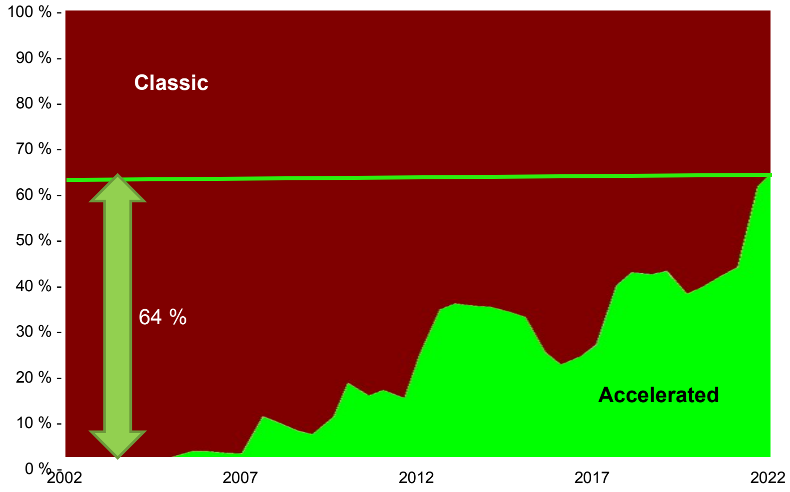
CExA project

Towards a Middleware to operate GPUs in Exascale context (Mission and Use Cases)

ADAC – 27 sept. 2023 – Fabien Baligand, CEA and Edouard Audit, CEA

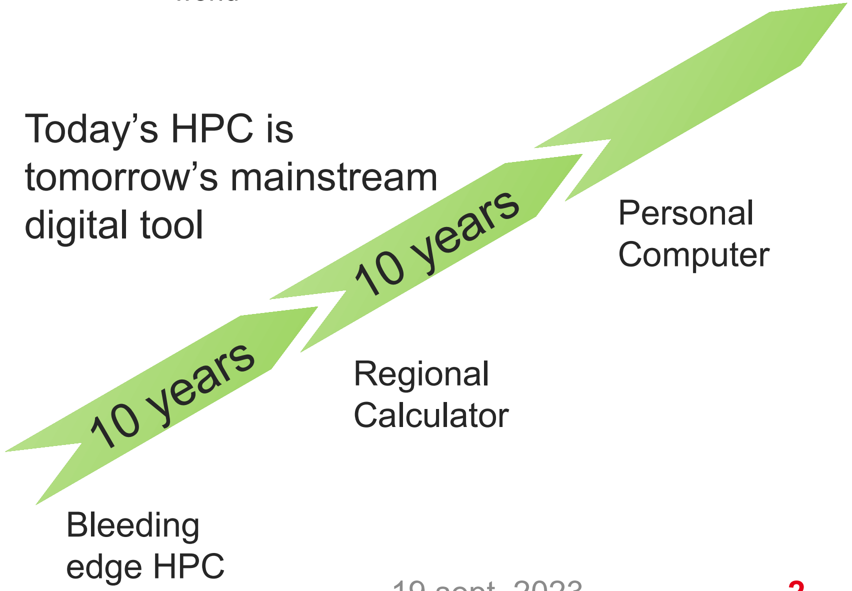
Context

- Intensive computing : a transversal tool for **sovereignty** and **competitivity**
  - Numeric twins, climte modelling, nuclear dissuasion, physics at extreme scales, multi scale design of materials, personnalized medecine, privacy, etc.
  - Infused in society, transversal to all directions of CEA
- **Exascale** era is starting (1st machine this year)
  - Accelerated architecture (GPU)
  - First supercomputers coming to Europe in 2024-2025
    - An Exascale computer in France at CEA/TGCC
  - Need to revelop applications to benefit from it
- GPU Middleware : **catalysts**
  - Performances portability
  - In the US : driven by the *Exascale Computing Project* (ECP) ⇒ Kokkos
    - OSS strategy to ease transfert to industries
  - In Europe and in France : research, but no technology yet
- **Yearning** for sovereign solution
  - Control roadmap, adapt to our specific needs (HW and SW)

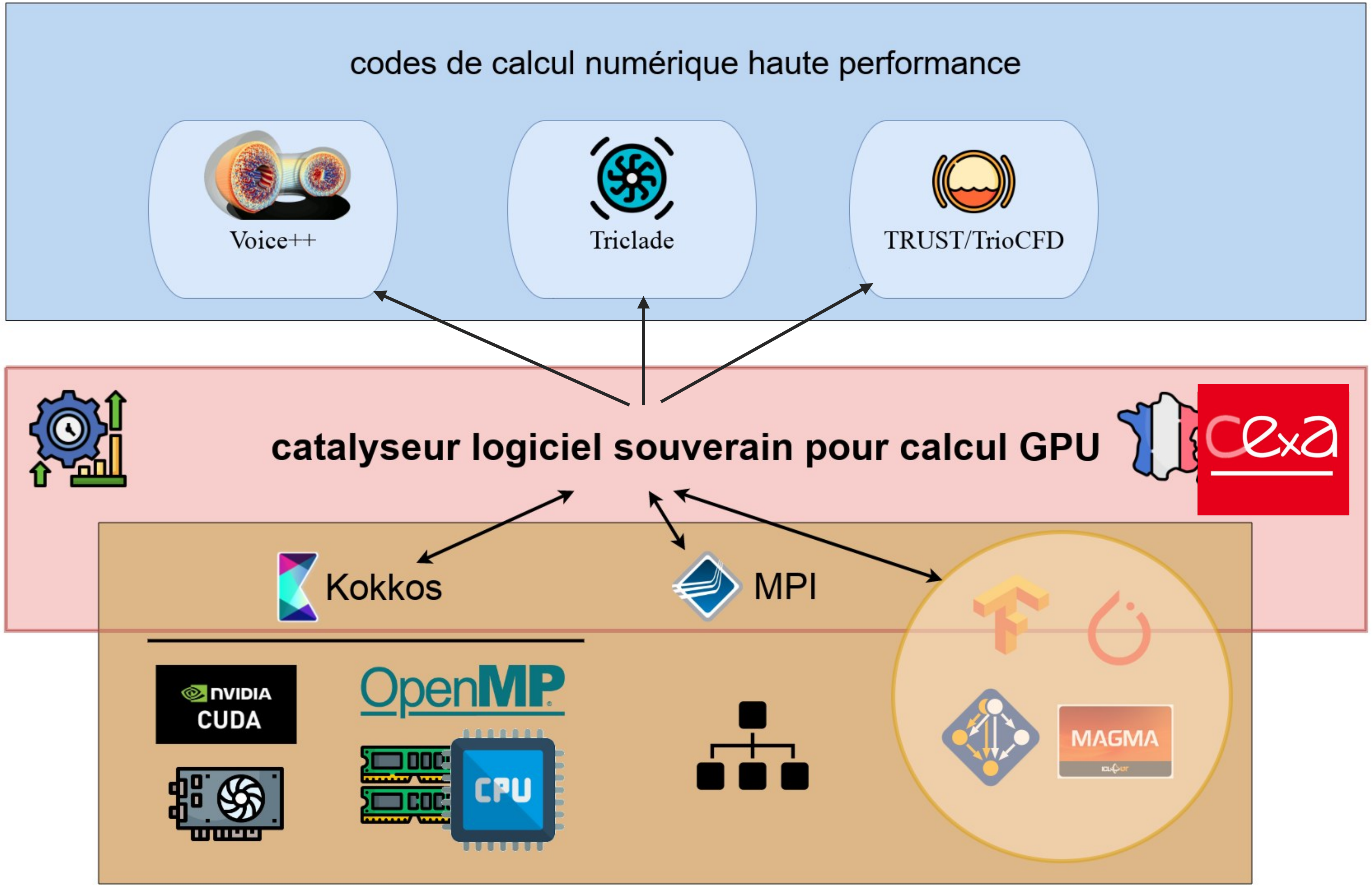


Computing power of the top 500 supercomputers in the world

Today's HPC is tomorrow's mainstream digital tool



The project



Communicate and Train to CExA inside CEA and outside

Adapt Use Case integrating CExA

Build a sovereign GPU computing software catalyst : CExA

CExA briefly

"adopt and adapt" Strategy based on Kokkos

- Kokkos : a **powerful platform**
  - Mature, free and open source
  - An architecture for performance portability
    - Ready for future machines
  - An integration step towards standard C++
    - Springboard towards standard C++
    - A preview of parallel C++

- Some **required adaptations**
  - For European hardware
    - No hardware sovereignty without software sovereignty
  - For CEA and European applications
    - Take specificities into account

Adequation to « distributed memory » applications

- Performance portability MPI+Kokkos CPU & GPU
- Efficient memory transfer
- Supports for GPU virtualization

Interface with 3rd party data processing tools

- Interface with Pytorch, Tensorflow
- Linear algebra batching

Heterogeneous architecture support

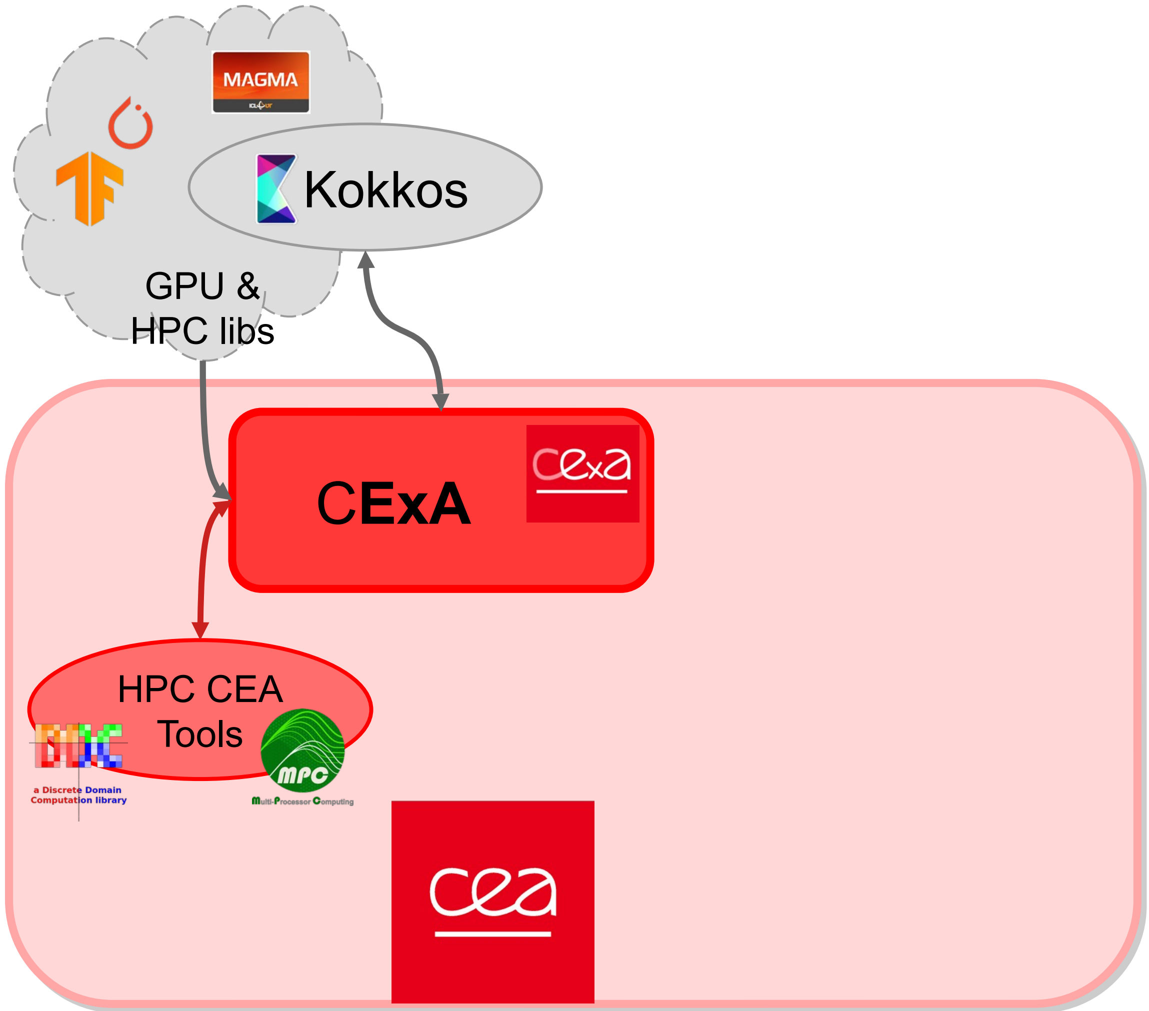
- Multi-architectures code support
- Multi space execution support
- EPI processors support

Ease of deployment on our computers

- Multi-device deployment management
- Continuous Integration and installation on our computers

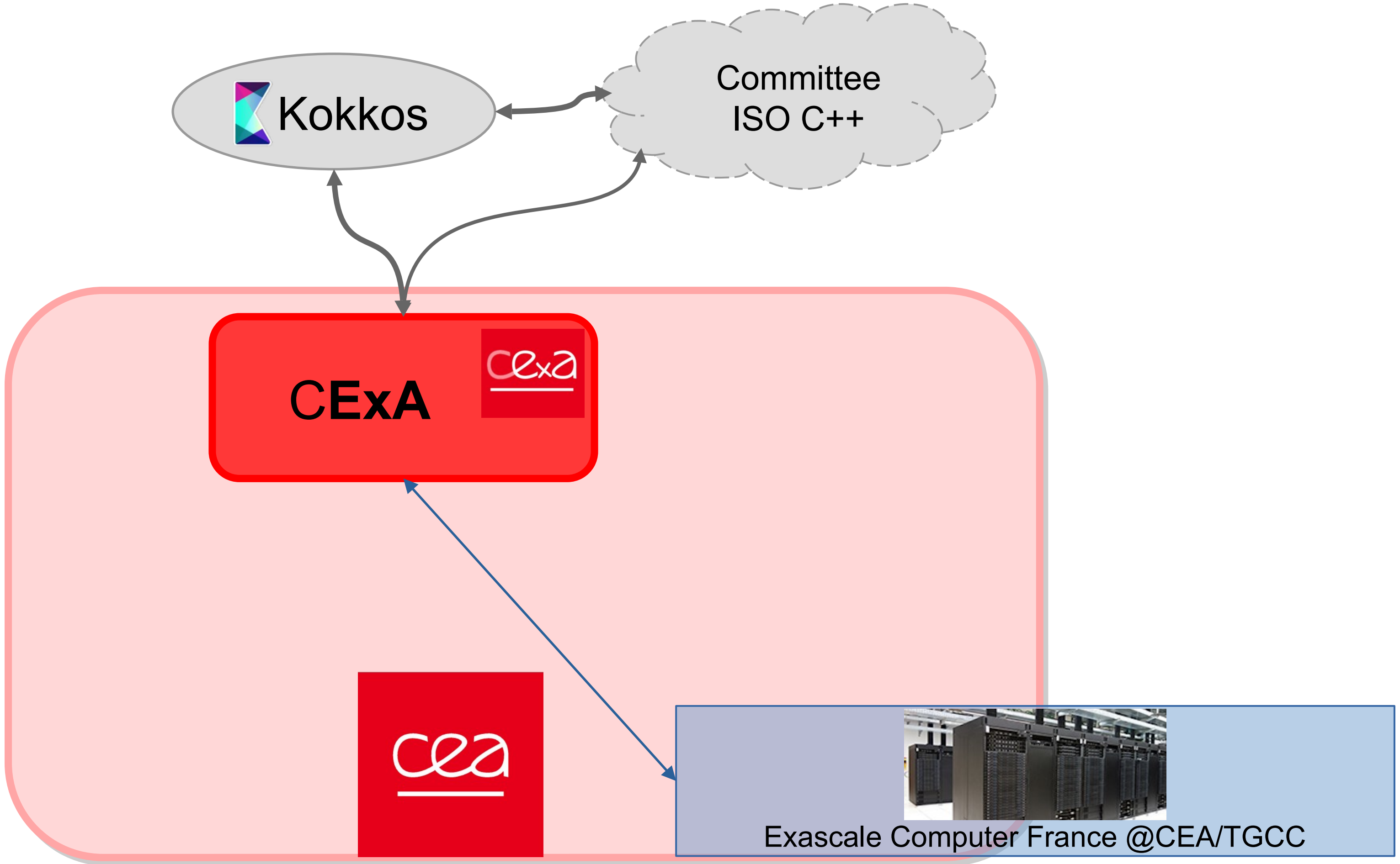
Hardware specificities | **Software specificities**

CExA Ecosystem: Upstream



- GPU & HPC Libraries
  - Tensorflow, Pytorch, MAGMA, etc.
  - Interface enabler with Open source software
- Kokkos development team
  - Strong connections
  - Here today (and at CEA last week)
- HPC CEA Libraries
  - MPC, DDC, Arcane, etc.
  - Integration and communications

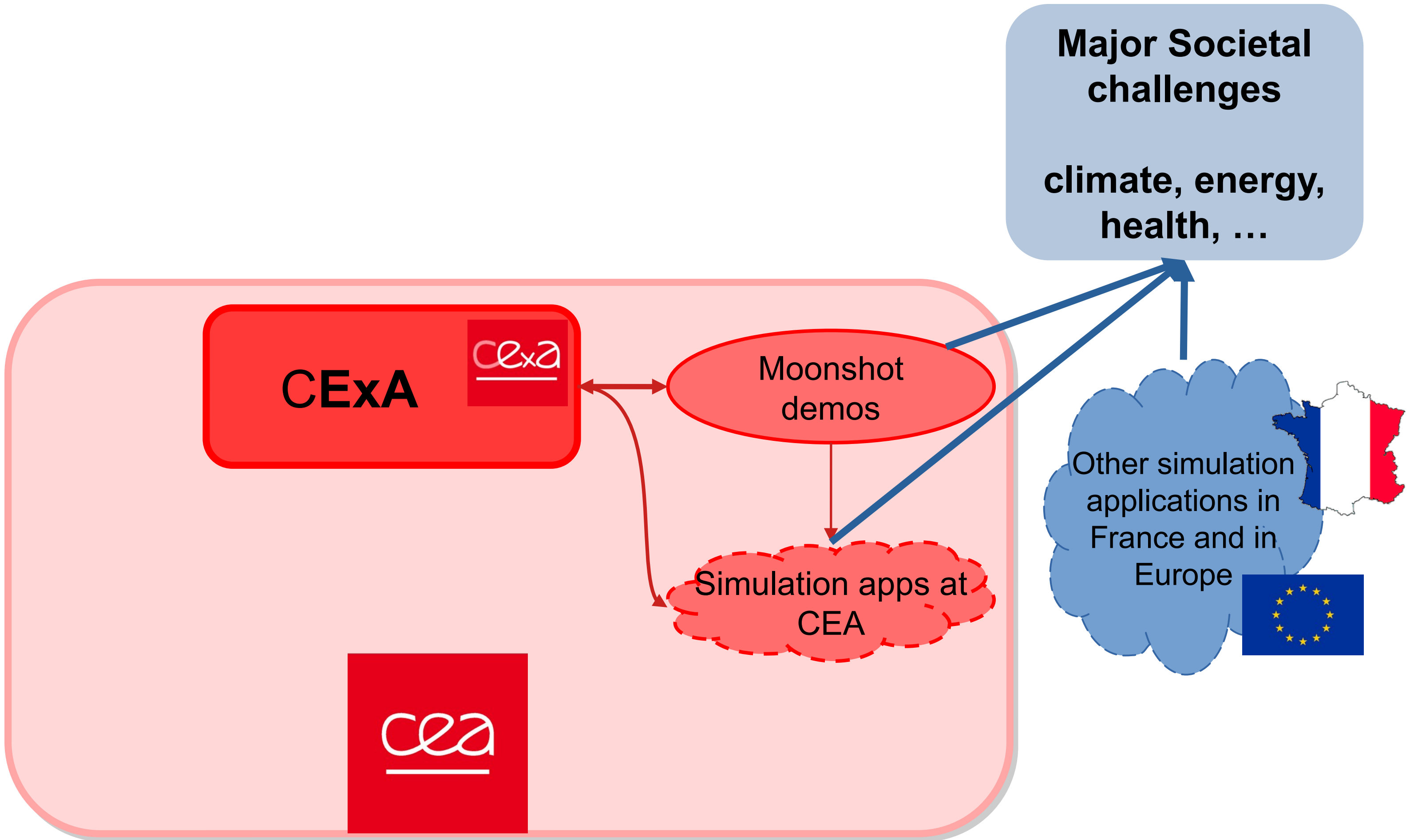
CExA Ecosystem: Partners



- Kokkos & ISO C++ committee
  - Standardization
    - Through Kokkos
  - Normalization et perennisation of CEA approaches

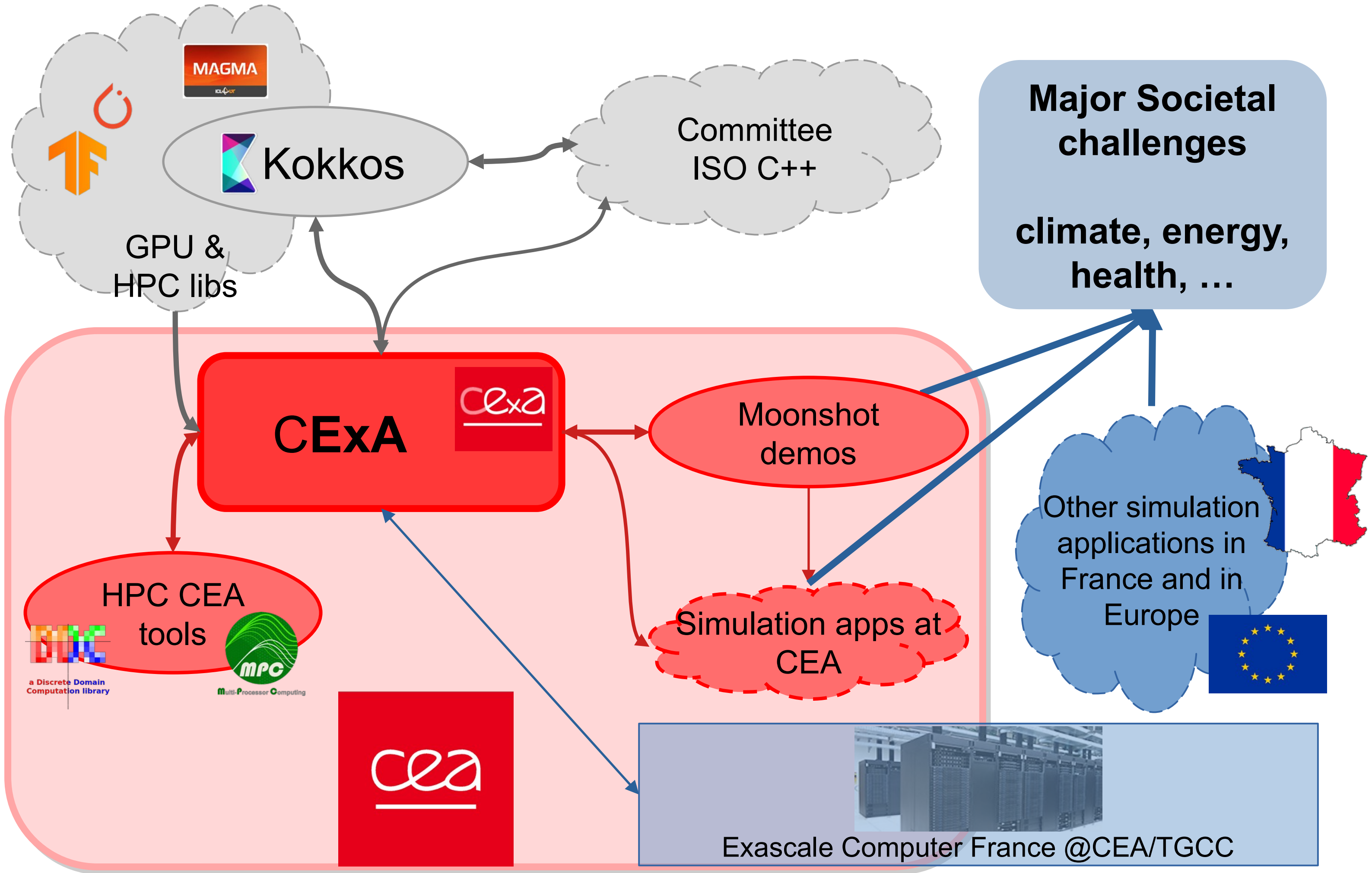
- Jules Vernes Project (Exa France)
  - Strong connections with GENCI, TGCC and NumPEx
  - AAP end of 2023
    - CExA requirements
  - Answer in 2024
    - Choice of architecture
  - Ship end of 2025
    - CExA production ready

CExA Ecosystem: Downstream



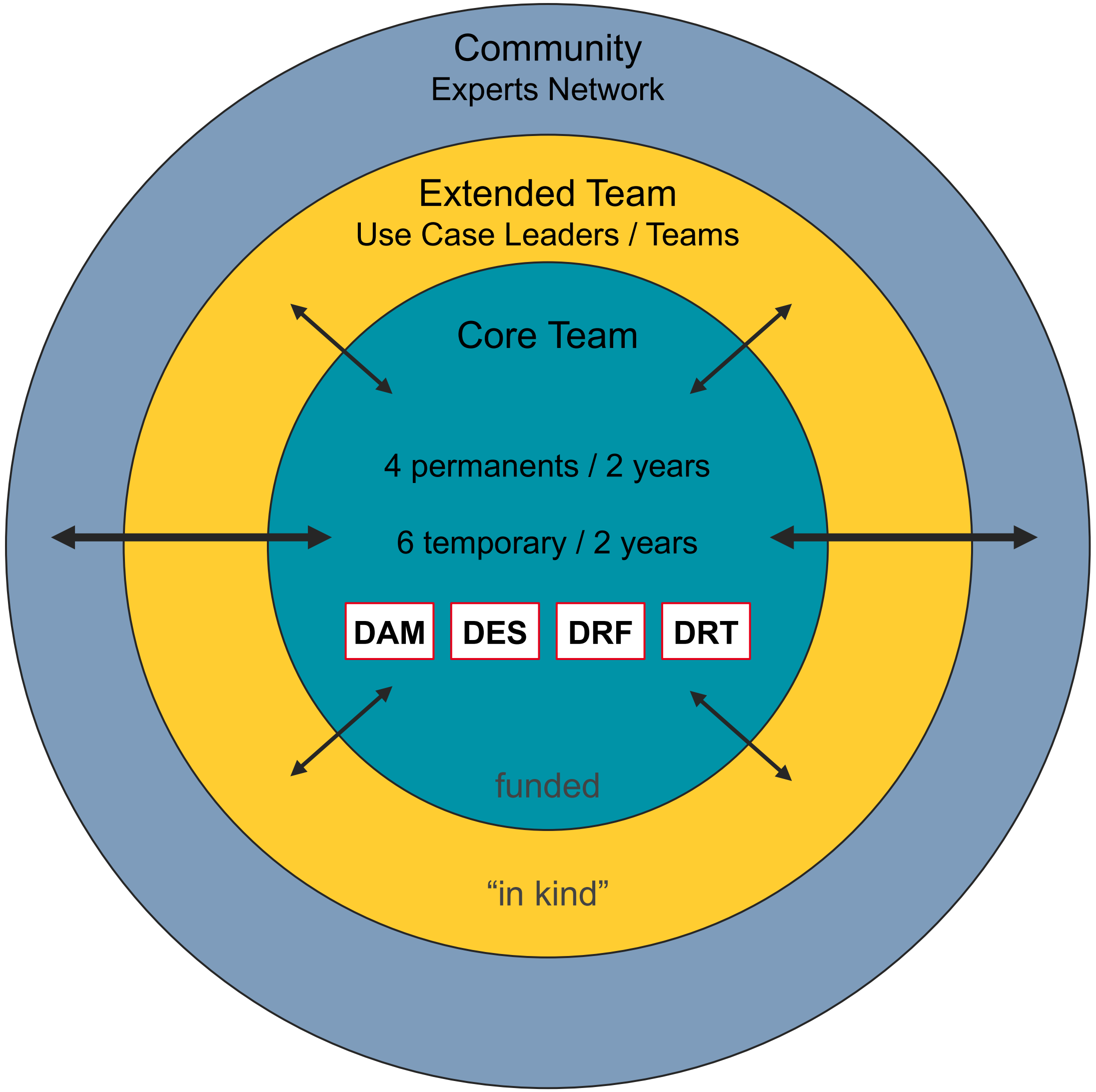
- Two layer downstream
  - Acceleration layer ⇒ applications
  - Second layer ⇒ societal challenges
- Integrated use cases
  - Co-design teams
  - Team training
  - Outcomes in critical domains
- CEA Applications
  - Training, hackathons, experience
  - Dynamic, opiniated
  - Community creation
  - CEA contributions ⇒ societal challenges
- FR and EU communities
  - CEA Visibility and role

CExA Ecosystem: Follow up



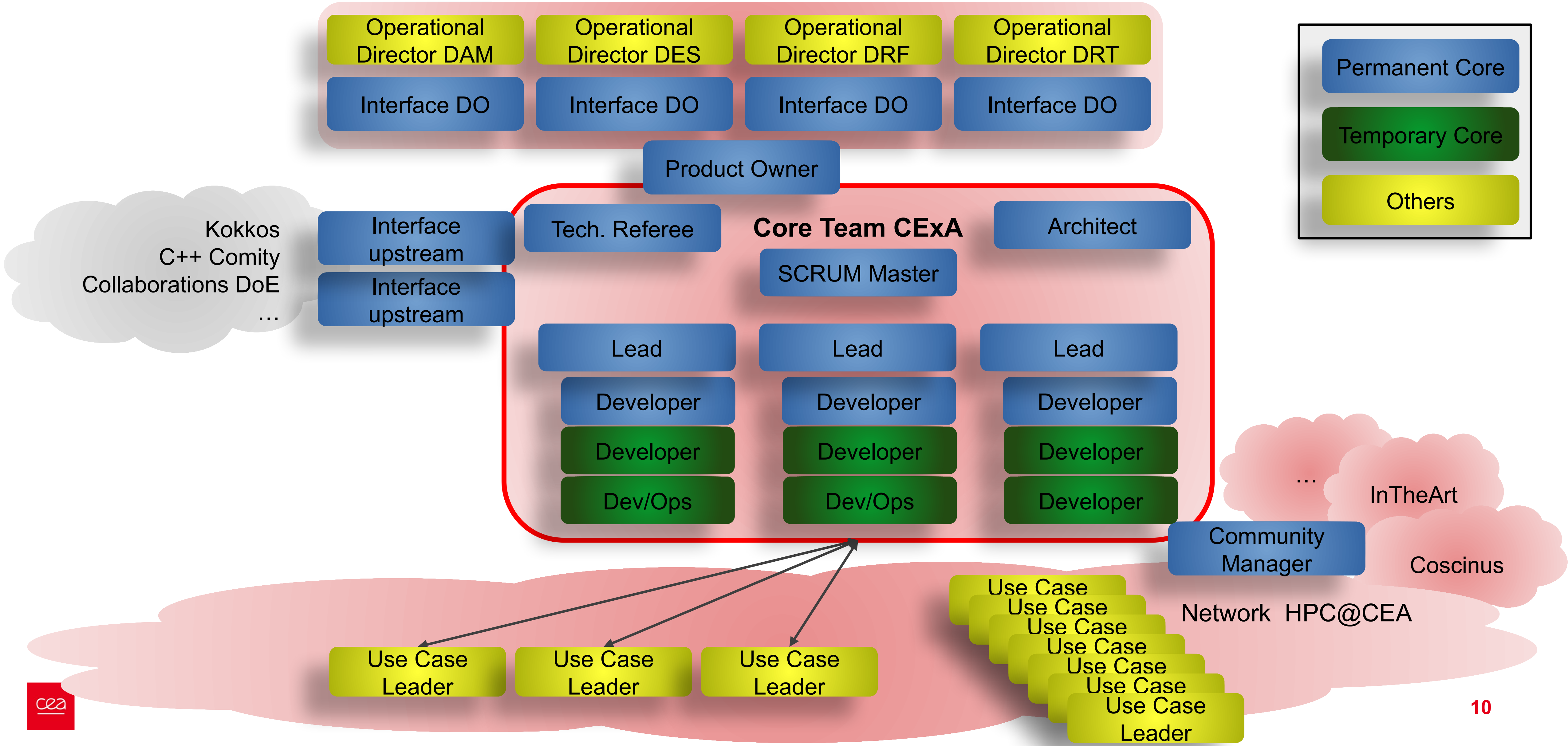
- Preparation of tooling for numerical computing on GPU
  - After graphism (1990's)
  - After neural networks (end of 2000)
- At the heart of the stack
  - Expertise on tooling
  - Bleeding edge
  - Suiting roadmap
- A unique competitive edge for years to come

Team Structure

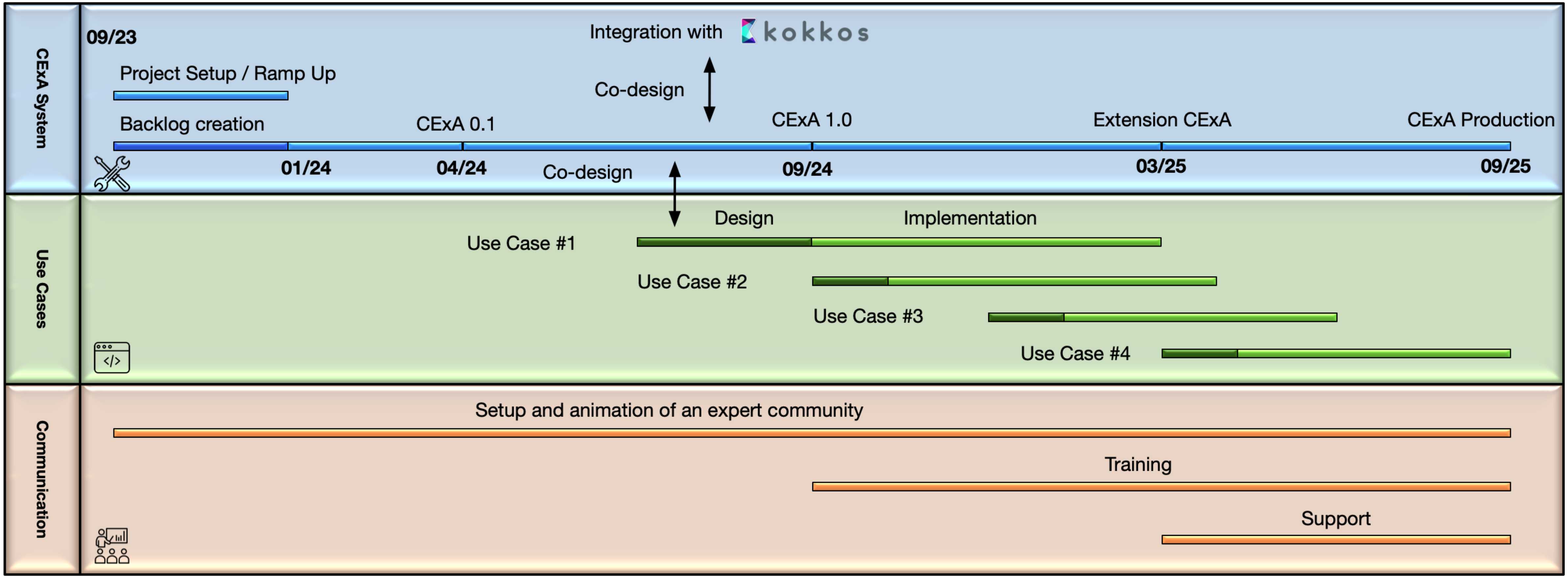


- Core Team
  - Management, Implementation and Dissemination
  - 4 permanent funded for 2 years
  - 6 temporary funded for 2 years
  - team organical growth
- Extended Team
  - Include use case leaders
  - 1 per DO (“in kind”)
  - ~6 months
- Community
  - Expert network federation
  - CExA co-design:
    - Backlog creation
    - CExA integration into applications
  - Dissemination targets
  - Work durability

Agile execution



Planning

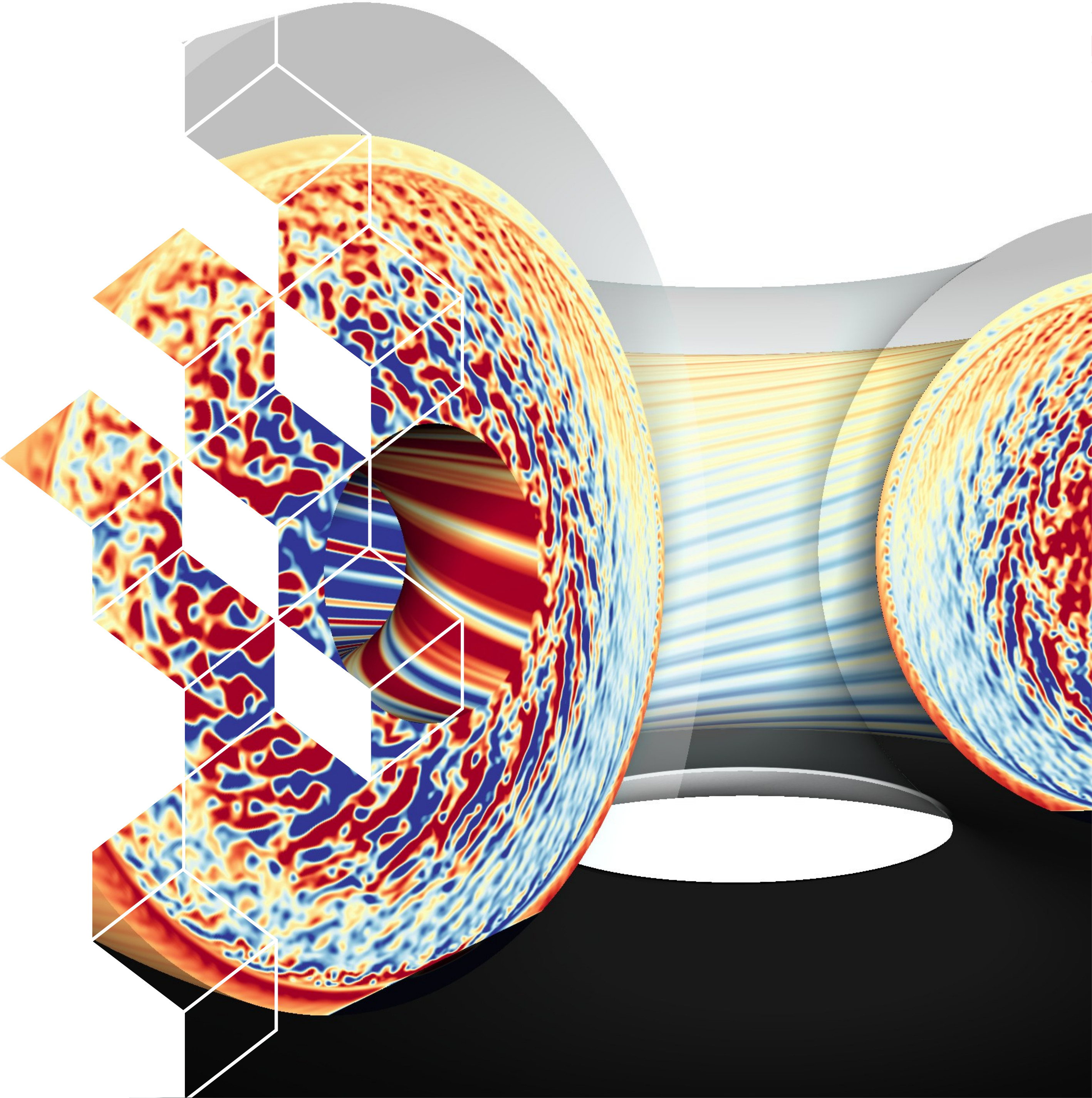

CExA System
09/23
Integration with kokkos
Project Setup / Ramp Up
Co-design
Backlog creation
CExA 0.1
CExA 1.0
Extension CExA
CExA Production
01/24
04/24
Co-design
09/24
03/25
09/25
Use Cases
Design
Implementation
Use Case #1
Use Case #2
Use Case #3
Use Case #4
Communication
Setup and animation of an expert community
Training
Support

Initial Backlog Epics

- Introduce physical variables management to write more robust simulation applications
- Facilitate port legacy of applications to accelerators hardware (GPU)
- Offer support to advanced and state of the art 3rd party functions/libraries (each vendor has its own library, plug to the right library via Kokkos level interfaces/adapters)
- Make full use of current and future European Exascale architectures
- Extend programming model to cover more usage scenarios
- Improve scientific applications Development by introducing Continuous Integration Facility
- Use Cases improvements (KPIs)
- Support CEA Technical Community

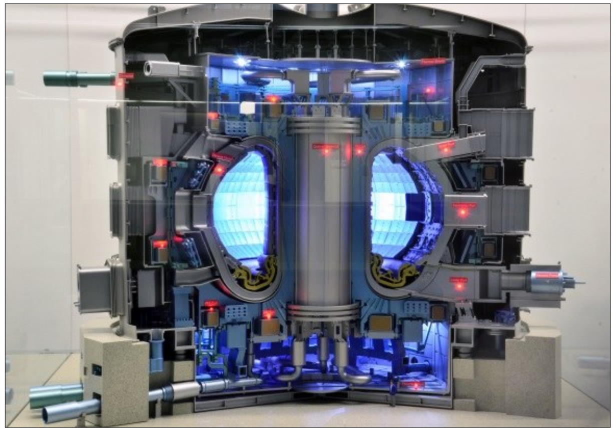
GyselaX++

Exascale Challenges for tokamak plasma turbulence simulations

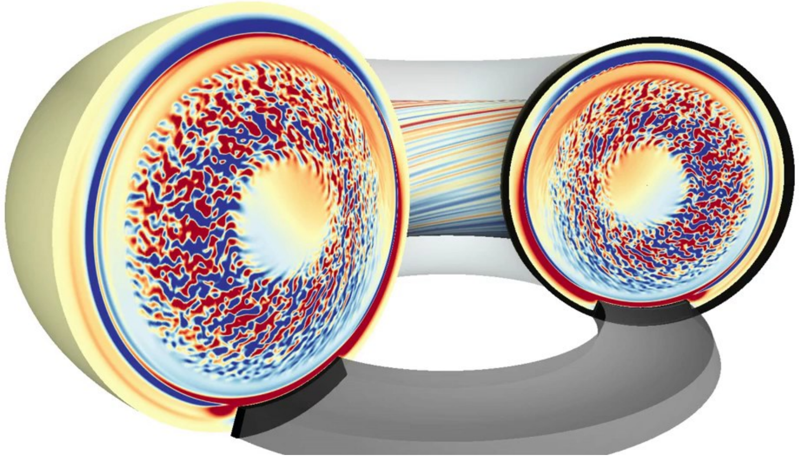


First principle simulations required for ITER

⇒ Gyrokinetic plasma turbulence simulations



ITER project

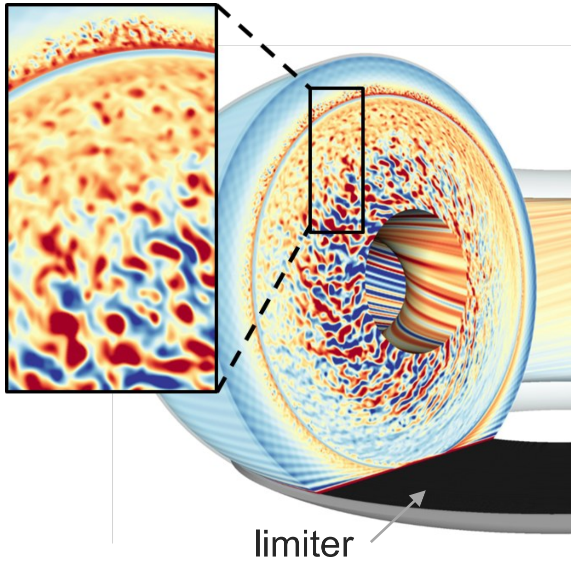


GYSELA simulation

- To optimize performance and minimize risks, each ITER scenario will have to be numerically validated.
- A complete chain of numerical tools will be required, ranging from scale models, which can be used in real time, to first-principles simulations, which are more costly but more reliable.
- Turbulent transport mainly governs confinement in Tokamaks
- Tokamak plasmas weakly collisional ⇒ Kinetic approach mandatory
  - Fusion plasma turbulence is low frequency ⇒ fast gyro-motion is averaged out
  - Gyrokinetic approach: phase space reduction from 6D to 5D

GYSELA: a highly parallelised code running at petascale

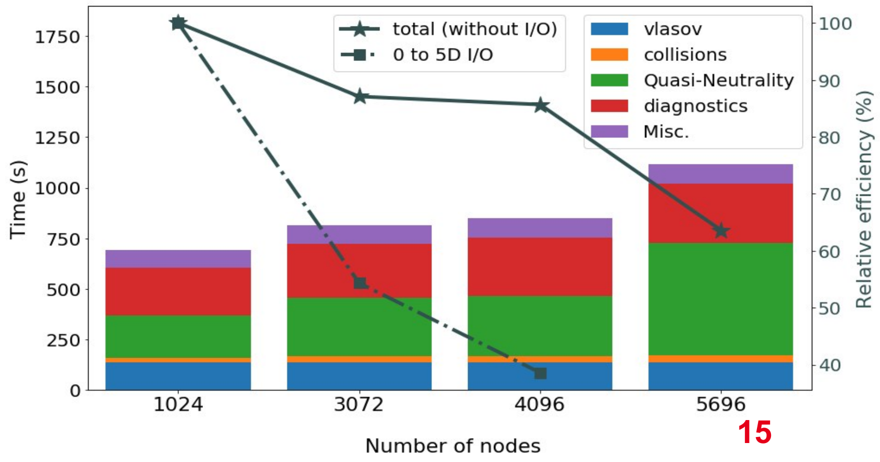
- Gyrokinetic codes require state-of-the-art HPC techniques and must run efficiently on several thousand processors
  - Non-linear 5D simulations (3D in space + 2D in velocity)
    + multi-scale problem in space and time



- Even more resources required when modelling both core & edge plasmas like GYSELA

- GYSELA = Fortran 90 code with hybrid MPI/OpenMP parallelisation optimized up to 730,000 cores
  - **Relative efficiency of 85% on more than 500k cores** and 63% on 730k cores on CEA-HF (AMD EPYC 7763)

Weak scaling of GYSELA on CEA-HF



- **Intensive use of petascale resources**: ~ 150 millions of hours / year
  - (GENCI + PRACE + HPC Fusion resources)

How to prepare GYSELA to HPC exascale architectures ?

⇒ Huge efforts of optimization and porting during EoCoE-II

- Target architectures:
  - 3 different architectures in the top 20

- Porting in 2021-2022 via CEA-RIKEN collaboration and GENCI support with ATOS

- Porting in 2022-2023 with HPE and EOLEN in the frame of ADASTRA Contrat de Progrès at CINES and with SCITAS-EPFL in the frame of EUROfusion Advanced Computing Hub

- May 2022: Opportunity to run during « Grand Challenge » campaign

TOP500 The List. NOVEMBER 2022

Rank	System	Cores	Rpeak (Pflop/s)
2	**Supercomputer Fugaku** – A64FX 48C, Fujitsu - RIKEN Center for Computational Science – **Japan**	7,630,848	537.21
11	**Adastra** – HPE Cray, AMD Instinct MI250X GENCI-CINES – **France**	319,072	46.10
20	**CEA-HF** – BullSequana XH2000, AMD EPYC 7763, Atos – CEA – **France**	810,240	23.24

- Operator refactoring (collisions, sources) + Performance optimization at node level (vectorization, blocking, asynchronous MPI communications) ⇒ Gain > 70%

Impossible without HPC experts

Good performance on the 3 architectures with same Fortran code via OpenMP directives
⇒ Not feasible without rewritting, duplication of most of the kernels

Roadmap for GyselaX++ towards exascale

→ Why do we choose to rewrite GYSELA ?

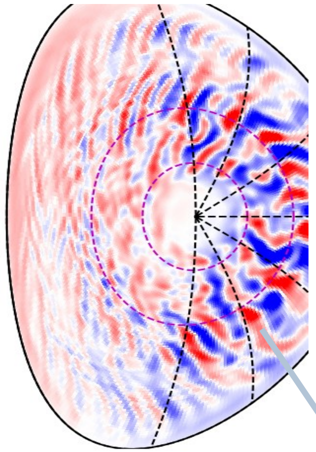
- 20 years-old code written in Fortran with hybrid MPI/OpenMP parallelism
- Unique code for both CPU (AMD milan or ARM-A64FX) and GPU with OpenMP directives is NOT optimal → extremely difficult to optimize on all architectures.
- Non-equidistant mesh mandatory for core-edge-SOL turbulence simulations

  → Modifying splines in GYSELA = rewrite most of the kernels

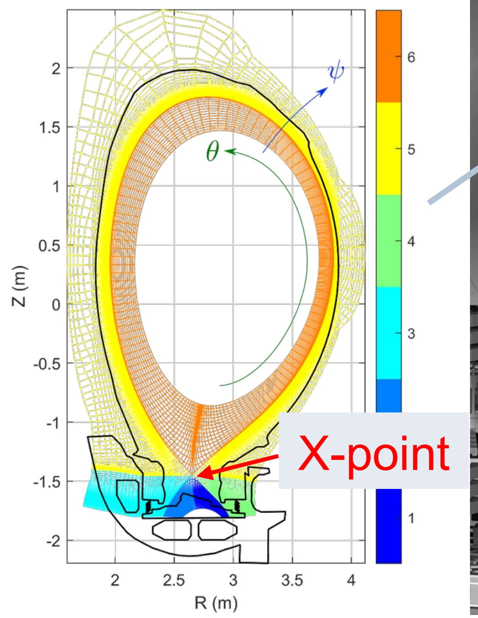
- X-point geometry

  → Development of new semi-Lagrangian scheme required to treat multipatches

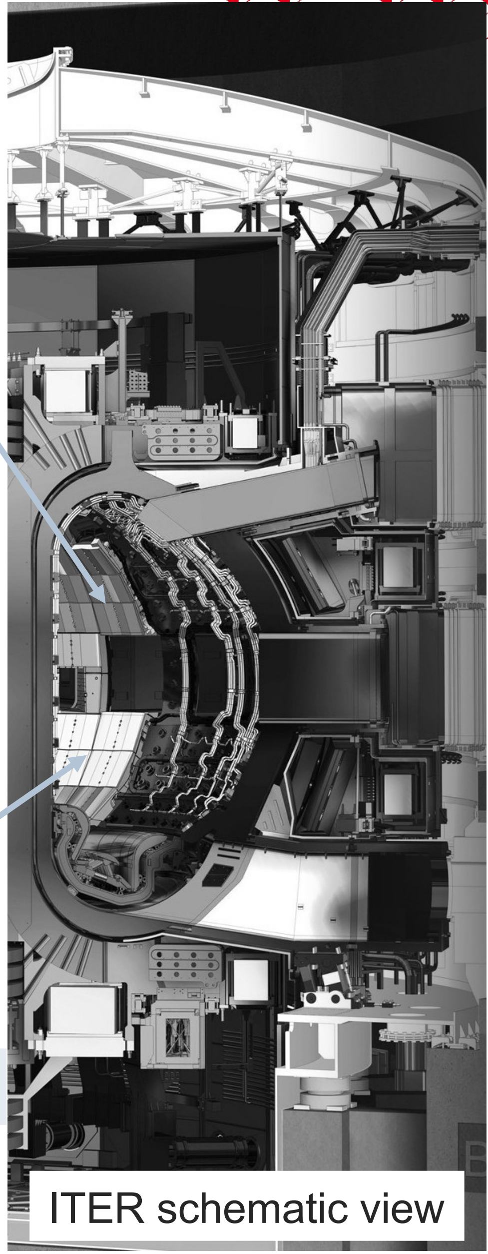
> Simpler to rewrite main kernels in modern C++ from scratch
> → GyselaX++ code



GYSELA
D-Shape geometry



SOLEDGE-3X
X-point geometry



ITER schematic view

Gysela-X towards exascale
Complete rewriting of the code in modern C++ (1/2)

- Proof of Concept: 2D prototype VOICE++ in modern C++ to address plasma-wall interaction problem

Input — YAML + Paraconf (developed at MDLS)

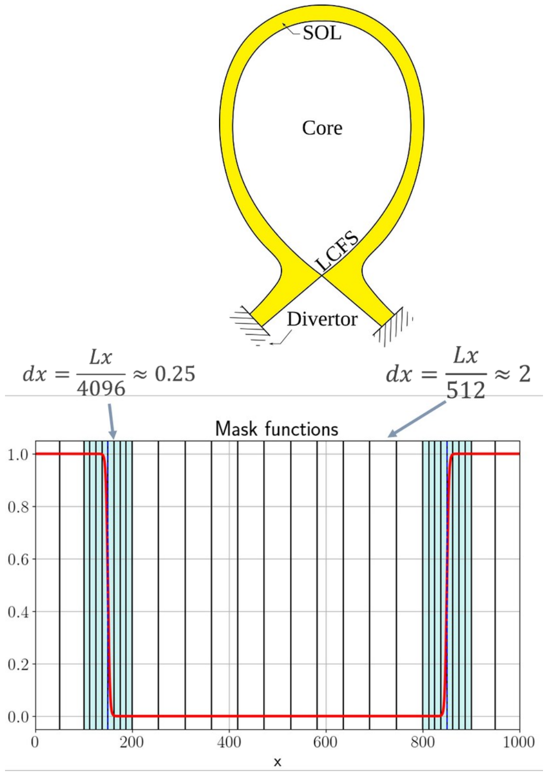
Geometry — a Discrete Domain Computation library — C++ template meta-programming domain decomposition library (developed at MDLS)

Vlasov — (1D,1V) Semi-lagrangian scheme with non-equidistant splines

Poisson — 1D solver based on spline finite elements

I/O — HDF5 via PDI

+ GITLAB + google tests + GITLAB-CI + SPACK installation

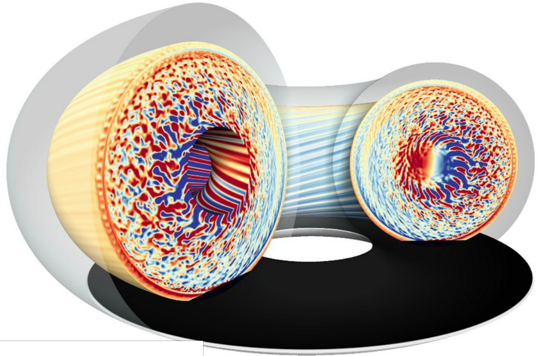


[E. Bourne et al., accepted JCP 2023]
[Y. Munschy et al., submitted to PoP]

https://github.com/gyselax/gyselalibxx

Gysela-X towards exascale
Complete rewriting of the code in modern C++ (2/2)

- 5D code in modern C++ scalable on exascale architectures



Input — YAML + Paraconf (developed at MDLS)

Geometry — DDC: a domain decomposition library — C++ template meta-programming domain decomposition library (developed at MDLS)

Vlasov — **5D semi-lagrangian** scheme with non-equidistant splines

Poisson — **3D solver** based on 2D spline FEM + 1D

I/O — **IMAS** via PDI

+ GITLAB + google tests + GITLAB-CI + SPACK installation

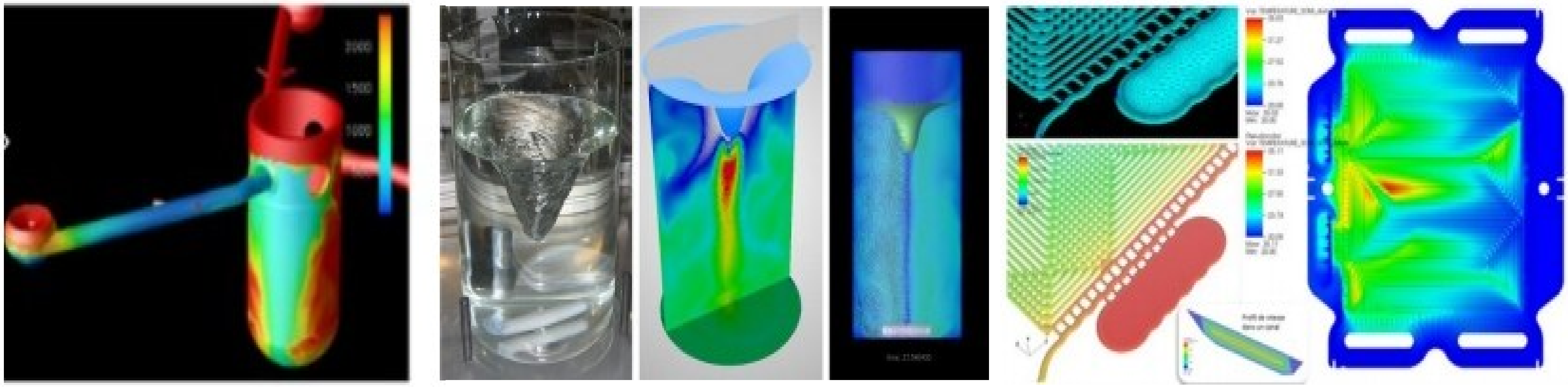
+ High level of parallelism MPI + KOKKOS

+ Adding of all operators (gyroaverage + sources + collisions)

+ Ampere (2D+1D) solver

+ In-Situ diagnostics

The TRUST / TrioCFD application

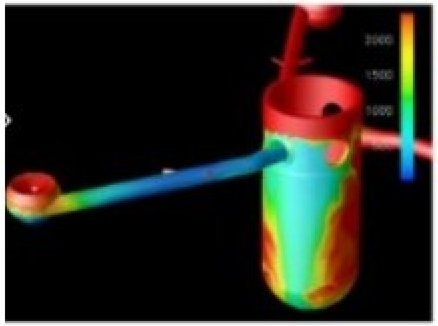


The TRUST/TrioCFD Application

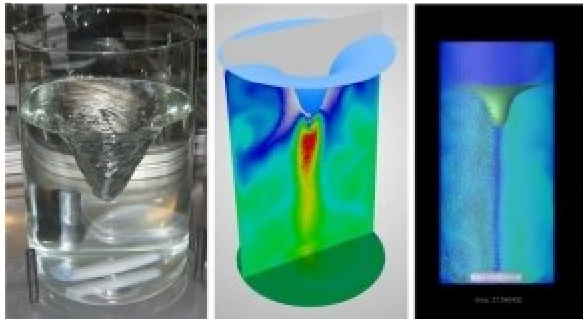
TRUST Thermohydraulic Plateforme (DES/DM2S/SGLS/LCAN)

TrioCFD Application dedicated to CFD build on TRUST

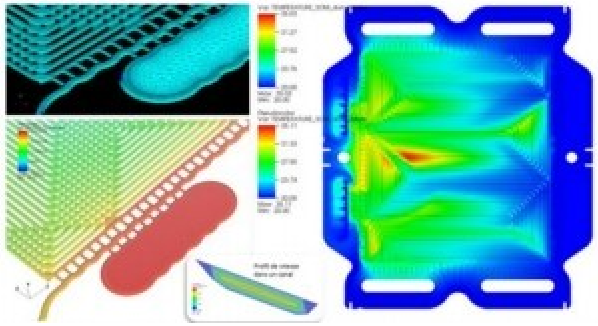
- Fluid mechanics :
  - Incompressible or slightly compressible;
  - Mono or diphasic
  - Front tracking
- Target applications area :



Reactor



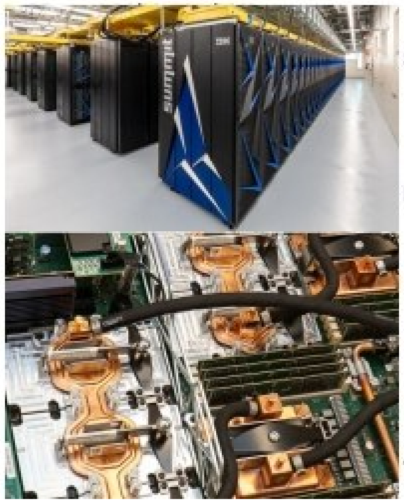
Vortex Mixing



Fuel Cells

- C++, MPI, OpenSource HTTPS://github.com/cea-trust-platform
- Many other applications build on TRUST : FLICA5, STT, CATHARE3D, TrioIJK, TrioMC, GENEPI+, PAREX+,…

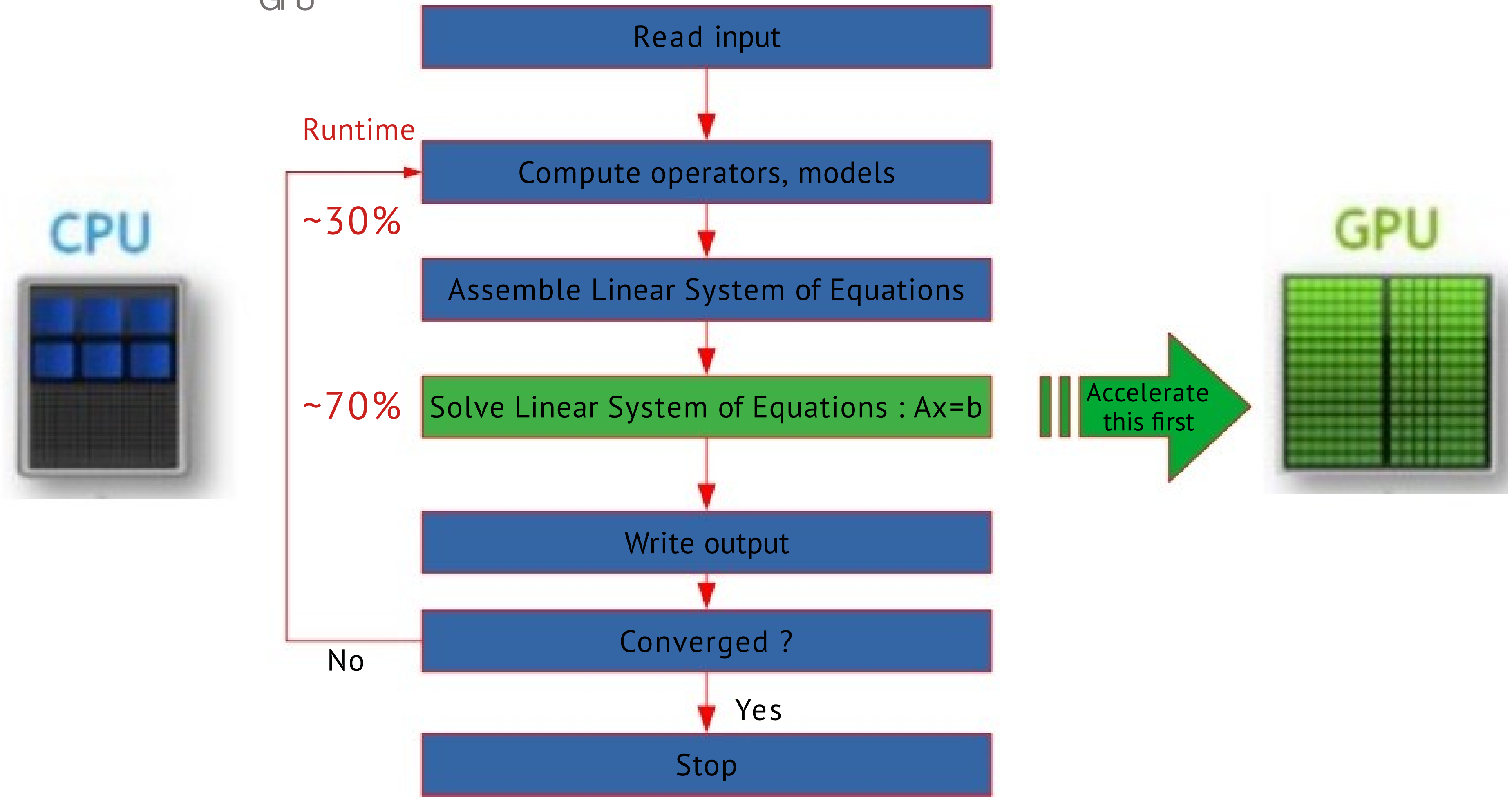
TRUST/TrioCFD roadmap for GPU computing



- 2014
  - First use of GPU in TRUST (PETSc)
    - Single node GPU, limited to one solver (GMRES/Jacobi)
- 2020
  - Test AmgX, Nvidia GPU library
    - Multi-node GPU, more solvers available (CG/Multigrid)
  - Porting of TRUST on ARM architecture
- 2021
  - Add AmgX library (Nvidia) to TRUST (1.8.3)
  - Nvidia Hackathon participation
    - TRUST team to evaluate OpenACC approach (parallel pragma directives)
- 2022
  - First study with a GPU partial accelerated TrioCFD (Jean-Zay)
  - Partial port on AMD GPU with OpenMP on Adastra (GENCI contract)
- 2023
  - First run with a fully GPU accelerated TRUST (Topaze)
- 2024
  - Enable CExA (Kokkos framework for CEA) in TRUST/TrioCFD
- 2025
  - French exascale supercomputer (ARM CPU/Nvidia GPU?)

Which strategy for TRUST computing on GPU ?

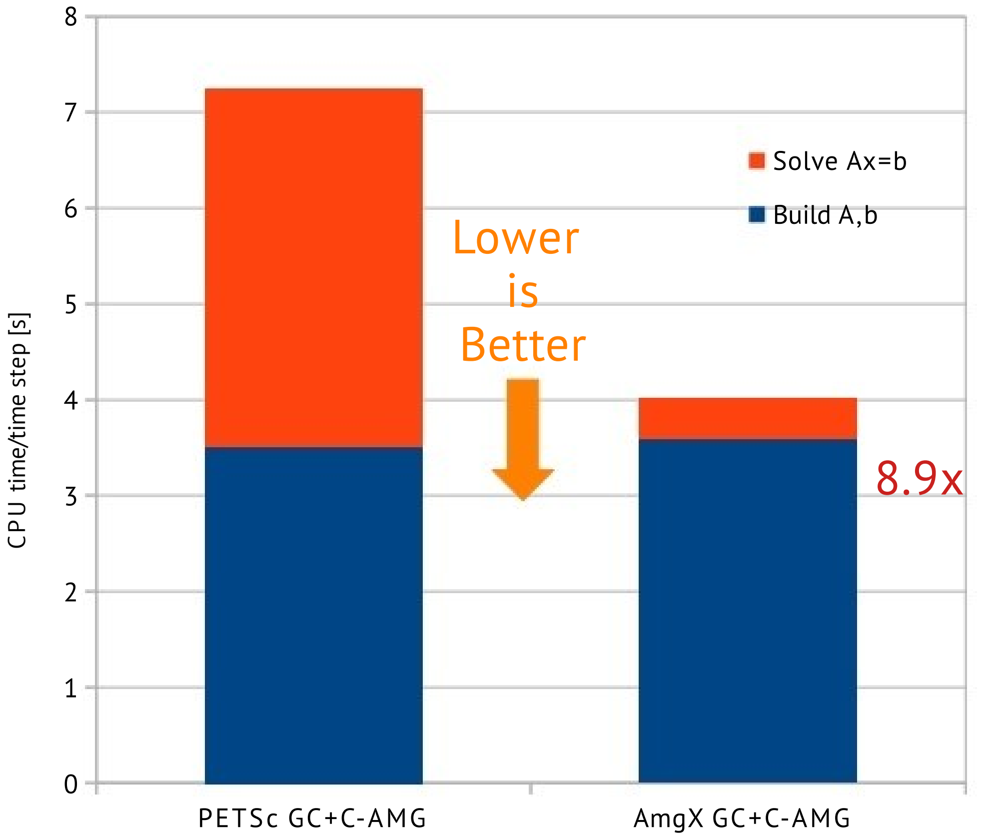
- Detect the most CPU expensive algorithms candidate to GPU



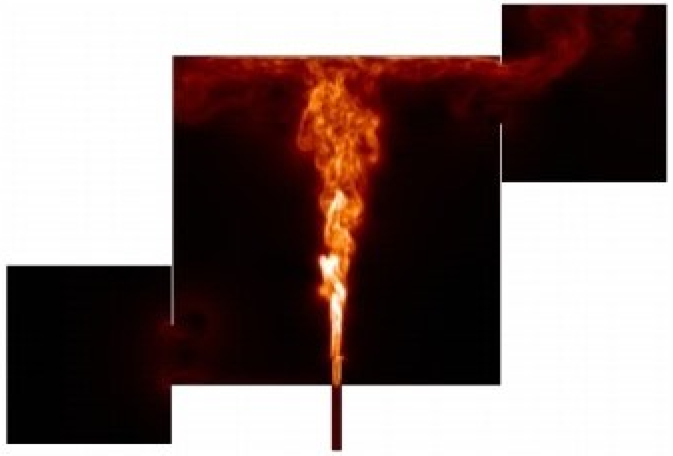
- Benefit firstly from dedicated linear algebra libraries (e.g. **AmgX** for GPU NVIdia)

Use of the AmgX solver (2021)

DNS simulation (TrioCFD 1.8.3) on Irene Joliot cluster (TGCC)



320 cores CPU | 320 cores CPU + 32 GPU

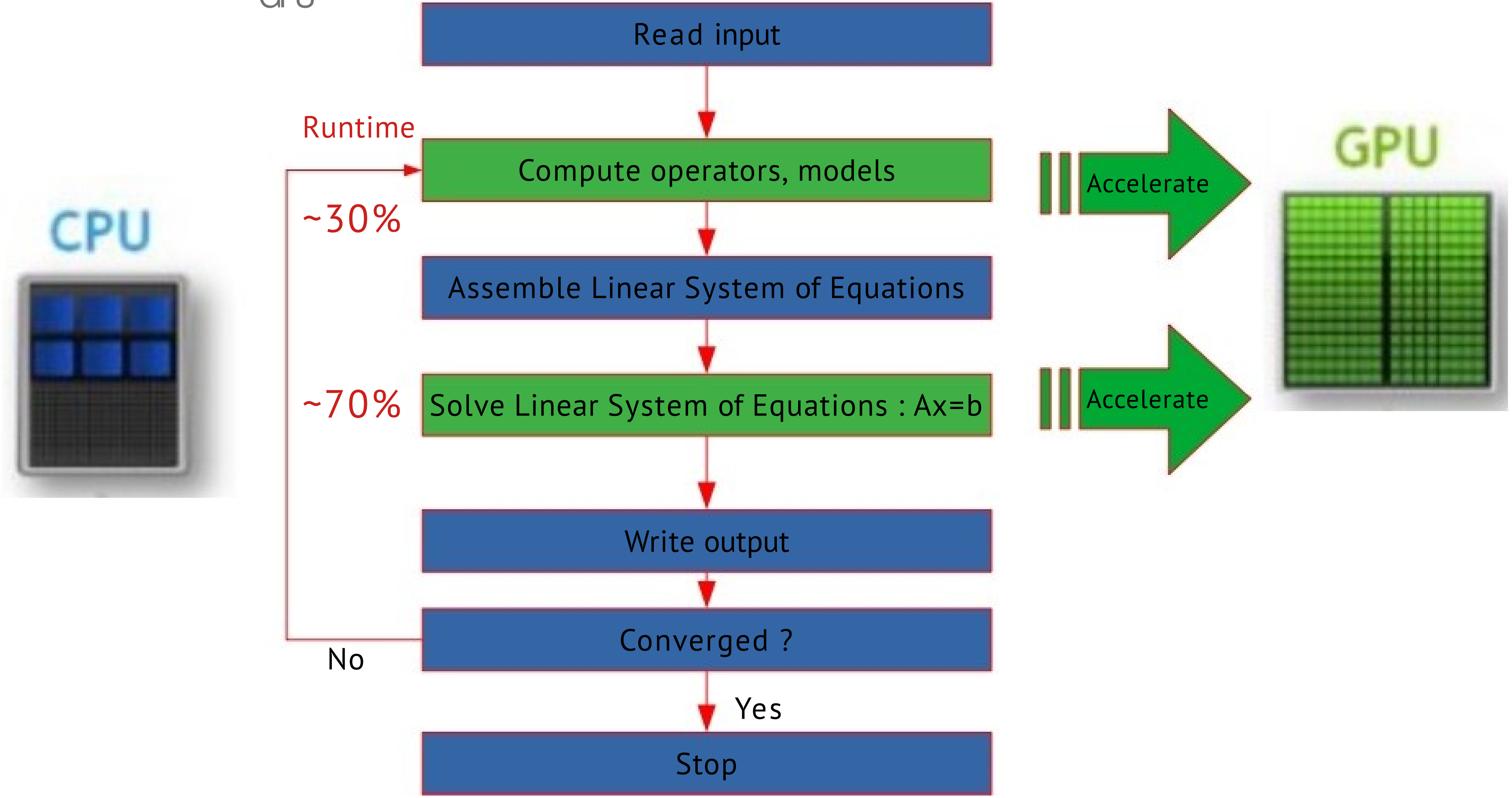


- Mini-GAMELAN geometry
- Structured mesh (VDF)
- 80M cells (250K/core)
- Unsteady DNS
- GC + C-AMG solver
- 50% time into solver

- **1.8x** acceleration for the simulation

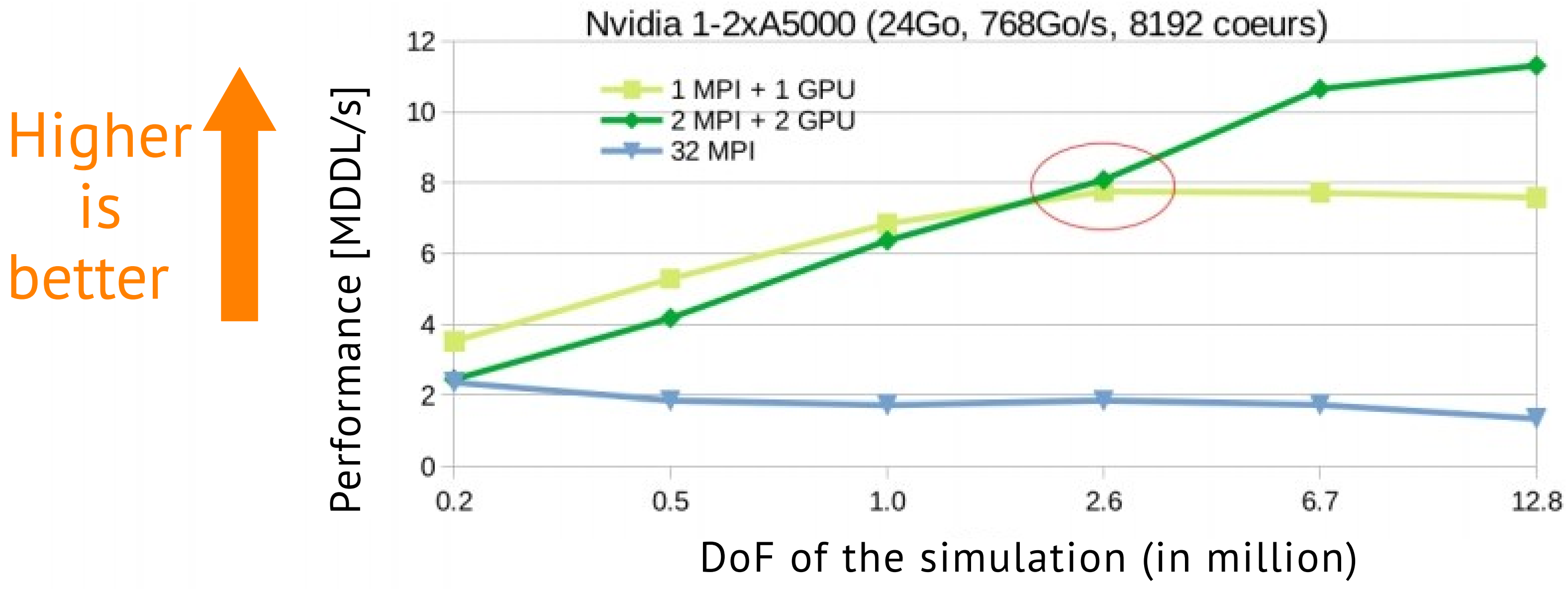
Which strategy for TRUST computing on GPU ?

- Detect the most CPU expensive algorithms candidate to GPU



- Benefit from dedicated linear algebra libraries (e.g. **AmgX** for Nvidia, **rocALUTION** for AMD)
- Introduce parallel directives (**OpenMP**) for the the most CPU expensive loops

Lesson #1 : Go Big !!



- GPU efficiency rise with the problem size
- 2-3 10^6 DoF per device seems optimal …
- … but it depends in the model, the device, communications,…

Lesson #2 : the code is memory bound

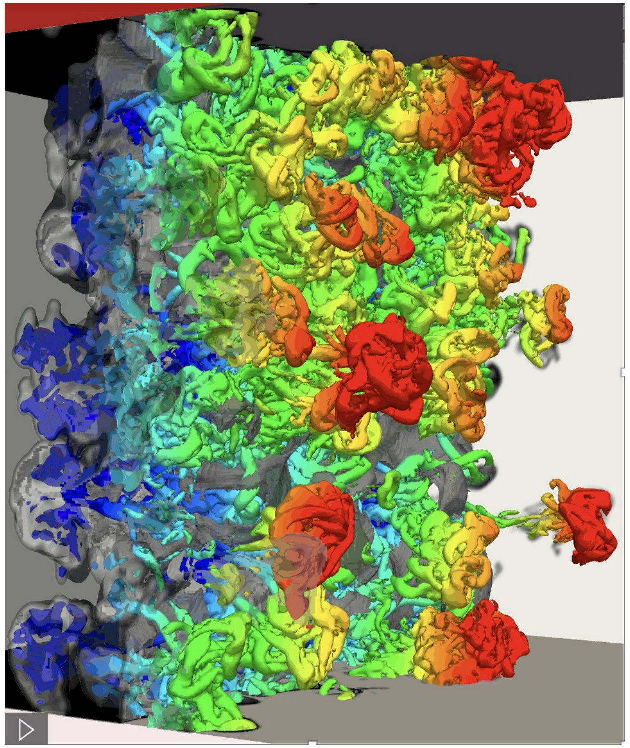
"roofline" analysis using Nsight Compute (Nvidia A6000):

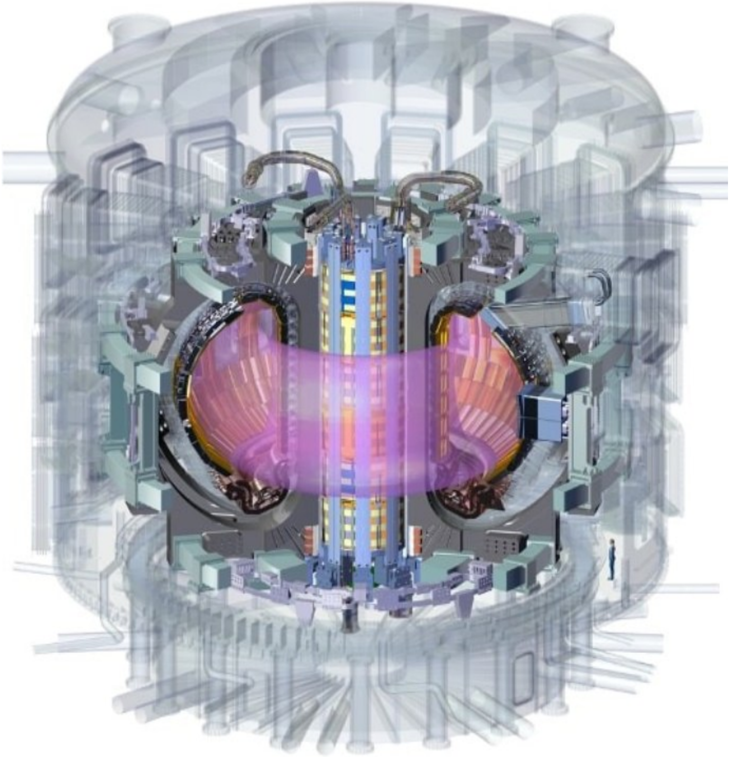
TRUST: Similar behavior on CPU and GPU

- The code is memory bound
- Only 15% of peak performance can be achieved on GPU

→ **Try to recompute some data instead of storing them**

The TRICLADE application



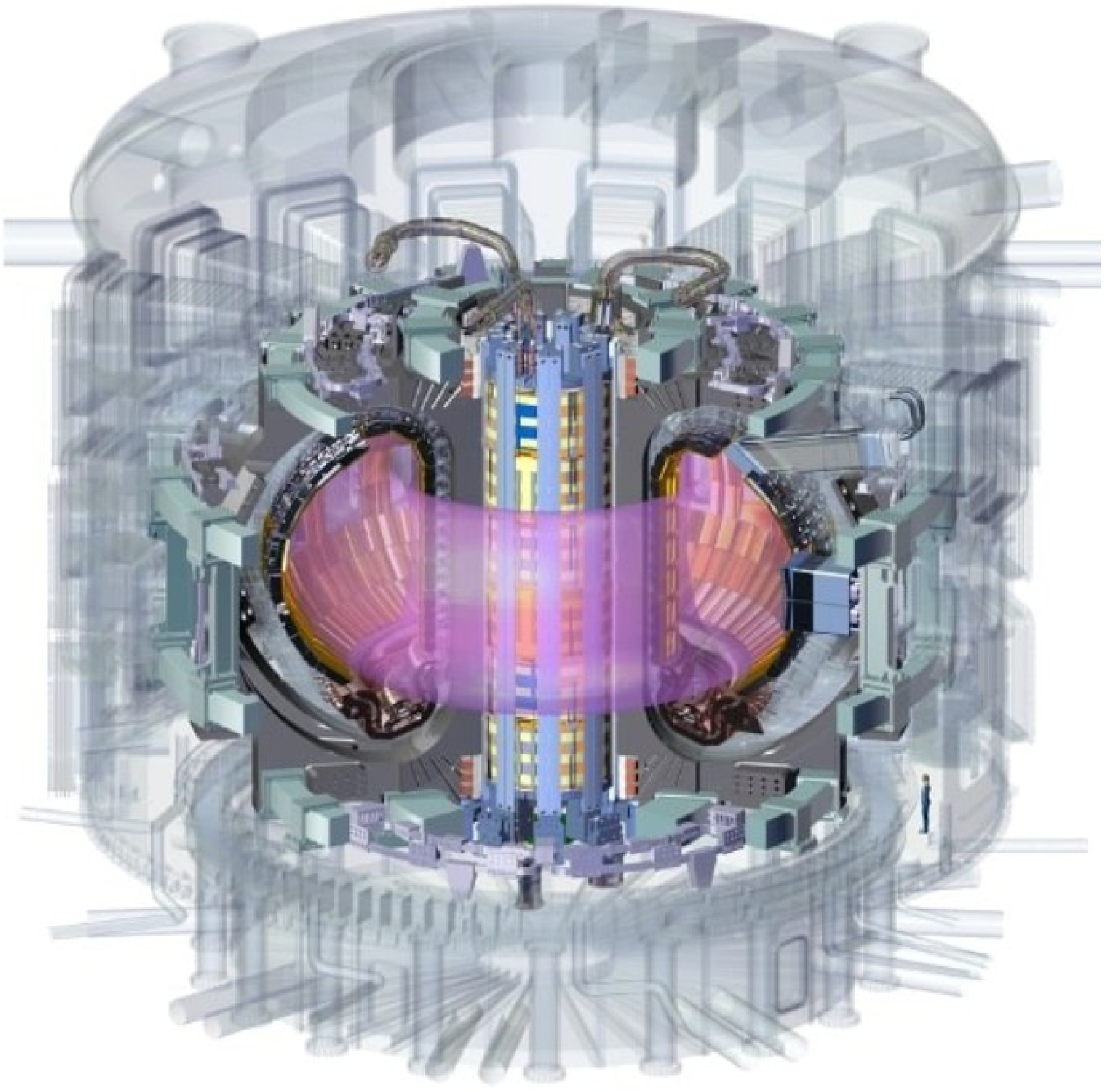


Introduction

Turbulent mixing

- Found in fields of interest to the CEA:
  - Astrophysics ;
  - Geophysics ;
  - Inertial Confinement Fusion ;
  - Etc.

- Very complex problem :
  - Intrinsically 3D ;
  - Multi-scale.

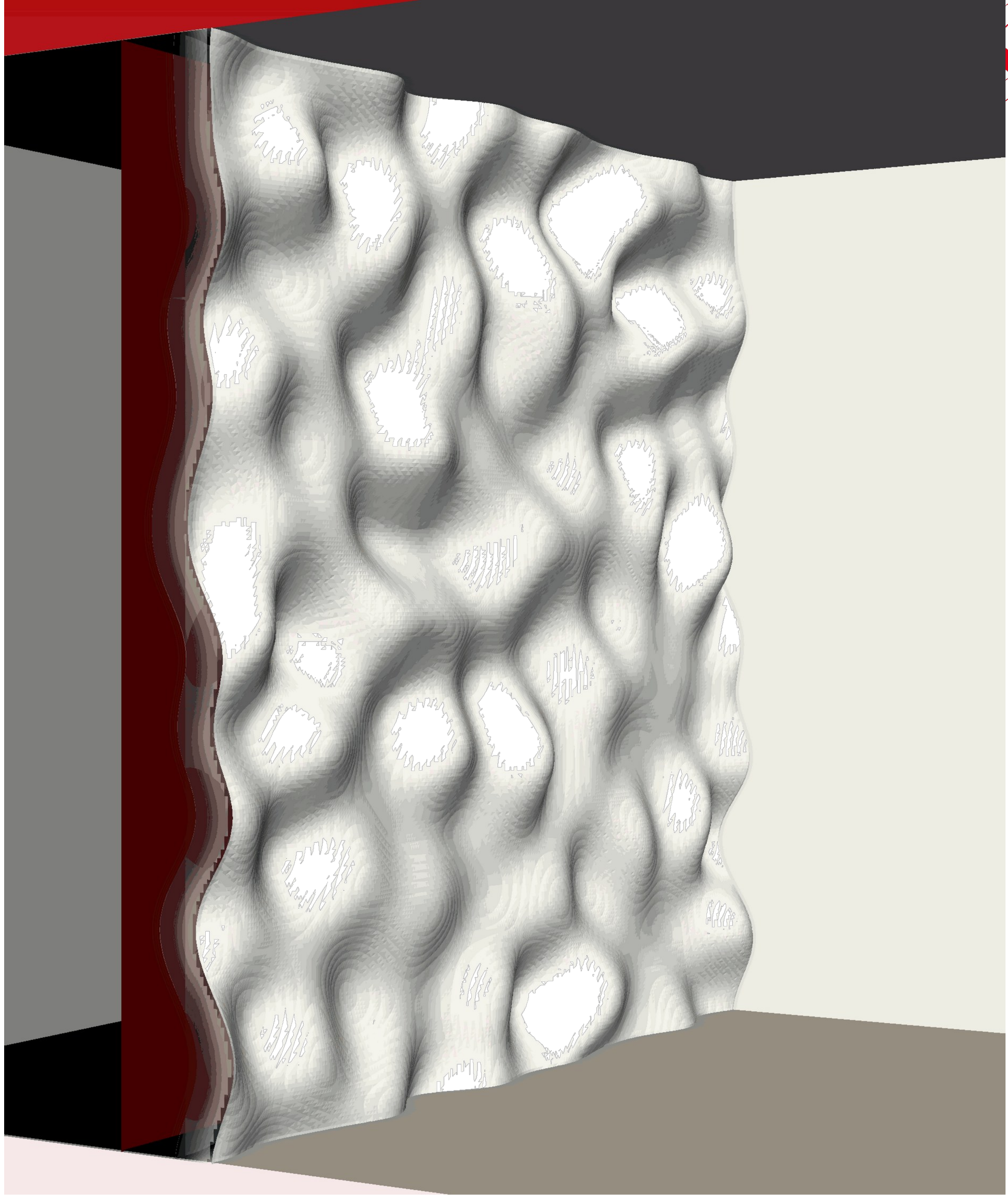


Some Context

- Study of Turbulent Mixing Zone:
  - Created and developed at fluids interface ;
  - From shock, expansion, acceleration, …
  - Dynamic and structure not fully understood.

- **TRICLADE**:
  - Turbulent binary mixing in a highly compressible environment
  - Navier-Stokes equations
  - Structured Cartesian Mesh
  - « Shock-capturing » numerical schemes

- Turbulence mixing problem = high complexity + multi-scale → need large mesh



Code information

- C++
  - Not really modern though…
  - ≈ 100 000 Lines of Code
  - MPI domain decomposition
  - Modular design
    - 1 module ≈ 1 numerical scheme
  - Depends on
    - Very little external libraries: MPI et FFTW
    - Lots of internal libraries for code environment

Porting Triclade to GPU

Triclade GPU port was decided

Impacted modules are roughly 10 000 LoC

M5
Numerical scheme

5th
Space order

3rd
Time order

few
Limiters and extrapolation methods

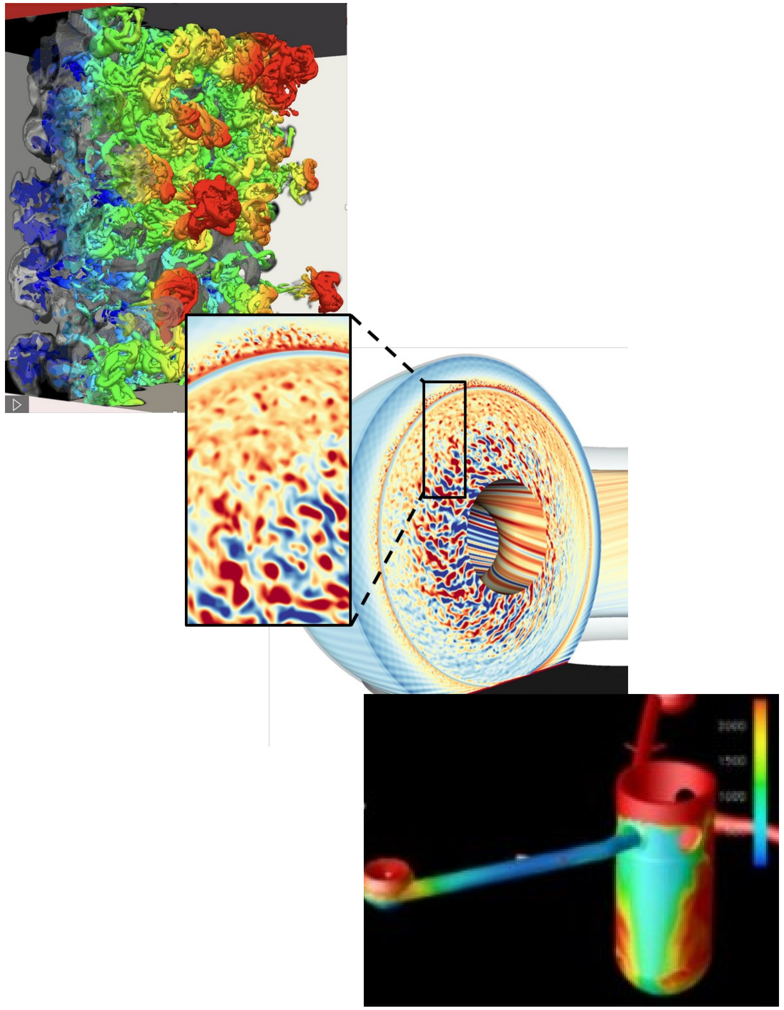
hllc2
flux

Regardless of the CExA initiative

Focusing on currently most use features

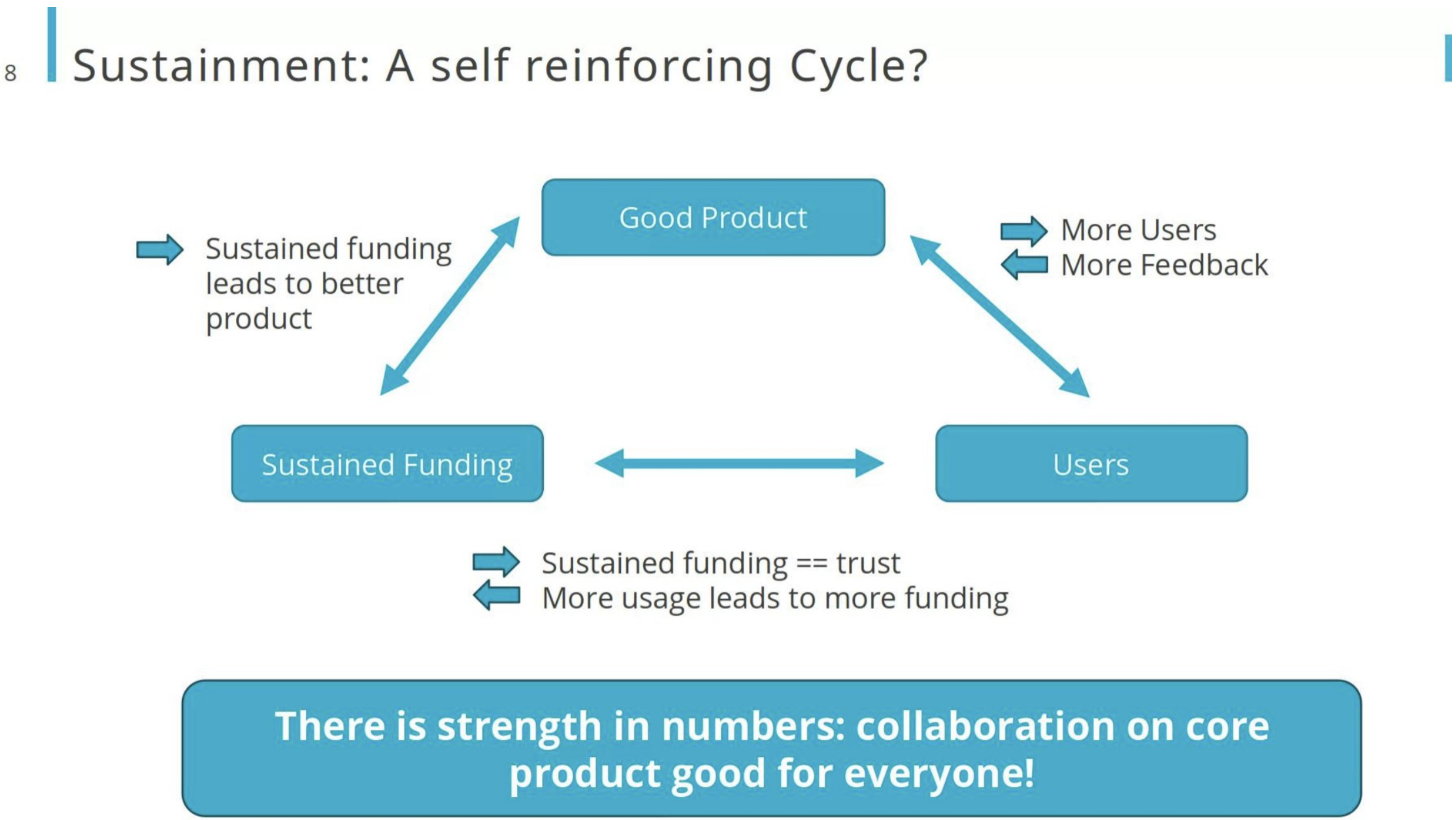
+ yet to be discovered dependencies...

Conclusion

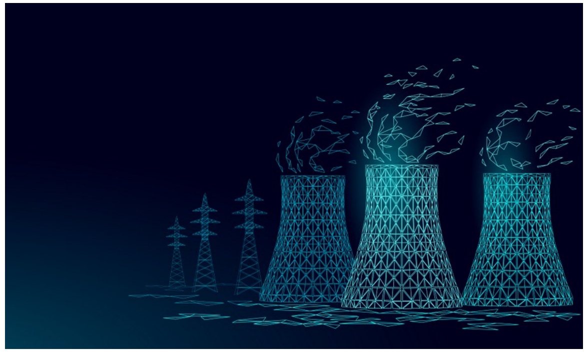


- Efficient way to port applications to GPU
- Performance portability
- Leverage on the strength of the Kokkos community
- Feed back to the Kokkos core team
- Long term sustainability of the codes

Contribute to a virtuous circle !!



Conclusion



- A sovereign tool to harness Exascale computers
- A large variety of applications across CEA
- CEA is building a community both around key applications and the Kokkos library development. The team is strongly motivated !!
- Strong collaboration with the Kokkos team
- A major impact on CEA programs and on many societal challenges
- Performance portability and code sustainability are the key challenges.
- Building a strong community is instrumental to meet the challenges
- Keep the application scientists onboard !

Thank you