NumPEx Project
Global Structure and Application Demonstrators

November, 2022
NumPEx Project
Global Structure

- NumPEx is a PEPR project, selected in the framework of the France2030 program
- NumPEx is composed of 5 “Target Projects” (PC = Projet Ciblé)
  - PC1 - ExaMA - Methods and Algorithms for Exascale
  - PC2 - ExaSoft - HPC Software and Tools
  - PC3 - ExaDoST - Exascale Data-Oriented Software and Tools
  - PC4 - ExaATOW - Architectures & Tools for Large-Scale Workflows
  - PC5 - ExaDIP - Application-driven Co-design Software Development & Productivity
    PC in charge of Application Demonstrators
**PC5 - ExaDIP**

- **Strategic PI:** Jean-Pierre Vilotte (CNRS)
- **Operational Co-PI:** Valérie Brenner (CEA)
- **Participants:** CNRS, CEA, INRIA
- **Coordinator:** CNRS

PC5 is composed of 4 work packages, executed by a Computational and Data Team (CDT) as follows:

<table>
<thead>
<tr>
<th>WP</th>
<th>Objectives</th>
<th>CDT FTEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WP-1</td>
<td>Select, steer, manage co-design activities across Exa-DIP</td>
<td>2</td>
</tr>
<tr>
<td>WP-2</td>
<td>Implement and manage co-design computational and communication motif projects</td>
<td>5</td>
</tr>
<tr>
<td>WP-3</td>
<td>Implement and manage co-design software development kit integration projects and NumPeX enabling software delivery technologies</td>
<td>5</td>
</tr>
<tr>
<td>WP-4</td>
<td>Engage in training and outreach Numpex materials and activities</td>
<td>1</td>
</tr>
</tbody>
</table>
Exa-DIP - WP1

- **Leader:** Jean-Pierre Vilotte (CNRS); **Co-Leader:** Jérôme Bobin (CEA)
- **CDT FTEs/5 years:** 2
- **Partners:** CNRS (1 FTE), CEA (1 FTE), INRIA

**T1.1: EXA-DIP Strategic Committee**
- Exa-DIP PIs, Exa-DIP WP leaders, representatives of each PC (1-4) for:
  - Review and prioritise NumPeX CSE Application Demonstrators (ADs)
  - Identify and prioritise ADs-driven computational and communication motifs
  - Identify and prioritise ADs-driven collection of software components to be integrated into software development kits
  - Hire and set up the CDT team to be allocated across the Exa-DIP activities

**T1.2: Exa-DIP Program Forum**
- CDT: program manager and ReleaseTrain Engineer, PCs’ and ADs: business owners
- steer and manage the Exa-DIP co-design activities (WP-2, WP-3) as Agile Release Trains

**T1.3: Key Performance Parameters (KPP3s)**
- to assess deliverables of the Exa-DIP activities;
- reviewed and approved by the Exa-DIP SC

**T1.4: Collaboration channels with national computing facilities, vendors, and other relevant organisations**

**Deliverables:**
- D1.1.1 (PM+2) First set of Application Demonstrators, to be updated each year
- D1.1.2 (PM+3): First identified computation and communication co-design motifs; to be revised and updated every year
- D1.1.3 (PM+12): first list of identified SDKs
- D1.3.1 (PM+6): First KPPs and metrics; to be revised in PM+12

**Milestones:** M1.2 (PM+6) Program Manage Forum in place and running, M1.1 (PM+12) CDT teams in place and running
Exa-DIP - WP2

- **Leader:** Mark Asch (U. Picardie); **Co-Leader:** Valérie Brenner (CEA)
- **CDT FTEs/5 years:** 5
- **Partners:** CNRS (3 CDT/FTEs), CEA (2 CDT/FTEs)

**T2.1: Co-design projects addressing a unique computational and communication motif**
- CDT members: product owner, scrum master, experts; PCs: experts from research and development teams, ADs: members from development teams
- Execute co-design projects as self-organised and self-managed agile teams
- Execute iterative program increments

**T2.2: Review-and-adapt meetings (after each program increment)**
- assess progress and methodologies
- assess developed software components according to KPP3s
- adapt and refine co-design projects’ backlogs (objectives, planning package, deliverables, milestones)

**T3.3: Review of developed co-designed software components**
- assess software components (libraries, tools, frameworks) according to KPP3s
- with WP-1 Strategic Committee and input from external SMEs

**Deliverables:**
- D2.1.1 (PM+3) First set of co-design projects; D2.1.2 (PM + 24): Updated set of co-design projects …
- D2.2.1 (PM+ 9): Review-and-adapt meeting reports; D2.2.2 (PM + 15) Review-and-adapt meeting reports …
- D2.3.3 (PM+ 15): First assessment report of developed co-designed software components

**Milestones:** M2.1.1 (PM+3) First set of co-design projects running, M1.2.1 (PM+9) First review-and-adapt meetings
Exa-DIP - WP3

- Leader: Bruno Raffin (INRIA); Co-Leader: Julien Bidot (CEA/Maison de la Simulation)
- CDT FTEs/5 years: 5
- Partners: CEA (2 FTEs), INRIA (3 FTEs)

T3.1: Establish common set of Community Software Policies

T3.2: SDK Co-design projects
  - CDT members: product owner, Release Train Engineer, experts; PCs: experts from research and software development teams; ADs:
  - AD-driven integration and testing to assess multi-layer interoperability for combined use of meaningful set of developed software components

T3.3: Enabling Team to broaden and harden common meta-builder and container technologies adapted to HPC systems
  - First level format targets: Spack, Singularity, Source
  - Second level format targets to be investigated: GUIX, Nix, Conda
  - Other optional targets: OpenHPC, Shifter, ...

T3.4: Software integration hub
  - Test/integration/release infrastructure (GitLab-CI/Jenkins/GitHub-inspired)
  - Partnering with national computing facilities, vendors, EuroHPC-CoEs and other projects for proving access to continuous integration and testing platform

Main Deliverables:
- D3.1.1 (PM + 6) First set of Community Software Policies, D3.1.2 (PM + 12) Revised Community Software Policies
- D3.2.1 (PM+ 12): First set of identified SDKs; to be revised and updated every year
- D3.3.1 (PM + 8): First report on common meta-builder and container technologies
- D3.4.1 (PM+12): First report on the Software integration hub and related policies

Milestones: M1.1 (PM+6) First set of Community Software Policies, M1.2 (PM+12) SDK co-design projects running ; M3.4.1 (PM + 18) Software integration hub
Exa-DIP - WP4

• **Leader:** France Boillod-Cerneux (INRIA); **Co-Leader:** Karim Hasnoui (CNRS/IDRIS/Maison de la Simulation)
• **CDT FTEs/5 years:** 1
• **Partners:** CEA (0,5 FTEs), CNRS (0,5 FTEs)

**T4.1: Collaboration Hub for better scientific software**
- community-driven information and resources on practices, techniques and experiences
- sharing tools to improve software development methodologies and software sustainability and related computing areas
- Practical guides for co-designed algorithmic methods, software components, and SDKs
- collaborative content creation and management system linked to GitHub

**T4.2: Webinar and tutorial materials, interdisciplinary technical meetings**
- best scientific software development practices; co-designed software components and SDKs, use of SPACK and Containers
- organise interdisciplinary technical meetings every year with contribution from the co-design projects and the WP-3 enabling team
- synergies with GENCI and the national computing facilities, PEPR and PIA projects, EuroHPC CoEs

**Deliverables:**
- D1.1 (PM+6) First report on the Collaboration Hub development and content, to be updated every 6 months
- D1.2 (PM+6): First set of webinar and tutorial materials; to be revised and updated every 6 months
- D1.3 (PM+8): First interdisciplinary technical meeting report; to be augmented

**Milestones:** M4.1 (PM+6) Collaboration Hub, M1.4.2 (PM+12) First interdisciplinary technical meeting
Application Demonstrators in NumPEx / PC5
Application Demonstrator
What is an Application demonstrator?

Objective:
Accelerate the development and enhance the capability and the performance of strategic CSE applications

- High-impact science and engineering exascale challenge problem
- Detailed criteria for assessing successful completion of challenge problem
- A figure of merit (FOM) formula quantifying performance or capability enhancement of challenge
- Demonstration and assessment of effective software integration with demonstrators

NumPeX Application Demonstrators
- Addressing on a science and engineering Exascale challenge problem
- Enabled by combined use of interoperable models, software components and technologies (crossing different NumPeX PCs)
- Driven by community research practices and scientific/engineering application development methodologies
- Assessed by measuring rate of science work enabled by successful and possibly inter-dependent developments.
Overview of AD process
How the process is developed (end-to-end)?

• Step 0 : PC5 discussions with each PC.

• (1) Form to be filled in by each Application Demonstrators (ADs) “candidate”.

• (2) Analysis and review of the propositions by the Strategic Committee of PC5.

• (3) Prioritise the Application Demonstrators and final selection.

• (4) Identify and prioritise co-design motifs (compute & communication patterns)

• (5) Execute associated developments as co-design projets in iterative increments.

• Iterate as needed for new ADs
1. **Document the science and engineering Exascale challenge**
   - why and how is it significant?
   - what new Exascale capacity and/or capability does it require?

2. **Document issues/barriers and targeted development needs**
   - physical models, algorithms, software components,
   - others: e.g., workflows and load balancing, data movements and I/Os, architecture and technologies, programming and execution environments

3. **Document Application team**
   - human resources and expertise
   - application development methodologies

4. **Document experience in leveraging HPC systems**
   - use of GENCI, PRACE/Euro-HPC and/or other international HPC centres
   - current level of performance and capability of the application

5. **Document a Figure of Merit**
   - measure rate of science work enabled by new performance and functionality
   - identify possible dependencies and trade-offs between targeted developments
   - set base and stretch application objectives
Application Demonstrator Form

How we will review and prioritise?

A. Significance
  - Science and Engineering impact of the challenge problem
  - Challenge problem requires Exascale (capacity and/or capability)
  - Aligned with the strategic priorities of the NumPeX stakeholders,
  - Map with European national and European strategic priorities

B. Breadth
  - Well-identified targeted developments and of their impact on the rate of science work
  - Map with the expertise and the planned research and software technology developments in the NumPeX PCs
  - Foster collaborative developments across a number of NumPeX PC teams

C. Team experience and confidence in
  - Model, software development and deployment
  - Leveraging HPC systems
  - Addressing ambitious Exascale targets

D. Impact into the Science and Engineering computational community
  - Ability to deliver translatable solutions that impact the broader community
  - Synergies with large international projects and/or consortium
Implementation
How Exa-DIP is implemented across NumPEx?

- **WP2/CDT**: steers and iteratively implement a co-design process for the developments across the PCs.

- **WP2/CDT**: important co-design workforce, enabling application-driven iterative coordination between initially loosely-coupled developments in the PCs.

- **WP3/CDT**: takes over integrating applications-driven collection of software components and delivering them using meta-build and container technologies.

- **WP3**: delivers SDKs to be easily deployed and to accelerate the development of Exascale CSE applications.

- **WP1/Strategic Committee**: decides, prioritises and allocates resources.

- **WP1/Program Forum**: steers and manages iterative increments of the global co-design process.
Co-Design Projects
Co-design projects

- Addressing unique cross-cutting algorithmic motif
- Mapping them into research and developments in the PCs
- Federating initially loosely-coupled development teams

Co-designed software technologies:

- Application-driven collection of software components (Libraries, tools, frameworks): interoperable modular instantiations that applications can combine and build upon to strive for performance portability, flexibility and scalability,
- Proxy applications (including physical models) to assess the meaningful application-driven collection of software components and their potential science-rate trade-offs.

Co-design projects executed as Agile Release Trains (ART) with composed of self-organising and self-managing Agile Teams

Co-designed projects:

- **Meaningful**: The meaningful algorithmic motifs and features identified, and the planning package to address them;
- **Capacity**: The number of new capability into an application demonstrator;
- **Impact**: The expected impact of the co-designed deliverables on CSE application development in exascale environments.
- **Breadth**: The number of CSE application demonstrators and NumPeX PCs that are to contribute to and benefit from.
- **Resource/Expertise**: CDT resources and expertise, together with commitment from ADs and PCs’ research and development teams.
Example of a Motif
AMR computational and communication motif

Application Demonstrators:

- Accelerator design, Astrophysics and cosmology, Combustion, Multi-phase flow, ...

A wide variety of AMR-based physical models and algorithms

- Multilevel electromagnetic PIC (plasma accelerators) — electrons as AMR particles, and electric/magnetic fields on hierarchical mesh — heavily leveraging particle-mesh functionality: particle data deposited onto the mesh and mesh data interpolated to particles each time step.

- Explicit compressible hydrodynamics (astrophysics) with Poisson solver for self-gravity, and time integration of stiff nuclear reaction networks.

- Compressible hydrodynamics with self-gravity coupled with N-body dark matter representation (cosmology). Dark matter particles interact with each other and with the mesh-based fluid only through gravitational forcing, stiff source terms.

- Multi-phase flow modelling leveraging both mesh and particle functionality — solid particles within a gas. Major role of particle-particle interactions in the dynamics that take place inside complex geometries (EB data structures). Multi-grid solvers for the dynamic pressure field in a projection formulation, implicit treatment of viscous terms.

- Compressible combustion code using EB methodology for problem geometry, and chemical kinetics solvers. Multi-grid solvers to solve for the dynamic pressure field in a projection formulation and for the semi-implicit treatment of viscous terms. Particles used both as tracer and sprays.

A number of different existing open-source AMR libraries and frameworks

- originally used basic MPI parallelisation: data distributed to MPI ranks, each rank performing operations on its own data.
- introduced some level of hierarchical parallelism through the use of OpenMP directives.
- Moving towards the exascale: increasingly widespread use of many-core CPU only nodes and hybrid nodes with GPU accelerators,
- different types of GPUs have different programming models.

=> A key behind the development of an AMR library and framework is:

- provide a high performance framework for applications to run on a variety of current- and next-generation systems
- minimise architecture-specific development/modification costs for each new platform.
An AMR co-design project shall

- extend and develop libraries and a software framework that provides a unified infrastructure with the functionality needed for AMR applications to effectively and efficiently utilise exascale systems.
- support algorithms that solve systems of partial differential equations (PDEs) in simple or complex geometries, and those that use particles and/or particle-mesh operations to represent component physical processes.
- provide Core elements of the AMR co-design framework including: data containers and iterators, several specialised operations to meet the needs of the application demonstrators, strategy to achieve highly performant codes across a range of accelerator-based architectures for a variety of different applications
- support application program models such as Fortran, C, C++ or other languages that can be linked to C++, together with OpenMP, OpenACC or native programming models to implement their own kernel launches if desired.

AMR co-design philosophy need to

- to separate the design of the data structures and basic operations from the algorithms that use those data structures.
- to provide core software components providing the flexibility to support the exploration, development and implementation of new algorithms for additional performance and science gains.
- allow application developers to interact with the software at several different levels of abstraction.
- to provide performance portability with light weight abstraction layer (such as Kokos) allowing developer/user to specify what operations they want to perform on a block of data.

Application developers must be able to

- simply use the AMR data containers and iterators and none of the higher-level functionality, or
- use the data structures and iterators for single- and multilevel operations but retain complete control over the time evolution algorithm, i.e., the ordering of algorithmic components at each level and across levels

- a layered co-design approach providing developers/users with the ability to have complete control over their algorithm or to utilise an application template providing higher-level functionality.
- stubs for necessary operations such advancing the solution on a level, correcting coarse grid fluxes with time and space-averaged fine grid fluxes, averaging data from fine to coarse and interpolating in both space and time from coarse to fine.
Co-design Software Development Kit and Software Delivery Technologies
Community Software Policies:

- Improve software quality, usability, access and sustainability;
- Inform that a software component can be used with other software components;
- Provide a foundation for deeper levels of interoperability;
- Establish a certification process to label software (maturity, portability, compliance);
- Separation of concern for service and support level by different actors

Co-designed SDK projects:

- Logical application-driven collections of value-added interoperable software components,
- Integrated and packaged using common meta-builder systems enabling combined deployment of software components as needed by CSE applications

Enabling Team: Software packaging and deployment technologies

- Common Meta-builder systems (e.g. SPACK) and container technologies (e.g. Singularity)
- Developing new capabilities to enable deployment of SDKs on HPC systems, including regression tests

Software Integration hub:

- Enabling access to externally managed continuous integration and testing platforms for deploying tests on a large variety of OS and hardware variants, including the ability to track performance and energy efficiency
- Synergies with national computing facilities, CoEs, vendors
- Building an open community gathering application teams, research and software technology developers and exascale system admins.