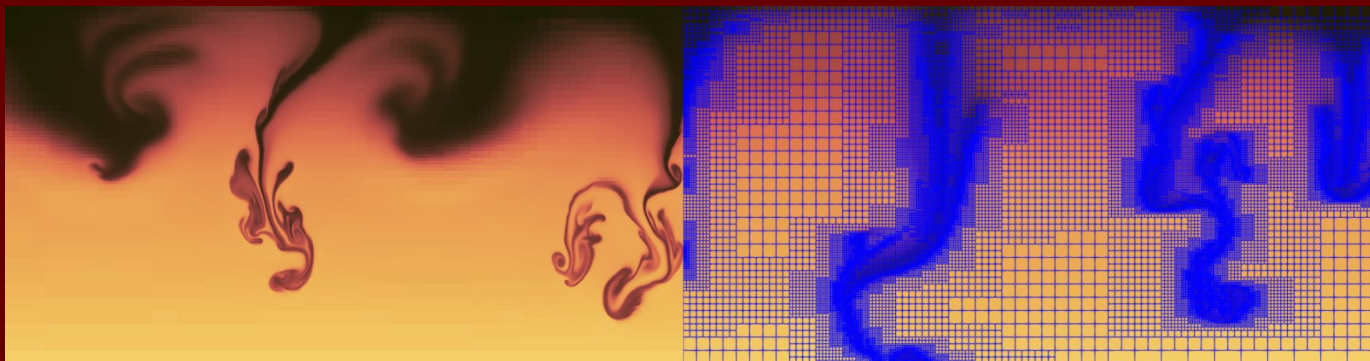# dyablo-Whole Sun
Un nouveau code de simulation sur grille AMR pour la simulation solaire sur architectures exascale
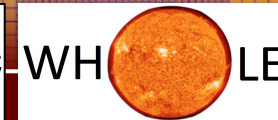
Maxime Delorme (maxime.delorme@cea.fr)
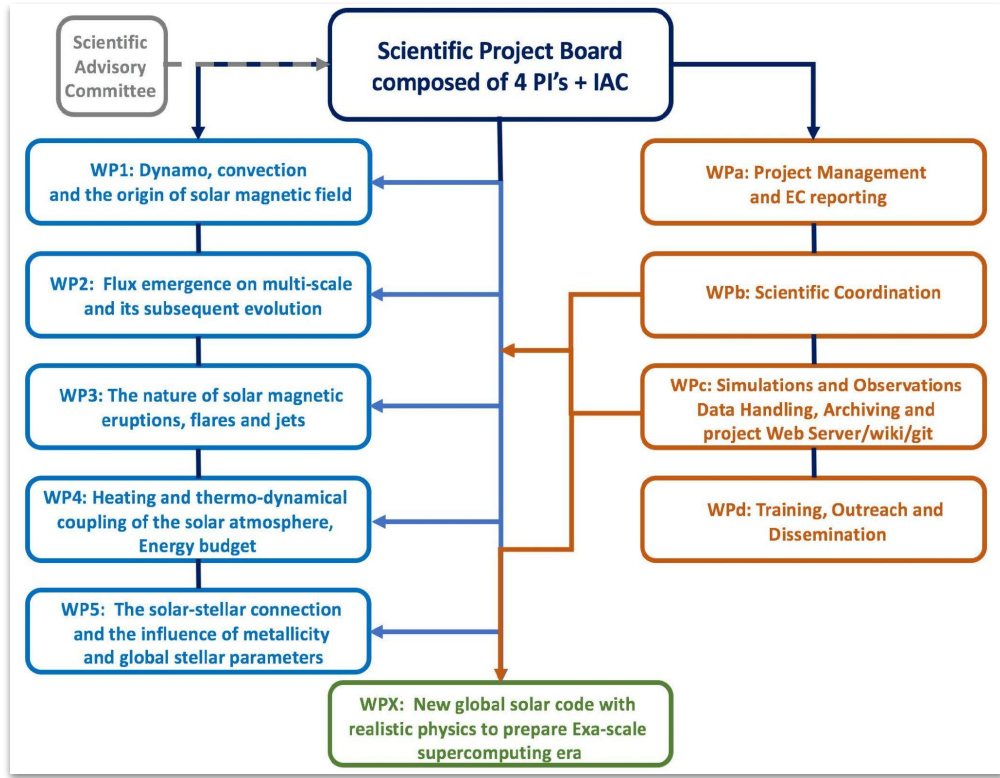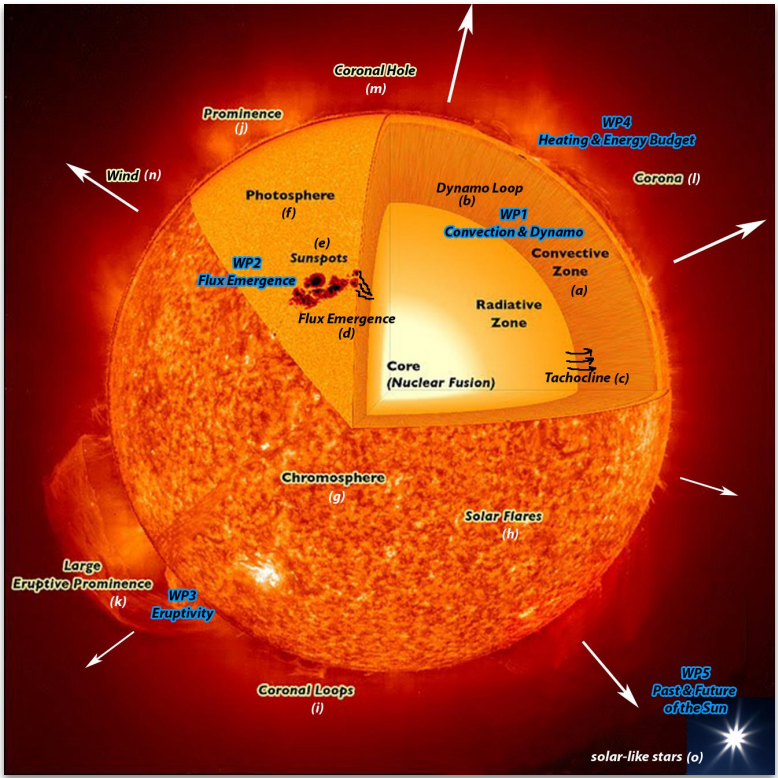**Colloque du PNST - Marseille - 19/05/2022**
Collaborateurs: Allan-Sacha Brun, Arnaud Durocher, Pierre Kestener, Antoine Strugarek

Whole Sun WPX

Source: Whole Sun website

Why a new code ?

Incentive:

- **Multi-scale/multi-physics dynamics**
  - Large variation of temporal and spatial scales
  - Different regimes corresponding to different regions
- **Global simulations of the Sun**
  - Ideally from the radiative interior out to the corona
  - At minima link the interior processes and phenomena close to the surface
- **Modularity and ease of use**
  - Testing and implementing new physics
- **Performance portability**
  - Being able to run and be efficient on "any" cluster

Limitation of present codes:

- Static meshes
- Limiting physics
- Limiting resolution
- "Expert codes"
- Not adapted for modern architectures

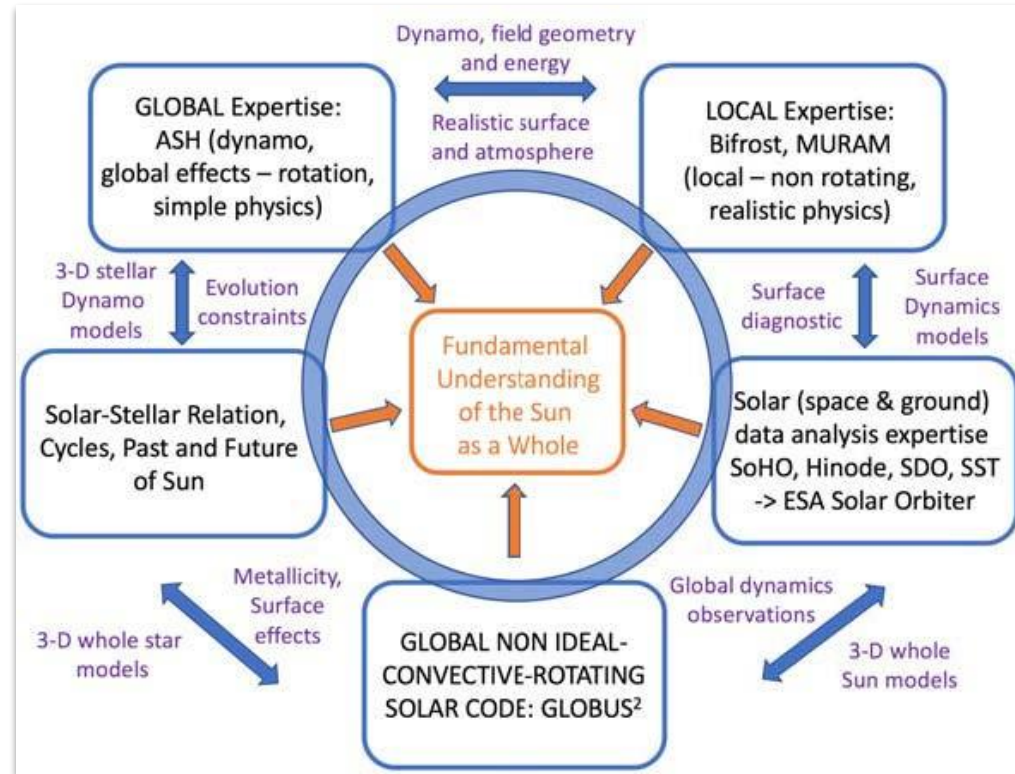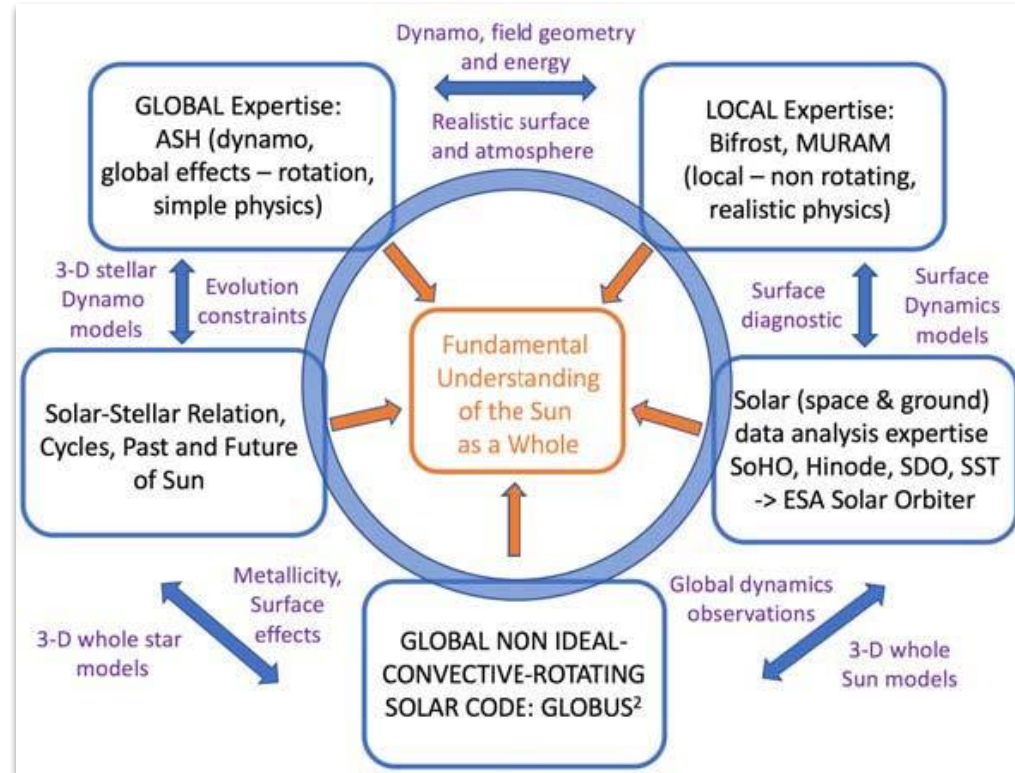*Source: Whole Sun website*

Why a new code ?

Incentive:

- **Multi-scale/multi-physics dynamics**
  - Large variation of temporal and spatial scales
  - Different regimes corresponding to different regions
- **Global simulations of the Sun**
  - Ideally from the radiative interior out to the corona
  - At minima link the interior processes and phenomena close to the surface
- **Modularity and ease of use**
  - Testing and implementing new physics
- **Performance portability**
  - Being able to run and be efficient on "any" cluster

Limitation of present codes:

- Static meshes
- Limiting physics
- Limiting resolution
- "Expert codes"
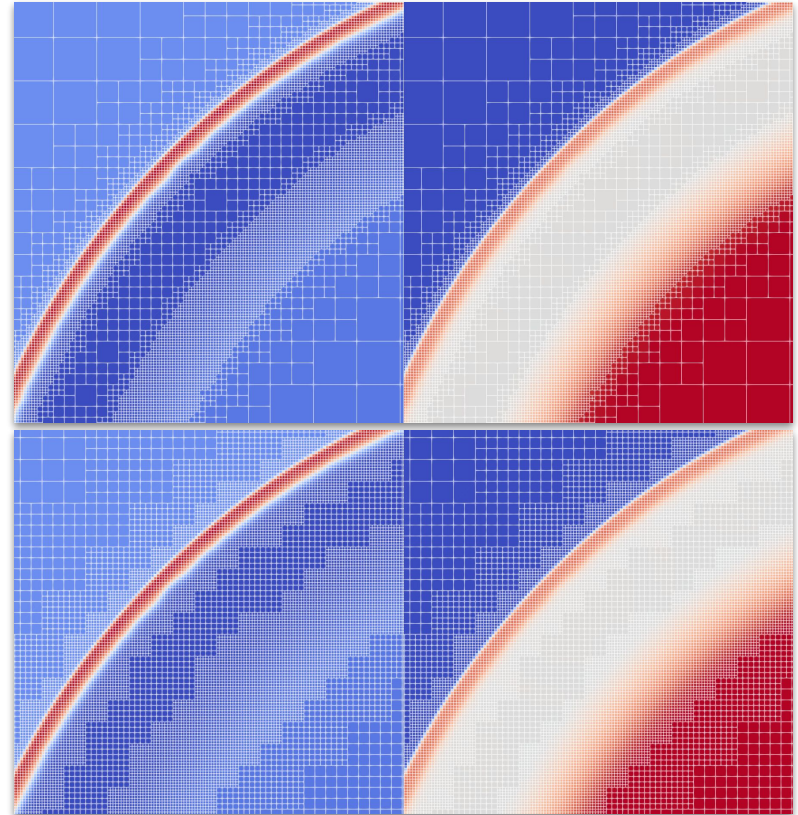- Not adapted for modern architectures

Source: Whole Sun website

A case for performance portability

Frontier : AMD CPUs + AMD GPUs

Aurora : Intel CPUs + Intel GPUs

Perlmutter : AMD CPUs + NVidia GPUs

Fugaku : Fujitsu CPUs

Summit : IBM CPUs + NVidia GPUs

Modern architectures are diverse and require adaptation and portability

AMR ?

Adaptive **M**esh **R**efinement :
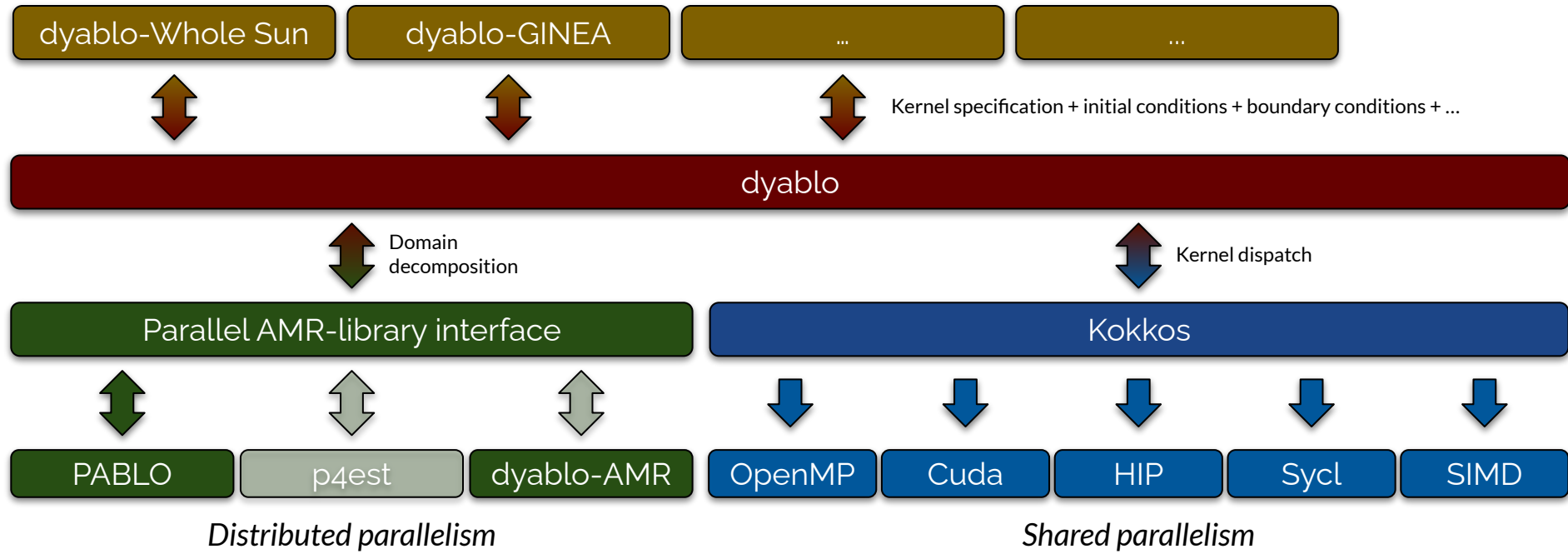
- Allocate more points in interesting[*definition needed*] regions
- Allows to fit large problems in memory
- Many flavors :
  - Cell-based
  - Block-based
  - Patch-based
- Main challenges :
  - More difficult algorithmics
  - More complex numerical schemes
  - Difficult to parallelize
  - Usually slower than regular grids
  - What's a sensible refinement criterion ?

Cell-based AMR

Block-based AMR

dyablo: a high-performance AMR framework

| dyablo-Whole Sun | dyablo-GINEA | ... | ... |

Kernel specification + initial conditions + boundary conditions + ...

dyablo

Domain decomposition

Kernel dispatch

| Parallel AMR-library interface | Kokkos |

| PABLO | p4est | dyablo-AMR | | OpenMP | Cuda | HIP | Sycl | SIMD |

Distributed parallelism   *Shared parallelism*

Historically, DYABLO := DYnamics Adaptive mesh refinement CFD applications with PABLO

dyablo-Whole Sun: design goals and wishlist [2022]

Physics

- **Objective:** Global simulation of the Sun, from the radiative interior to the corona
- **Ingredients:** MHD, viscosity, gravity, thermal conduction, radiative-transfer, rotation, all-Mach

## Numerical methods

- Geometry: Adaptive mesh refinement, multiple geometries
- Finite-volumes, with godunov-type method, multiple solvers (muscl-hancock, rk2/rk3, euler)
- Explicit integration of sources (purely explicit, STS, RKL) or IMEX methods
- Adaptive time-stepping
- [Exploratory items : time-parallel methods (parareal), PINNs]

Software engineering

- Performance portable: MPI + shared parallelism
- *"Separation of Concerns"*: Generic AMR tree traversals/reductions
- Modularity: Plugins and factories system

dyablo-Whole Sun: current state [2022]

Physics

- **Objective:** Global simulation of the Sun, from the radiative interior to the corona
- **Ingredients:** MHD, viscosity, gravity, thermal conduction, radiative-transfer, rotation, all-Mach

Numerical methods

- Geometry: **Adaptive mesh refinement**, multiple geometries
- Finite-volumes, with godunov-type method, multiple solvers (**muscl-hancock, rk2**/rk3, **euler**)
- Explicit integration of sources (**purely explicit**, STS, RKL) or IMEX methods
- Adaptive time-stepping
- Exploratory items : time-parallel methods (parareal), PINNs

Software engineering

- Performance portable: **MPI + shared parallelism** [**CPU intel/AMD; GPU Nvidia**]
- *Separation of Concerns*: **Generic AMR tree traversals/reductions**
- Modularity: **Plugins and factories system**

dyablo-Whole Sun: Hydrodynamics tests
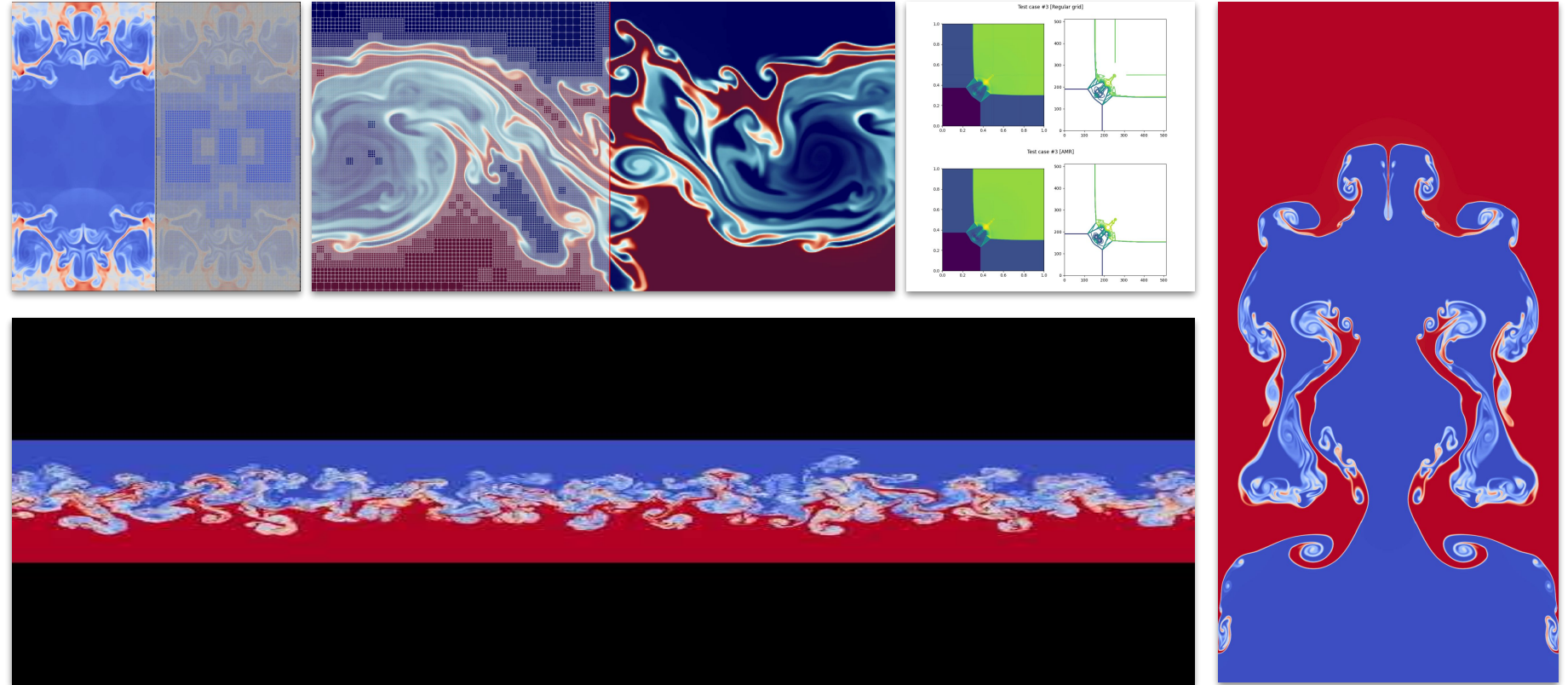
Convective hydrodynamics benchmark
Setup

- Inspired from [Hurlburt 1984](#), [**Cattaneo et al 1991**](#), [Brummell et al. 1996](#) and [2002](#)
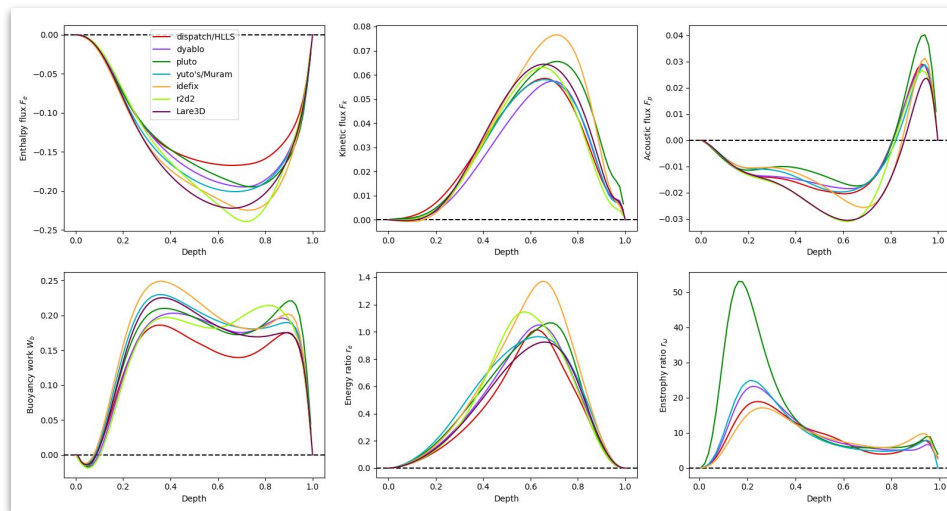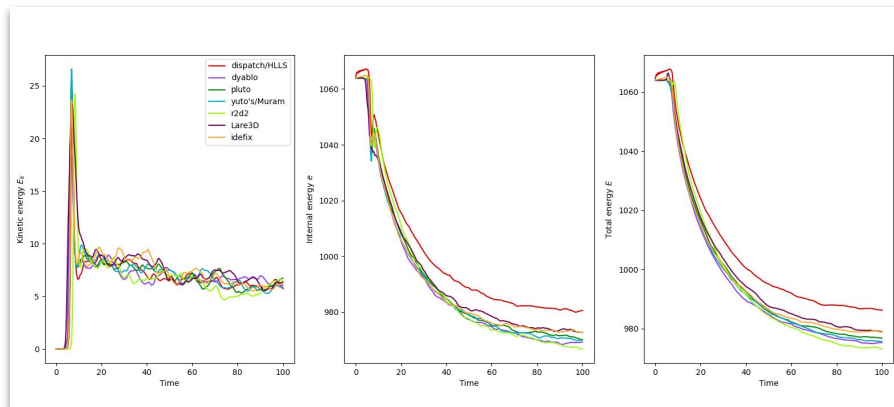
TURBULENT COMPRESSIBLE CONVECTION

FAUSTO CATTANEO, NICHOLAS H. BRUMMELL, AND JURI TOOMRE
Joint Institute for Laboratory Astrophysics and Department of Astrophysics, Planetary, and Atmospheric Sciences,
University of Colorado, Boulder, CO 80309-0440

- **Ingredients**: Compressible hydrodynamics, viscosity, gravity and thermal conduction
- **Domain:** Convective near-surface slab. Highly stratified spanning multiple density scale-heights.
  - Horizontal dimension spans 4 times the vertical dimension
  - Fixed grid resolution: $256^2 \times 64$
  - Initial conditions: Polytropic model, hydrostatic equilibrium, random perturbation on pressure
  - Horizontal BCs: periodic
  - Vertical BCs:
    - Imposed temperature at top, Imposed temperature flux at bottom
    - Stress-free impenetrable walls
    - Density recovered from continuity
- Benchmark inputs:
  - Stratification θ
  - Prandtl number σ
- 9 codes involved : dedalus, dispatch, dyablo, hps, idefix, lare3d, muram, pluto, r2d2

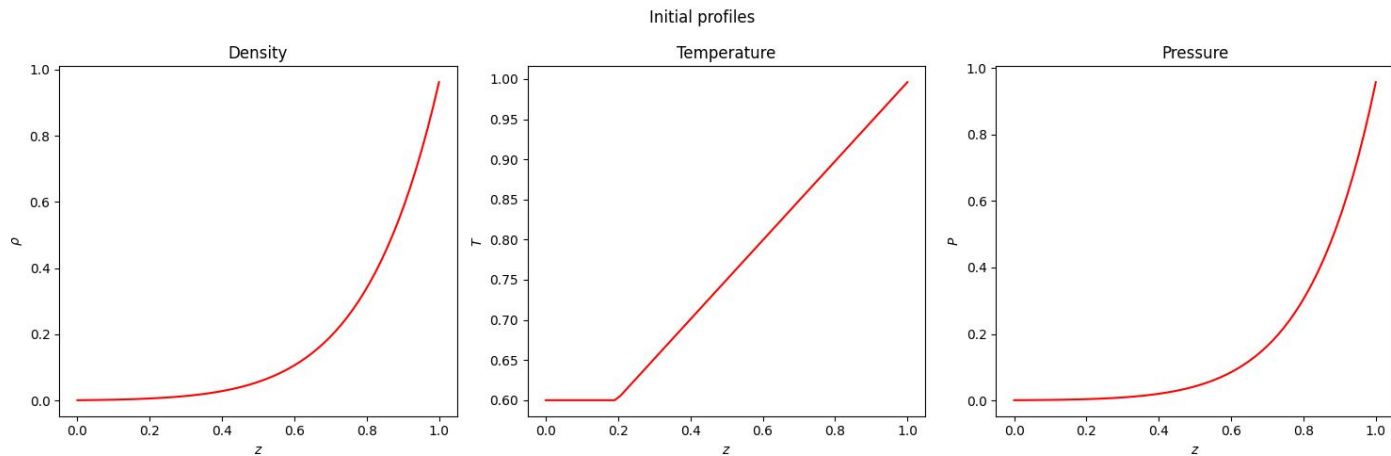# Convection benchmark

Increasing Ra

Horizontal cuts at z=0.1

| θ=10; σ=1.0	θ=10; σ=0.3	θ=20; σ=1.0	θ=20; σ=0.3
Ra = 4.9×10^4	Ra = 1.6×10^5	Ra = 3.6×10^5	Ra = 1.6×10^6
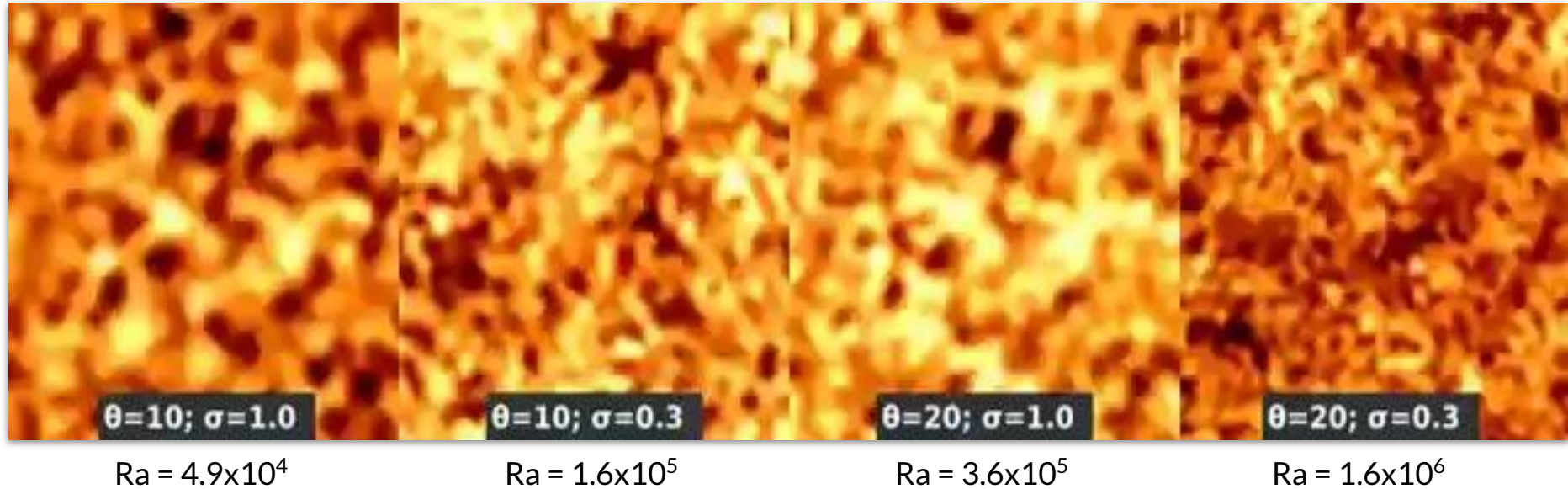
Surface cooling driven convection benchmark

Setup

- Derived by Åke Nordlund in the context of Whole-Sun. Coordinated by Mikolaj Szydlarski
- **Ingredients**: Compressible hydrodynamics + Newtonian cooling
- **ICs** :
  - Polytropic model from the base of the convection zone to the cooling layer,
  - Constant temperature above
  - Deterministic perturbation to trigger instability
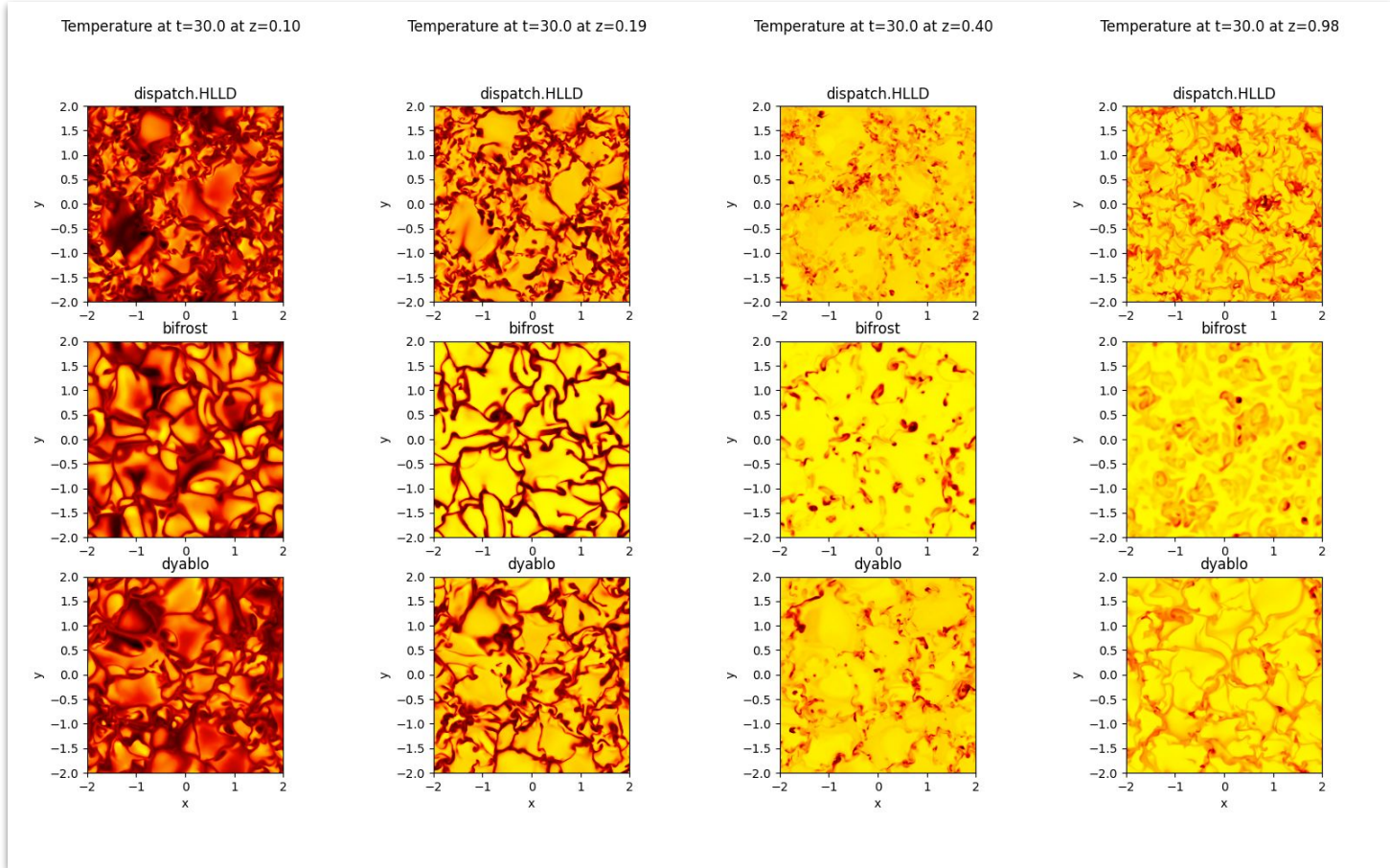- Participating codes : bifrost, dispatch, dyablo, (CO)-Mancha

Initial profiles

Surface cooling driven convection benchmark

Hydrostatic equilibrium

- No cooling, no perturbation, only hydro + gravity
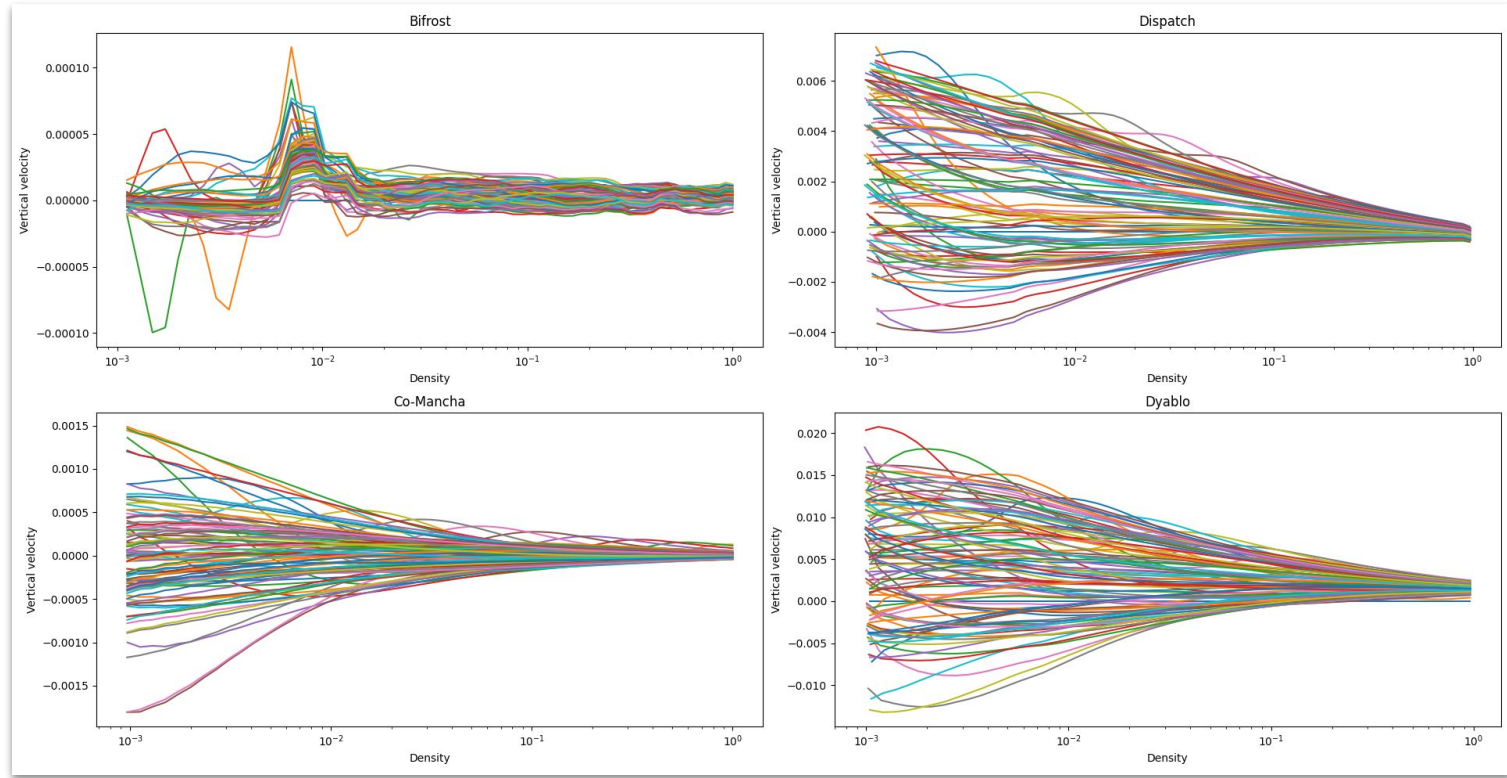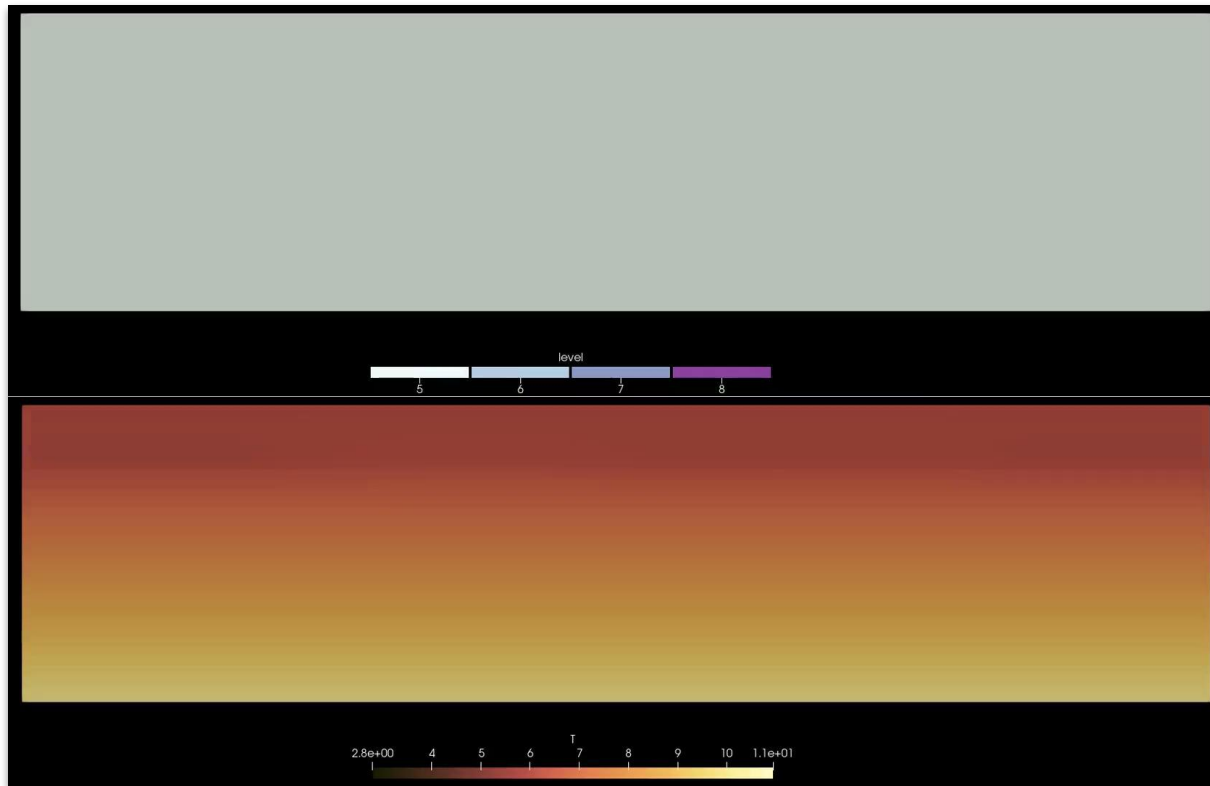
Surface cooling driven convection benchmark

Runs

Surface cooling driven convection benchmark
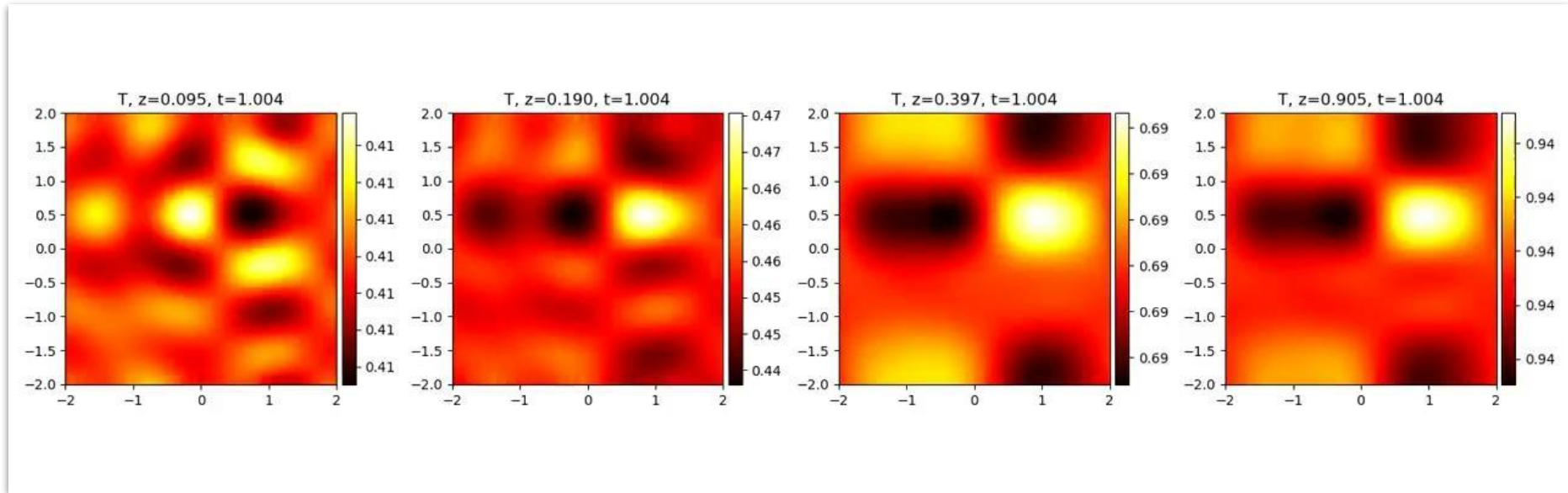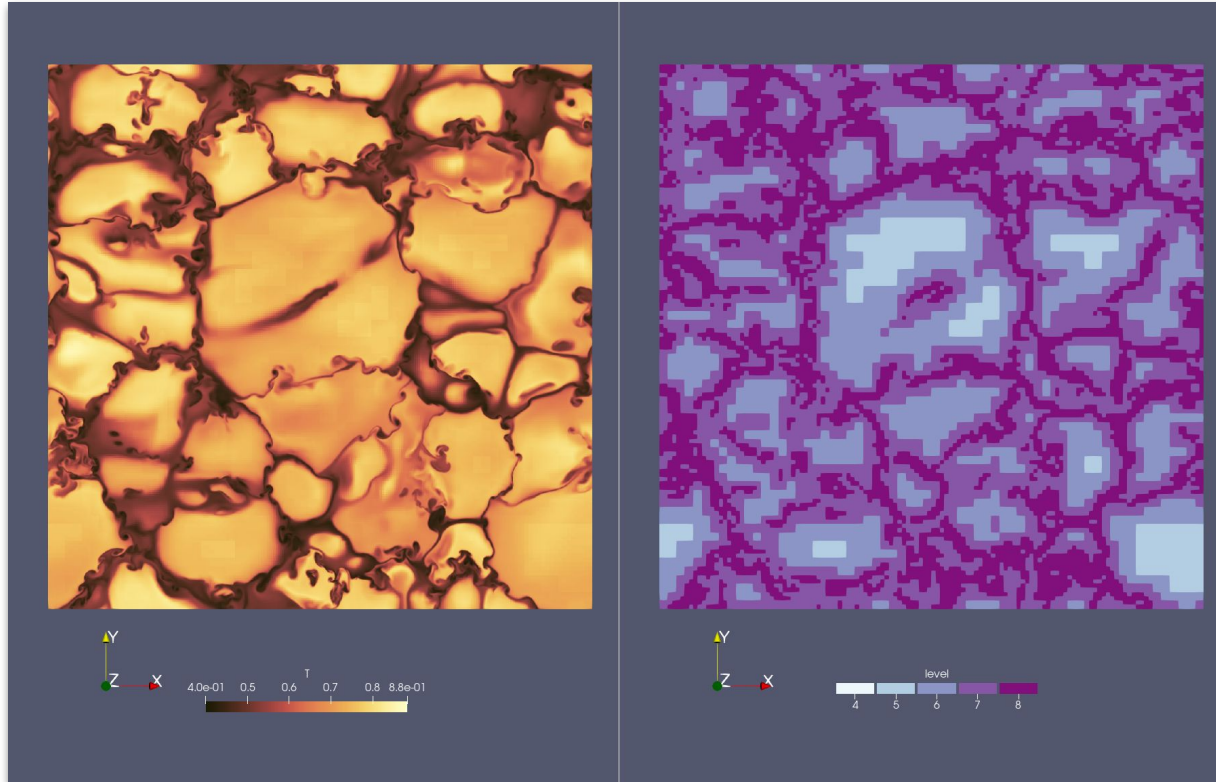
Runs

Surface cooling driven convection benchmark

AMR Runs (2d)

Base resolution:
128x32

Max resolution:
1024x256

Surface cooling driven convection benchmark
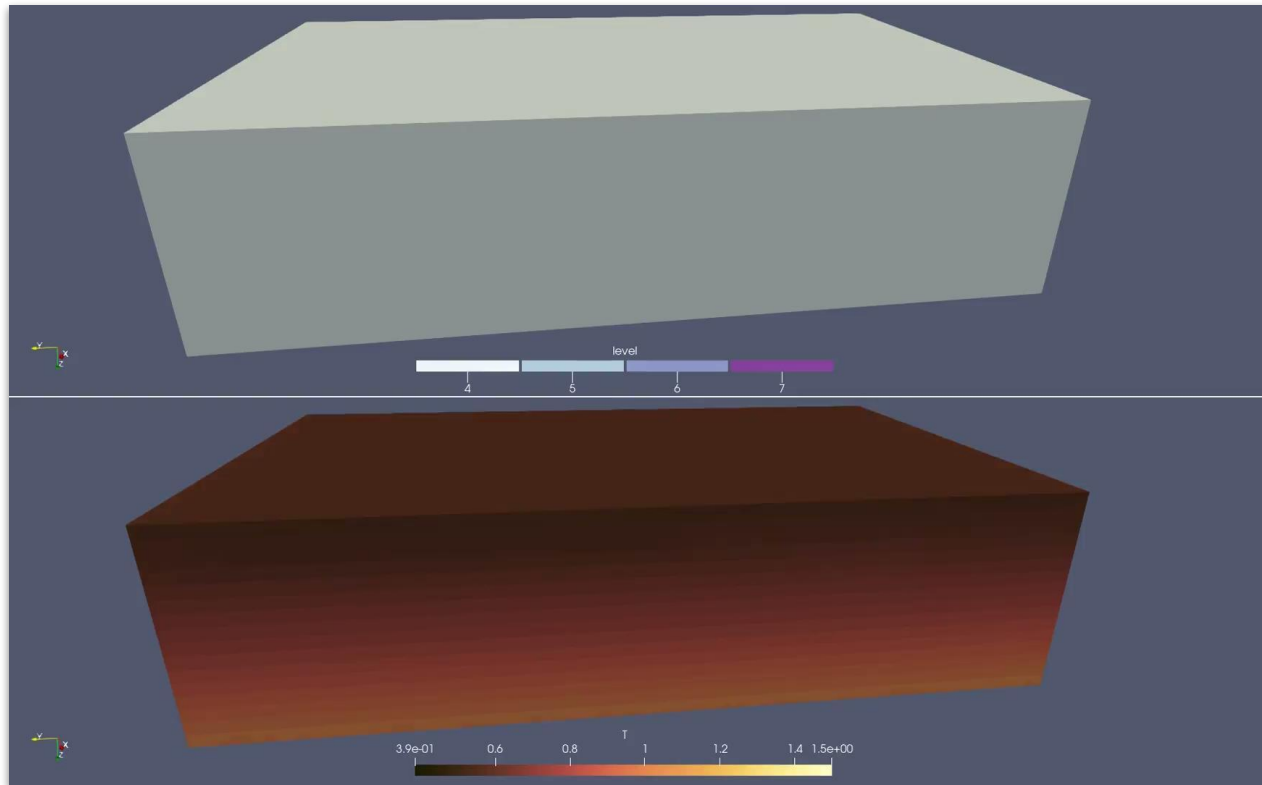
AMR Runs [base level of fixed run is 6]

Base resolution:
64x64x16

Max resolution:
1024x1024x256

Surface cooling driven convection benchmark
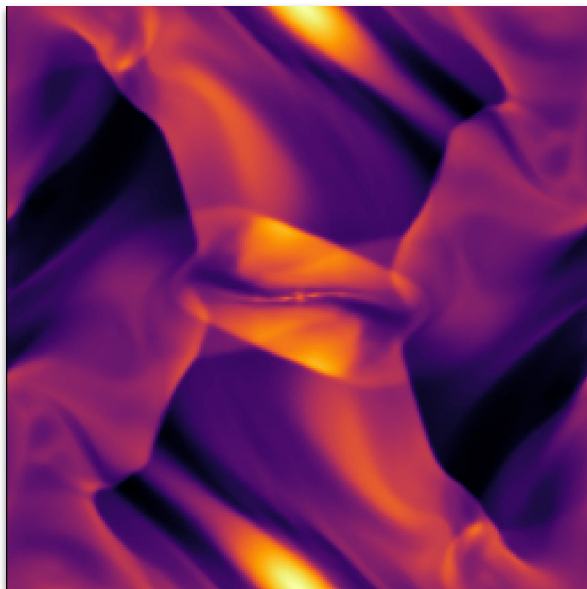
AMR Runs [base level of fixed run is 6]

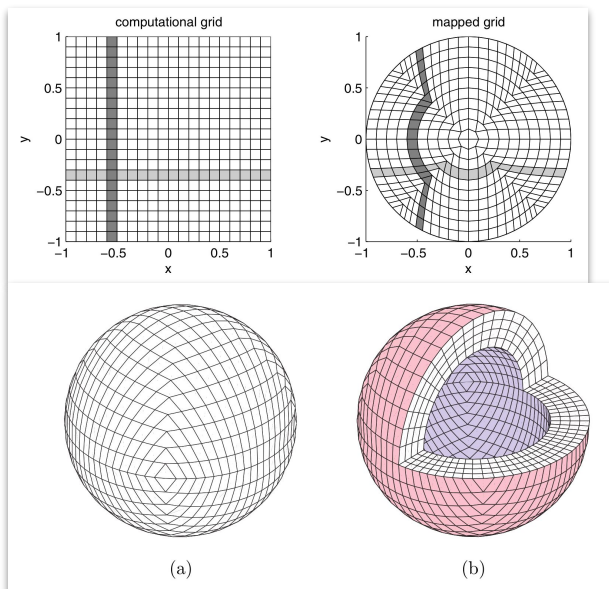**Base resolution:**
64x64x16

Max resolution:
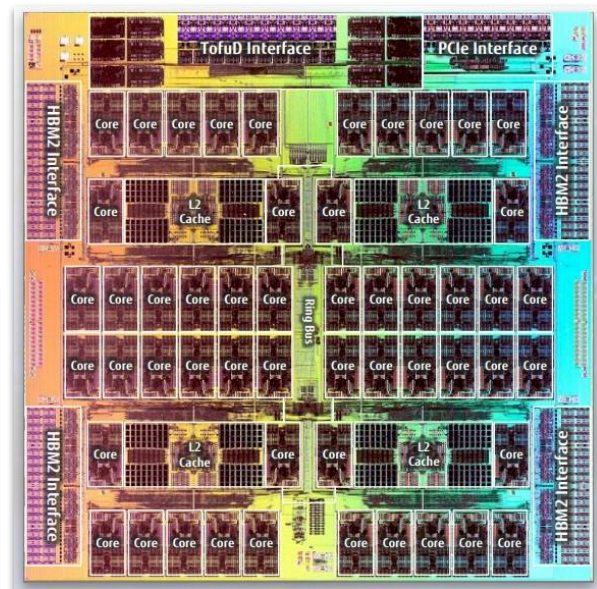512x512x128

dyablo-Whole Sun:

What's next ?

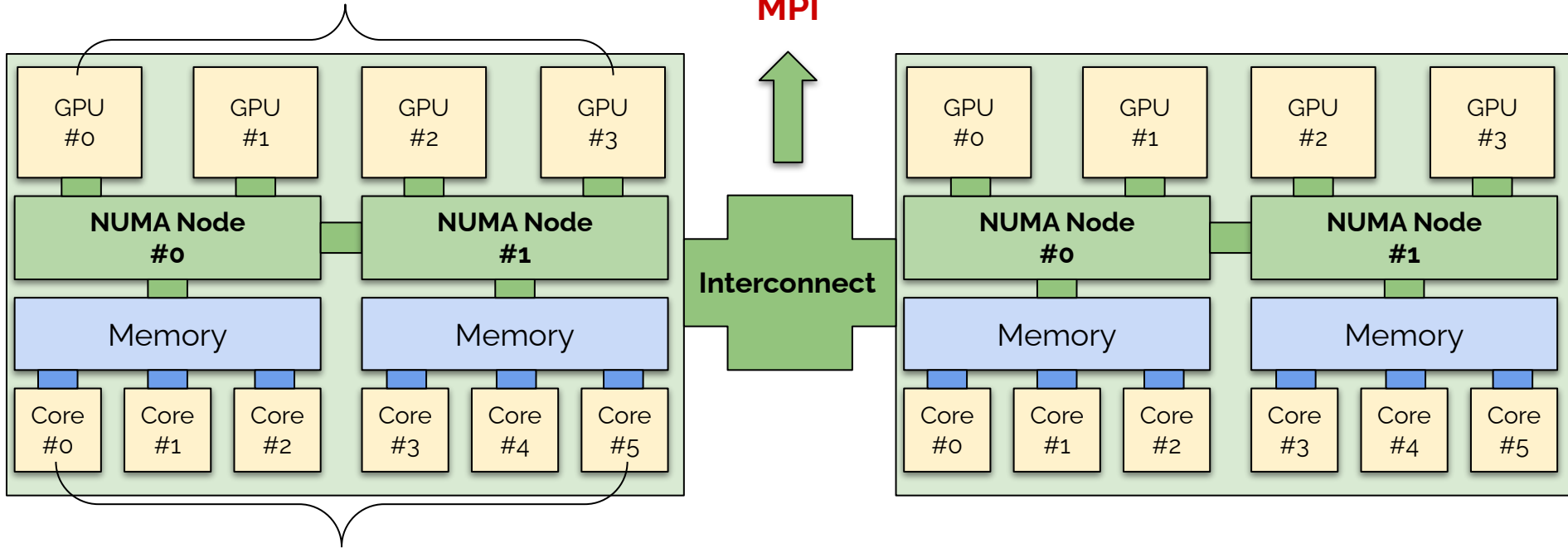Tremblin et. al (in prep)

Calhoun et. al 2008

(+ Tons of debugging/improvements/testing)

Thank you
for your attention

# Questions ?

The scope of performance portability

Cuda, Sycl, OpenACC, OpenMP, HIP, [...]

MPI

| GPU #0 | GPU #1 | GPU #2 | GPU #3 |

| NUMA Node #0 | NUMA Node #1 |

Interconnect

| Memory | Memory |

| Core #0 | Core #1 | Core #2 | Core #3 | Core #4 | Core #5 |

| GPU #0 | GPU #1 | GPU #2 | GPU #3 |

| NUMA Node #0 | NUMA Node #1 |

| Memory | Memory |

| Core #0 | Core #1 | Core #2 | Core #3 | Core #4 | Core #5 |

OpenMP, AVX/SVE, Sycl, OpenACC, [...]

Kokkos: performance portability in C++

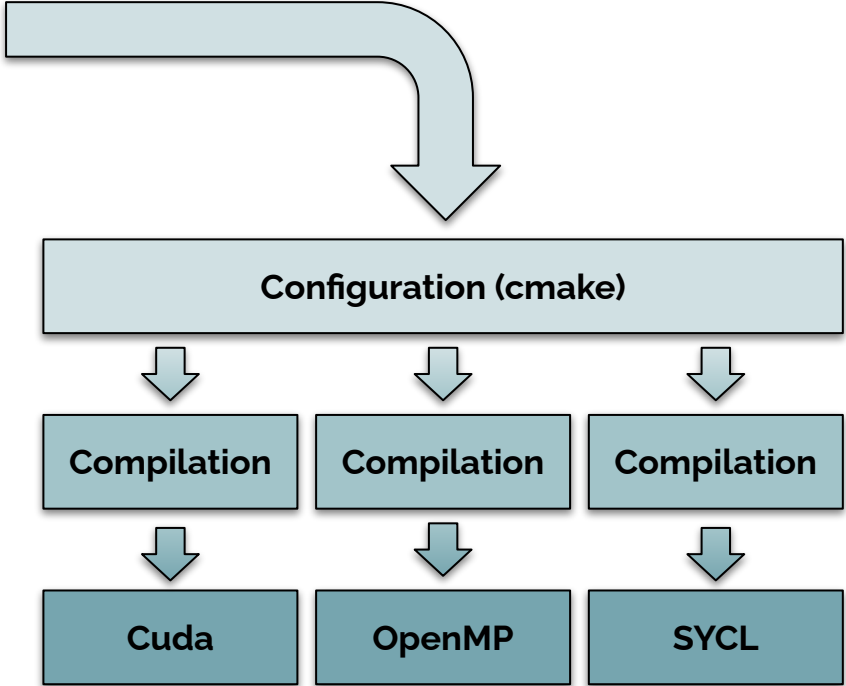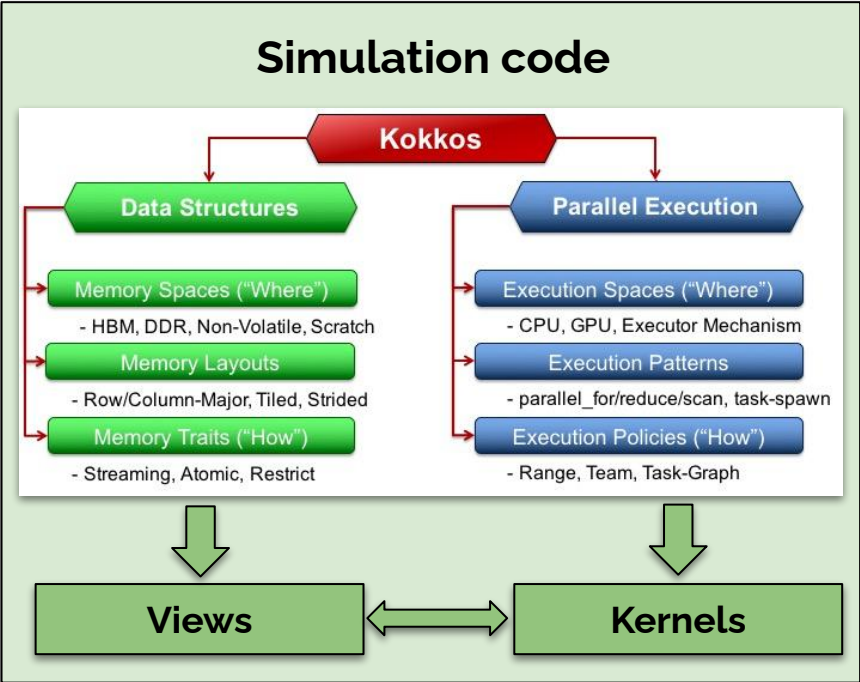## A solution to heterogeneous systems
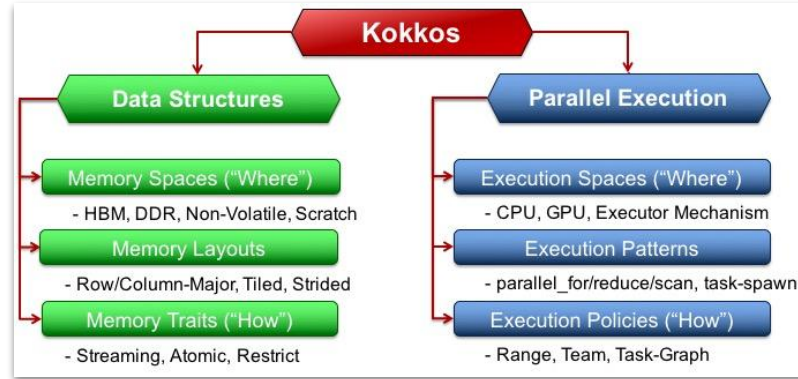
- Open-source modern C++ metaprogramming library
- Developer picks the memory structure, the type of algorithm and provides computation kernels
- Kokkos provides backends to automatically adapt the code to target architectures with minimum overhead

https://github.com/kokkos/kokkos
Carter Edwards, H., Trott, C., Sunderland, D., "Kokkos: Enabling manycore performance portability
through polymorphic access patterns", Journal of Parallel and Distributed Computing, 2014

Using the Kokkos ecosystem

Simulation code

Kokkos

Data Structures
- Memory Spaces ("Where")
  - HBM, DDR, Non-Volatile, Scratch
- Memory Layouts
  - Row/Column-Major, Tiled, Strided
- Memory Traits ("How")
  - Streaming, Atomic, Restrict

Parallel Execution
- Execution Spaces ("Where")
  - CPU, GPU, Executor Mechanism
- Execution Patterns
  - parallel_for/reduce/scan, task-spawn
- Execution Policies ("How")
  - Range, Team, Task-Graph

Views ⟷ Kernels

Configuration (cmake)

Compilation → Cuda

Compilation → OpenMP

Compilation → SYCL

SoC : Separation of Concerns

"We all have a specific job"

- **Physicists do physics**

  - Corollary #1 : Physicists don't do Software engineering, code optimization, GPU code, [...]

  - Corollary #2 : The parts of the code physicists modify should :

    - 1. Have access to simple interfaces to implement/add functionalities

    - 2. Hide all the complexities of the algorithmic machinery

    - 3. Avoid as many side effects as possible, especially on performance.

Plugin system and Factories
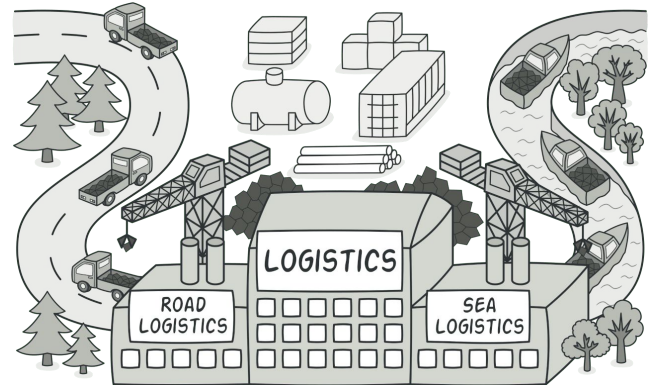
(Source : Refactoring Guru)

- Abstraction of common parts of the code
- **Factory** : Let the system create the right object at startup
- **Plugin** : Factory + Concrete Products

  - (M)HD solver, Parabolic Terms, Parabolic Solver, Refinement method, IO methods, etc.

Plugins/Factory example

Code

⬇

Compilation

⬇

Runtime Parameters

```
class ParabolicUpdate_implicit {
  ParabolicUpdate_implicit() {
    /** Constructor :
     * initialising object,
     * reading required parameters, etc.
     **/
  }

  void update(
    /** Parameters required for parabolic update **/)
  {
    // Code of the update
  }
};
FACTORY_REGISTER( dyablo::muscl_block::ParabolicUpdateFactory,
                  dyablo::muscl_block::ParabolicUpdate_explicit,
                  "ParabolicUpdate_implicit")
```

```
class ParabolicUpdate_explicit {
  ParabolicUpdate_explicit() {
    /** Constructor :
     * initialising object,
     * reading required parameters, etc.
     **/
  }

  void update(
    /** Parameters required for parabolic update **/)
  {
    // Code of the update
  }
};
FACTORY_REGISTER( dyablo::muscl_block::ParabolicUpdateFactory,
                  dyablo::muscl_block::ParabolicUpdate_explicit,
                  "ParabolicUpdate_explicit")
```

```
[parabolic]
thermal_conduction=ParabolicUpdate_explicit
viscosity=ParabolicUpdate_explicit
uniform_kappa=true
viscosity_type=dynamic
uniform_viscosity_coefficient=true
```

```
[parabolic]
thermal_conduction=ParabolicUpdate_implicit
viscosity=ParabolicUpdate_explicit
uniform_kappa=true
viscosity_type=dynamic
uniform_viscosity_coefficient=true
```

AMR-Cycle

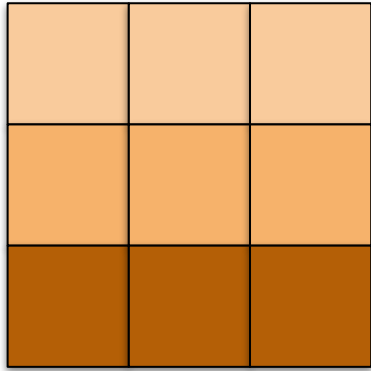| AMR cycle step	PABLO backend	Hashmap backend
Cell marking	On device + transfer	On device*
Mesh adaptation	On host	On device*
Mesh remapping	On device	On device
Load balancing	On host	On device

* CPU <-> GPU transfers due to backward compatibility
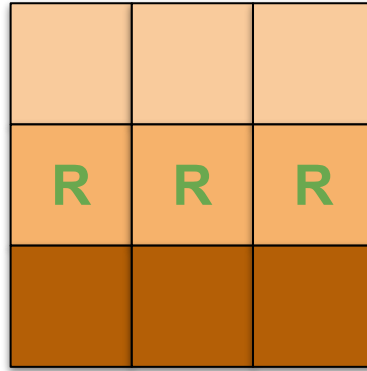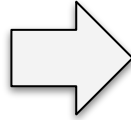
Hydrostatic equilibrium refinement disaster
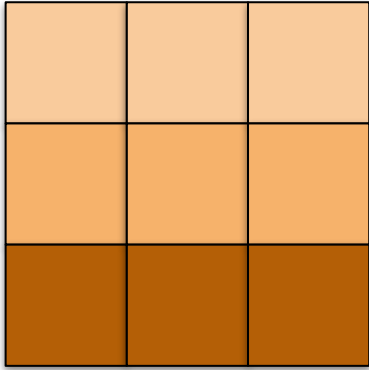
In hydrostatic equilibrium
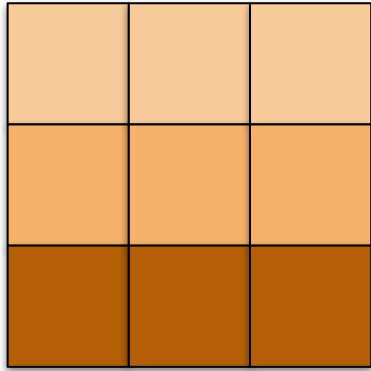
Hydrostatic equilibrium refinement disaster

In hydrostatic equilibrium

Marking for refinement

Hydrostatic equilibrium refinement disaster
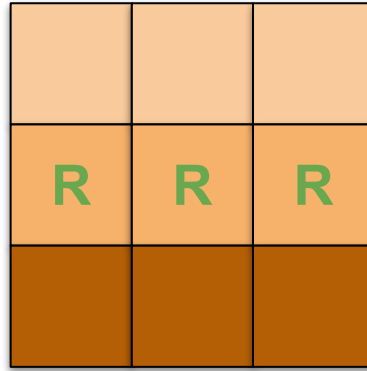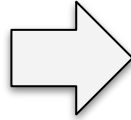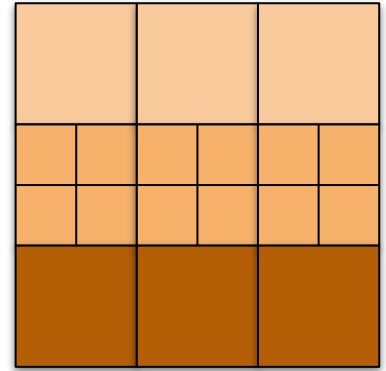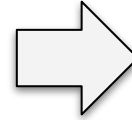

In hydrostatic equilibrium

Marking for refinement

Hydrostatic equilibrium refinement disaster