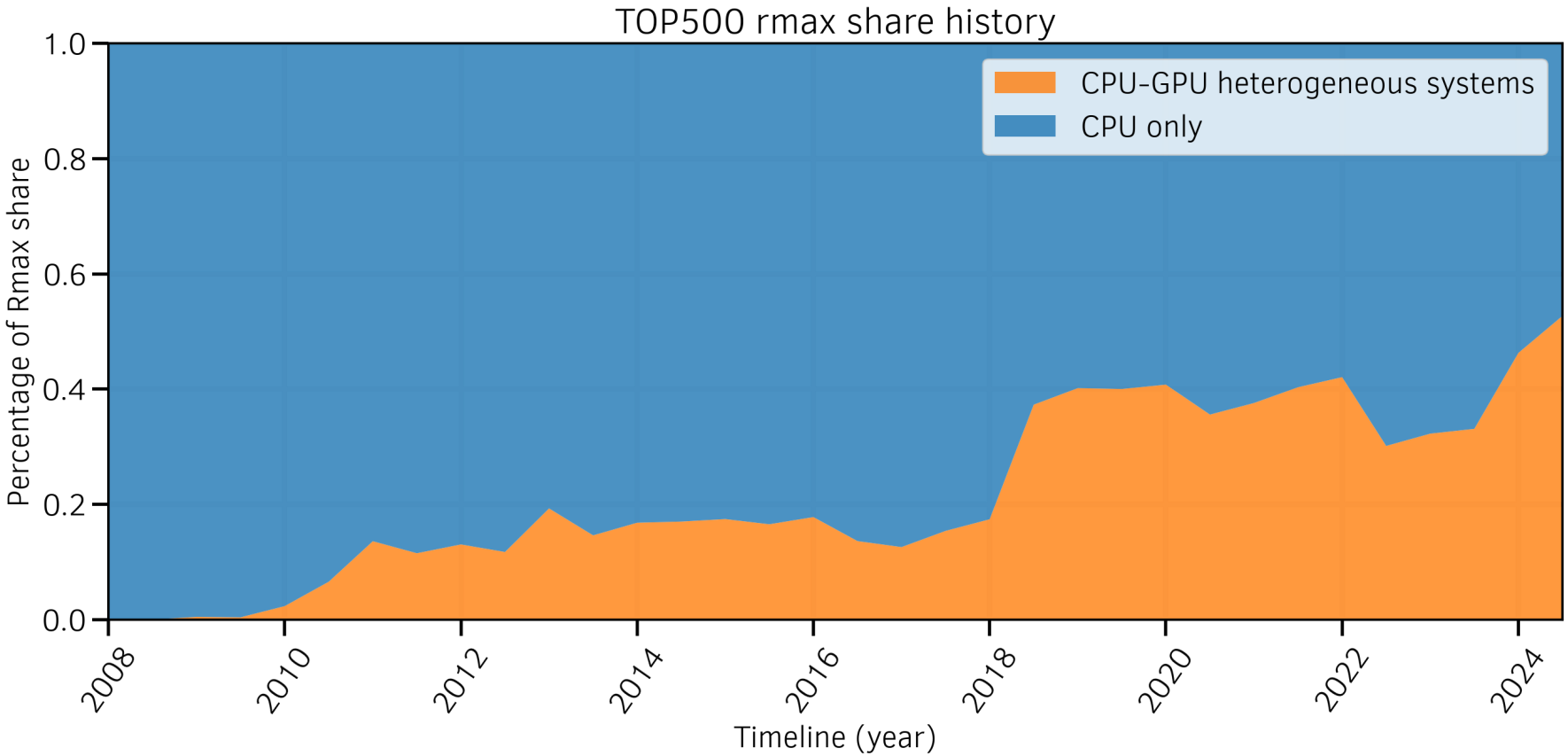
Developers must :

- Update their knowledge to handle new hardware specificity and programming models
- Rewrite or duplicate part of their applications
- Bet on a technology without long-term vision



HPC landscape

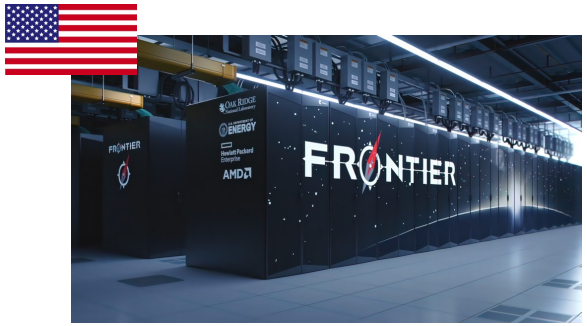
- More and more computational power in TOP500 super-computers comes from accelerated systems equipped of GPU
- Rmax is the benchmarked computational power



*heterogeneous system excludes Xeon Phi accelerators

HPC landscape in the world

- Almost all most powerful super-computers are equipped of GPU today



Frontier – 1.2 Eflops
AMD CPU and GPU

Aurora – 1 Eflops
Intel CPU and GPU

Fugaku – 440 Pflops
FUJITSU ARM A64FX

LUMI – 379 Pflops
AMD CPU and GPU

ALPS – 270 Pflops
NVIDIA (ARM) CPU and GPU

LEONARDO – 241 Pflops
INTEL CPU and NVIDIA GPU

Current GENCI Super-computers in France

- National super-computers accessible to academics
- Also heterogeneous at the French scale

Jean-Zay – 125 Pflops
Intel CPU and
NVIDIA H100 GPU

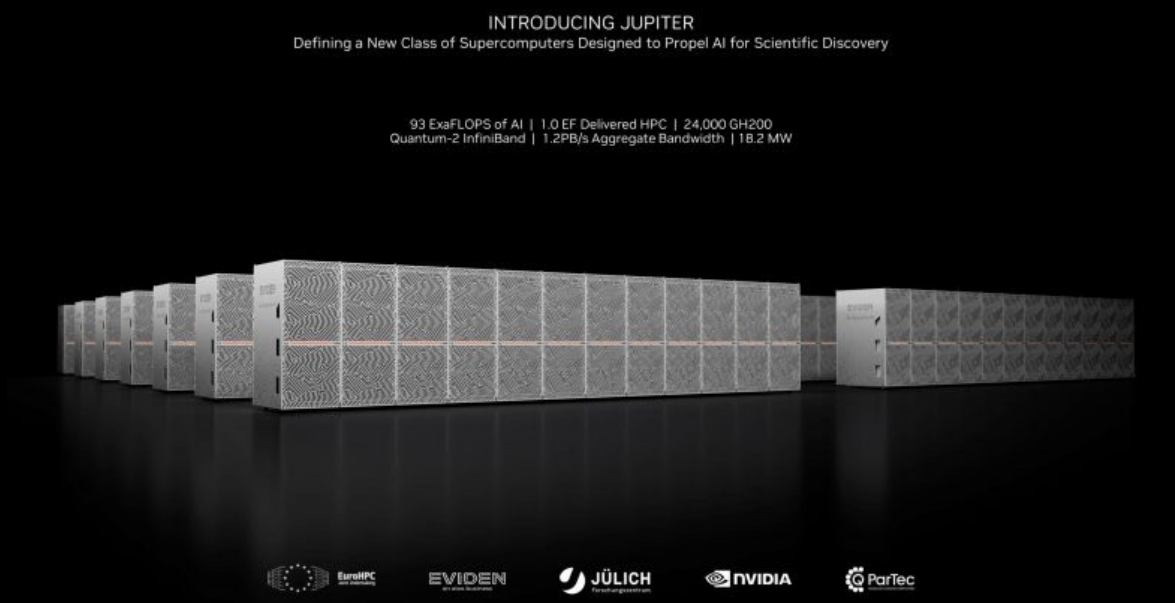
Adastra – 46 Pflops
AMD CPU and
MI250 GPU

Joliot-Curie
ROME – 7 Pflops
AMD CPU

CEA HE – 57 Pflops
NVIDIA GH200
CPU/GPU

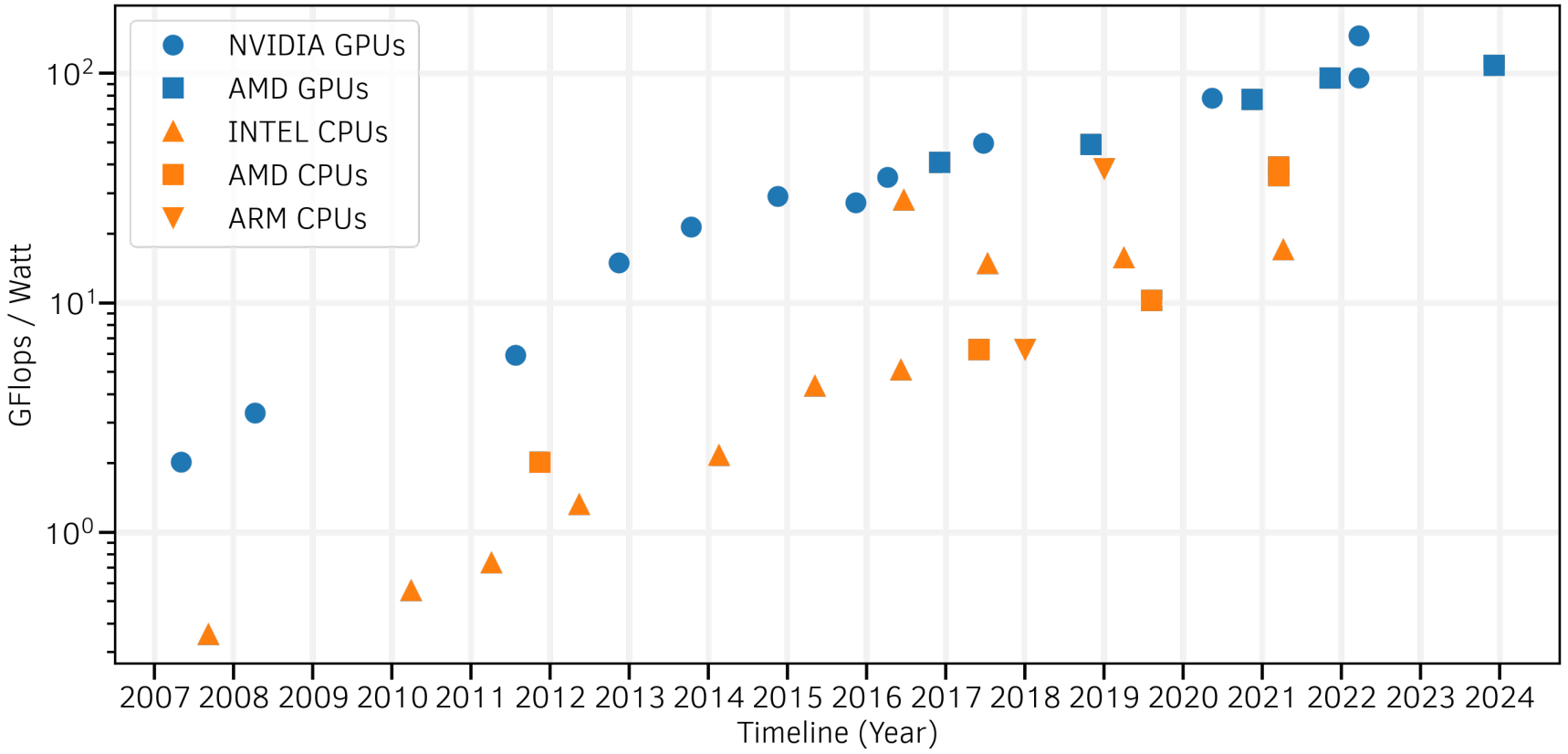
Coming Exascale super-computers

- Exascale coming in Europe :
  - European pre-Exascale systems: Mix of AMD & Nvidia
  - First Exascale machines planned in Europe for 2024/2025
  - First European Exascale machine: Jupiter at Jülich Super-Computing Center (Germany): Nvidia GH200 GPUs and Rhea ARM CPUs
  - Second European Exascale machine: CEA/TGCC (Jules Vernes consortium)

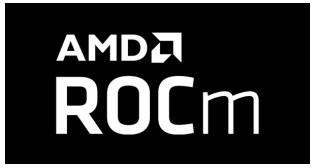


Why GPUs have conquered the HPC ecosystem?

- CPUs are general purpose computing units
- GPU are specialized chips to achieve highly parallel arithmetic operations



Current state of HPC hardware



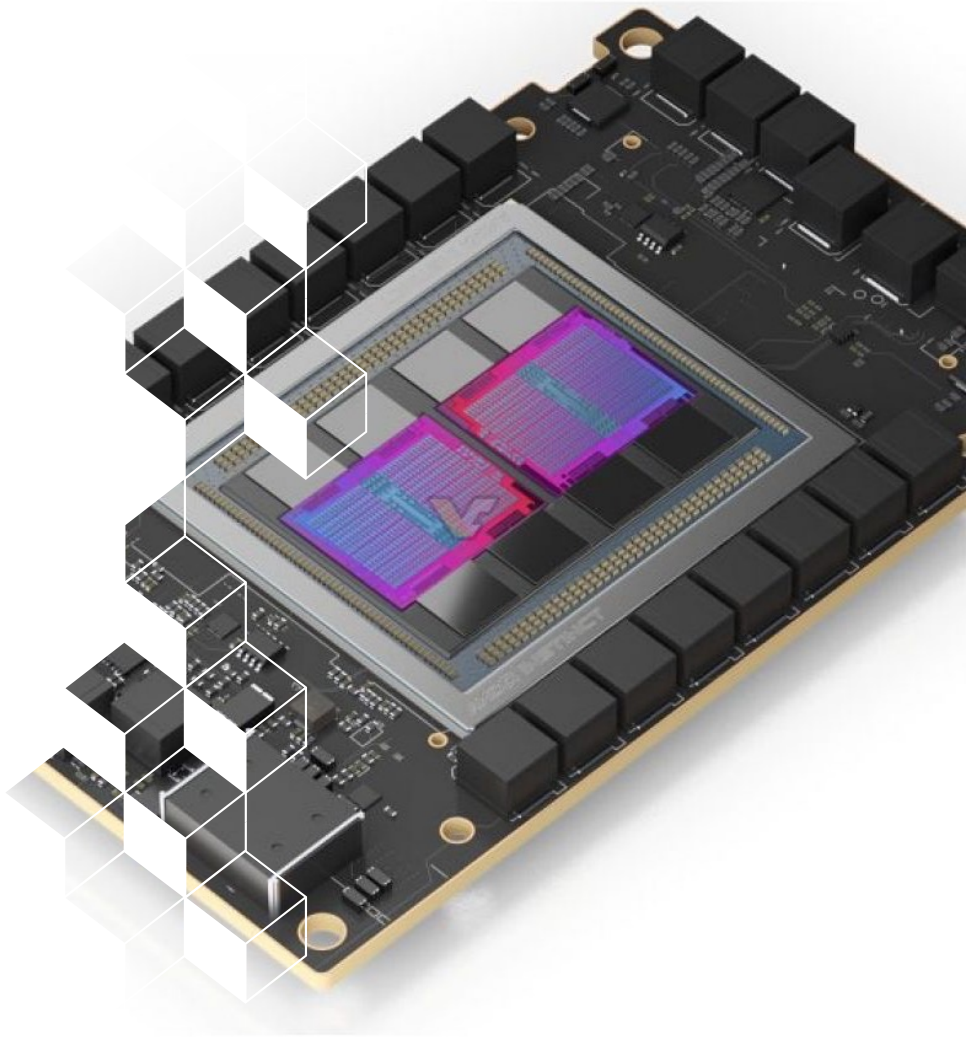
- NVIDIA used to dominate the HPC GPU market (as Intel used to be for CPUs)
  - Good for technology stability (programming models, optimization, etc)
  - Bad for market price, innovation, etc

- Today's landscape is composed of many vendors:
  - NVIDIA
  - AMD CPUs and GPUs
  - ARM based processors and accelerators
  - Intel GPUs

- Super-computers tend to be more and more heterogeneous:
  - CPUs + GPUs today,
  - RISC-V, TPUs, FPGA and quantum accelerators in the future?

Challenges for porting applications at Exascale

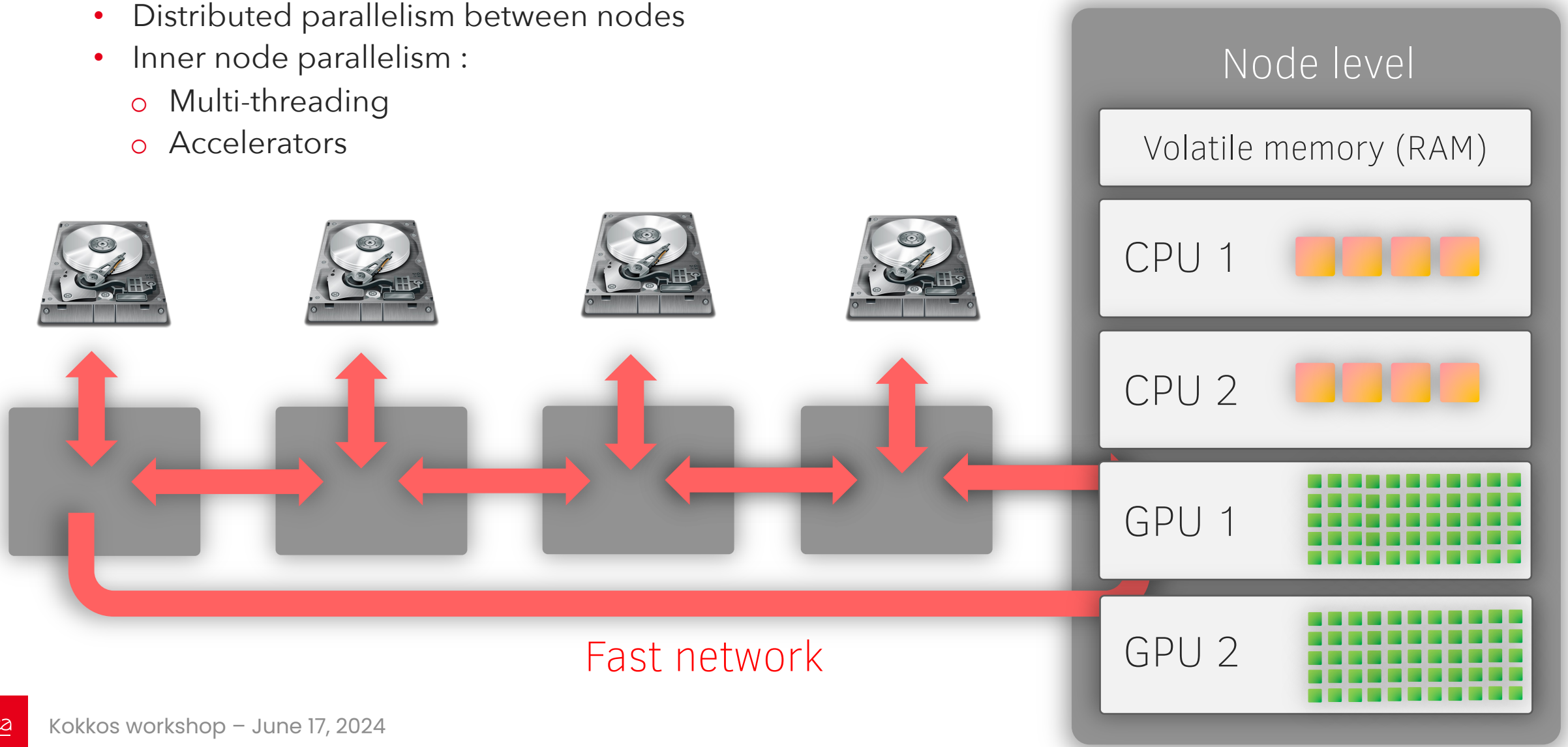
- **Engineering skills** – complex architectures, expertise to tune and optimize applications, more significant for small developer teams
- **Optimization** – some applications may be partially ported or ported but not optimized
- **Software stack** – Many different programming models and libraries (CUDA, OpenACC, OpenMP, Kokkos, StarPU). Not a clear view about which one to choose.
- **Hardware zoology** – HPC technologies evolve very fast and may be more and more heterogeneous (GPU accelerators, FPGAs, TPUs, quantum accelerators).
- **Long term maintainability** – application life longer than hardware, applications written by scientific should be maintainable by physicist



How to program heterogeneous systems

Super-computers still have multiple parallelism layers:

- Distributed parallelism between nodes
- Inner node parallelism :
  - Multi-threading
  - Accelerators

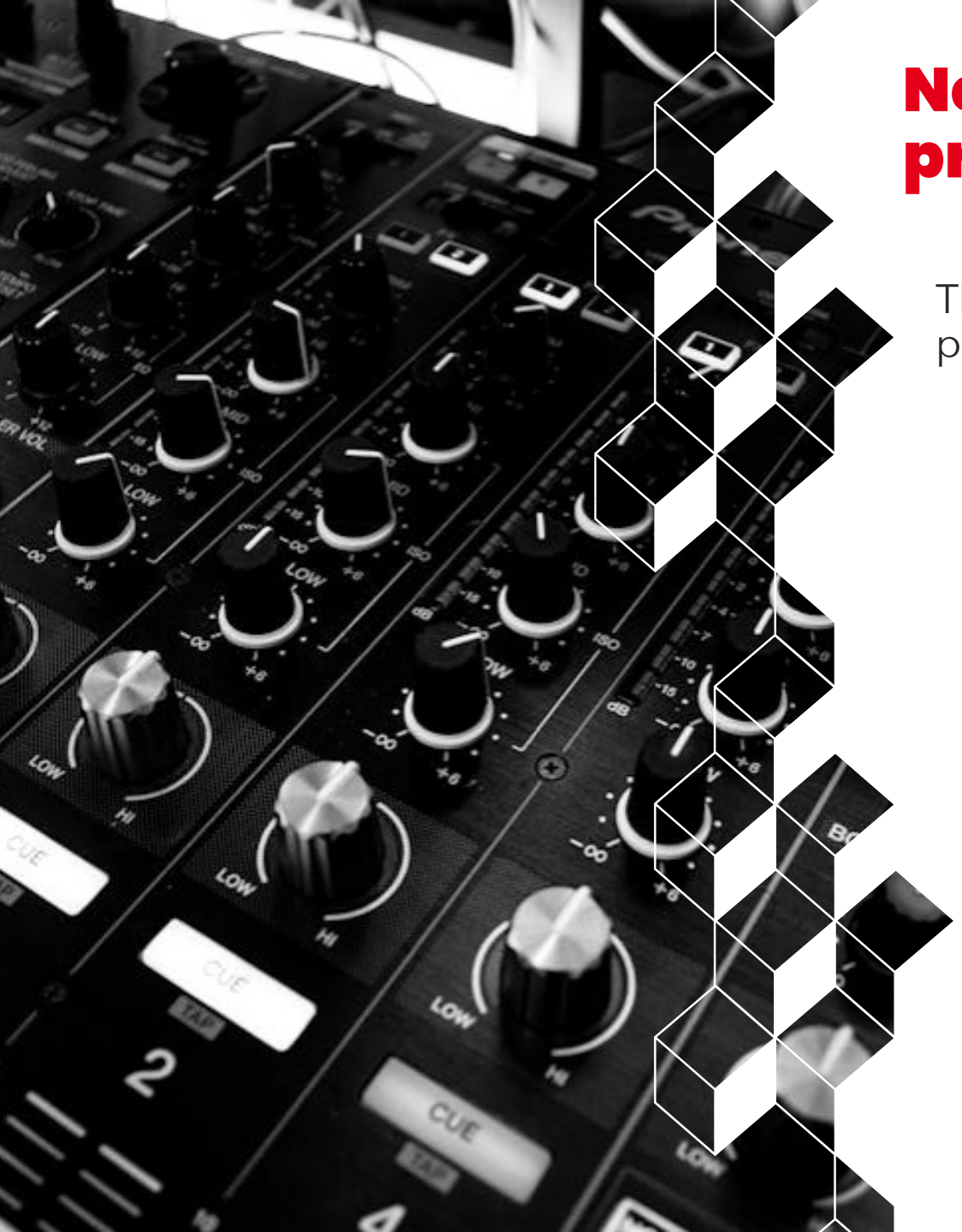


What is performance portability?

- Ideal goal, write one single implementation that:
  - Compiles and runs on multiple architectures
  - Obtains performant memory access patterns across architectures
  - Can leverage architecture-specific features where possible.
- There's a difference between portability and performance portability.

Example: implementations may target particular architectures and may not be thread scalable.

(e.g., locks on CPU won't scale to 100,000 threads on GPU)



Need for performance portable programming models

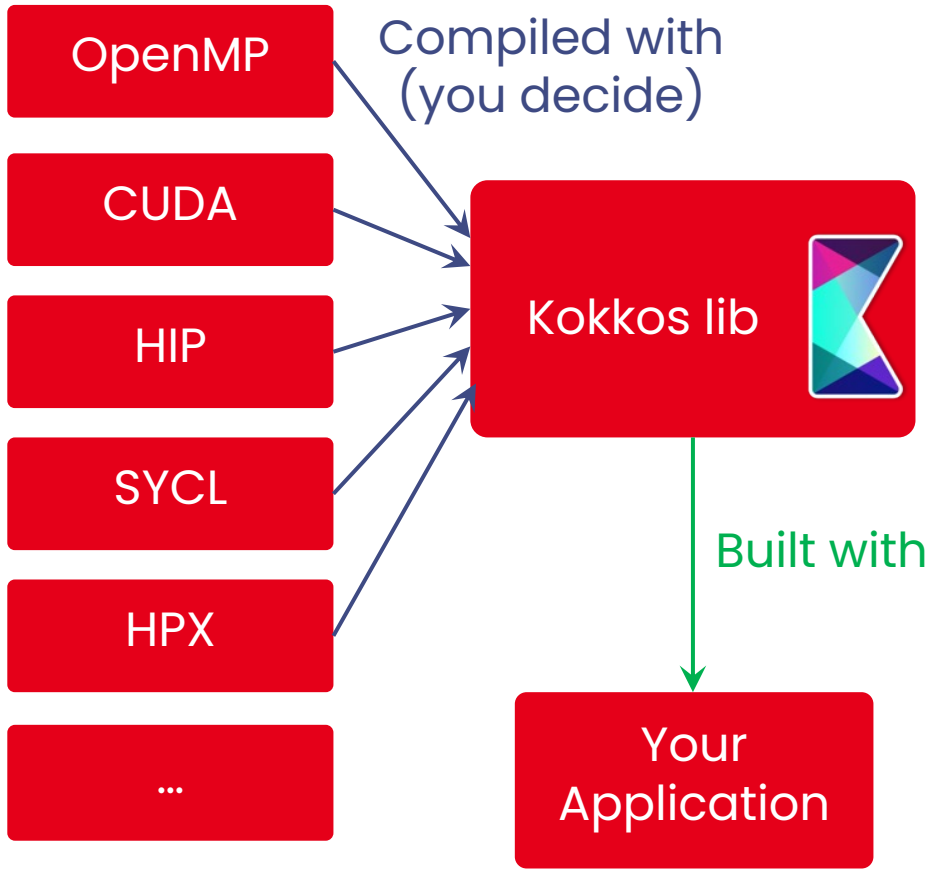
The choice depends on the value of the cursor between many parameters :

- Performance across a wide range of architectures
- Portability across a wide range of architectures
- Maturity (bugs, advanced features)
- Long-term support
- Code maintainability
- Programing complexity (required programming skilled)
- Ecosystem and interoperability with existing libraries (linear algebra, input/output, machine learning, etc)

3. What is Kokkos ?

What is Kokkos?

- A C++ Programming Model for Performance Portability
  - Implemented as a template library on top CUDA, HIP, OpenMP, …
  - Aims to be descriptive not prescriptive
  - Aligns with developments in the C++ standard
- Expanding solution for common needs of modern science and engineering codes
  - Math libraries based on Kokkos
  - Tools for debugging, profiling and tuning
  - Utilities for integration with Fortran and Python
- It is an Open Source project with a growing community
  - Maintained and developed at
  - Hundreds of users at many large institutions



Kokkos main capability

Basic features:

- Parallel loop: one-dimension, multi-dimensions, reduction patterns and more like OpenMP
- Multidimensional arrays like Fortran or Python
- Memory and execution policy to decide where data is located and where kernels are run
- Implicit data layout and data access management for performance

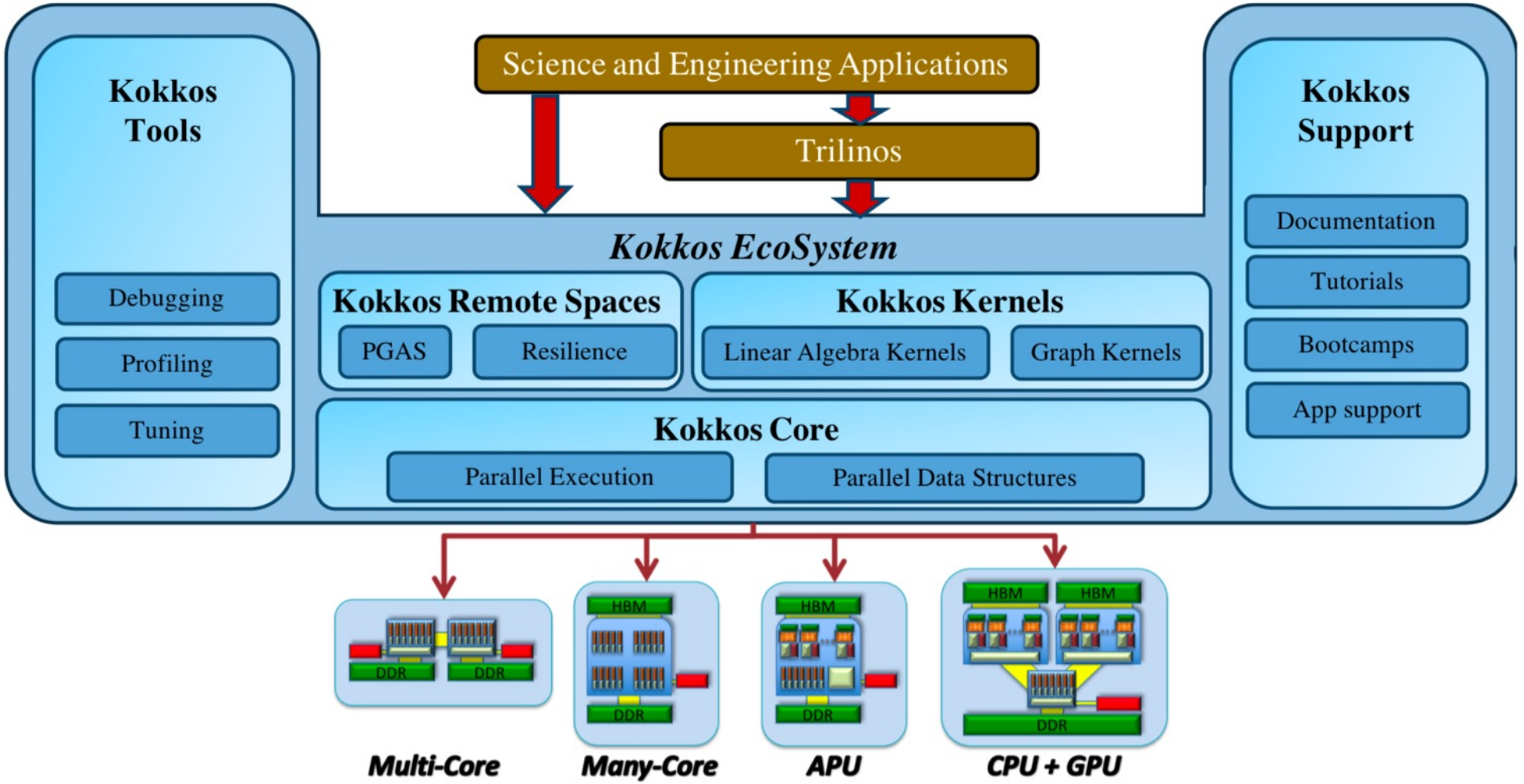
More advanced features:

- Thread safety, thread scalability and atomic operation
- Hierarchical parallelism (threading, vectorization, SIMT, etc.)
- Optimization capability

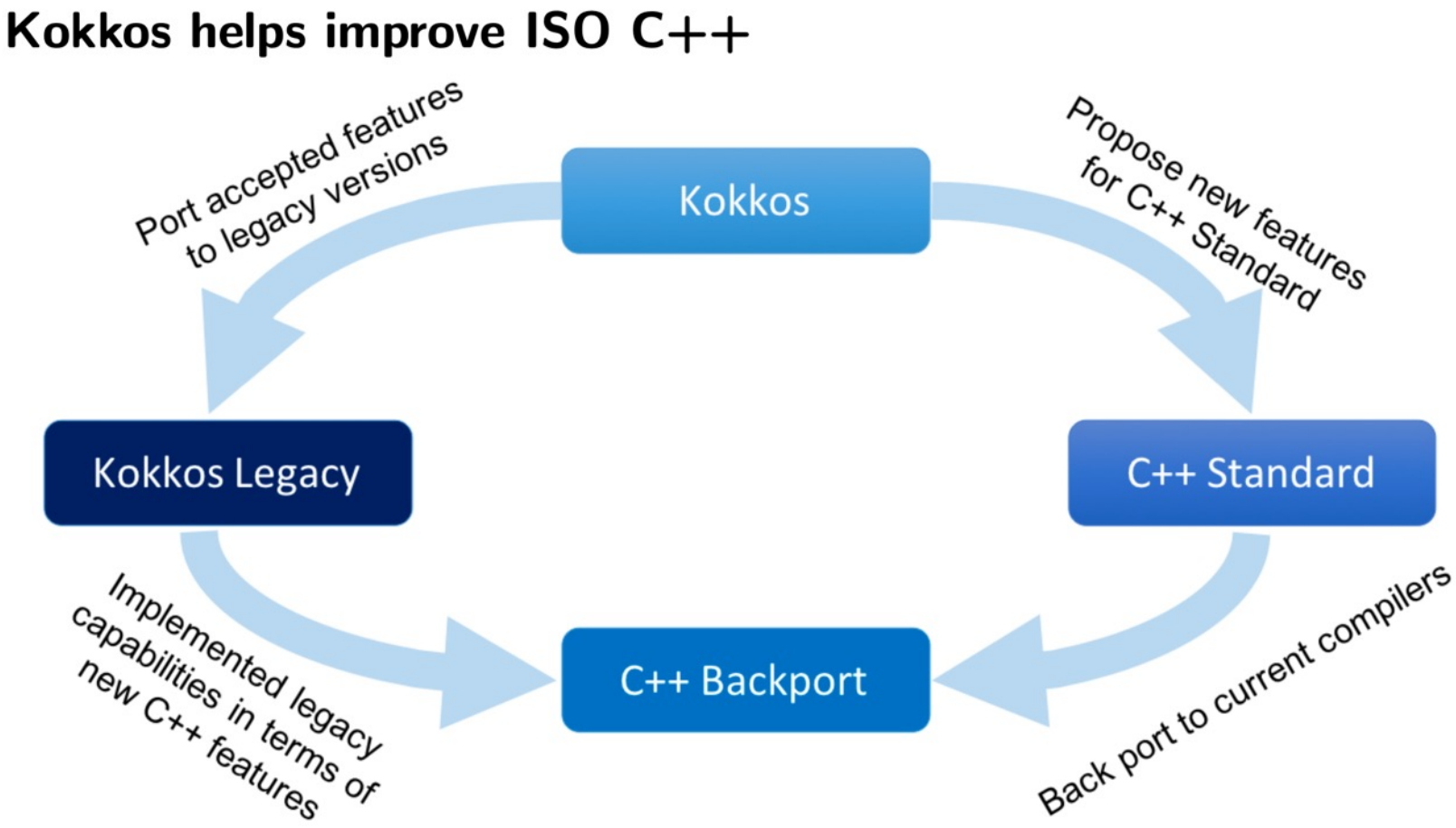
Tools:

- Compatibility with classical debuggers and profilers
- Build-in algorithm (sorting) like Thrust and mathematic features (linear algebra)
- Interoperability with Python, Fortran and other programming models

The whole ecosystem picture



Kokkos helps improve ISO C++



Ten current or former Kokkos members are members of the ISO C++ standard committee

C++20 std::atomic_ref

C++11 std::atomic insufficient for HPC

- Objects, not functions, with only atomic access
- Can't use non-atomic access in one operation, and then atomic access in the next

C++20 std::atomic_ref adds atomic capabilites as in Kokkos

- Can wrap standard allocations
- Works also for sizes which can't be done lock-free (e.g. complex<double>)
- Atomic operations on reasonably arbitrary types

```
// Kokkos today
Kokkos::atomic_add(&a[i],5.0);

// atomic_ref in ISO C++20
std::atomic_ref(a[i]) += 5.0;
```

C++23 std::mdspan

C++ does not provide multi dimensional arrays

- Every scientific programming language has them: Fortran, Matlab, Python, ...

C++23 std::mdspan adds Kokkos::View like arrays

- Reference semantics.
- Compile time and runtime extents (also mixed)
- Data layouts to allow for adapting hardware specific access patterns.
- Subviews!

```
// Kokkos today
View<float**[5],LayoutLeft> a("A",10,12); a(3,5,1) = 5;

// mdspan in ISO C++23
using ext = extents<int,dynamic_extent,dynamic_extent,5>;
mdspan<float,ext,layout_left> a(ptr,10,12); a[3,5,1]+=5;
```

High Performance Software Foundation

Goal: build, promote, and advance an open and portable core software stack for high performance computing.

HPSF aims to make life easier for high performance software developers through a number of focused initiatives, including:

- Continuous integration resources tailored for HPC projects
- Continuously built, turnkey software stacks
- Architecture support
- Performance regression testing and benchmarking
- Collaborations with other LF projects, such as OpenSSF, UEC, UXL Foundation, and CNCF

- https://hpsf.io/

HIGH PERFORMANCE SOFTWARE FOUNDATION

Spack

kokkos

Viskores

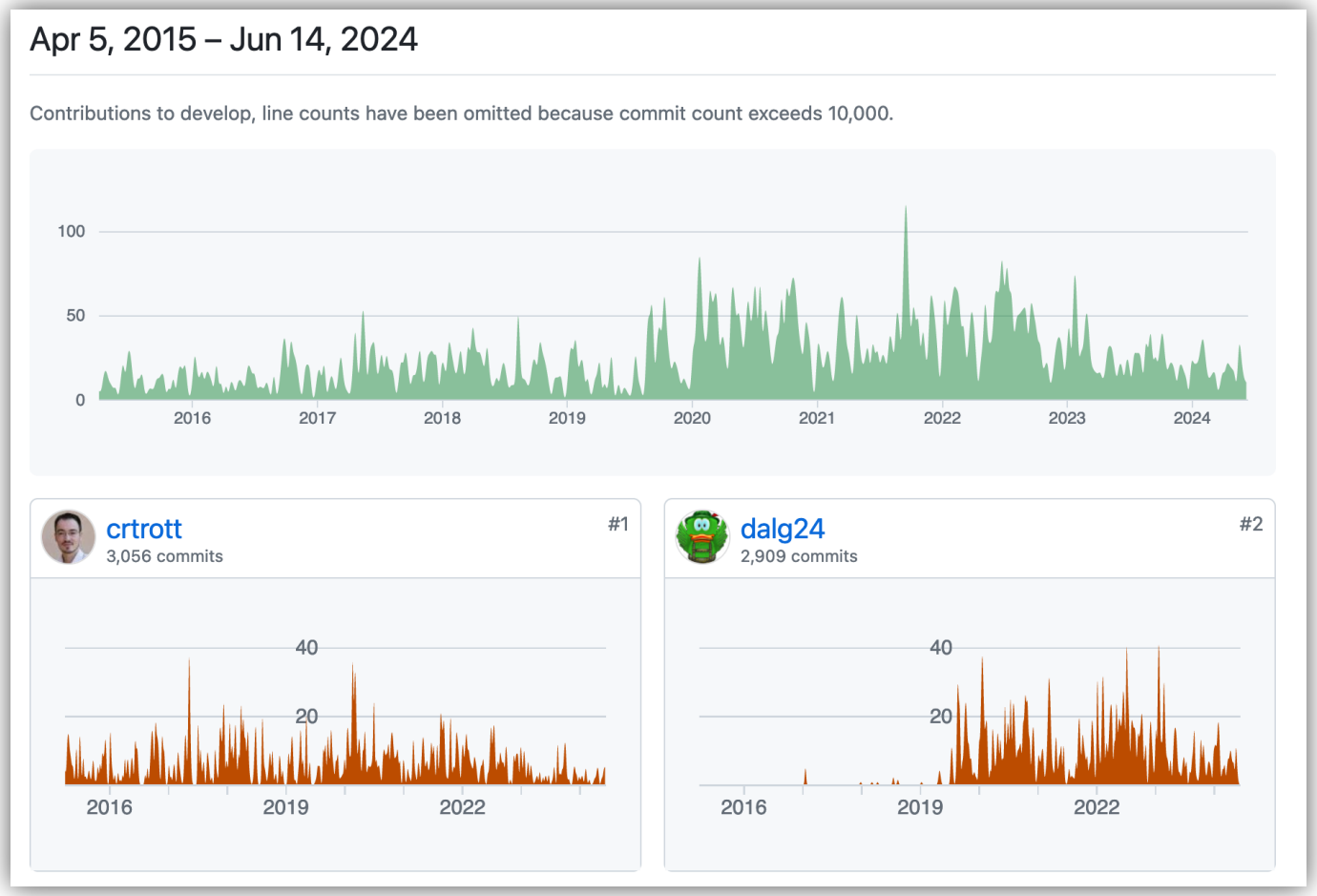
HPCToolkit

E4S

A worldwide community

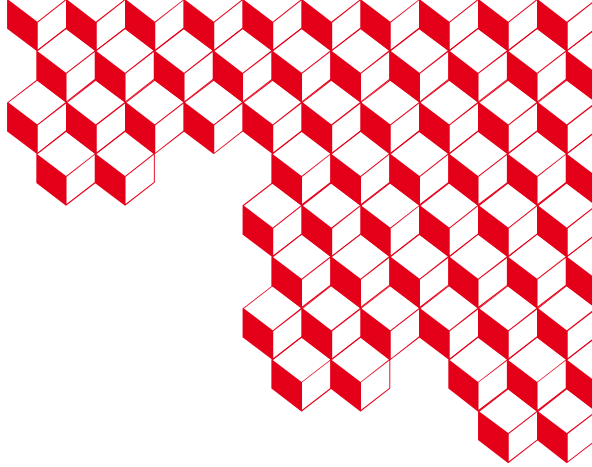
An active project for more than 10 years

- 1.8 K stars on GitHub
- 49 participants
- Many Slack channels to interact with developers and the community (even in French)
- https://kokkosteam.slack.com/



To conclude about Kokkos

- Kokkos enables Single Source Performance Portable Codes
- Simple things stay simple - it is not much more complicated than OpenMP
- Advanced performance optimizing capabilities easier to use with Kokkos than e.g. CUDA or HIP
- Kokkos provides data abstractions critical for performance portability not available in other programming models Controlling data access patterns is key for obtaining performance
- The Kokkos Ecosystem comes with tools (profiling, debugging, tuning, math libraries, etc.) needed for application development in professional settings



Thank you for your attention

Kokkos workshop – June 17, 2024