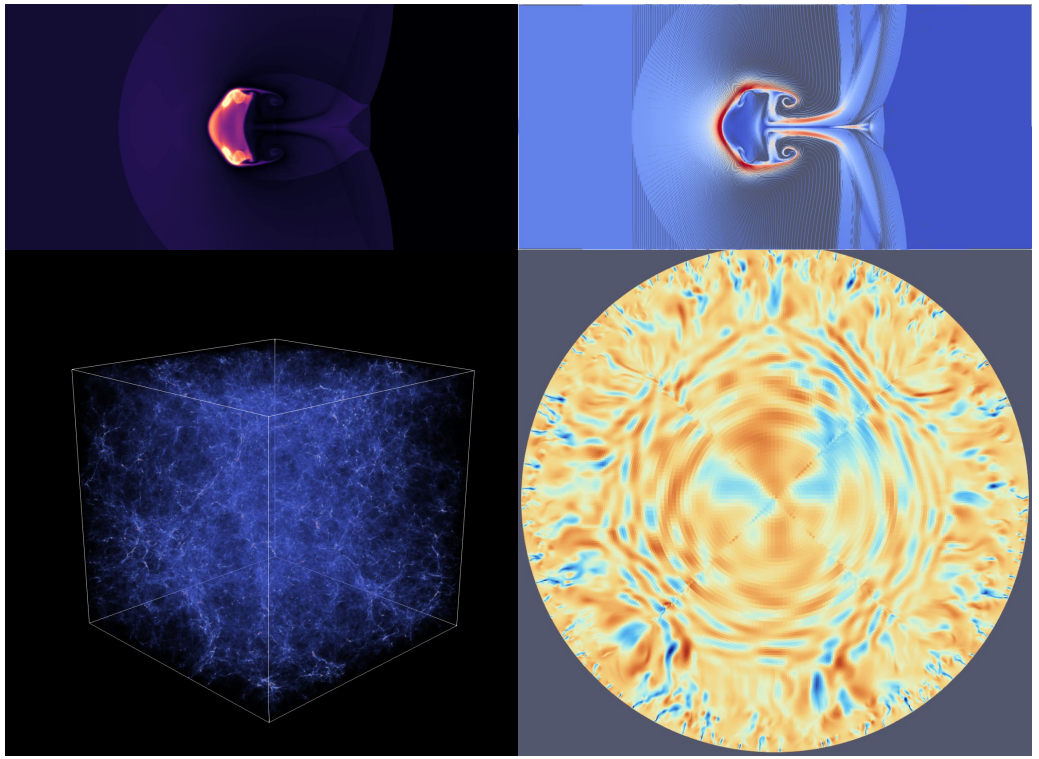
Dyablo:
An adaptive mesh refinement code at the era of Exascale computing



Journées de l'action spécifique numérique

15/12/2025

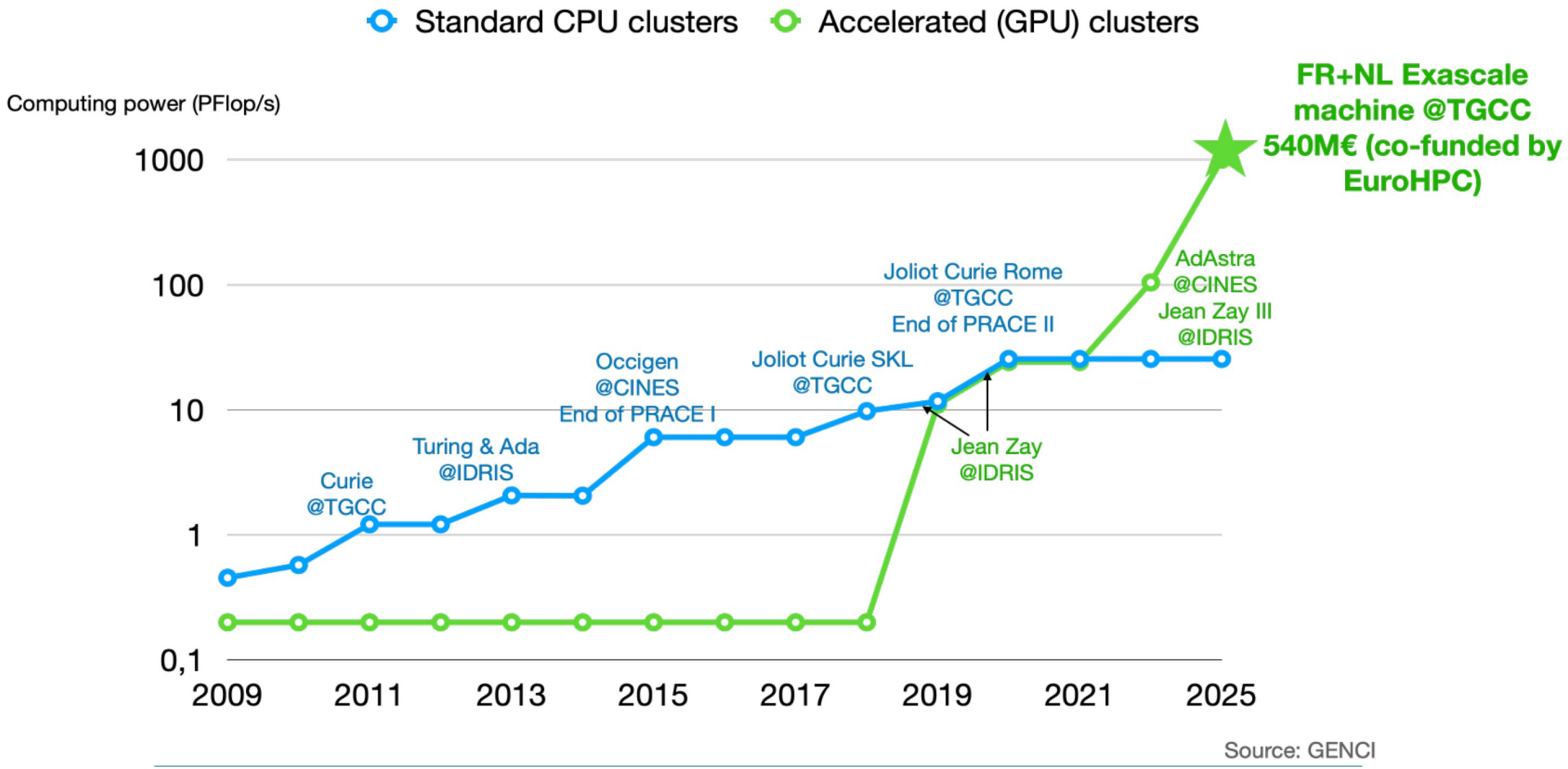
Maxime.Delorme@cea.fr

Arnaud.Durocher@cea.fr

Commit authors: Dominique Aubert (ObAS), Lucas Barbier-Goy (CEA), Catherine Blume (CU Boulder), Michel-Andrès Breton (CEA), Corentin Cadiou (IAP), Grégoire Doebele (CEA), Adam J. Finley (ESTEC), San Han (IAP), Olivier Marchal (ObAS), Mike Petrault (CRIStAL), Leodasce Sewanou (CRAL), Guillaume Tcherniatinsky (IAP)

The problem with Exascale

(Thanks Julien)



Graph courtesy of G. Lesur/PEPR Origins

The problem with Exascale

Variety of architectures/vendors

- GPU is mandatory to reach target resolution/time-to-solution/efficiency
- Each vendor has their own formalism
- CUDA does not work on AMD, HIP does not work on Intel, etc.
- So what do we do ?
- The solution is performance portability:
  - Kokkos (Julien has talked about it before me)
  - SYCL (see Timothée's talk on Wednesday)
- Dyablo is an (one) answer to this.

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,340,000	1,809.00	2,821.10	29,685
2	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	2,055.72	24,607
3	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
4	**JUPITER Booster** - BullSequana XH3000, GH Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, RedHat Enterprise Linux, EVIDEN EuroHPC/FZJ Germany	4,801,344	1,000.00	1,226.28	15,794
5	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	

What is Dyablo ?

The outline:

- Modern code for the modeling of multi-scale/multi-physics astrophysical fluids
- Block-based AMR formalism
- Written in modern C++ (20 in a couple of days !)
- Relying on Kokkos for performance portability and MPI
- Open-source and community-centered
- Modern software engineering concepts
- Benefit from common needs between various astrophysics fields
- Let’s start from scratch and question everything “we do as we always did”

Open source and community centered

Open-source:

- Developments are made on CEA hosted gitlab: https://drf-gitlab.cea.fr
  - Anyone can create an account
  - Three repositories: Dyablo, pyablo, documentation
  - Protected branches (dev, main) are mirrored automatically on github: https://github.com/Dyablo-HPC/
  - Documentation is automatically built with readthedocs: https://dyablo.readthedocs.io/en/latest/
  - Collaboration on day-to-day basis via Slack
  - *Dyablo dev meeting* once every month

Collaboration:

- GINEA: Groupement d'Instrumentation Numérique pour l'Exascale en Astrophysique
- ERC Synergy WholeSun
- PEPR Numpex: Exa-DoST (PC3); Exa-DI (PC5) (see Julien's talk @11:00)
- PEPR Origins: MHD@Exascale (see Guillaume's talk @11:30)
- EUPEX (WP3)
- Moonshot CExA

Software engineering concepts

Separation of concerns

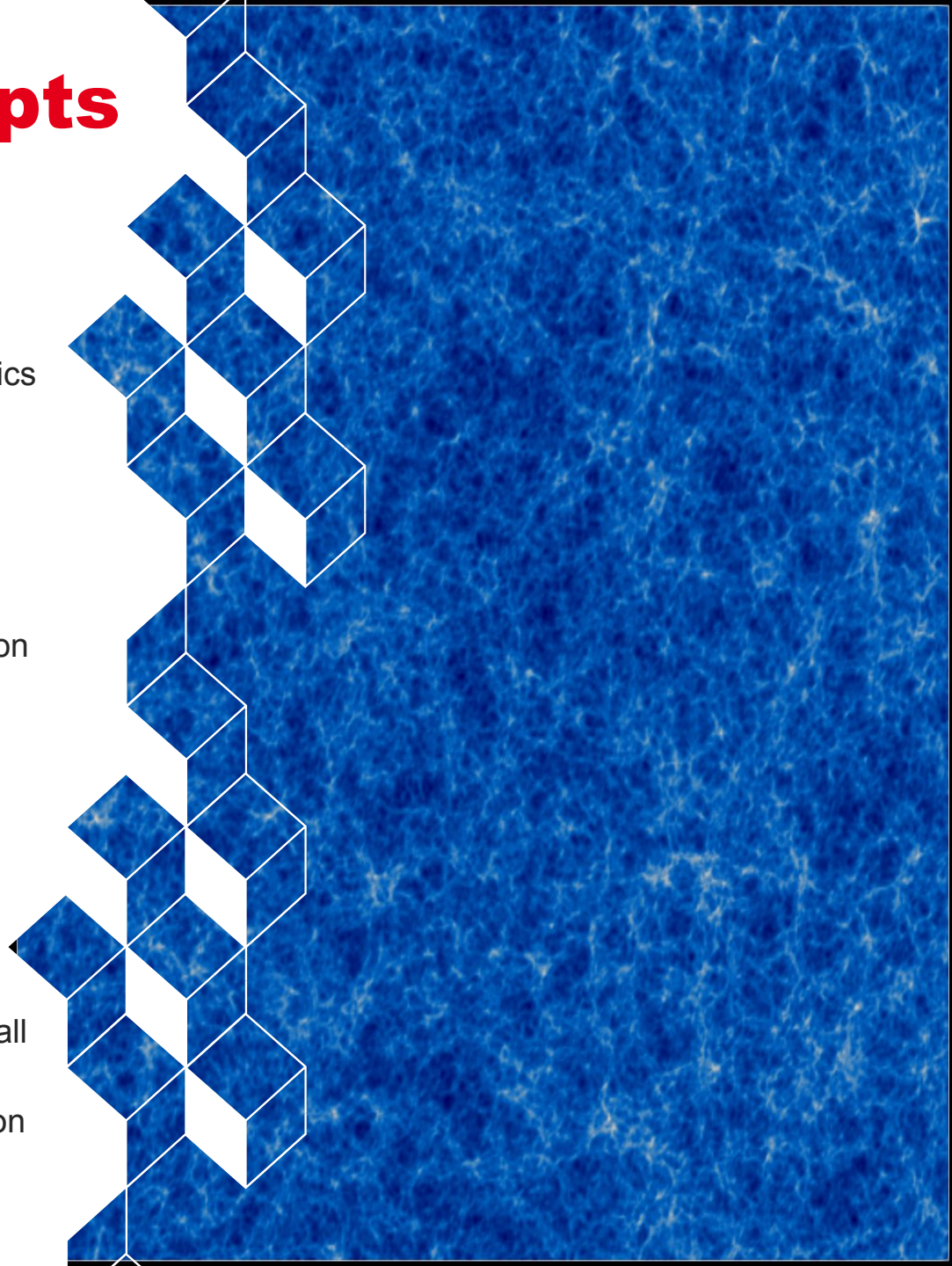
- Physicists implementations should not interfere with HPC
- Back-end implementation should be modifiable without rewriting the physics
- Interface between back-end and physics kernels should stay as stable as possible

High-level abstractions

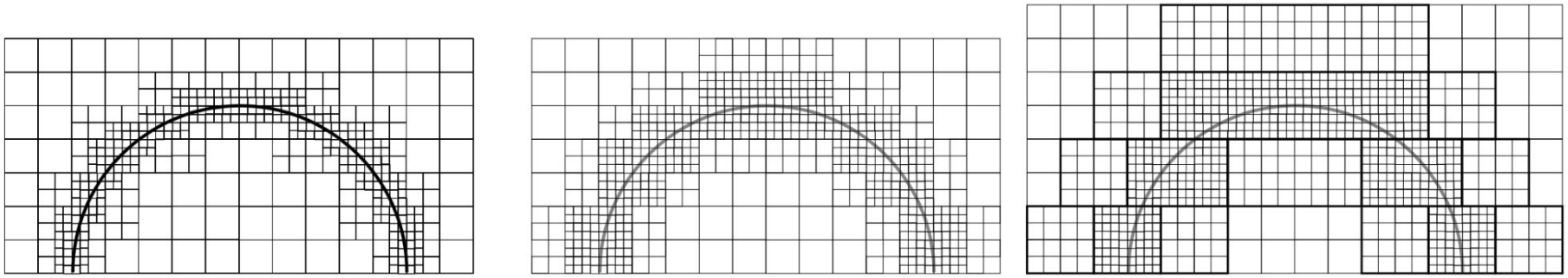
- Having access to high-level classes hides the complexity of implementation
- Easier implementation of physics without having to bother about memory layout/management
- Example: The `CellIndex` class manages neighbor finding for AMR

Modularity

- C++ templating + performance portability = long compilation times
- To avoid having to recompile the code (de)activating modules à la Pluto, all major features of Dyablo are *“plugins”*
- Plugins are compiled once and can be activated/deactivated at initialization instead of compilation.



Block-based AMR

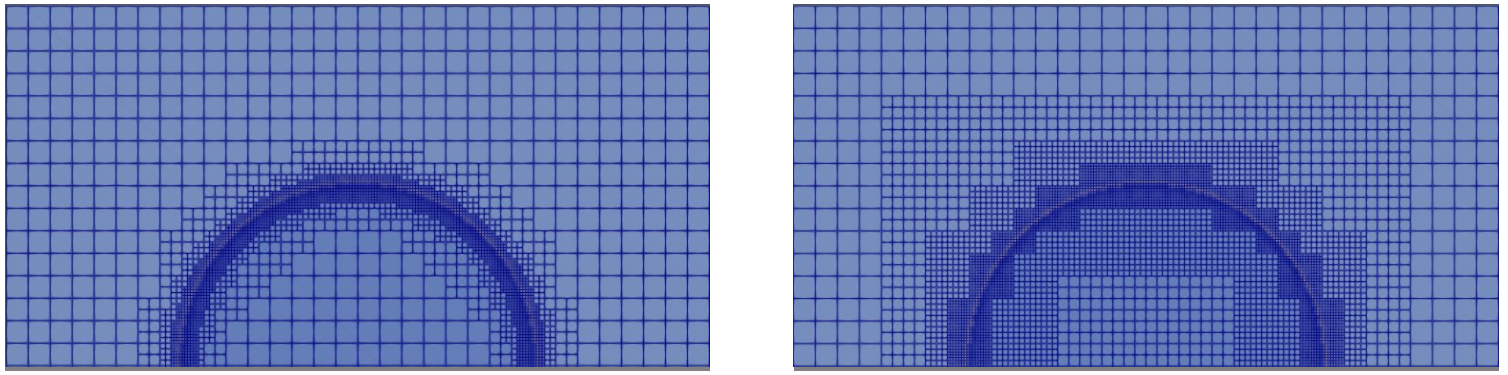


Source: Dunning et al. 2019; "Adaptive Mesh Refinement in the Fast Lane."

Advantages of block-based AMR:

- **Smaller tree:** AMR cycle is faster
- **Increased regularity**: More conformal faces = easier streamlining for the GPU
- Cell-based still possible by taking blocks of size 1

Cell-based
600k octants
600k cells

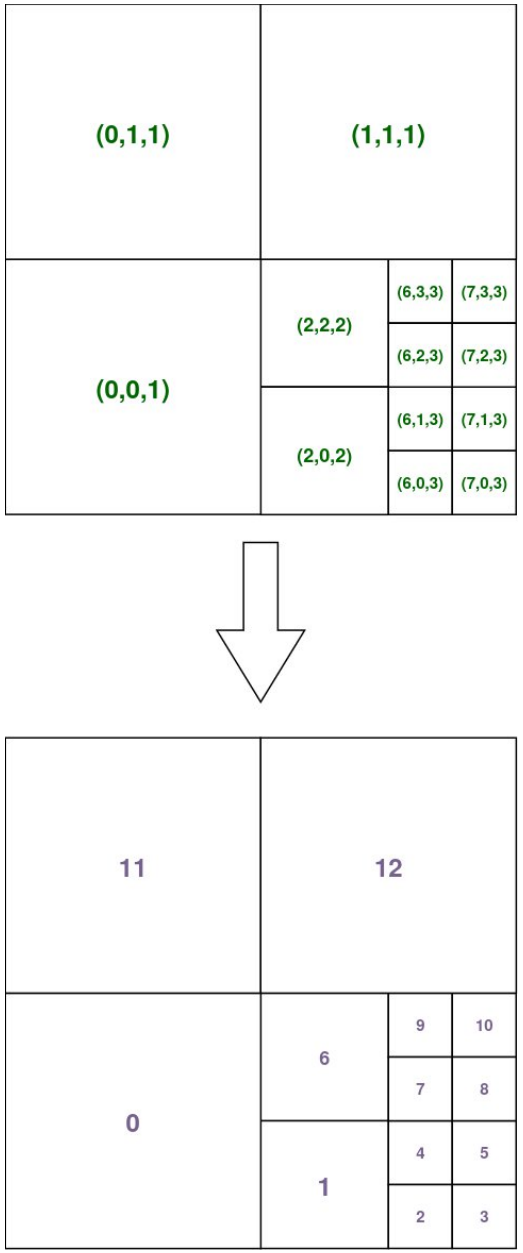


Block-based
18k octants
4x4x4 blocks
1100k cells

Hashmap based neighbor search

Memory layout:

- Linear octree: only the leaves (ie blocks) are stored
- Space-filling curve: octants are ordered along the Morton Z-curve
- All octants are referenced in a hashmap: (i, j, k, level) -> octant_id
- Looking for a neighboring octant is simply looking for matching keys in hashmap:
  - Construct the neighbor key, if it exists return the stored id
  - If not, construct the neighbor key at level-1
  - If still nothing, construct the neighbor key at level+1
- Allows the data to live at all times on GPU memory for computation



Plugins

Plugins are feature “bricks”

- Every aspect of the code that should be modular is made as a plugin
- Static inheritance and a *factory design pattern* to instantiate the plugins at runtime
- Plugins can simple simple objects:
  - Dt calculation, refinement criterion, IO backends, etc.
- Or more complex, multi-layered classes
  - HydroState + HLLC + Slope limiters + BoundaryConditions = Hydro policy
  - Hydro policy + RK2 scheme = Hydro_RK2 plugin
- Plugins are registered in generic factories at compile time
- And selected in .ini file at runtime

```
FACTORY_REGISTER( dyablo::HyperbolicUpdateFactory,
                  dyablo::HydroUpdate_RK2,
                  "HydroUpdate_RK2")
```

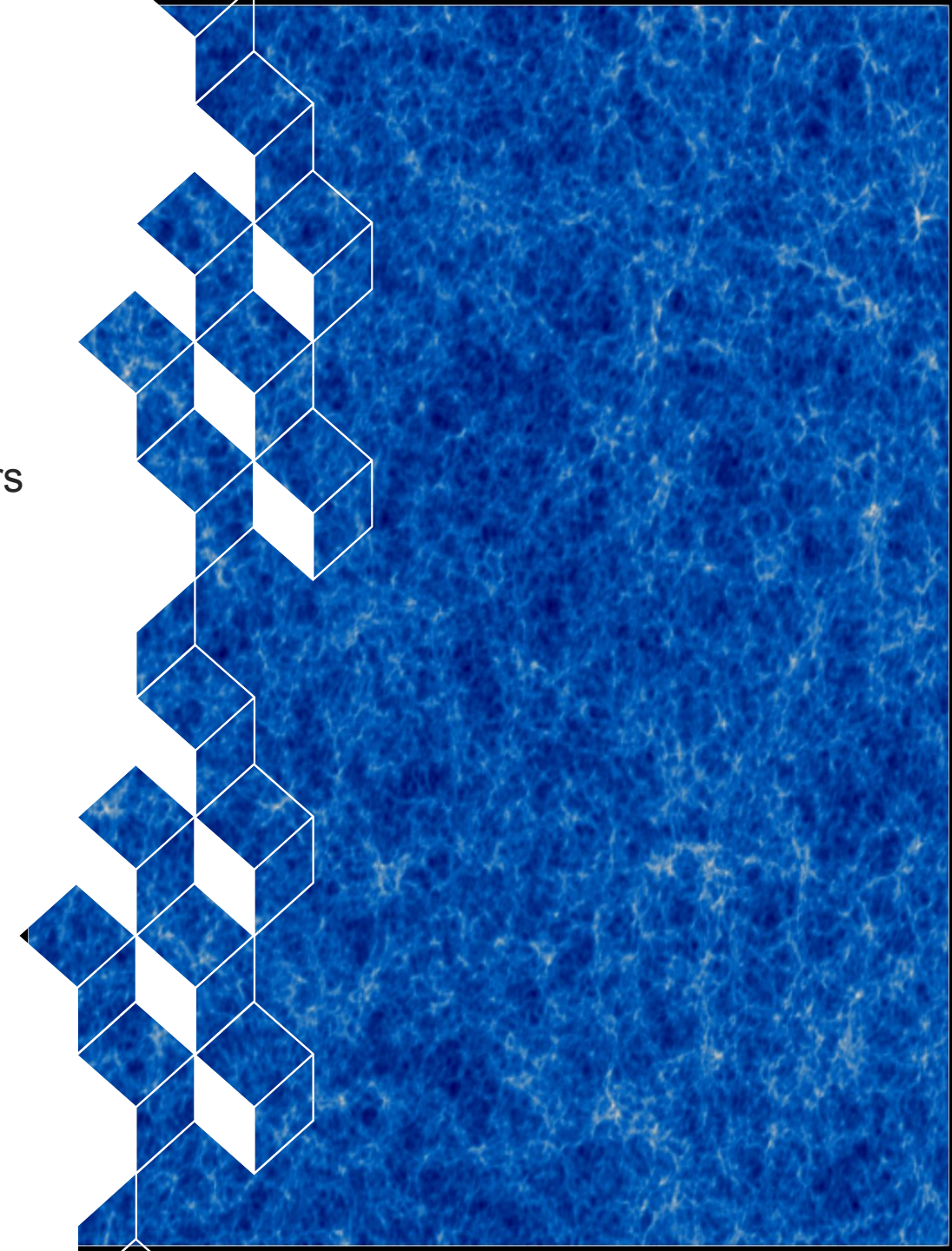
```
std::string godunov_updater_id =
  configMap.getValue<std::string>("hydro", "update", "HydroUpdate_hancock");
this->godunov_updater = HyperbolicUpdateFactory::make_instance( godunov_updater_id,
  configMap,
  this->m_foreach_cell,
  timers
);
```

```
[hydro]
update=HydroUpdate_RK2
```

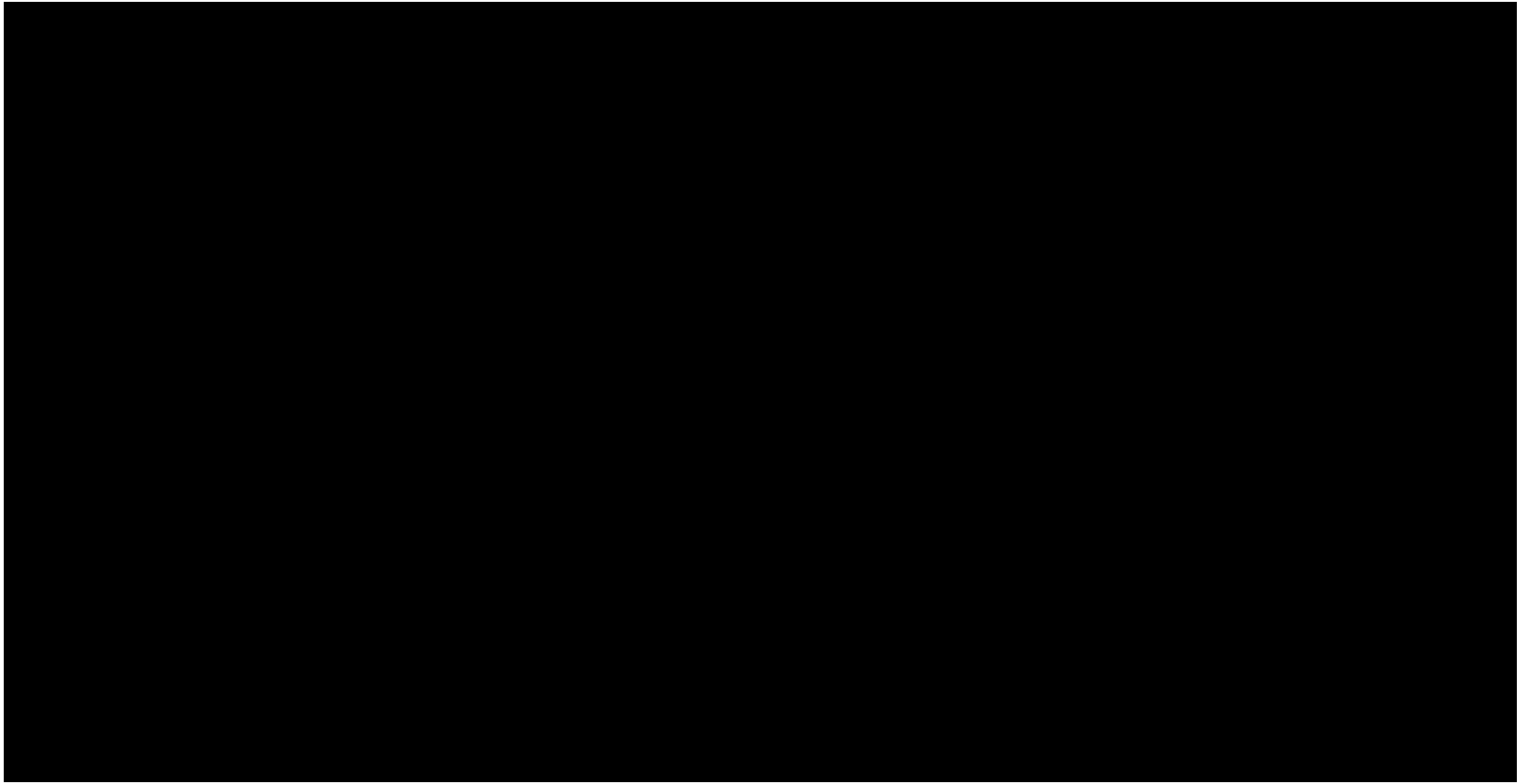
What is in Dyablo ?

Multiple branches:

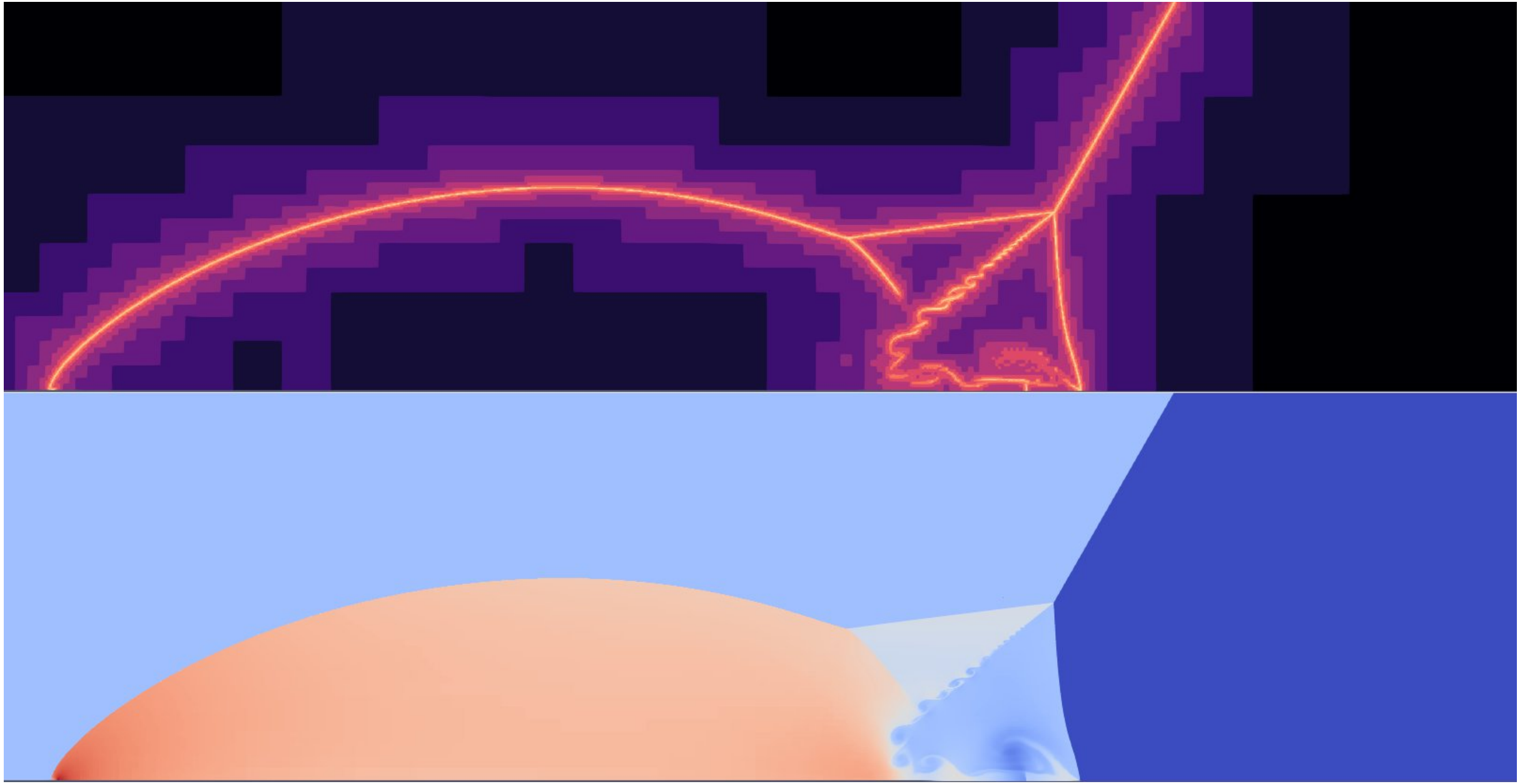
- Dev branch with common useful plugins:
  - Hydro/MHD(div cleaning)/RHD(M1 explicit) solvers
  - Thermal Conduction/Viscosity explicit parabolic solvers
  - (self) Gravity update with CG
  - Particles CIC, NGP, tracers
  - Source terms: basic cooling,
- Community branches:
  - Branches led by community applications
  - Two main community branches for now
- Feature branches:
  - Sub-branches made for implementing features



Double Mach reflection



Double Mach reflection



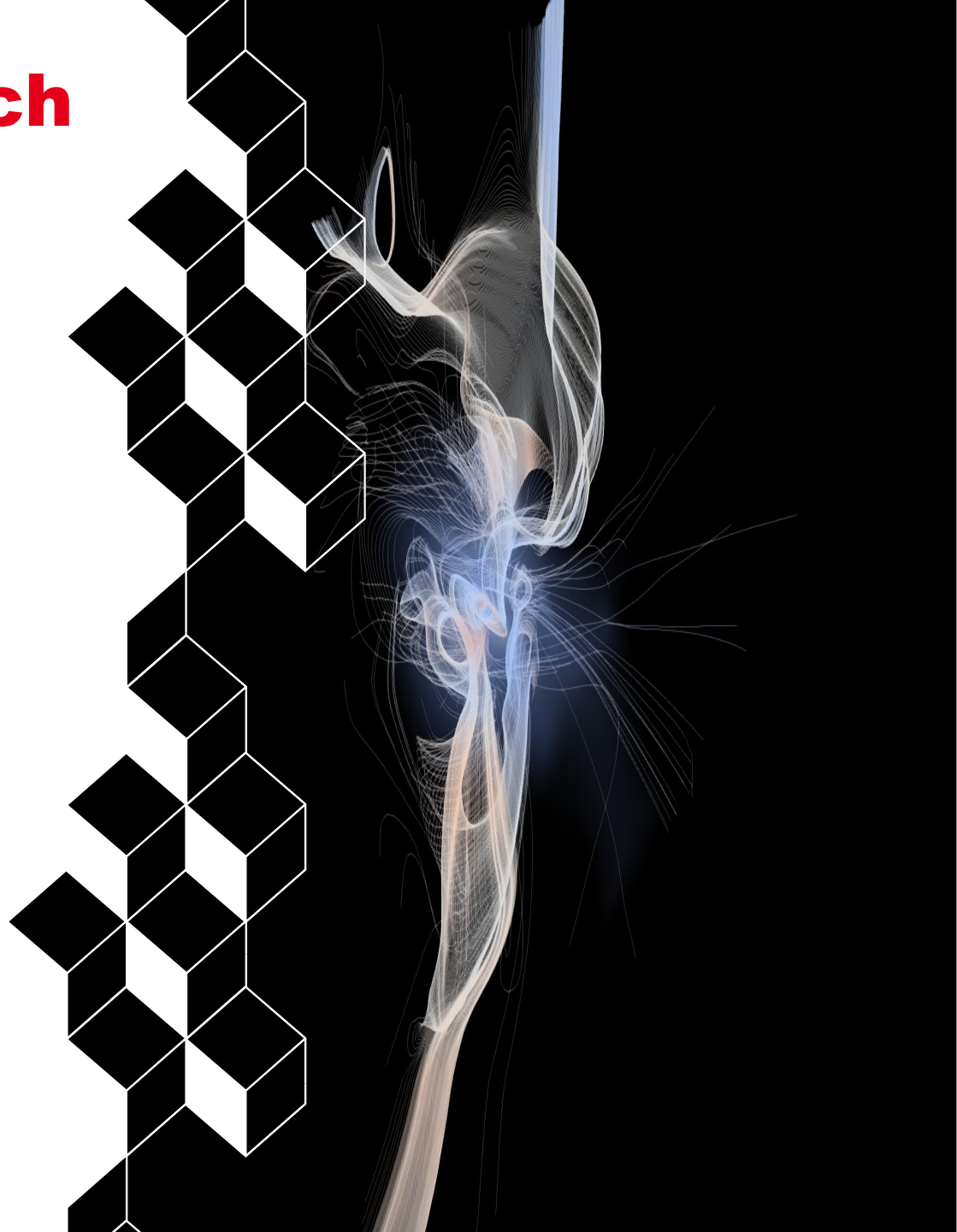
Developments: `cosmo` branch

Exclusive features:

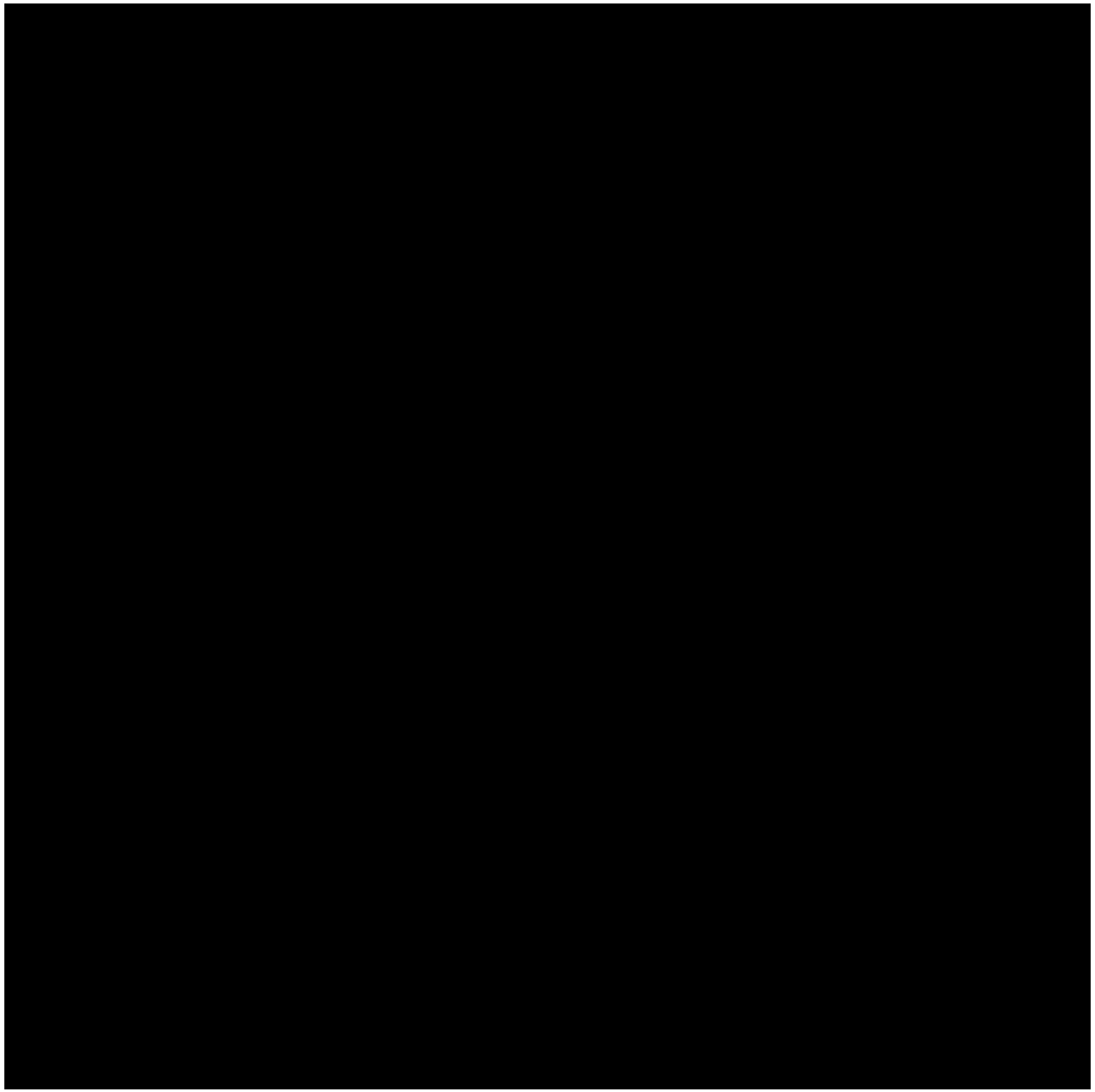
- **Hydro:** Pressure fix (not finalized)
- **Source Terms:** Cooling using Grackle public tables, turbulent forcing term (not finalized)
- **Particles:** Stellar feedback, Star formation
- **Gravity**: Multigrid Poisson solver
- **IOs:** Reading from Gadget snapshot (not finalized)

Contributors:

- Dominique Aubert – ObAS
- Michel-Andrès Breton – CEA Saclay
- Corentin Cadiou – IAP
- Olivier Marchal – ObAS



A first galaxy with Dyablo



Video courtesy of: Dominique Aubert, Michel-Andrès Breton, Corentin Cadiou, Olivier Marchal,

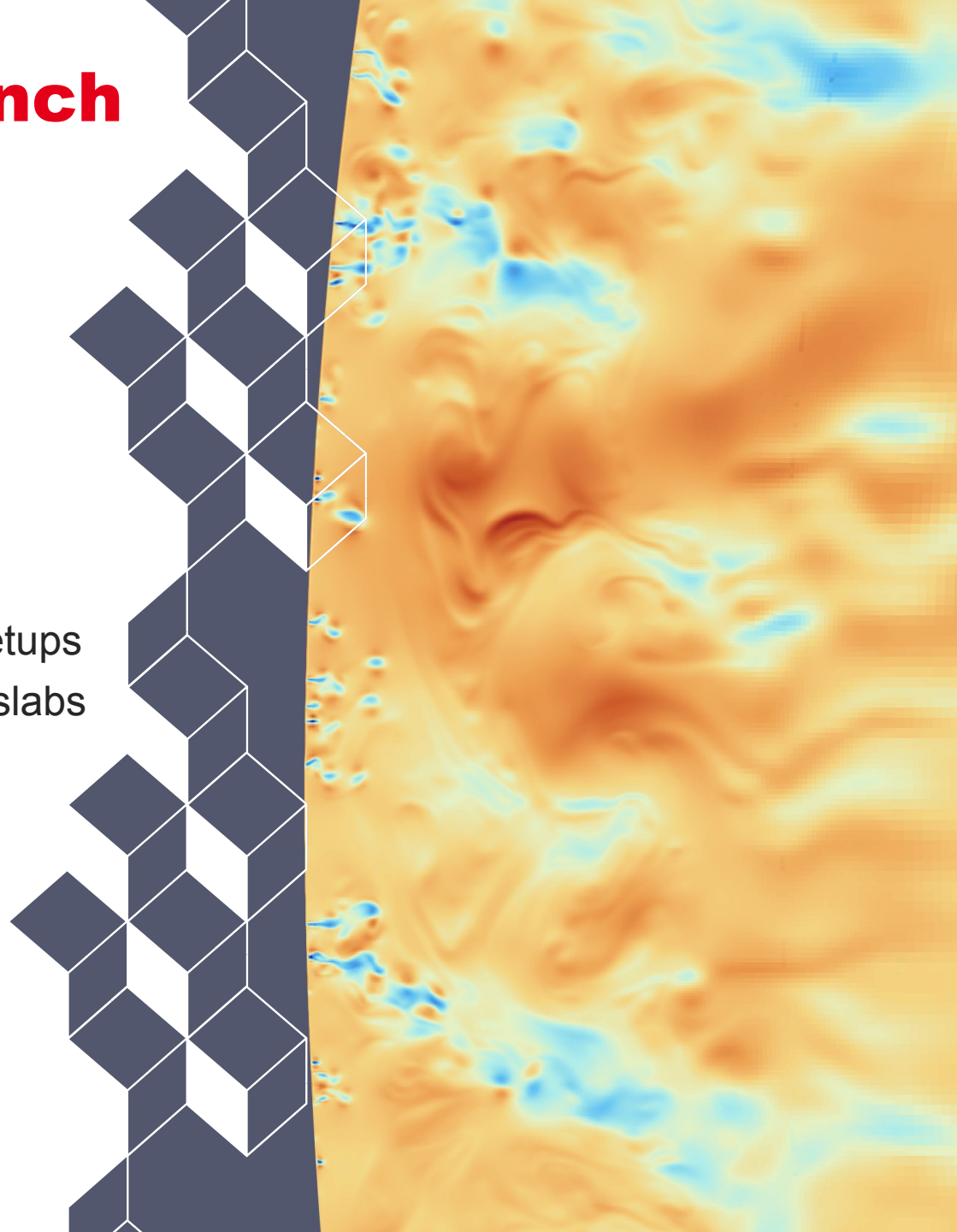
Developments: `wholesun` branch

Exclusive features:

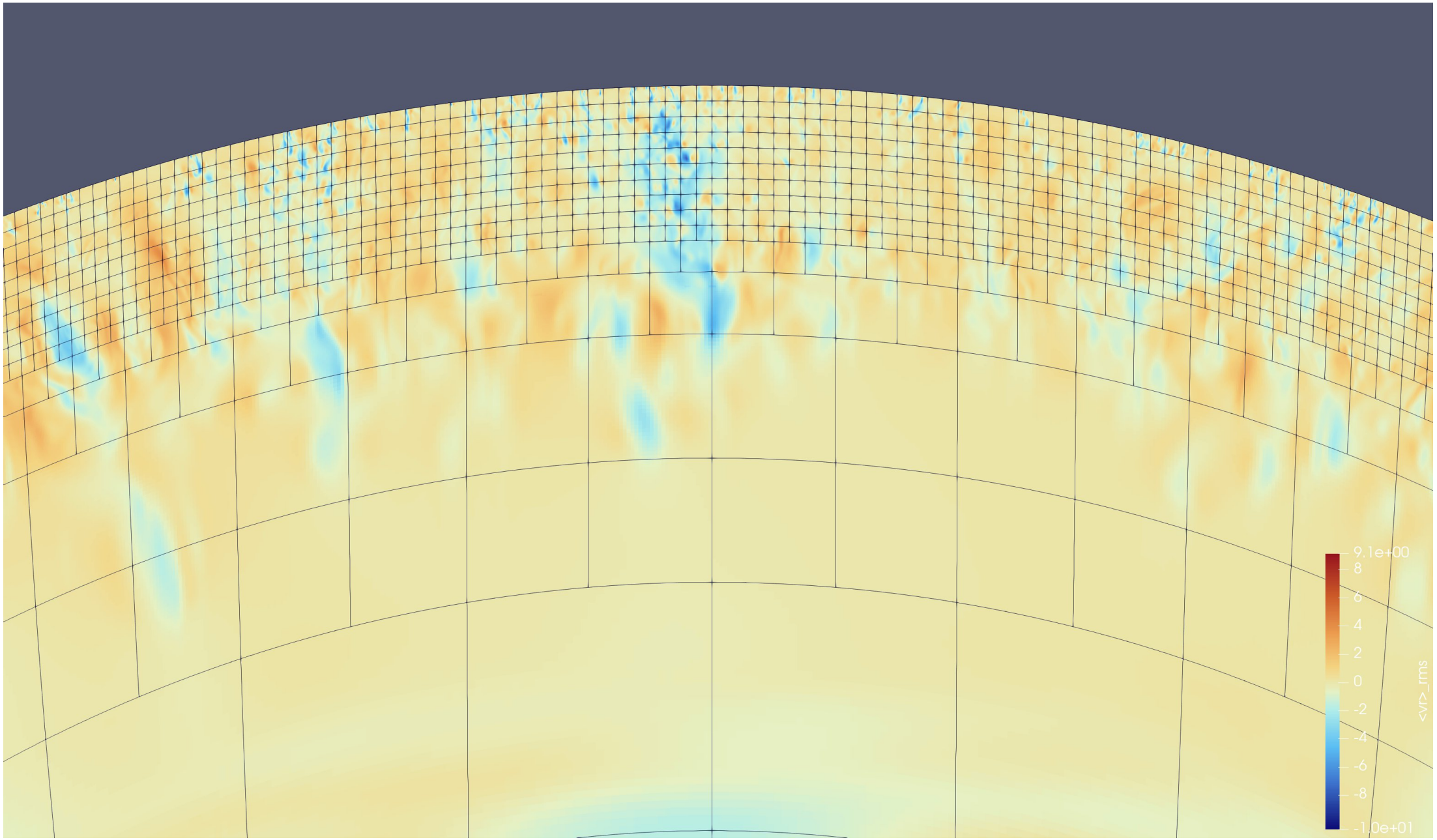
- **Geometry:** Isometric mappings (See Grégoire's talk on Wednesday)
- **MHD:** Five-Waves solver from MDLS
- **Source Terms:** Isothermal atmosphere cooling
- **Well-balancing:** Alpha-beta well balancing for radial setups
- **Setups:** Various setups for solar physics ranging from slabs to radial profiles

Contributors:

- Lucas Barbier – CEA Saclay
- Catherine Blume – University of Colorado Boulder
- Grégoire Doebele – CEA Saclay
- Adam Finley – ESTEC Leiden

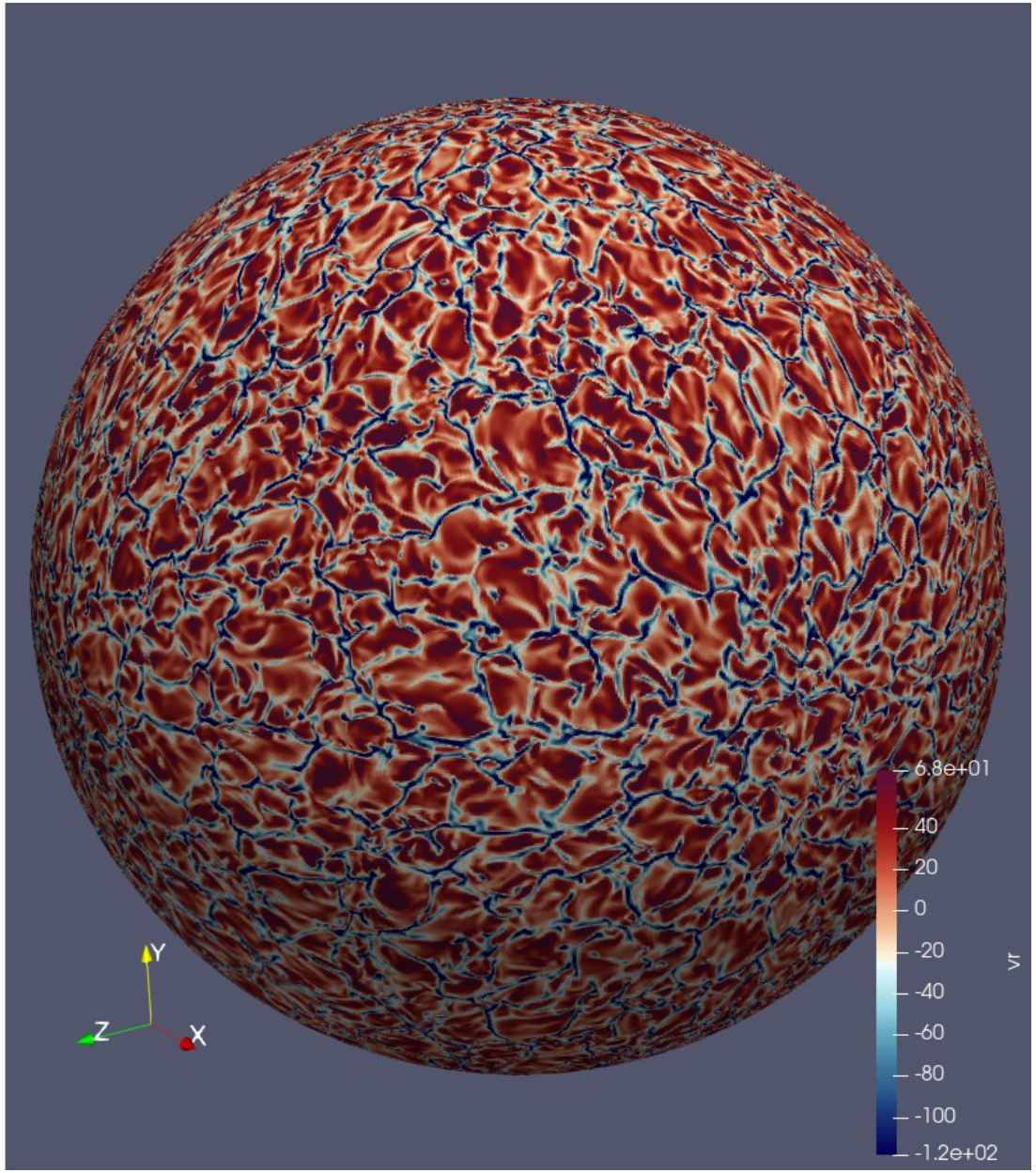


Solar-like setup with geometric module simulation (and AMR)



Edges indicate blocks, not cells ! Running on 1 h100 on Jean-Zay

Solar-like setup with geometric module simulation (in 3D)



$N = 512\text{^}3$ *fixed resolution run; 64 MI250X on AdAstra*

Developments: other branches

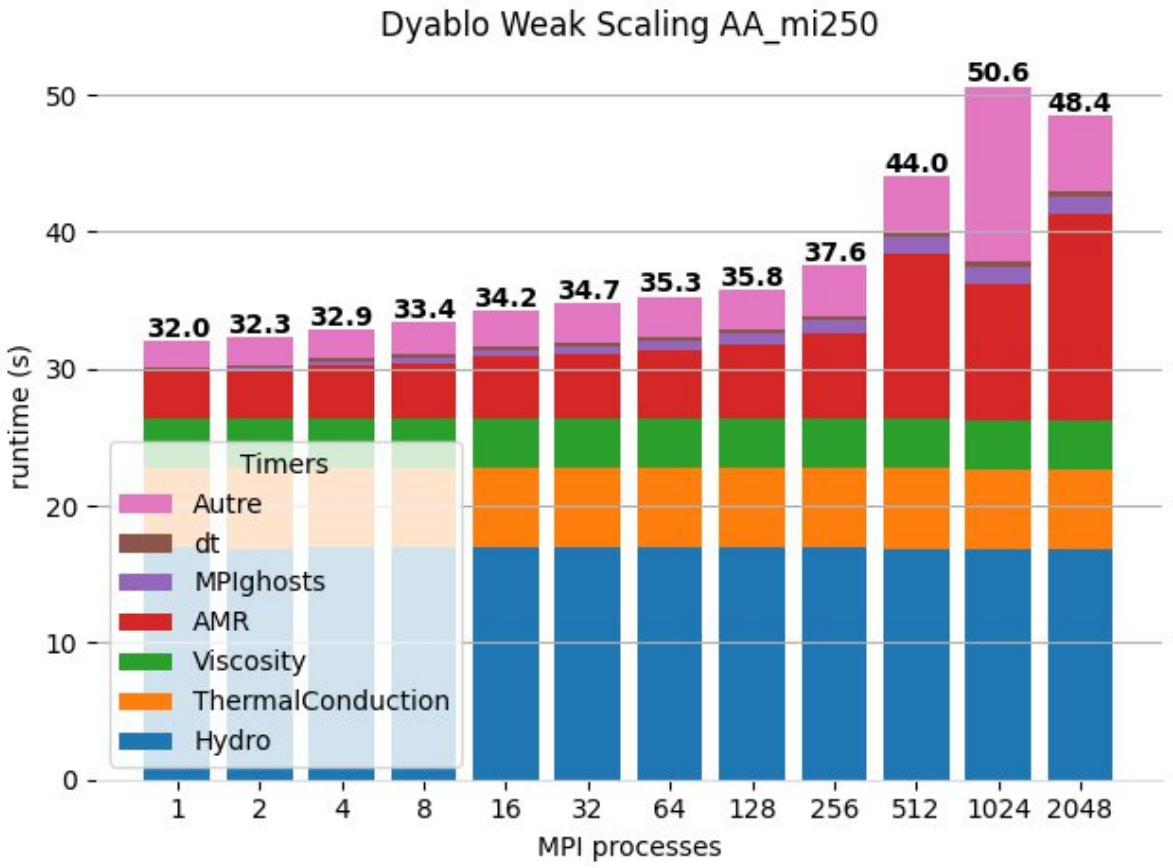
Other branches (development or future community):

- **Multigrid and TSC scheme:** Michel-Andrès Breton
- **Dust/ISM:** Benoît Commerçon, Leodasce Sewanou
  - Base for a future community branch (see maybe Leodasce's talk on Wednesday)
  - A collapse with barotropic equations has been tested in another branch -> Works but BG is killing the time-to-solution
- **Subgrid physics/comparison with Ramses:** Florent Bréhard, Mike Petrault, Jenny Sorce
- **Cosmic rays:** Yohan Dubois, San Han, Guillaume Tcherniatinsky
- **Core features** - Passive scalars, hierarchical timestepping, intermediate level storages: Maxime Delorme, Arnaud Durocher

What about exascale ?

Weak scaling benchmark:

- **Solar physics setup**: Navier-Stokes + Gravity + Viscosity + Thermal conduction
- 3-7 refinement levels: Max Resolution: 2048x2048x512
- 30.6Mcells per domain (1 GPU)
- 100 iterations
- No load-balancing
- Scalability tested on French infrastructures:
  - Up to 128 a100 on Jean-Zay
  - Up to 2048 mi250X on AdAstra (~60 billion cells)
- Data is quite old now (november 2023): needs to be updated.



Roadmap for Dyablo v1.0

MULTIGRID	PASSIVE SCALARS	HIERARCHICAL TIMESTEPPING	DOCUMENTATION	BENCHMARKING	DYABLO V1.0
• Integration of the multigrid branch • Adaptation of the general structure for intermediate level storage/communication	• Passive scalar advection regardless of scheme used	• Partial integration of levels • AMR cycle in the middle of an integration cycle	• Finalizing contributor guide • Doxygen/Breathe • CI integration	• Re-run of all benchmarks: hydro, MHD, cosmo, solar. • Comparison with state of the art	• Public tag • Method paper

A lot of work is still needed to reach a first stable official version of the code !

Current and future work

Peripheral developments/activities:

- **Numpex:**
  - ExaDOST (PC3): Sylvain Joube (postdoc) is working on a new solution for compressed AMD data-formats, visualization and analysis.
  - ExaDI (PC5): Dyablo and Samurai (Polytechnique/CMAP) are leading a working group on a series of benchmarks for modern AMR codes to pinpoint common difficulties
- CExA/PTC: Jean-François David (postdoc) is building analysis tools for the profiling and optimization of large kernels in Kokkos applications

Future work for the core team:

- Python front-end:
  - Being able to fully control the time-loop and let C++ handle the computations
- In-situ analysis:
  - Use the supercomputers to calculate diagnostics and first step analysis while the code is running

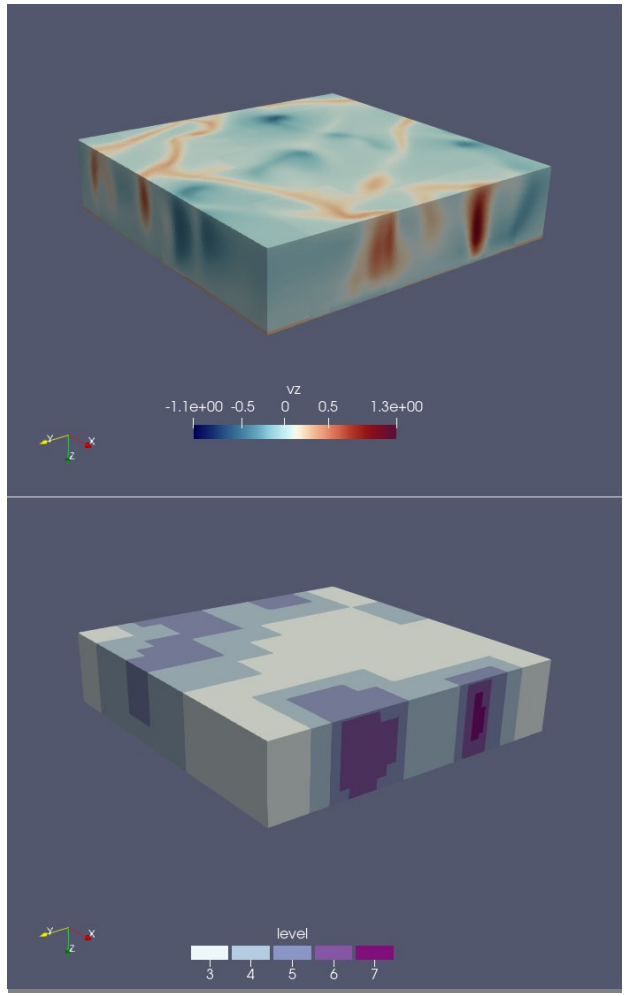
Thank you

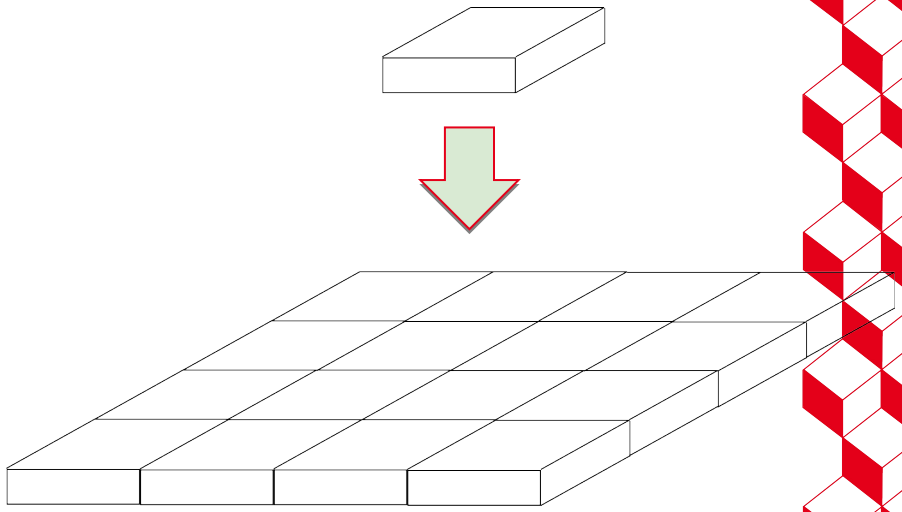
Weak scaling benchmarks

Use case

Solar convection slab :

- 3-7 refinement levels
  - Base resolution 128x128x32
  - Max resolution 2048x2048x512
  - 30.6M cells per domain
- Horizontal tiling per MPI process
- 100 iterations
- 1 AMR cycle per iteration
- No Load-balancing
- Scalability tested on Jean-Zay and Ad-Astra [Tier 1 french SC]
  - CPU : Intel CSL, AMD Genoa
  - GPU : v100, a100, MI250X
  - Tested up to 2048 GPUs ~62 billion cells

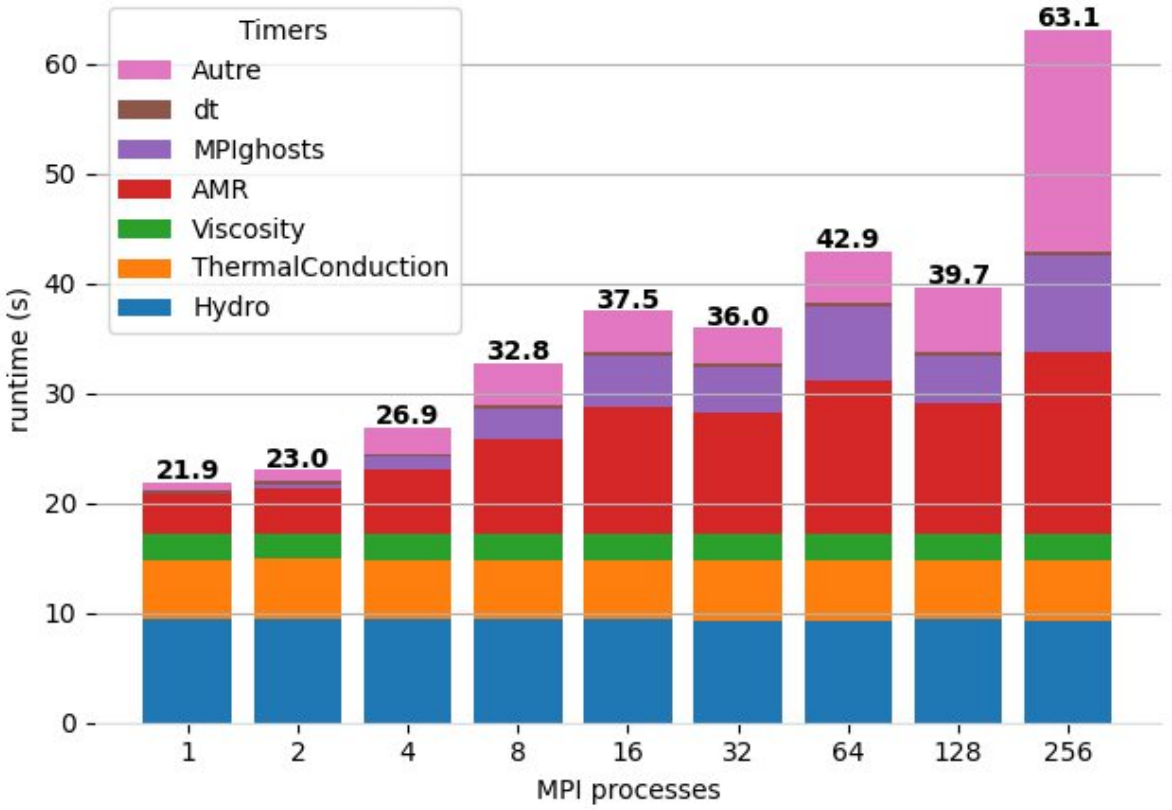




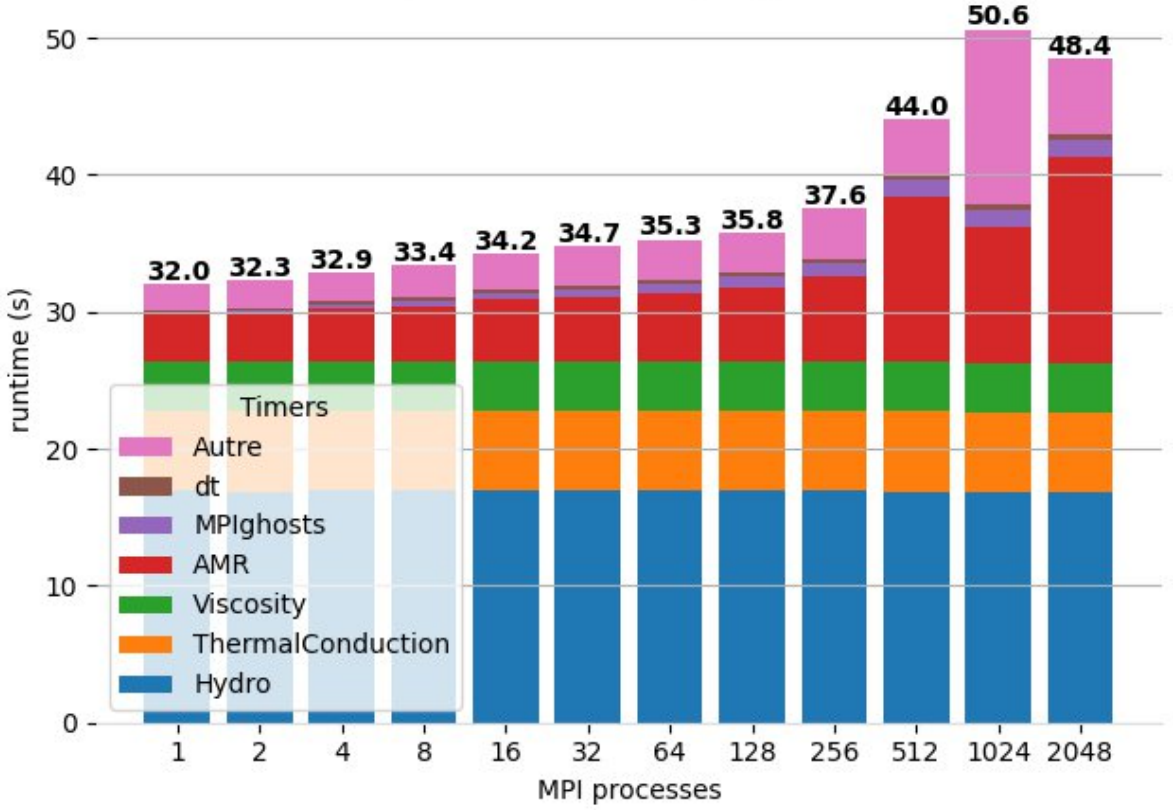
Replication on N MPI processes

Weak scalability for convection runs

Jean Zay GPU *(8 x Nvidia A100)*



Ad Astra GPU *(8 x AMD MI250x)*



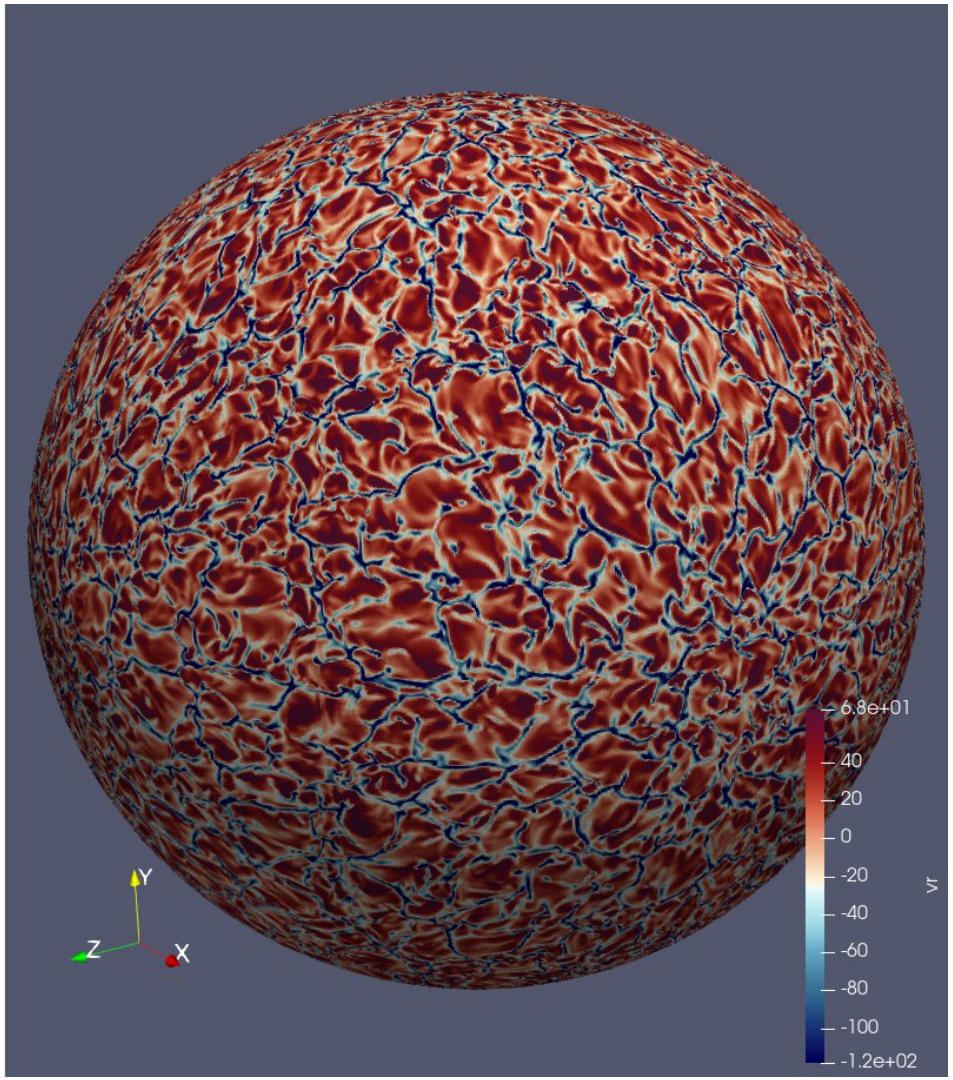
Strong scalability benchmark

Use case

Convection on a sphere:

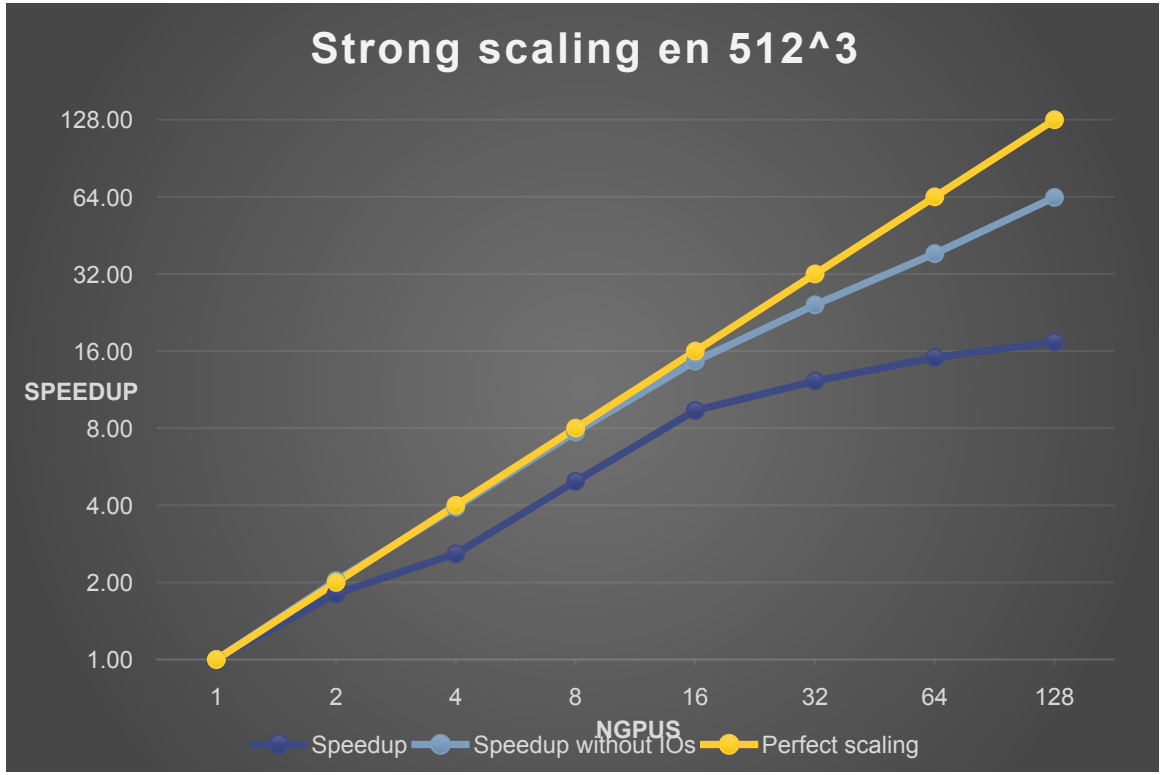
1. Equations:
    1. Navier-stokes
    2. Thermal conduction
    3. Gravity
    4. Heating at the center
2. Boundary conditions: Fixed energy flux at surface
3. 512^3 fixed resolution (130Mcells), solved on a radial mapping

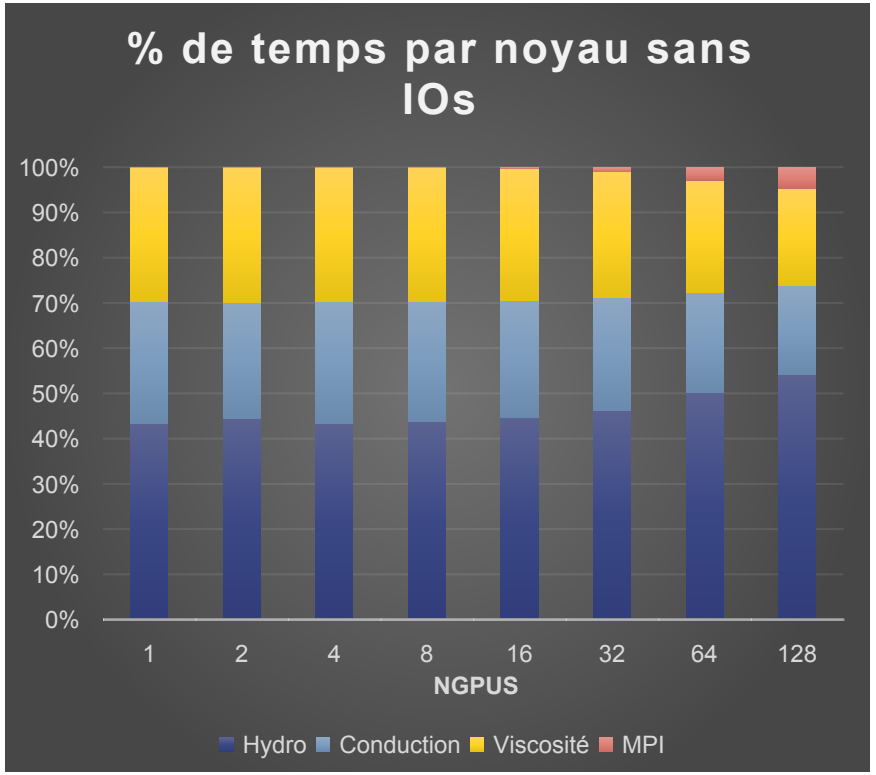
Warning: Geometry module is still very experimental and numbers should be taken with a pinch of salt.



Strong Scalability

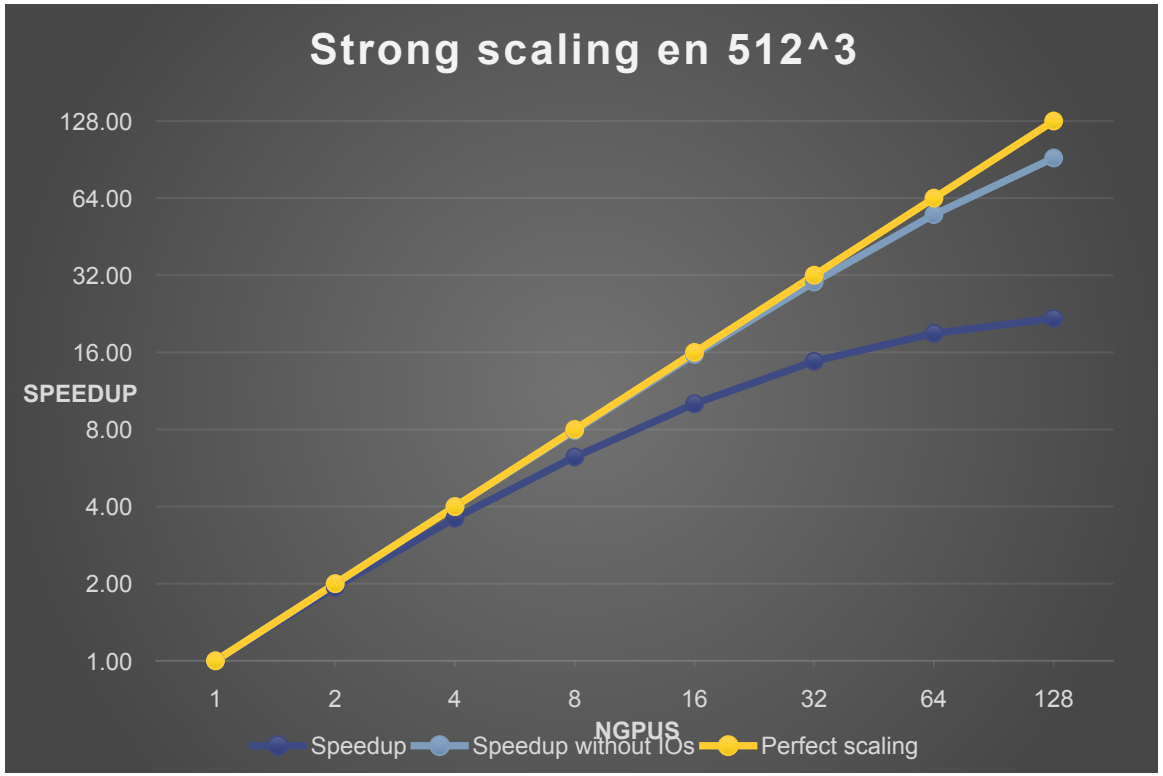
Jean-Zay H100

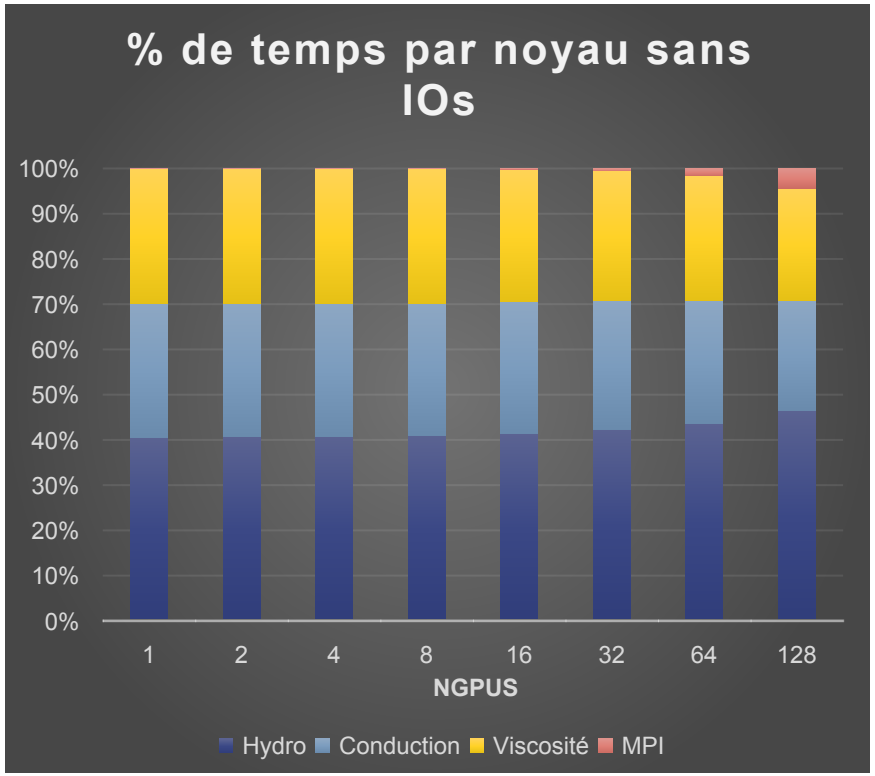




Strong Scalability

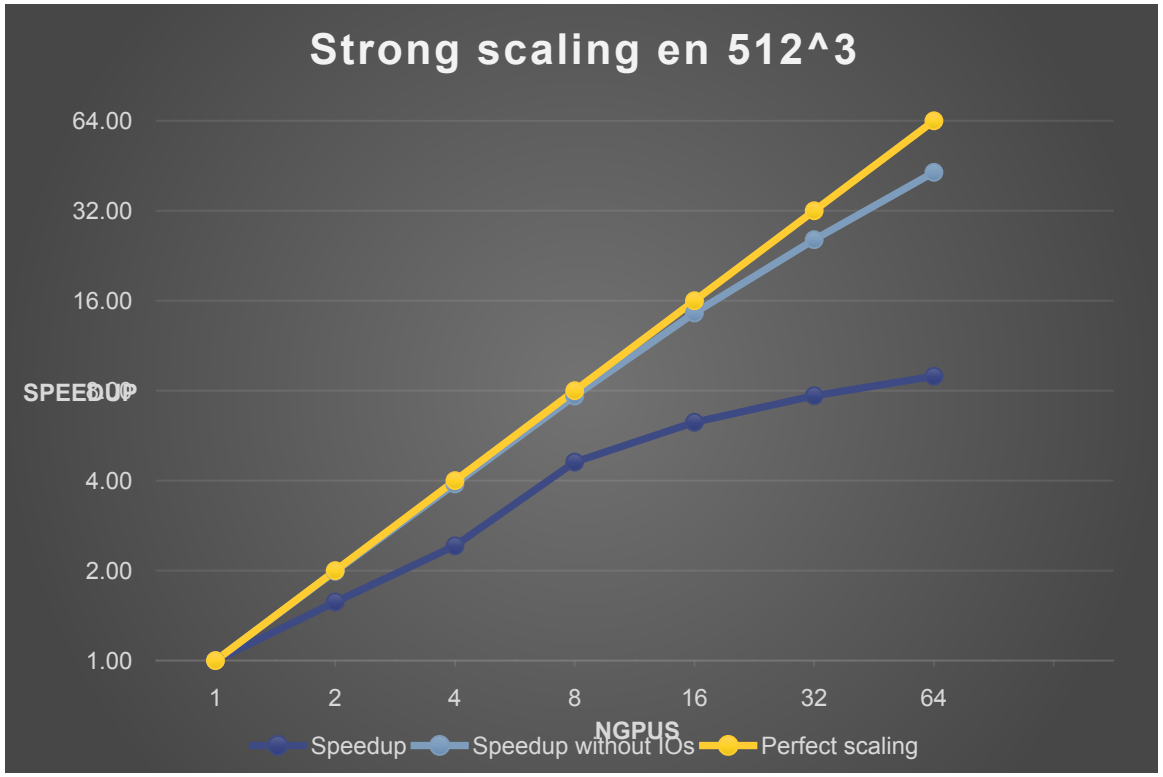
AD-Astra MI250X

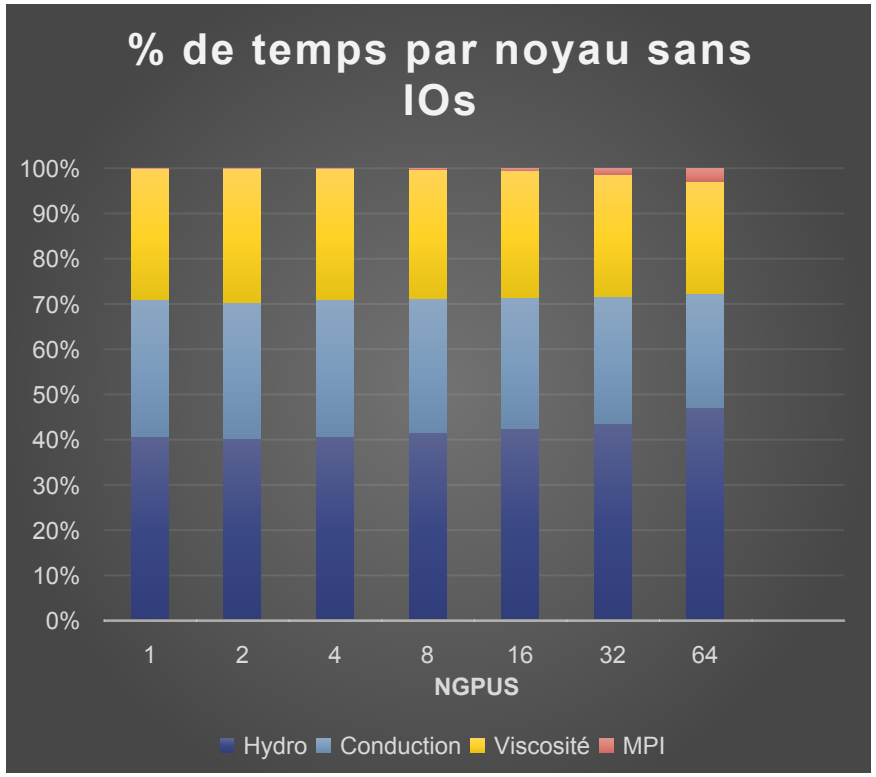




Strong Scalability

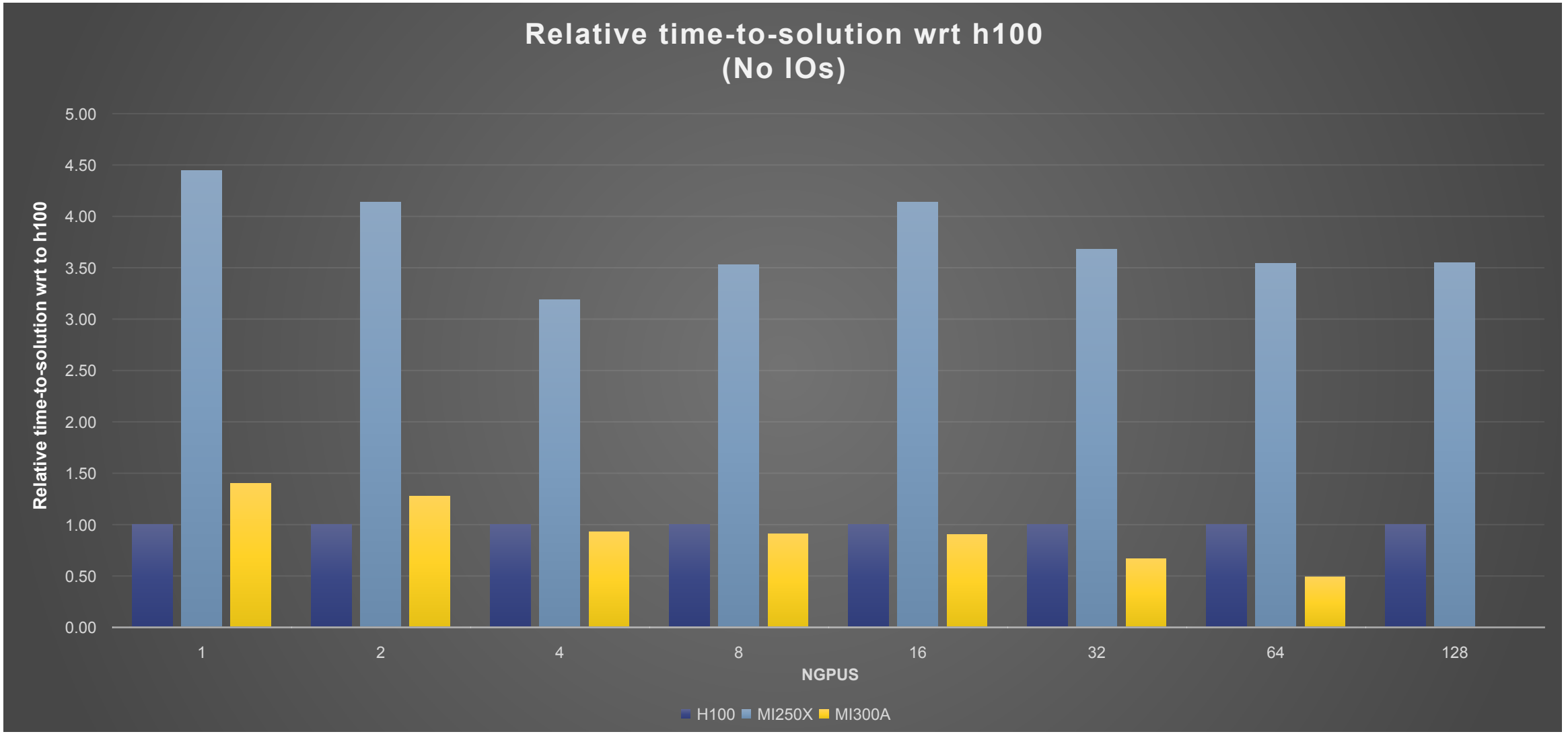
AD-Astra MI300A





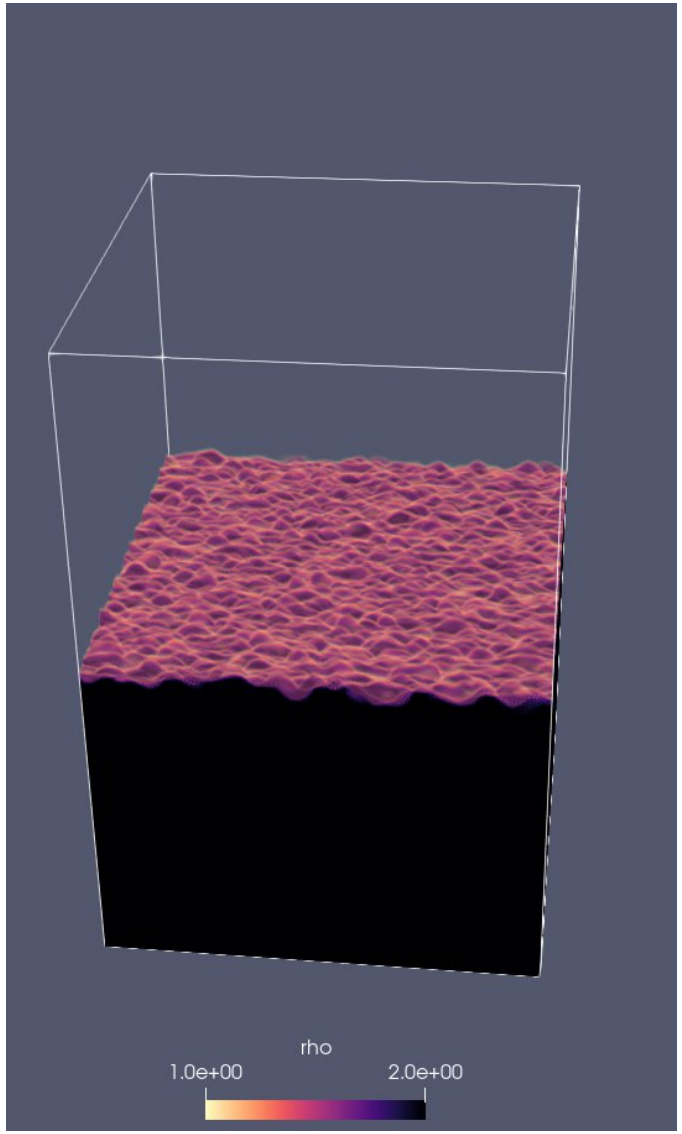
Strong Scalability

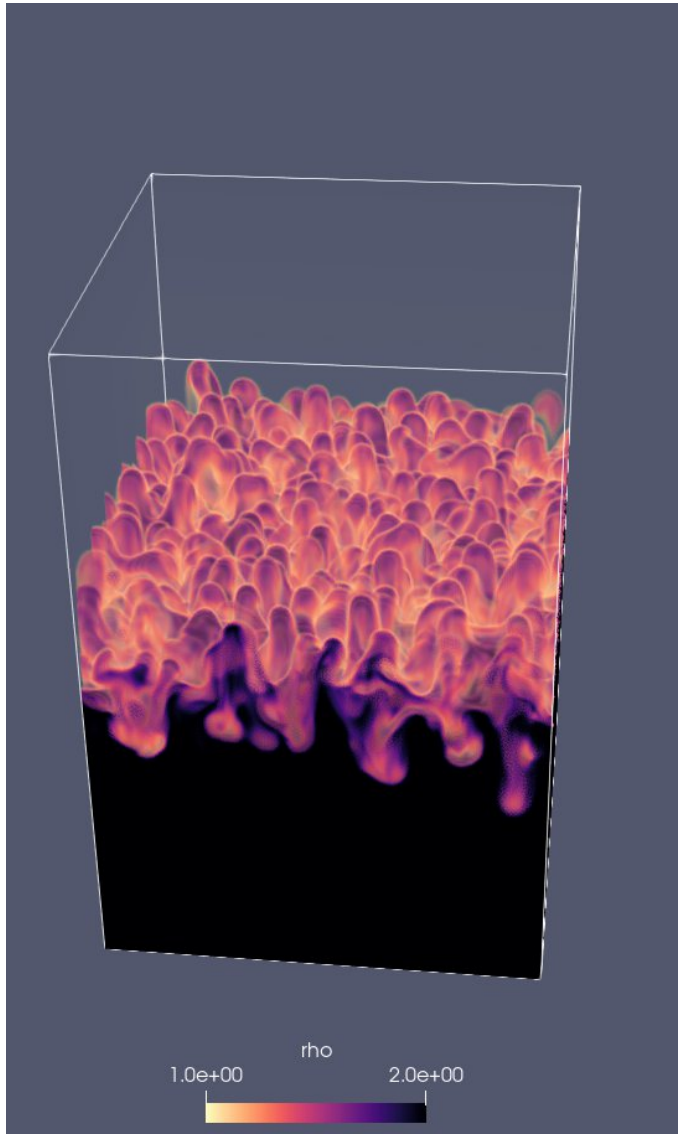
Architecture comparison



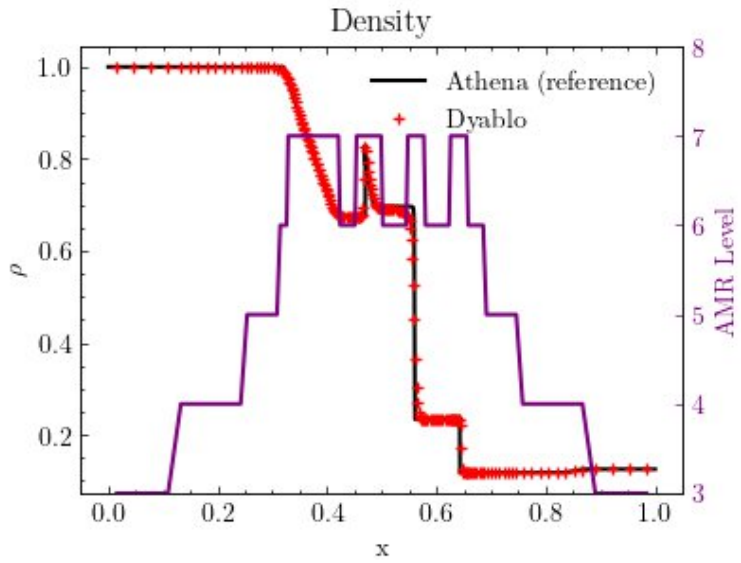
Cloud-shock interaction

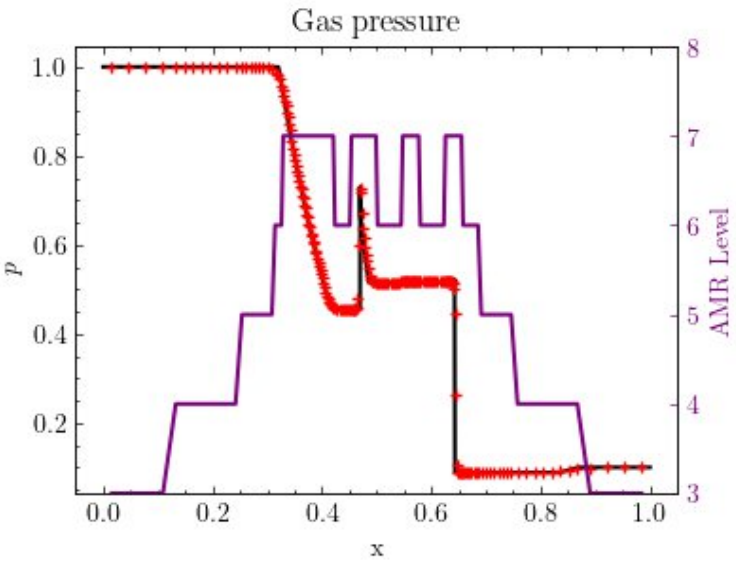
Rayleigh-Taylor 3D

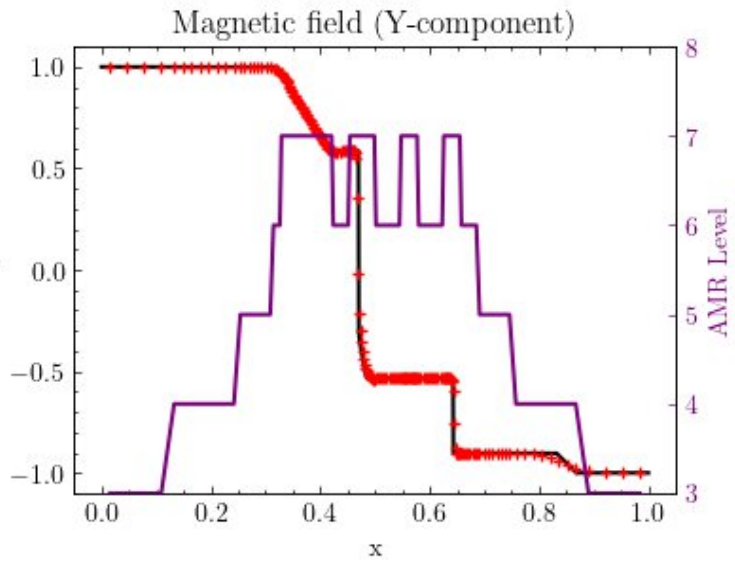




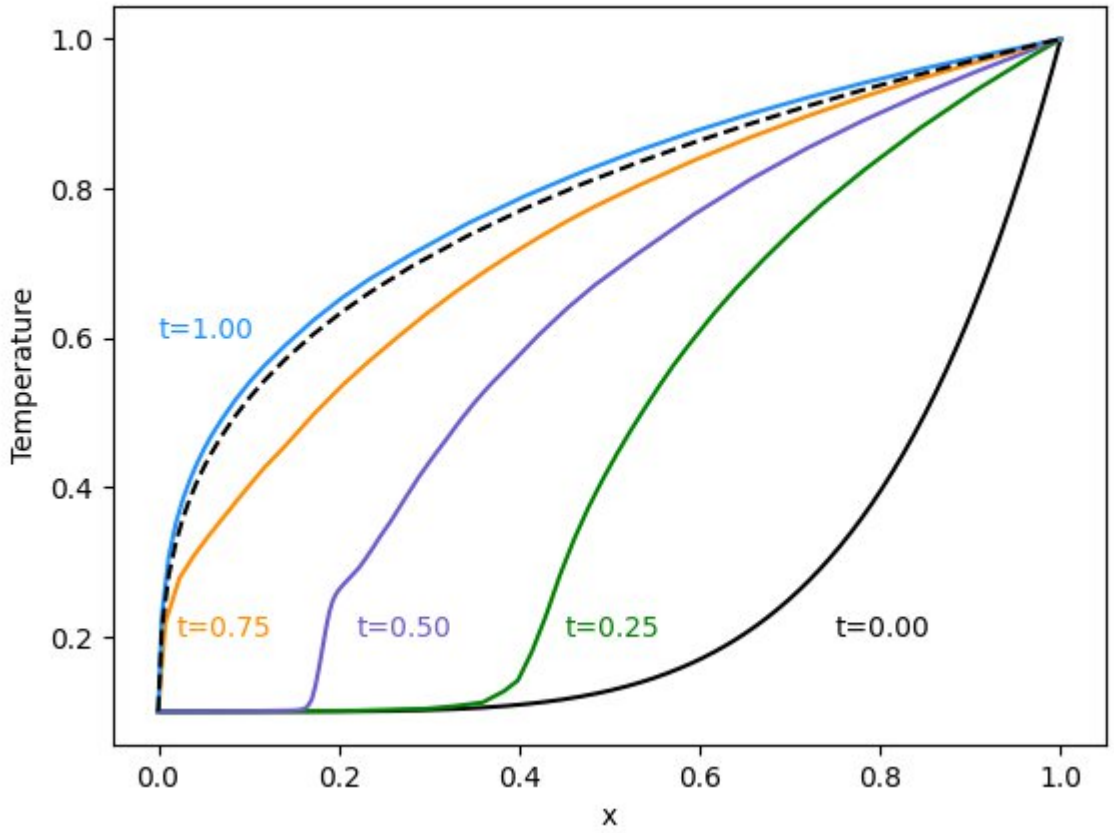
Brio-Wu


Density
Athena (reference)
Dyablo
ρ
AMR Level
x

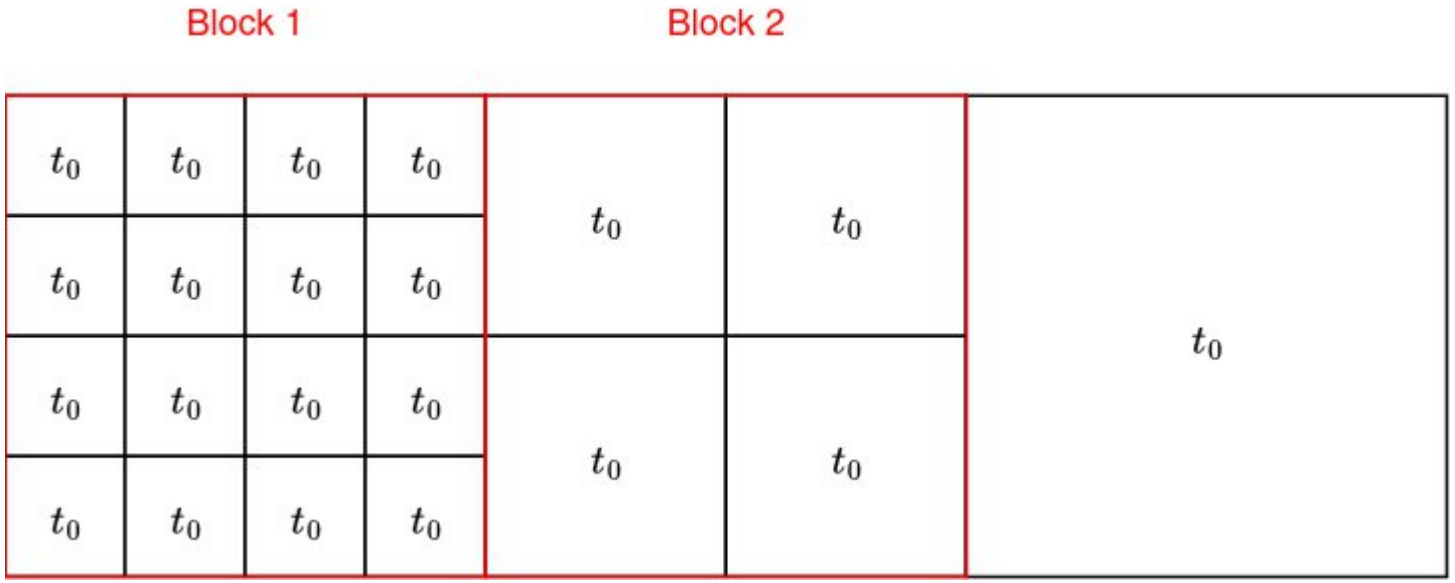

Gas pressure
p
AMR Level
x


Magnetic field (Y-component)
B_y
AMR Level
x

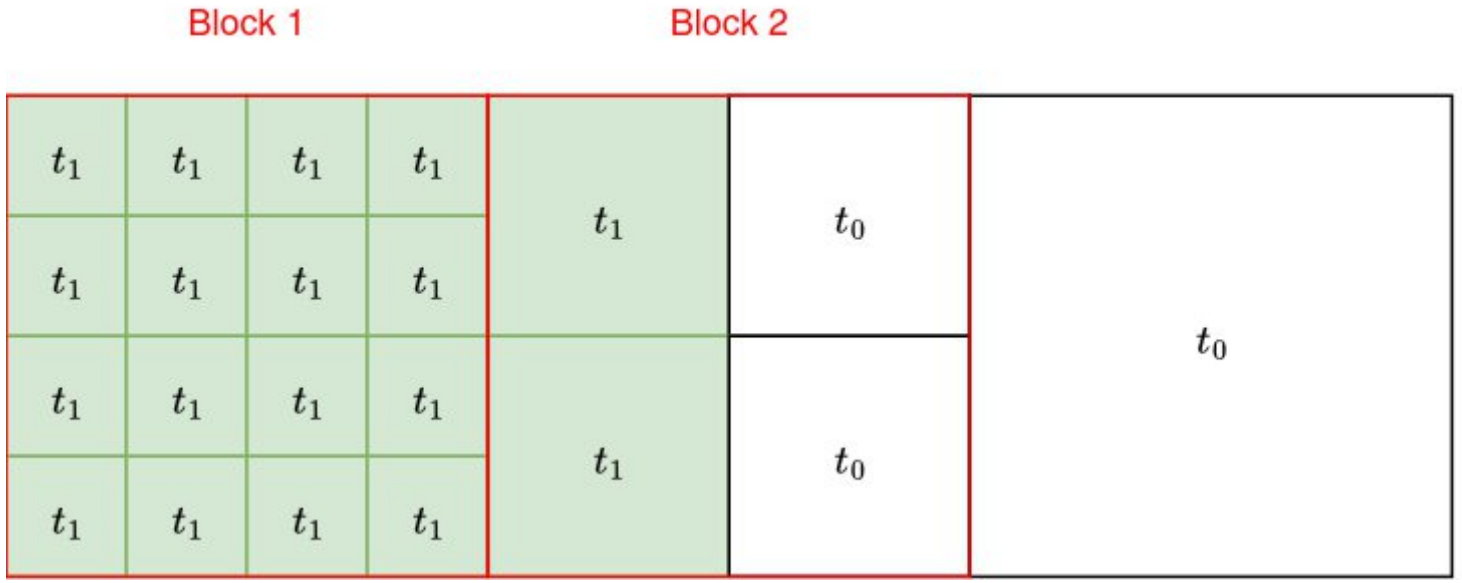
Heat conduction – Rempel et al 2016



Hierarchical Timestepping



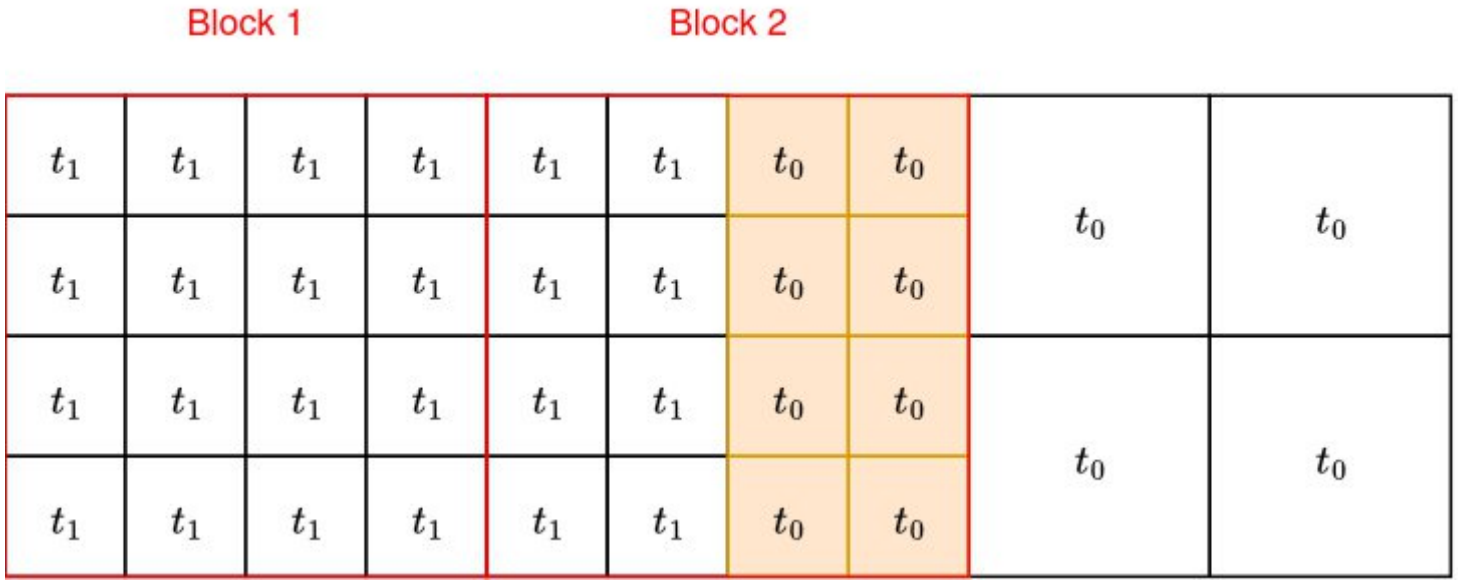
Hierarchical Timestepping



Advancing time

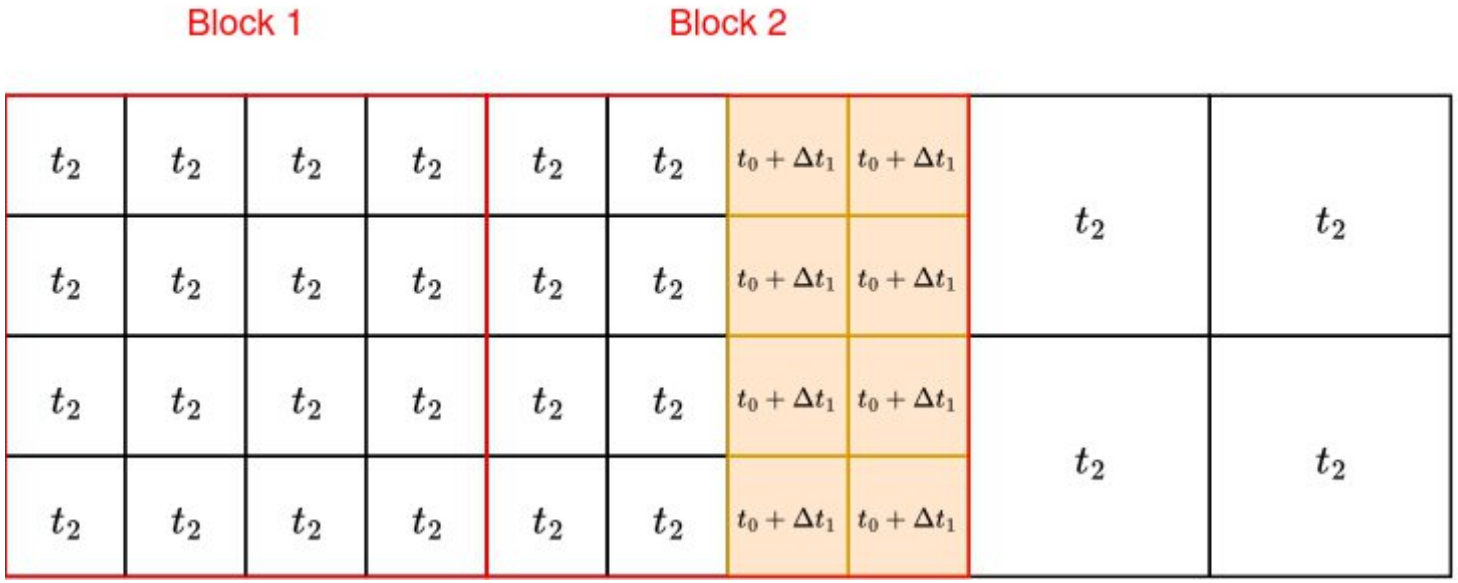
$t_1 = t_0 + \Delta t_0$

Hierarchical Timestepping



AMR Cycle

Hierarchical Timestepping



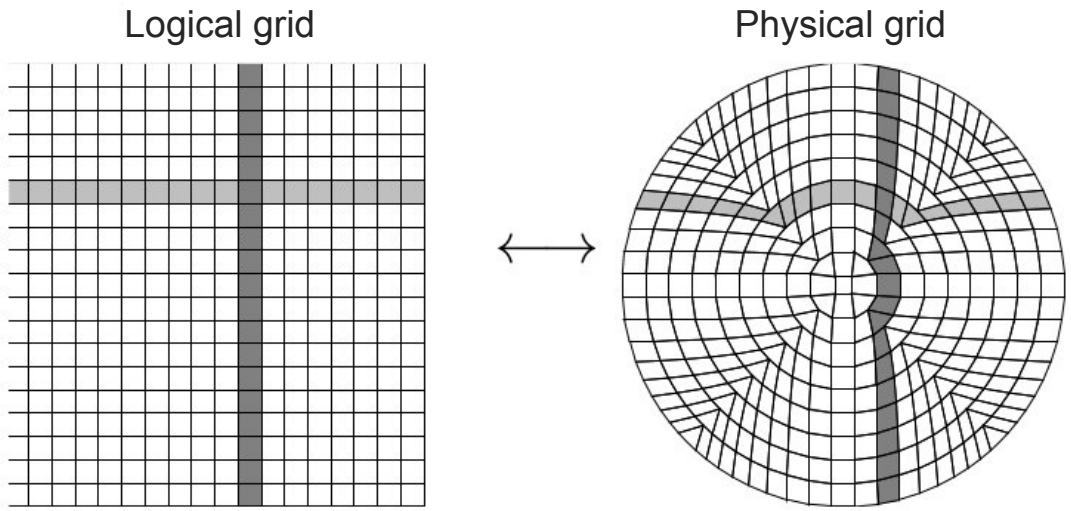
Advancing time

$$t_2 = t_1 + \Delta t_1$$

Geometry module

Isomorphism between a logical grid (cartesian) and a physical grid:

- Based on *Calhoun et al. 2008*: *"Logically Rectangular Grids and Finite Volume Methods for PDEs in Circular and Spherical Domains"*
- Easily to implement, hard to master
- One unique formulation for any type of geometry (no source terms)
- No singularity for spherical geometry
- Less variation in cell aspect-ratio for a given level



Geometry module

Some other examples

- All that is needed is a mapping from logical to physical
- The module calculates areas, volumes and projections accordingly