Kokkos, HPSF & CExA

Or: How We Learned To Stop Worrying And Love Productive, Performance-portable GPU Programming

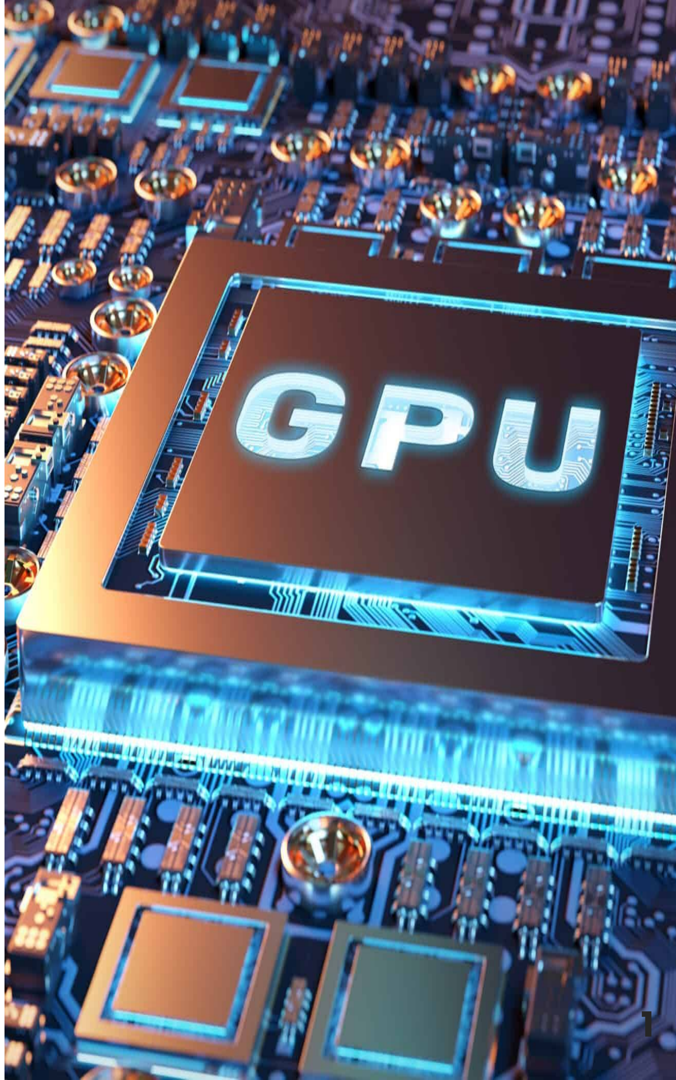
HPSF
HIGH PERFORMANCE SOFTWARE FOUNDATION

cexa-project.org

Computing at **Ex**ascale with **A**ccelerators at the **CEA**

P3HPC'24 Workshop
Atlanta
November 18th 2024

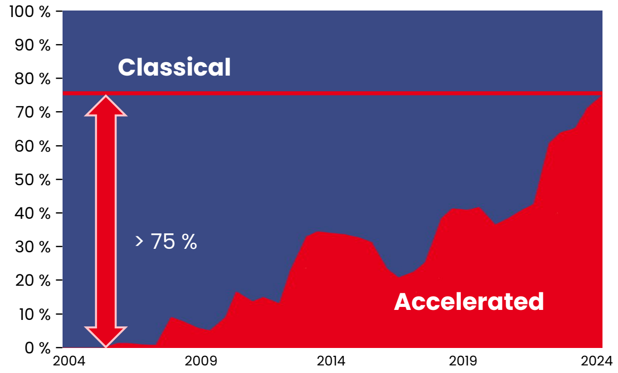
Julien Bigot, the CExA & Kokkos team



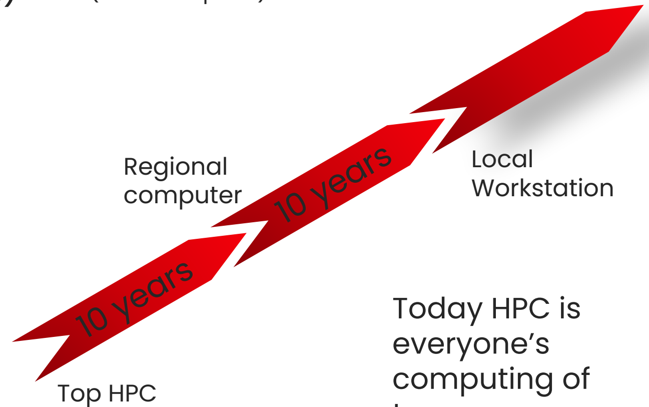
Context (2 years ago)

Come visit us at booth 4143

- **CEA**: French Atomic Energy Commissary ("French DoE")
  - Around 20k researchers, 9 research centers all over France
  - Organized in **4 divisions**: military applications (**DAM**), energies (**DES**), fundamental research (**DRF**) & technological developments (**DRT**)
  - **HPC** is a tool largely used **all over CEA**
- We just entered the **Exascale** era, that means **GPU**
  - **US Exascale**: **AMD** & **Intel**, **EU pre-Exascale**: **AMD** & **Nvidia**
  - 2 **Exascale** machines planned in **EU** for 2025
    - Jupiter machine in **Germany**, at Jülich => **Nvidia** + **SiPearl**(Rhea)
    - Jules Vernes machine in **France**, at CEA/TGCC (**open** call)
  - Need to re-develop applications with Performance portability
- GPU middleware: **software catalysts**
  - France and Europe: great research but no production tool
  - App developers are sitting on Buridan's ass
- A **need** for a long-term sustainable solution
  - Adapted to our hardware and software specificities
  - Trust in the roadmap

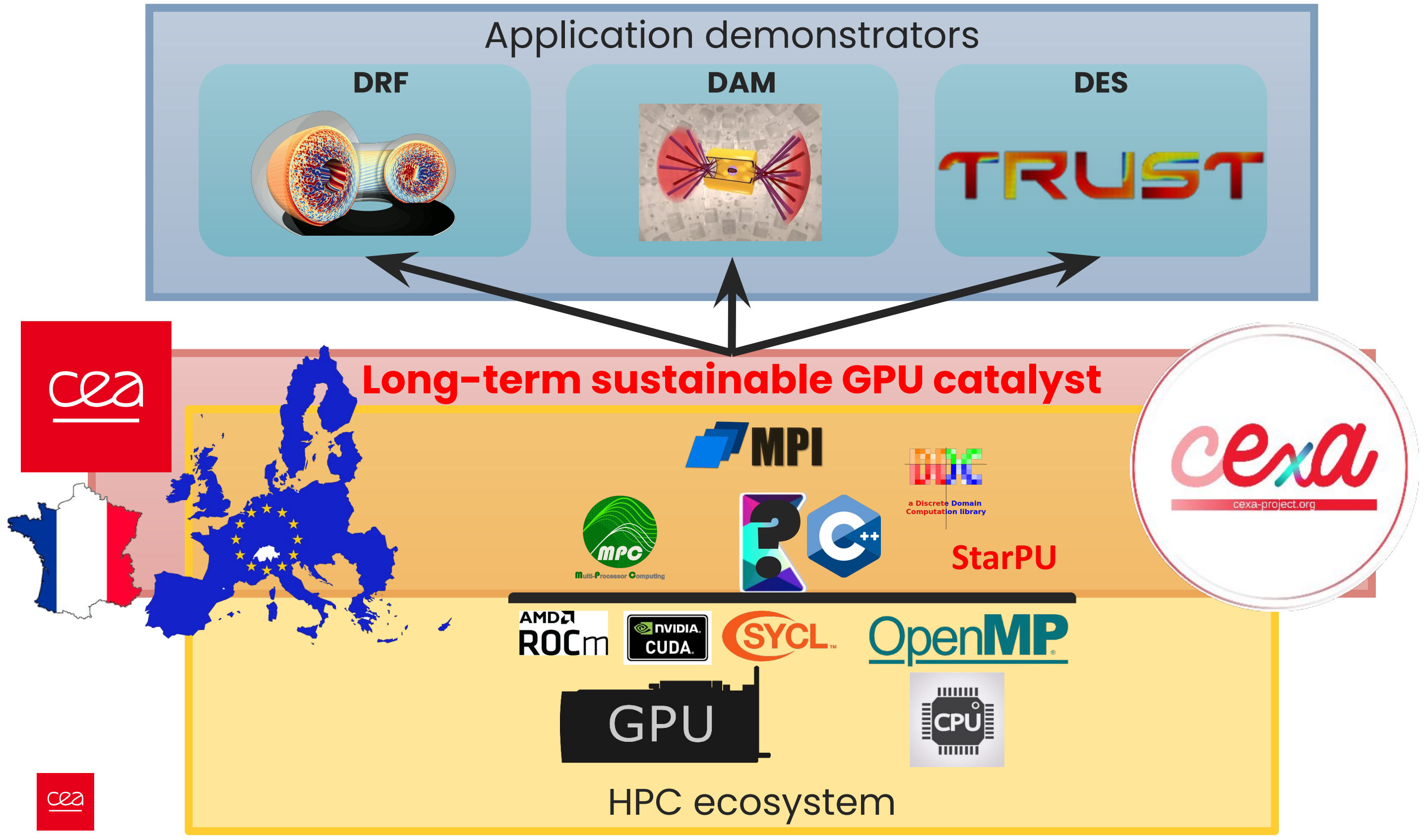


Computing power of the top 500 supercomputers from june 2004 to june 2024 (source Top500)



Today HPC is everyone's computing of tomorrow

CExA project: goals


Application demonstrators
DRF
DAM
DES
TRUST
Long-term sustainable GPU catalyst
MPI
a Discrete Domain Computation library
MPC
Multi-Processor Computing
StarPU
cexa
cexa-project.org
AMD ROCm
NVIDIA CUDA
SYCL
OpenMP
GPU
CPU
HPC ecosystem

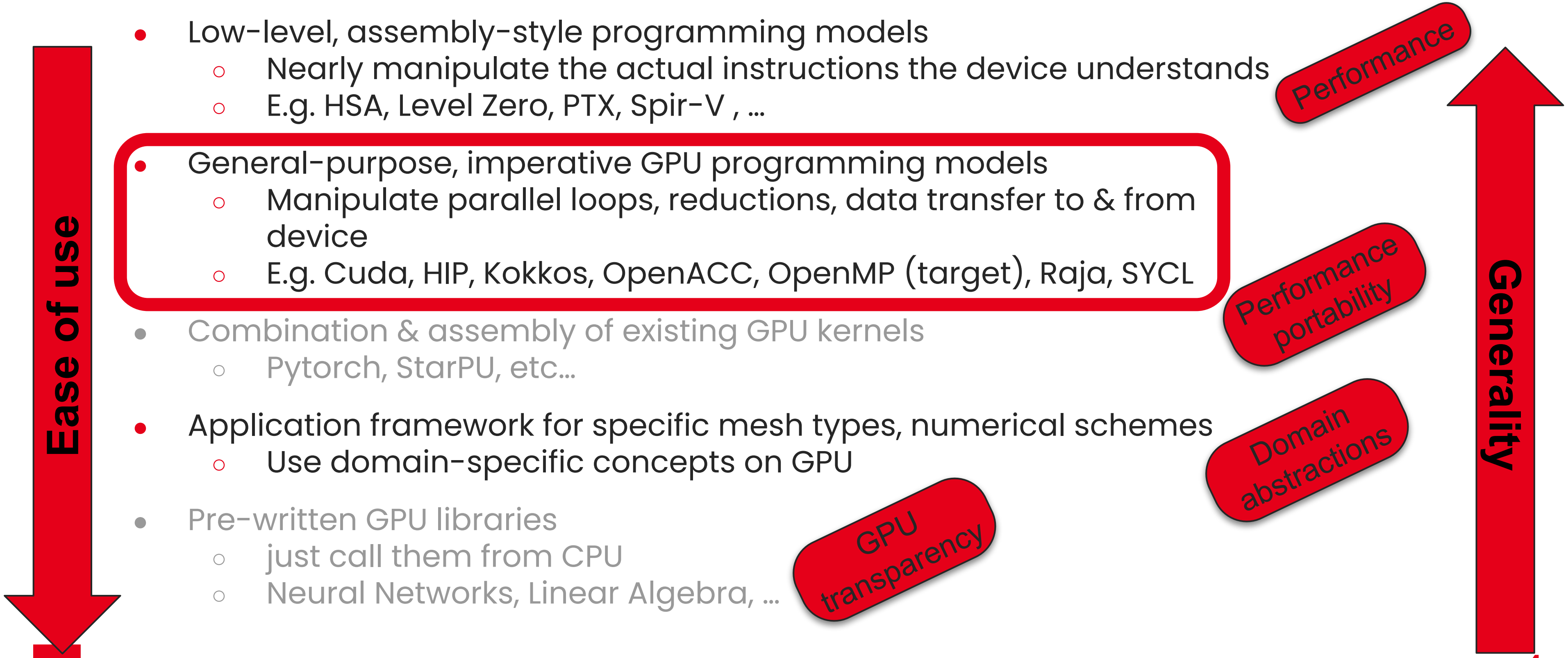
Disseminate and offer **training** at large

Adapt application demonstrators

Provide a long-term **sustainable** software **catalyst** for **GPU** computing

GPU programming, a vast choice of approaches

- Low-level, assembly-style programming models
  - Nearly manipulate the actual instructions the device understands
  - E.g. HSA, Level Zero, PTX, Spir-V , …
- General-purpose, imperative GPU programming models
  - Manipulate parallel loops, reductions, data transfer to & from device
  - E.g. Cuda, HIP, Kokkos, OpenACC, OpenMP (target), Raja, SYCL
- Combination & assembly of existing GPU kernels
  - Pytorch, StarPU, etc…
- Application framework for specific mesh types, numerical schemes
  - Use domain-specific concepts on GPU
- Pre-written GPU libraries
  - just call them from CPU
  - Neural Networks, Linear Algebra, …



Imperative GPU programming, a vast choice of approaches

- Cuda
- HIP
- Kokkos
- OpenACC
- OpenMP (target)
- Raja
- SYCL
    - OneAPI/DPC++
    - AdaptiveC++/OpenSYCL/hipSYCL

Imperative GPU programming, a vast choice of approaches

- Cuda
- HIP
- Kokkos
- OpenACC
- OpenMP (target)
- **Raja**
- SYCL
  - OneAPI/DPC++
  - **AdaptiveC++ (was OpenSYCL/hipSYCL)**

- **Production grade, with public support**

Imperative GPU programming, a vast choice of approaches

- **Cuda**
- **HIP**
- Kokkos
- **OpenACC**
- OpenMP (target)
- Raja
- SYCL
  - **OneAPI/DPC++**
  - AdaptiveC++/OpenSYCL/hipSYCL

- Production grade, with public support
- **Vendor neutral**

Imperative GPU programming, a vast choice of approaches

- Cuda
- HIP
- **Kokkos**
- OpenACC
- **OpenMP (target)**
- Raja
- SYCL
  - OneAPI/DPC++
  - AdaptiveC++/OpenSYCL/hipSYCL

- Production grade, with public support
- Vendor neutral

OpenMP & Kokkos : the simplest GPU loop

```
for (int j = 0 ; j < Nj ; ++j) {
        // [...]
}
```

Sequential

```
#pragma omp teams distribute parallel for
for (int j = 0 ; j < Nj ; ++j) {
        // [...]
}
```

OpenMP Target

```
parallel_for(Nj, KOKKOS_LAMBDA(int j) {
        // [...]
});
```

Kokkos

Execute in parallel, on a separate GPU thread each,

the same workload *[. . .]*

identified by a unique identifier `j`

`Nj` times between `0` and `Nj-1`

OpenMP & Kokkos : memory transfer

```
double* x = malloc(Ni*sizeof(double));
double* y = malloc(Nj*sizeof(double));
double* A = omp_target_alloc(
      Ni*Nj*sizeof(double),
      omp_get_initial_device());

#pragma omp target data \
      map(to: x[0:Ni]) \
      map(from: y[0:Nj])
{
#pragma omp teams distribute parallel for
for (int j = 0 ; j < Nj ; ++j) {
      for (int i = 0 ; i < Ni ; ++i) {
            y[j] += x[i] * A[j*Ni+i];
      }
}
```

OpenMP Target

```
View<double*, Kokkos::HostSpace> x(Ni);
View<double*, Kokkos::HostSpace> y(Nj);
View<double*> A(Nj, Ni);

{
auto dx = create_mirror_view_and_copy(dev, x);
auto dy = create_mirror_view(dev, y);
parallel_for(Nj, KOKKOS_LAMBDA(int j) {
      for (int i = 0 ; i < Ni ; ++i) {
            dy(j) += dx(i) * A(j,i);
      }
});
deep_copy(y, dy);
}
```

Kokkos

Copy x to GPU from device before kernel
and y from GPU to device after kernel
Keep A on the device

Compilation

Kokkos

- A C++ template library
  - No direct code generation
  - rely on vendors C++-like languages
- Multiple “backends”
  - Selection at compile time
  - OpenMP, Cuda, OneAPI, HIP, …
- Maximum 3 backends enabled at once
  - Serial backend
  - 1 Host parallel backend (openmp)
  - 1 Device parallel backend (cuda, HIP, Sycl)

OpenMP Target

- Use an OpenMP compiler
  - Compatible with the target construct
  - Compatible with the hardware you target
- Each vendor provides its own OpenMP compiler
  - Usually based on LLVM infra
- Default Clang/LLVM & GCC also try to support this
  - For some hardware

Imperative GPU programming, a vast choice of approaches

- Cuda
- HIP
- **Kokkos**
- OpenACC
- **OpenMP (target)**
- Raja
- SYCL
  - OneAPI/DPC++
  - AdaptiveC++/OpenSYCL/hipSYCL

- Production grade, with public support
- Vendor neutral
- Annotations
  - Works best with imperative languages: C, Fortran, …
  - Requires to re-design applications for GPU
  - Compiler integration: potential for additional optimizations
- Library
  - Suited to language with deep encapsulation: C++
  - Requires to re-design applications for GPU
  - On top of vendor backends: easier to port to new hardware

Kokkos parallel patterns

```
parallel_for(Nj, KOKKOS_LAMBDA(int j) {
      // [...]
});
```

Kokkos parallel patterns

```
parallel_for(Nj, KOKKOS_LAMBDA(int j) {
      // [...]
});
```

```
parallel_reduce(Nj, KOKKOS_LAMBDA(int j, double& accumulator) {
      // [...]
      accumulator += /* [...] */ ;
}, result);
```

```
parallel_scan(Nj, KOKKOS_LAMBDA(int j, double& result, bool isfinal)
{
      // [...]
      accumulator += /* [...] */ ;
      if(is_final) {
            // [...]
      }
}, result);
```

- For
  - independent iterations
- Reduce
  - Accumulate into a single value
- Scan
  - N independent prefix reduction

Kokkos parallel patterns: easy debug

```
parallel_for(“loop1”, Nj, KOKKOS_LAMBDA(int j) {
      // [...]
});
```

- Naming loops ease debugging & profiling
- Integrated with kokkos-specific tools
- Get a trace with names includes
- Get a name in debug messages
- Omitted in the presentation, but a good practice overall

Kokkos parallel patterns: Policies

```
parallel_for(RangePolicy(1, Nj, chunk_size), KOKKOS_LAMBDA(int j) {
        // [...]
});
```

Beyond simple 1D execution

- RangePolicy for 1D iteration
  - Begin / end iteration boundaries
  - Chunk_size hint for improved performance
- MDRange policy for multi-dimensional iterations
  - Multi-D begin / end iteration boundaries
  - Tiling hint hint for improved performance

Kokkos parallel patterns: ExecutionSpace

```
parallel_for(RangePolicy(DefaultExecutionSpace(), 0, Nj), KOKKOS_LAMBDA(int j) {
        // [...]
});
```

- ExecutionSpace defines where to run
  - `Cuda, HIP, SYCL, HPX, OpenMP, OpenMPTarget, Threads, Serial`
  - 3 exec spaces per execution max: Serial + parallel Host + parallel Device
- Choose where to run at compile time with a `#define`
  - Usually set from CMake
- 2 predefined aliases are often enough
  - DefaultExecutionSpace: parallel Device, or parallel Host, or Serial
    - Most of the time, this is the default
  - DefaultHostExecutionSpace: parallel Host, or Serial
    - When using host-only code

Kokkos parallel patterns: hierarchical parallelism

```
parallel_for(TeamPolicy(Nj, team_size), KOKKOS_LAMBDA(const team_handle& team) {
      // [...]
      parallel_for(TeamThreadRange(team, Ni, chunk_size), KOKKOS_LAMBDA(int i) {
            // [...]
      });
      // [...]
});
```

- Default loops can not be nested
- 2-level nesting is supported by teams of threads
    - Matches groups / threads support in GPU
    - But also available on CPU
    - Intermediate (scratch) memory allocation available

Kokkos parallel patterns are asynchronous

```
parallel_for(Nj, KOKKOS_LAMBDA(int j) {
        // [...]
});
parallel_for(Nj, KOKKOS_LAMBDA(int j) {
        // [...]
});
fence();
```

- Asynchronous execution
- Result visibility is only assured after a fence
- Or between kernels running on the same execution space

Kokkos views: multi-dimensional arrays

```
View<int**, MemorySpace> my_matrix("matrix", Nx, Ny);
```

- Multi-dimensional arrays
  - Type & dimensionality specified: `int**` => 2D integer array
  - Dynamic sizes are parameters: `Nx, Ny`
  - Static sizes are also possible: `int*[4]` => 2D array, 4 × dynamic
- Behaves like a C++ `shared_ptr`
  - Shared ownership with reference counting (like in python)
- With a name for debugging/profiling
- MemorySpace is part of the type, defaults should be used
  - `CudaSpace, CudaHostPinnedSpace, CudaUVMSpace, HIPSpace, HIPHostPinnedSpace, HIPManagedSpace, SYCLDeviceUSMSpace, SYCLHostUSMSpace, SYCLSharedUSMSpace,` **`HostSpace`**`,` **`SharedSpace`**`, SharedHostPinnedSpace`
  - Check of accessibility between MemorySpace & ExecutionSpace

Kokkos views copies & co.

```
auto dview = subview(oview, pair(start, end), ALL, slice_idx);
```

- Make a new reference to a subset of an existing view
    - Modifying the result modifies the source
    - pair: select a subrange, ALL: keep the dimension, integer: slice the dimension

```
void deep_copy(const ExecSpace &exec_space, const ViewDest &dest, const ViewSrc &src);
```

- Copy data between 2 views
    - Potentially on distinct memory spaces
    - An asynchronous operation

```
auto dview = create_mirror(mspace, a_view); // allocates & copy a new view of same size
auto dview = create_mirror_view_and_copy(mspace, a_view); // allocates & copy if necessary
```

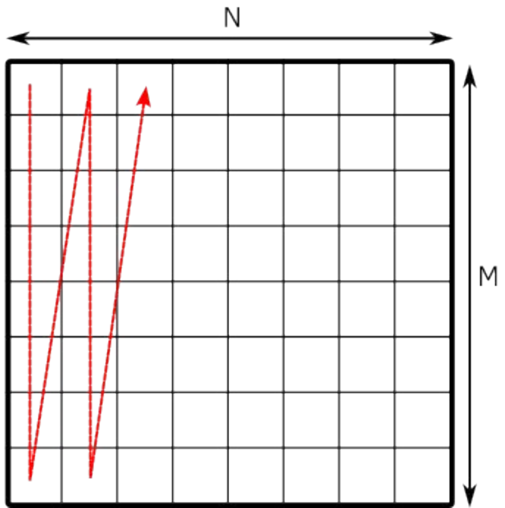
- Allocates & copy to a new memory space

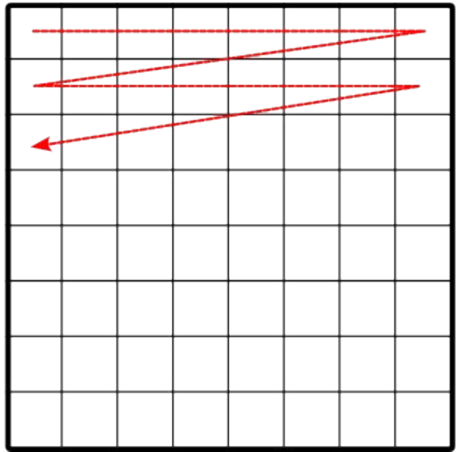
Kokkos views layout

```
View<double**, LayoutLeft> A("A", M, N);
```

- Layout specifies the linearization of multi-D indices into memory
  - `LayoutLeft` (a.k.a Fortran, default on GPU)
  - `LayoutRight` (a.k.a C, default on Host)
  - `LayoutStride` (generic, useful for subviews)

Layout left
Column major in 2D
(Fortran)
Device default layout





Layout right
Raw major in 2D
(C, C++, Python, Java)
Host default layout

What’s in Kokkos (core library)?

Multi-dimensional arrays

- Layout auto change for performance

Parallel patterns w. asynchronous support

- Independent interactions, Reductions, Scans

Iteration strategies

- Tiled, Hierarchical, …

What's in Kokkos (core library)?

Multi-dimensional arrays

- Layout auto change for performance

Other containers

- Key-value maps, ScatterView …

Automatic ref-counted Host/Device memory allocation & management

Host/device memory transfers

Support of "dual" arrays with one version on each side

- Up-to-date tracking & automatic transfers when required

Scratch memory

- Using "team-local" fast memory on the device

Parallel patterns w. asynchronous support

- Independent interactions, Reductions, Scans

Iteration strategies

- Tiled, Hierarchical, …

Algorithms

- Sorting
- Random number generation
- Many of STL parallel algorithms
- …

QoL features: portable printf, etc.

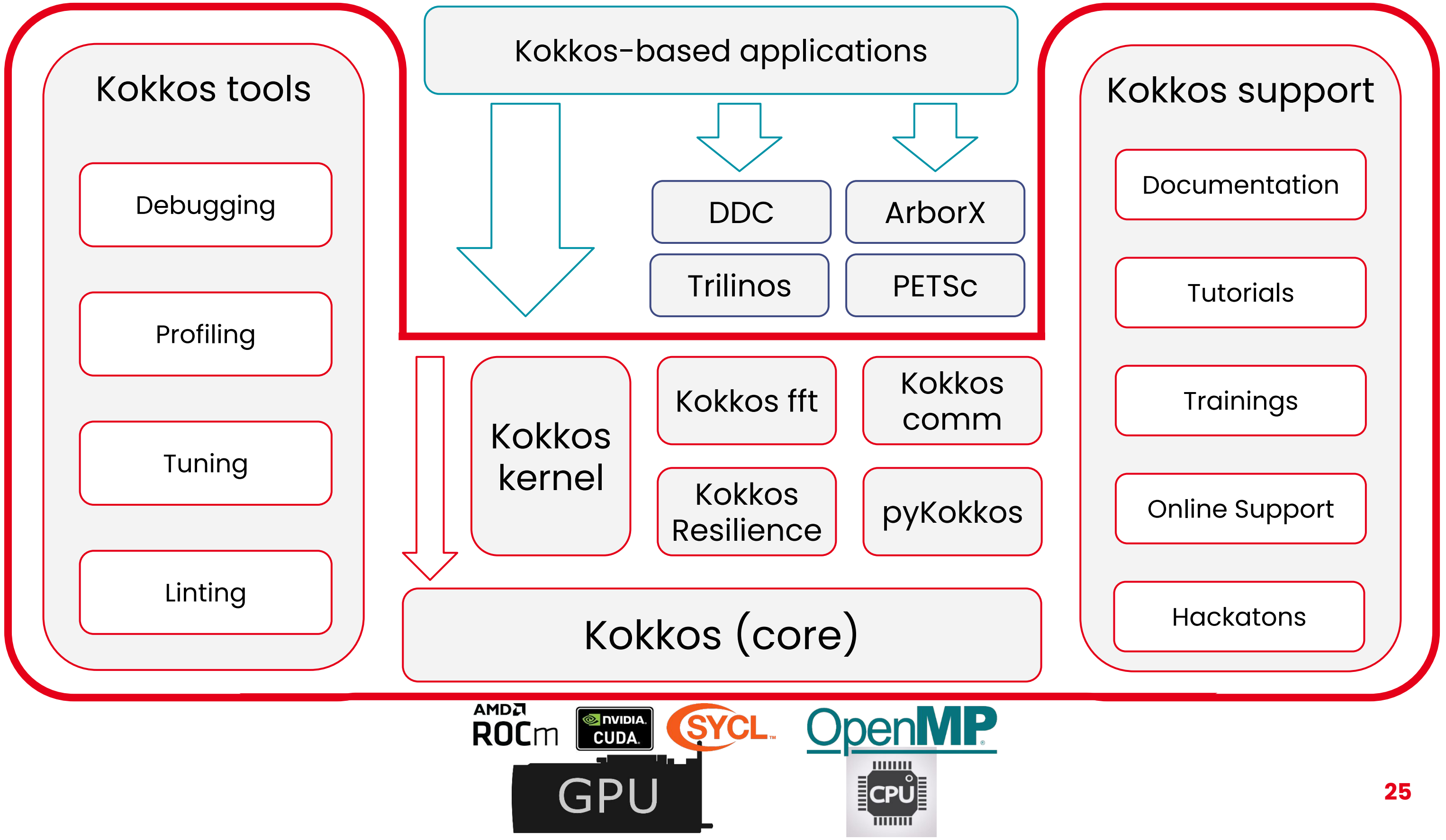
Portable atomic operations

SIMD

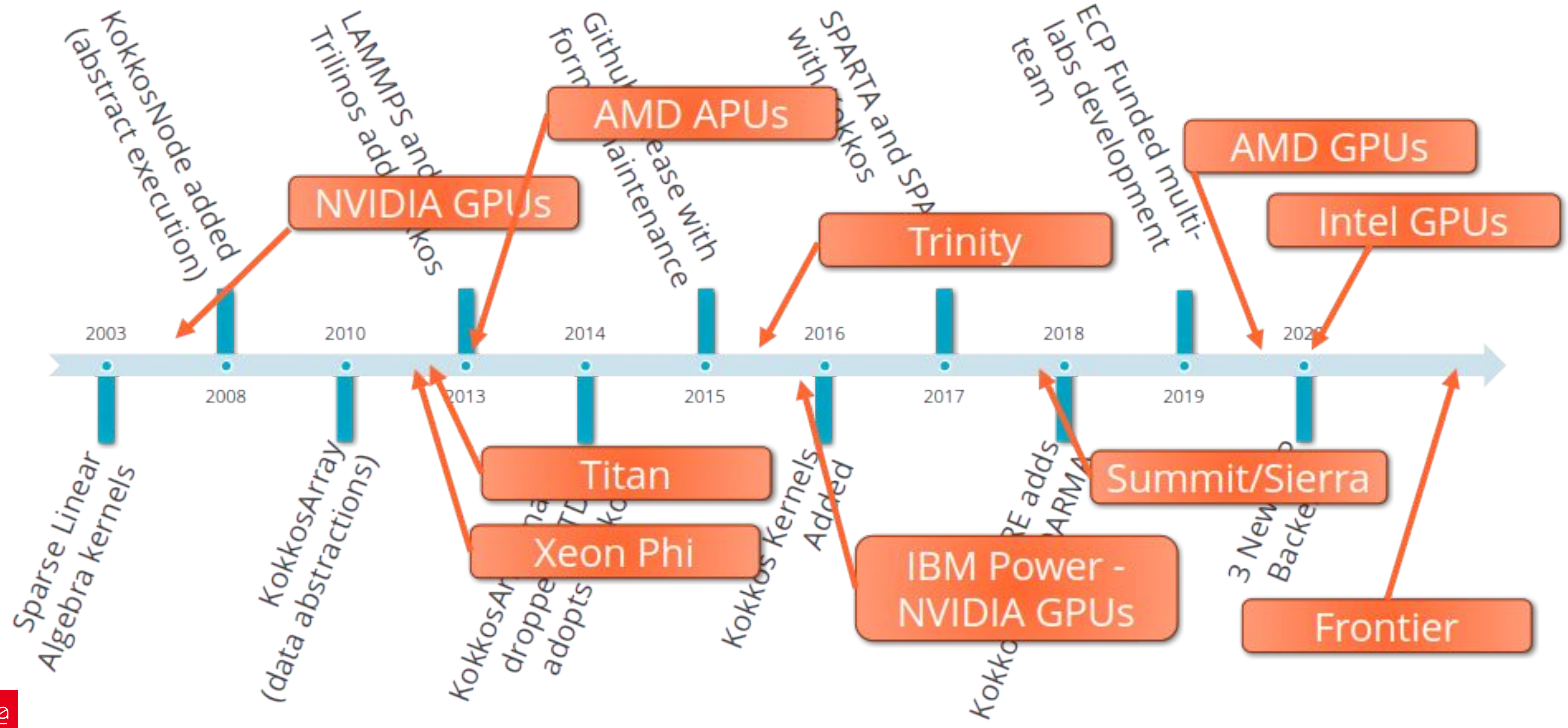
Coarse & fine-grain tasks

And much more…

Kokkos Ecosystem, beyond just the Kokkos project



Kokkos: a library with a history



KokkosNode added (abstract execution)
LAMMPS and Trilinos add Kokkos
Github release with formal maintenance
SPARTA and SPARC with Kokkos
ECP Funded multi-labs development team
2003
2008
2010
2013
2014
2015
2016
2017
2018
2019
2020
Sparse Linear Algebra kernels
KokkosArray (data abstractions)
KokkosArray dropped, adopts
Kokkos Kernels Added
Kokkos adds ARM
3 New Backends
NVIDIA GPUs
AMD APUs
Trinity
AMD GPUs
Intel GPUs
Titan
Xeon Phi
IBM Power - NVIDIA GPUs
Summit/Sierra
Frontier

Kokkos an anteroom for standard C++

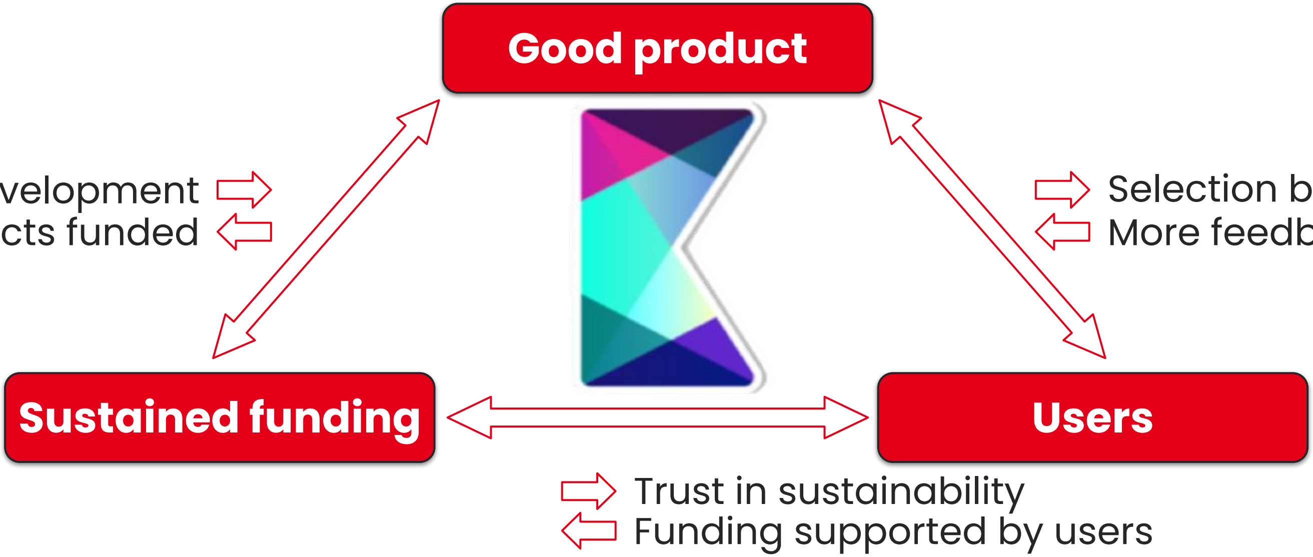
ISO C++ is **standardizing** base tools for **HPC**

- Parallel programming is entering the **ISO C++ language**
  - Parallel algorithms, sender/receivers, etc.
- The Kokkos team spearheads the standardization of many **features**
  - Multi-D arrays (`std::mdspan`)
  - Vectorization (`std::simd`)
  - Linear algebra (`std::linalg`)
  - And much more to come (mixed precision, etc.)

Kokkos offers a **stable API today** for the features of the **C++ of tomorrow**

- Standardization is slow (9 years for `mdspan`)
  - Consensus with all communities
- Kokkos offers the features **today**
  - And keeps maintaining a **stable API** on top of standardized ISO C++
  - With added interoperability layers (Cf. `kokkos::view` / `std::mdspan`)
  - And in a **GPU-compatible** implementation (Cf. `kokkos::array`)

Kokkos at the center of a virtuous cycle

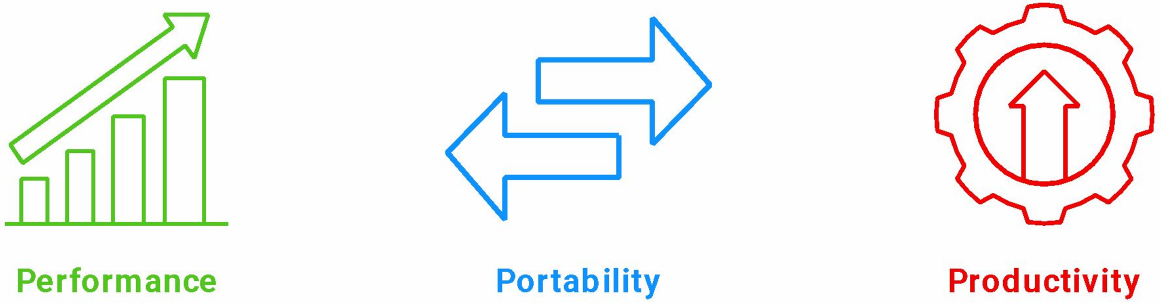


There is strength in numbers:
collaboration on core products is good for everyone

Here comes HPSF

Come visit us at booth 4648

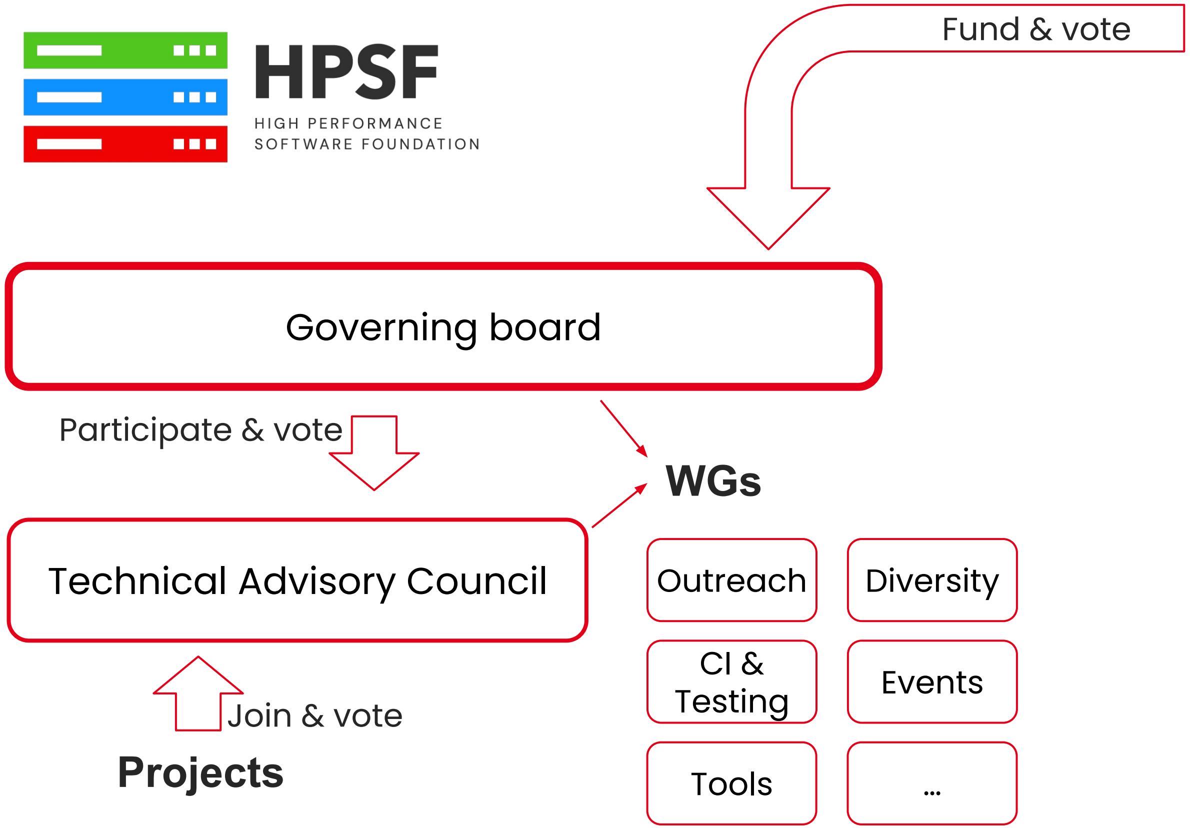
HPSF
HIGH PERFORMANCE SOFTWARE FOUNDATION



1. A neutral hub for open source, high performance software.
2. HPSF supports projects that advance portable software for diverse hardware by:
   - Increasing adoption
   - Aiding community growth
   - Enabling development efforts
3. Lowering barriers to productive use of today's and future high performance computing systems.

Under the Linux Foundation

HPSF
HIGH PERFORMANCE SOFTWARE FOUNDATION



Projects

Viskores
WarpX
HPCToolkit
Spack
TRILINOS
AMReX
Charliecloud
kokkos
APPTAINER
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

सी डैक CDAC
UNIVERSITY OF OREGON
UNIVERSITY OF MARYLAND

cea

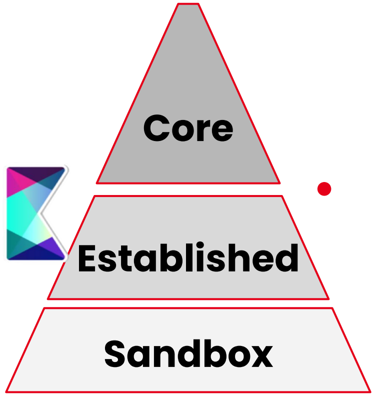
Two (independant) ways to participate

Find out more:
HPSF BoF,
Tuesday 5:15PM,
B309

- Joining as a member (for institutions)
  - You need to join the Linux Foundation (Non-profit/academic, as associate for $0)
  - Joining HPSF at one of three levels:
    - Premier: $175k / year
    - General: $2.5k - $50k / year depending on size of organization
    - Associate: $0 for non-profit / academic
  - Take a stand, fund it & get a say on where the funding goes to

- Joining as a project (for software project)
  - For the High Performance Computing ecosystem
  - That need a neutral home to facilitate multi-institutional collaborations
  - Providing vendor neutral solutions to engineering and science computational needs
  - Committed to building an open developer and user community

HPSF Software life-cycle



- **Core** projects have a **reliable** & **sustainable development** process
  - The developer base is strong and diverse
  - The funding sources are multiple
  - The governance is well specified
    - No single institution has a majority in the project lead
  - The project also fulfils all Established requirements
- **Established** projects are **production-strategic** for a **wide base of users**
  - The user base is wide and diverse
  - The development process is well documented and newcomers-friendly
  - The development is strong and steady
  - The project also fulfils all Sandbox requirements
- **Sandbox** projects are **free**, **open**, **neutral**, and **aim for the above**
  - Are free, libre, open-source HPC-related projects
  - With a code of conduct
  - And an aim to widen developer and user-base beyond a single institution

What kind of software is in HPSF so far?

Build & Deploy

- Build your software with tools that support all major computing architectures
- Deploy with cloud-ready packaging and container technologies on everything from your laptop to the largest exascale supercomputers

Develop & Sustain

- Leverage performance-portable software technologies
- Reuse high-quality scientific computing libraries including programming models, solvers, and visualization
- Foster community development for modeling and simulation applications

Analyze & Tune

- Profile your software with tools targeted at HPC environment
- Tune your software using information that connects performance data to how your software leverages HPSF projects

Spack WarpX AMReX E4S APPTAINER

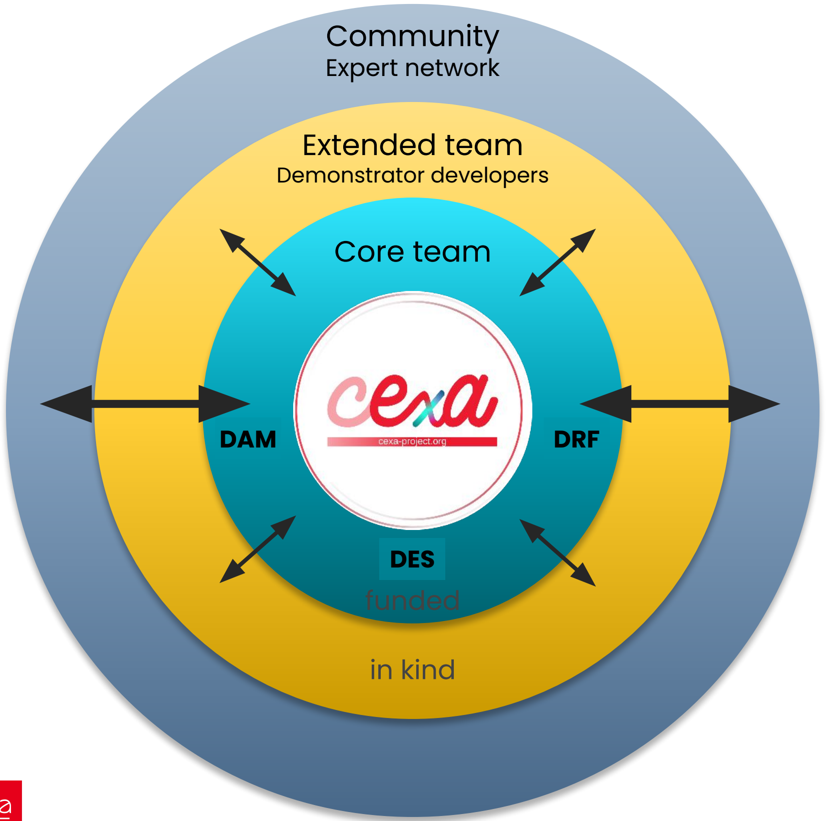
kokkos Charliecloud TRILINOS Viskores HPCToolkit

With CExA, CEA goes for Kokkos!

“adopt and adapt” strategy based on Kokkos

- Kokkos : a **strong technical basis**
  - A software architecture ready for the future
  - Mature, free, libre, and open-source
  - An **independent foundation** to own the product
    - HPSF under the Linux Foundation
  - A standardisation effort in ISO C++
    - A **stepping stone** one step ahead toward **HPC C++**
- Some **adaptations required**
  - For European **hardware**
    - There is no real hardware sovereignty without software sovereignty
  - For **applications** from CEA, France and Europe
    - Take our specificities into account

CExA project in practice



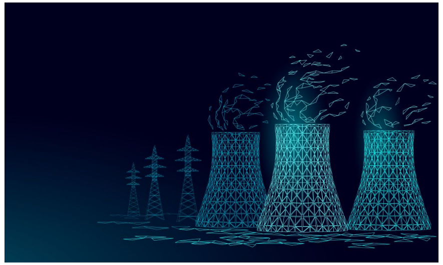
- **Core team**
  - Management, implementation and dissemination
  - Fully integrated in the Kokkos team
  - 13 researchers from all over CEA
  - 3 recrutements done, 5 more funded
  - Funding for 3 more hires expected next year
- **Extended team**
  - Demonstrator developers
    - Not funded
    - Find their own interest in the participation
  - 2-3 new demonstrators every 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

CExA: what's going on?

- Help with **documentation**
  - Website, Cheat-sheets, …
- **Trainings**, lots of training!
- **Support** our applications
  - Test **unified memory** viability & performance
  - Add required solvers to **Kokkos-kernels**
- Improve software **quality**
  - Work on **GPU CI**
  - Co-maintaining Kokkos **Spack recipes**
- Ease **code migration**
  - From **Fortran**
  - From **C (with classes)**
  - From **OpenMP (CPU)**

- Test **hardware** & improve kokkos for it
  - **Intel PVC** backend improvement
  - **Nvidia Grace Hopper** memory management handling
- Add **our contributions** to Kokkos ecosystem
  - **DDC**
    - Discrete data & computation
  - **kokkos-fft**
    - Performance portable FFT with a Kokkos API
  - **Kokkos-comm**
    - Message passing integrated with Kokkos

To conclude



- Kokkos is a strong vendor-neutral, performance portable Exascale programming model with GPU support
- CExA & HPSF ensure it is a sovereign and sustainable approach that can be relied on for the foreseeable future
- A strong dynamic all over the CEA and beyond
- A knock-on effect with new synergies identified every weeks with code developers

The core team

Julien Bigot
Principal investigator

Ansar Calloo
Group leader

Cedric Chevalier
Group leader

Mathieu Lobet
Group leader

Yuuichi Asahi
Senior developer

Rémi Baron
Senior developer

Thomas Padioleau
Senior developer

Paul Zehner
Developer

Hariprasad Kannan
Developer

The extended team

Pierre Ledac

Trust/TrioCFD lead

Virginie Grandgirard

GyselaX++ lead

François Letierce

Triclade lead

Julien Jaeger

DAM link

Édouard Audit

Network animator

Samuel Kokh

DES link

Patrick Carribault

DAM link

Join us & join the fun!

Come visit us at booth
4143: **CEA**
4648: **HPSF**

HPSF **BoF**, Tuesday 5:15PM, B309

2-years HPC DevOps Engineer position

Deployment and CI on supercomputers for the C++ Kokkos library within the "Moonshot" CExA project

CEA is recruiting DevOps engineers for a 2-year period to join the CExA "Moonshot" project team, which is setting up CEA's GPU computing software stack around the Kokkos C++ library, to contribute to innovative packaging, deployment and continuous integration approaches for supercomputers, based in particular on Spack. A team of more than 10 people is currently being set up. The positions will be based at the CEA Saclay site near Paris.

2-years C++ expert engineer position

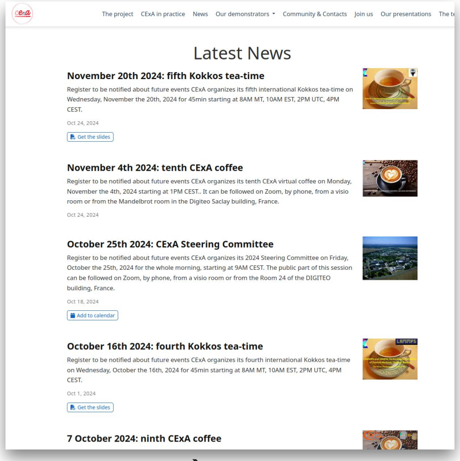
Contribution to the development of the Kokkos GPU computing library within the CExA "Moonshot" project

Join the CEA's ambitious "Moonshot" project, CExA, and contribute to the development of the Kokkos GPU computing library. We are recruiting six talented and enthusiastic C++ development engineers for a period of 2 years to work at our CEA Saclay site near Paris.

https://cexa-project.org

Kokkos training & community animation

- Many Kokkos trainings
  - September 2023 with C. Trott & D. Lebrun Grandié in Saclay
  - March 2025 Hackathon at IDRIS
  - September 2024 w. D. Lebrun Grandié & L. Berger-Vergiat
  - November 2024 Mission Numérique CEA in Grenoble
  - January 2025 CEA/Riken winter school in Barcelona
  - January 2025 Hackathon w.
  - February 2025 Mission numérique in Cadarache
  - Summer school 2025 w. EDF & Inria
- Kokkos slack now has a `#general-fr` channel (~10% of the whole community)
- CExA virtual café once a month
  - Informal presentations & discussions, in French about Kokkos, its ecosystem & GPU at large
- Kokkos virtual tea-time once a month
  - Informal presentations & discussions, in English about Kokkos, its ecosystem & GPU at large
  - With our US partners



Latest News

November 20th 2024: fifth Kokkos tea-time

Register to be notified about future events CExA organizes its fifth international Kokkos tea-time on Wednesday, November the 20th, 2024 for 45min starting at 8AM MT, 10AM EST, 2PM UTC, 4PM CEST.

Oct 24, 2024

Get the slides

November 4th 2024: tenth CExA coffee

Register to be notified about future events CExA organizes its tenth CExA virtual coffee on Monday, November the 4th, 2024 starting at 1PM CEST.. It can be followed on Zoom, by phone, from a visio room or from the Mandelbrot room in the Digiteo Saclay building, France.

Oct 24, 2024

October 25th 2024: CExA Steering Committee

Register to be notified about future events CExA organizes its 2024 Steering Committee on Friday, October the 25th, 2024 for the whole morning, starting at 9AM CEST. The public part of this session can be followed on Zoom, by phone, from a visio room or from the Room 24 of the DIGITEO building, France.

Oct 18, 2024

Add to calendar

October 16th 2024: fourth Kokkos tea-time

Register to be notified about future events CExA organizes its fourth international Kokkos tea-time on Wednesday, October the 16th, 2024 for 45min starting at 8AM MT, 10AM EST, 2PM UTC, 4PM CEST.

Oct 1, 2024

Get the slides

7 October 2024: ninth CExA coffee