RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

FRANCE 2030 PROGRAMME DE RECHERCHE NUMÉRIQUE POUR L'EXASCALE

Proxy geos

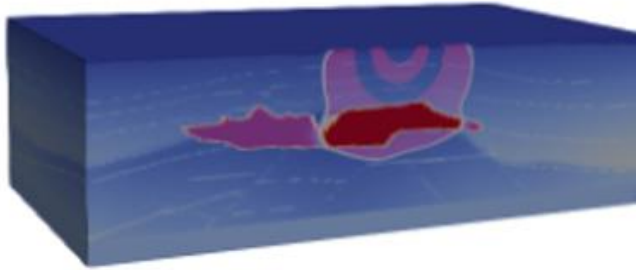
https://github.com/numpex/proxy-geos-hc/

https://gitlab.inria.fr/numpex-pc5/wp2-co-design/proxy-geos-hc

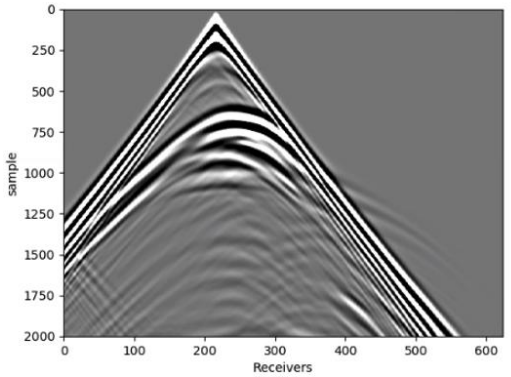
Plan

- Overview of Proxy-Geos
- Current state of the development environment
- Possible improvements and future developments
- Conclusion and way forward

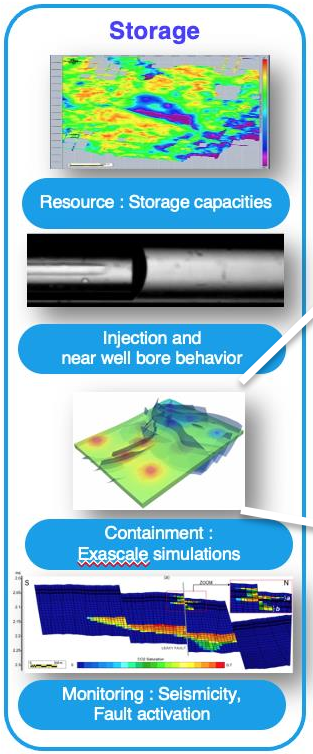
Proxy-GEOS



- Focus on spectral finite element wave equation kernels implementations and performances
  - The current version features includes:
    - Time discretization using the Leap Frog method
    - 3D scalar acoustic wave equation
    - Sponge boundary absorbing condition
    - Unstructured mesh Gauss-Lobato Spectral Finite Element implementation
    - Two integral formulations:
      - "Classic" standard SEM implementation
      - "optim" an enhanced SEM implementation based on GEOS
    - Programming models:
      - OpenMP, RAJA and Kokkos



Background of Proxy-GEOS



GEOSX
Multiphysics Simulator on HPC

TotalEnergies

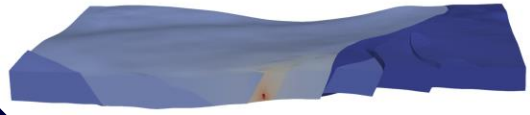
Stanford University

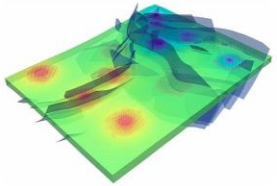
Lawrence Livermore National Laboratory

CO_2 Storage enhanced by next-gen supercomputing

Meant to improve **the management and safety of large-scale geological storage**

Open-source tool to support the entire industry in developing CCUS technologies which are essential for **achieving global carbon neutrality objectives**





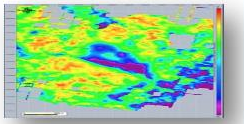
CONSUMER LAPTOP

PRODUCTION HPC

EXASCALE SYSTEMS

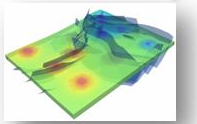
Background of Proxy-GEOS

Storage

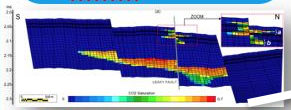


Resource : Storage capacities

Injection and
near well bore behavior



Containment :
Exascale simulations



Monitoring : Seismicity,
Fault activation

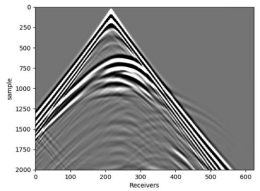
GEOSX
Multiphysics Simulator on HPC

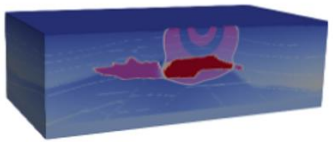
TotalEnergies

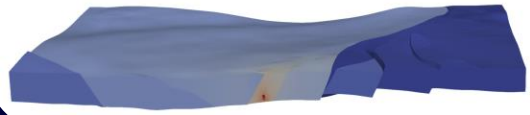
Stanford University

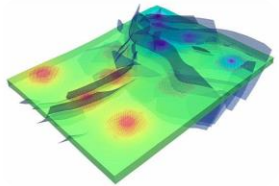
Lawrence Livermore National Laboratory

GEOSX: multi-physics CO_2 Storage enhanced by next-gen supercomputing









CONSUMER LAPTOP

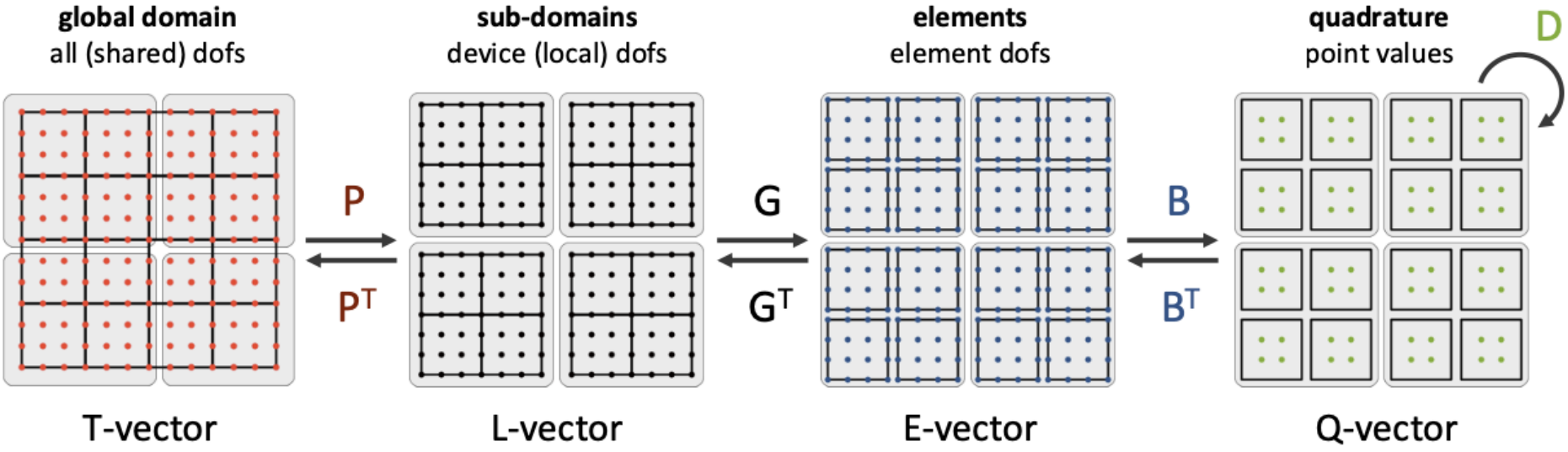
PRODUCTION HPC

EXASCALE SYSTEMS

FEM operator decomposition

FEM operator assembly/evaluation can be split into **parallel**, **mesh**, **basis**, and **geometry/physics** parts:

$$A = P^T G^T B^T D B G P$$



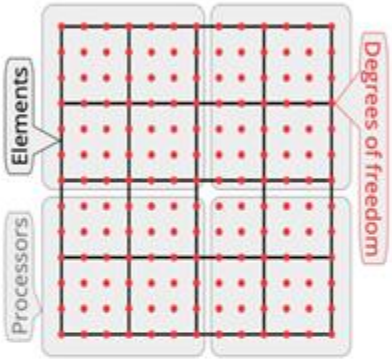
- ✔ **partial assembly** = store only **D**, evaluate **B** (tensor-product structure)
- ✔ better representation than **A**: *optimal memory*, *near-optimal FLOPs*
- ✔ purely algebraic
- ✔ AD-friendly

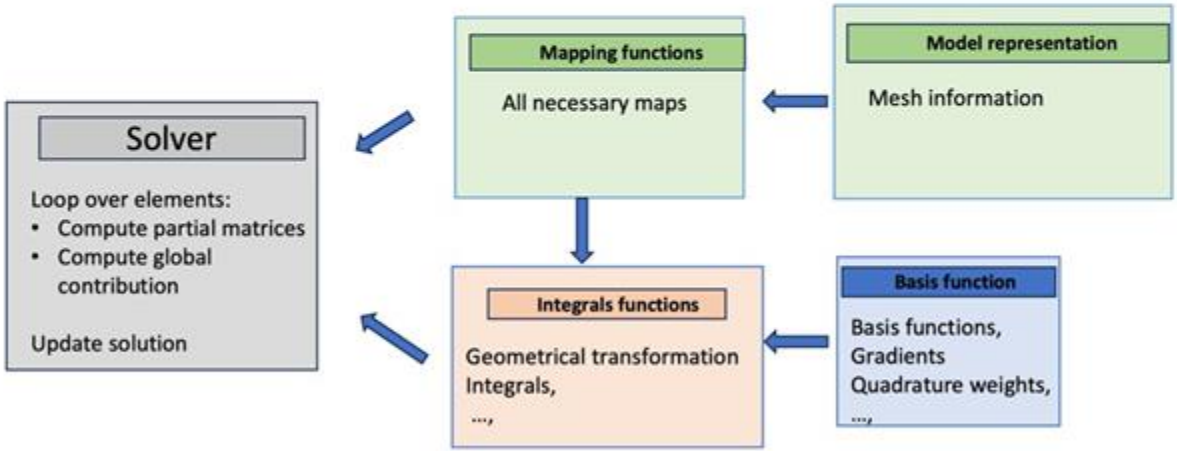
Proxy-Geos:

- Motif inspired by GEOS a multiphysics platform for fluid flow, geomechanical simulation and seismic wave propagation modeling
- Actual proxy-app version is targeting the scalar wave equation based on explicit time marching spectral finite element implementation.
- Discretization performed over a high order hexahedral representation
- Integration over Gauss-Lobatto quadrature points leads to diagonal mass matrix representation and algorithm complexity reduction from $O(r^9)$ to $O(r^5)$, r order or polynomial approximation, for the stiffness matrix computation.

$$\sum_{e=1}^{Nelem} \left\{ \int_{K_e} \frac{1}{c^2} \frac{d^2 p_h}{dt^2} v_h d\mathbf{x} + \int_{K_e} \nabla p_h \cdot \nabla v_h d\mathbf{x} + \int_{\partial K_e^{ext}} \alpha \frac{dp_h}{dt} v_h d\sigma(\mathbf{x}) - \int_{K_e} f v_h d\mathbf{x} \right\} = 0, \ \forall v_h \in U_h^r,$$

$$p_h(\mathbf{x}, 0) = p_{0,h}(\mathbf{x}), \ \frac{\partial p_h}{\partial t}(\mathbf{x}, 0) = p_{1,h}(\mathbf{x}) \in \Omega$$





Stiffness matrix $\int_{K_e} \nabla p_h \cdot \nabla v_h d\mathbf{x} = \sum_{i=1}^{(r+1)^3} p_i \int_{K_e} \nabla \varphi_i \cdot \nabla \varphi_j d\mathbf{x}$

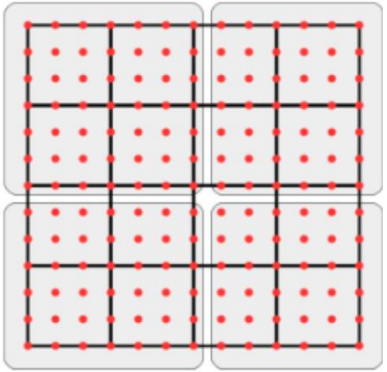
Mass matrix $\int_{K_e} \frac{1}{c^2} p_h v_h d\mathbf{x} = \sum_{i=1}^{(r+1)^3} p_i \int_{K_e} \frac{1}{c^2} \varphi_i \varphi_j d\mathbf{x}$

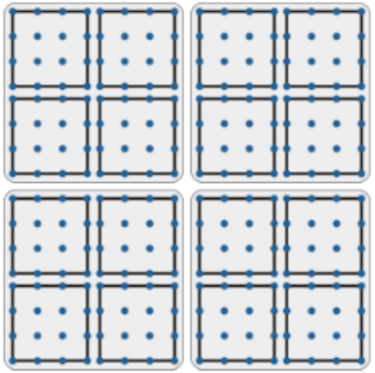
Development Environment Status

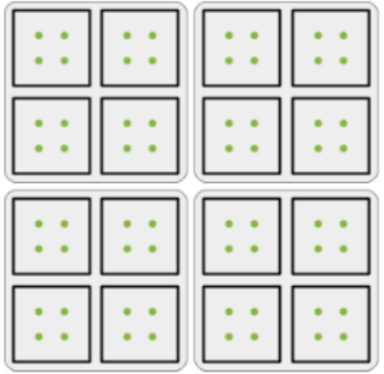
proxy-Geos-HC : https://gitlab.inria.fr/numpex-pc5/wp2-co-design/proxy-geos-hc

- Programming Language:
  - C++ 17
- Supported programming models:
  - OpenMP [https://www.openmp.org/]
  - RAJA [https://raja.readthedocs.io/en/develop/]
  - KOKKOS [https://kokkos.github.io/kokkos-core-wiki/]
  - RAJA and KOKKOS facilitate exposing computational kernels to accelerated hardware
- Data containers available:
  - LvArray [https://lvarray.readthedocs.io/en/latest/]
  - C++ std::vector
  - Kokkos
- Software package management tools:
  - GUIX
  - SPACK

Potential evolution/optimization





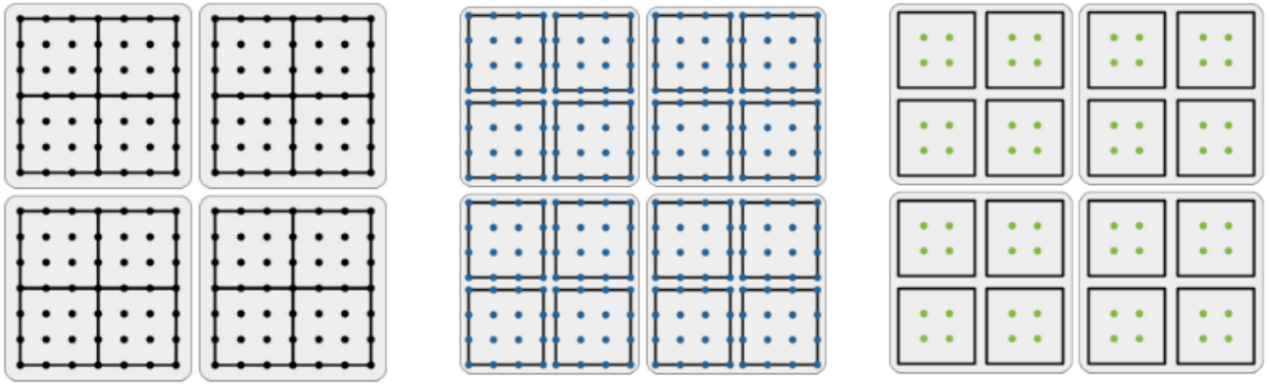


Existing implementation

Computational methodology:
Computations are performed per element, with one thread assigned to each element.

- **Potential evolution/optimization**:
  - Explore other strategies like **multi-streaming** to distribute computations across both quadrature points and elements, potentially leveraging **Kokkos teams** for efficiency.
  - Std C++ parallel loop ?
  - Task programming model ?

Potential evolution/optimization



Existing implementation

Computational methodology:
Computations are performed per element, with one thread assigned to each element.

- **Potential evolution/optimization**:
  - Multi device implementation,
  - Subdomain decomposition: MPI, ...
  - Task programming model ?
- **Other evolution (not directly related to Exa-Soft)**
  - Take advantage of tensor product structure to evaluate local integrals
  - Implement scalable IO strategy

Conclusion/Way forward

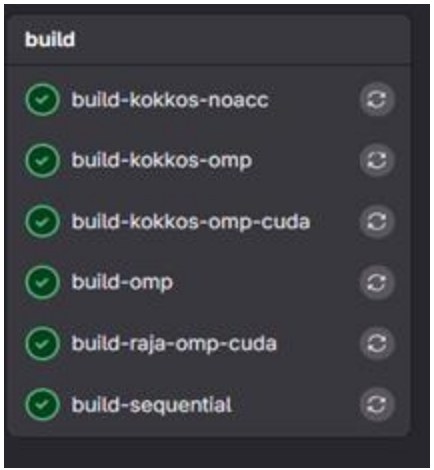
- Proxy-app “first release” achieved: dev aligned with Software Production Guidelines
  - ✅ Public Repo: > gitlab.inria.fr/numpex-pc5/wp2-co-design/proxy-geos-hc
  - 🚀 Release v0.1
  - ✅ CI
  - ✅ Packaging : spack, guix-hpc
  - ✅ Testsuite
  - ✅ Documentation

- Next steps in collaboration with Exa-Soft ?
  - Profiling tools: EzTrace,…
  - "deeper" kernel exposition to GPU (multi-streaming)
  - Multi-GPU implementation
  - Subdomain decomposition,
  - Other parallel paradigm?

BACK UP SLIDES

Automated build pipeline in Gitlab CI

- Docker image with all TPLs created by Guix
- Build multiple configurations inside this environment on Inria shared runners



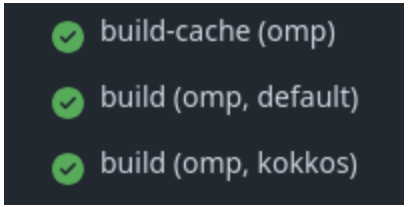
- Next steps : run tests (on non-gpu configurations), performance testing on Grid'5000

Automated test and deployment with GitHub Actions

- Set-up Spack environment from Numpex mirror
- Build inside this environment

```
build:
        needs: build-cache
    runs-on: ubuntu-24.04
    strategy:
        matrix:
                env_variant: [omp]
                preset: [default,
kokkos]
```

Example of configuration of the CI with GitHub : one variant of environment (`omp`) and two preset of compilation for Cmake (`default` and `kokkos`)



- Run the tests

Test Suite

- Environment Testing: to verify the Third-party libraries are properly installed
- Quick Functional Testing: to verify in a ultra-short run that the Proxy-app provides “sane” results

```
# SEM order
2
2000.000 2000.000 2000.000
# Mesh information
8120601 1000000 27 7880599
# Source information
1001.000 1001.000 1001.000 505050
0.010 0.001 3
3
0 0.000 0.010
3 0.003 0.018
6 0.006 0.030
# PnSource
0 0.000 0.041
3 0.003 0.229
6 0.006 0.386
```

← Reference output for 6 steps at 10^{-3} precision

To be added next: unit tests, performance testing, numerical testing…

Generalization of CMake Presets

- Simplification of deployment process using presets

```
cmake -DUSE_KOKKOS=ON \
-DBUILD_FROM_TPLMIRROR=ON \
-C configs/config_proxy-app.cmake \
-B ./build \
-DCMAKE_INSTALL_PREFIX=./install \
-S .
```

Example of CMake command for the Proxyapp

→

```
cmake --preset kokkos-my_machine_cuda-tpl_from_source
```

Example with CMake Presets

- Presets are generated by script: accorded to user/developers's usage
- All Proxy-Geos configurations are covered

```
Available configure presets:

"kokkos-mirror-my_machine_cuda"   "kokkos-mirror-my_machine_hip" "kokkos-mirror-
noacceleration"    "kokkos-guix-my_machine_cuda"  "kokkos-guix-my_machine_hip"
"kokkos-guix-noacceleration" "raja-mirror-my_machine_cuda"     "raja-mirror-
my_machine_hip" "raja-mirror-noacceleration"      "raja-guix-my_machine_cuda"
"raja-guix-my_machine_hip"        "raja-guix-noacceleration"
"omp-mirror-my_machine_cuda"      "omp-mirror-my_machine_hip"
"omp-mirror-noacceleration"       "omp-guix-my_machine_cuda"
"omp-guix-my_machine_hip"         "omp-guix-noacceleration" "sequential-mirror-
my_machine_cuda" "sequential-mirror-my_machine_hip" "sequential-mirror-noacceleration"
"sequential-guix-my_machine_cuda"  "sequential-guix-my_machine_hip"  "sequential-guix-
noacceleration"
```

- Script is not "Proxy-Geos dependant": can be used for any proxyApp

The guix-numpex channel

```
(channel
    (name 'guix-numpex)
    (url "https://gitlab.inria.fr/numpex-pc5/guix-numpex.git")
    (branch "master"))
```

Host:

- The PROXY-GEOS packages

  proxy-geos-<LIB>-<GPU>-<ARCH>

- The required Raja (camp, chai) and Kokkos variants

Using OMP:

- OMP can be activated on demand using guix package transformation functionality

Abstraction **LIB**rary	**GPU** backend model	**ARCH**itecture
KOKKOS	**CUDA**	Volta V100
		Turing T4
		Pascal P100
RAJA		Kepler K40
		Ada Lovelace
		Ampere A40
		Ampere A100

On going Dev

Using `feelpp::feelpoly` as another FEM in the proxy-app

- Integrate the polynomial library of Feel++ `feelpoly`, currently CPU only
- Port `feelpoly` to GPU :
  - use Eigen: replacement of Boost::ublas
  - Enable Kokkidio, middleware between Kokkos and Eigen to simplify coding
  - implement fast HO methods on tensor product convex
- Build using Spack environment defined in https://github.com/numpex/spack.numpex (feelpp, kokkidio, proxy-geos-hc)

Listing 3: Parallel SAXPY in *Eigen*, using OpenMP

```
float a (0.5);
int size (10);
/* 1D data structure in Eigen */
Eigen::VectorXf
  x (size),
  y (size),
  z (size);


/* We only open a parallel region,
 * as the loop is inside Eigen */
#pragma omp parallel
{
/* We compute the ranges of work
 * items. For simplicity, we are
 * neglecting remainders here. */
int nproc = omp_get_num_threads();
int procItems = size / nproc;
int start = procItems * nproc;

/* Operate on contiguous segments,
 * to facilitate vectorisation */
z.segment(start, procItems) = a *
  x.segment(start, procItems) +
  y.segment(start, procItems);
}
```

Listing 4: SAXPY in *Kokkidio*

```
float a (0.5);
int size (10);
/* 1D data structure in Kokkidio */
Kokkidio::ViewMap<Eigen::ArrayXf>
  x (size),
  y (size),
  z (size);


/* Parallel dispatch looks the same
 * as in Kokkos. On the host, this
 * opens a parallel region. */
Kokkidio::parallel_for( size,

/* Computing the ranges happens
 * in a target-specific way when
 * constructing ParallelRange */
  KOKKOS_LAMBDA(ParallelRange<> rng){

/* ParallelRange::operator() returns
 * an Eigen::Block expression,
 * e.g. 'segment' in 1D, making the
 * statement much more concise: */
  rng(z) = a * rng(x) + rng(y);
 }
);
```

From Steffen, Lennart and Hinkelmann, Reinhard, Kokkidio: Fast, Expressive, Portable Code, Based on Kokkos and Eigen. Available at SSRN: https://ssrn.com/abstract=5013361 or http://dx.doi.org/10.2139/ssrn.5013361

Next Steps ?

Programming models	PC1 PC2	Covers extending beyond Kokkos/RAJA/OpenMP to new abstractions or DSLs (e.g. SYCL, Alpaka) as part of the software stack evolution.
Compare kokkos vs std c++ (kokkos::View becomes mdspan in C++23)	PC2	Software modernization topic – migration or interoperability with evolving C++ standards.
Task runtime (StarPU, Specx)	PC2	Task-based execution models
Subdomain decomposition (multi GPUs + MPI)	PC1 PC2	Numerical implications (PC1) and runtime-level management (PC2), especially for hybrid shared/distributed parallelism.
I/O asynchrone + compression + I/O processes (eg Damaris)	PC3	High-performance I/O, reduced memory footprint, and postprocessing fall under PC3's focus on data, UQ, and ML pipelines.

Next Steps ?

Agnosticity to mesh	PC1 PC2	PC1 for algorithmic generalization (e.g., supporting FEM on hybrid meshes), PC2 for infrastructure changes required in mesh handling libraries.
Use new convexes (tetrahedra, prism, pyramids)	PC1	Mathematical/algorithmic support for new element types in spectral or finite elements.
Use new basis functions (Bernstein, Dubiner)	PC1	Involves changing polynomial spaces and quadrature rules for better numerical performance or compatibility with various convexes.
Support for HDG	PC1	High-order method that requires local solvers and trace spaces, deeply tied to numerical algorithm design.
High order geometries	PC1	Geometrical mapping of curved domains, especially for high-order elements; relates to accurate representation of boundaries in FEM.