CENTRE OF EXCELLENCE FOR HPC
ASTROPHYSICAL APPLICATIONS

gPLUTO LUMI Hackathon Status (Oslo May 2025)

S. Truzzi, A. Suriano University of Turin, Turin, Italy

A. Romeo, N. Shukla CINECA, Casalecchio di Reno Bologna, Italy

Co-funded by the European Union

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Belgium, Czech Republic, France, Germany, Greece, Italy, Norway, and Spain under grant agreement No 101093441.

EuroHPC Joint Undertaking

What is gPLUTO?

- GPU-enabled version of PLUTO: a multi-algorithm framework for solving the equations for gas and compressible plasma dynamics with high Mach numbers flows (i.e. compressible Navier-Stokes equation, ideal MHD, relativistic MHD (RMHD) and resistive relativstic MHD (ResRMHD)).
- It is a Godunov-type finite volume grid code solving hyperbolic and parabolic magneto-hydrodynamic conservation laws up to three dimensions on a static grid or mapped grids.
- Freely distributed PLUTO at *http://plutocode.ph.unito.it* (v. 4.4)
- Public gPLUTO at *https://gitlab.com/PLUTO-code/gPLUTO*
- gPLUTO is part of Scalable Parallel Astrophysical Codes for Exascale (https://www.space-coe.eu) European project

written in:

- ***C and C++ (core code)***
- *OpenACC for GPU shared memory*
- MPI for multiGPU / CPU support

Available Physics Modules (~60 % ported on GPU version)

Advection Physics (Hyperbolic PDE)
- Hydrodynamics (HD)
- Magnetohydrodynamics (MHD)
- Relativistic Hydrodynamics (RHD)
- Ideal and resistive relativistic MHD (RMHD - ResRMHD)

Source Terms
- Gravity / Body forces
- Cooling
- Heating / optically thin
- Chemical networks

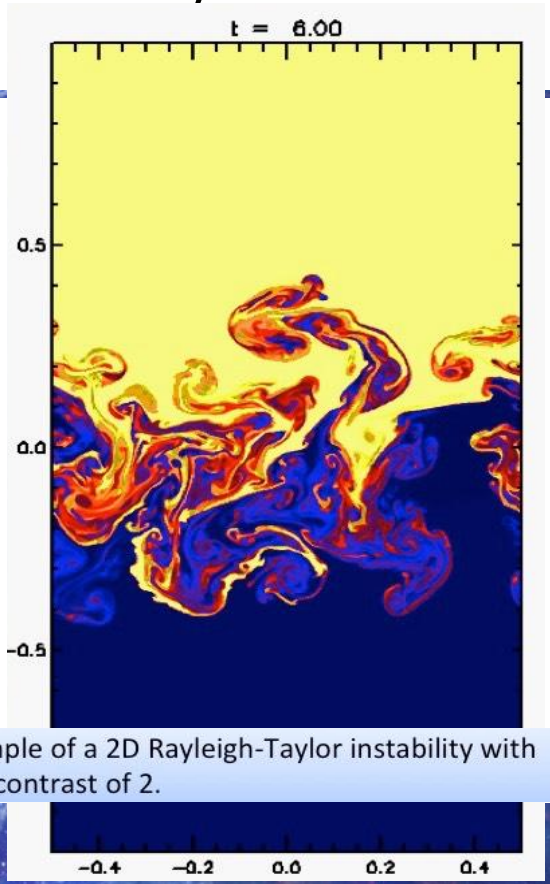
Geometry
- Cartesian
- Cylindrical
- Spherical

Dissipation Physics (Parabolic PDE)
- Viscosity (Navier-Stokes)
- Thermal conduction (hydro and MHD)
- Hall MHD, Ambipolar diffusion, Magnetic resistivity
- Radiation Hydrodynamics (FLD, 2 temp)

Particle Physics
- Lagrangian particles
- Cosmic Ray
- Dust

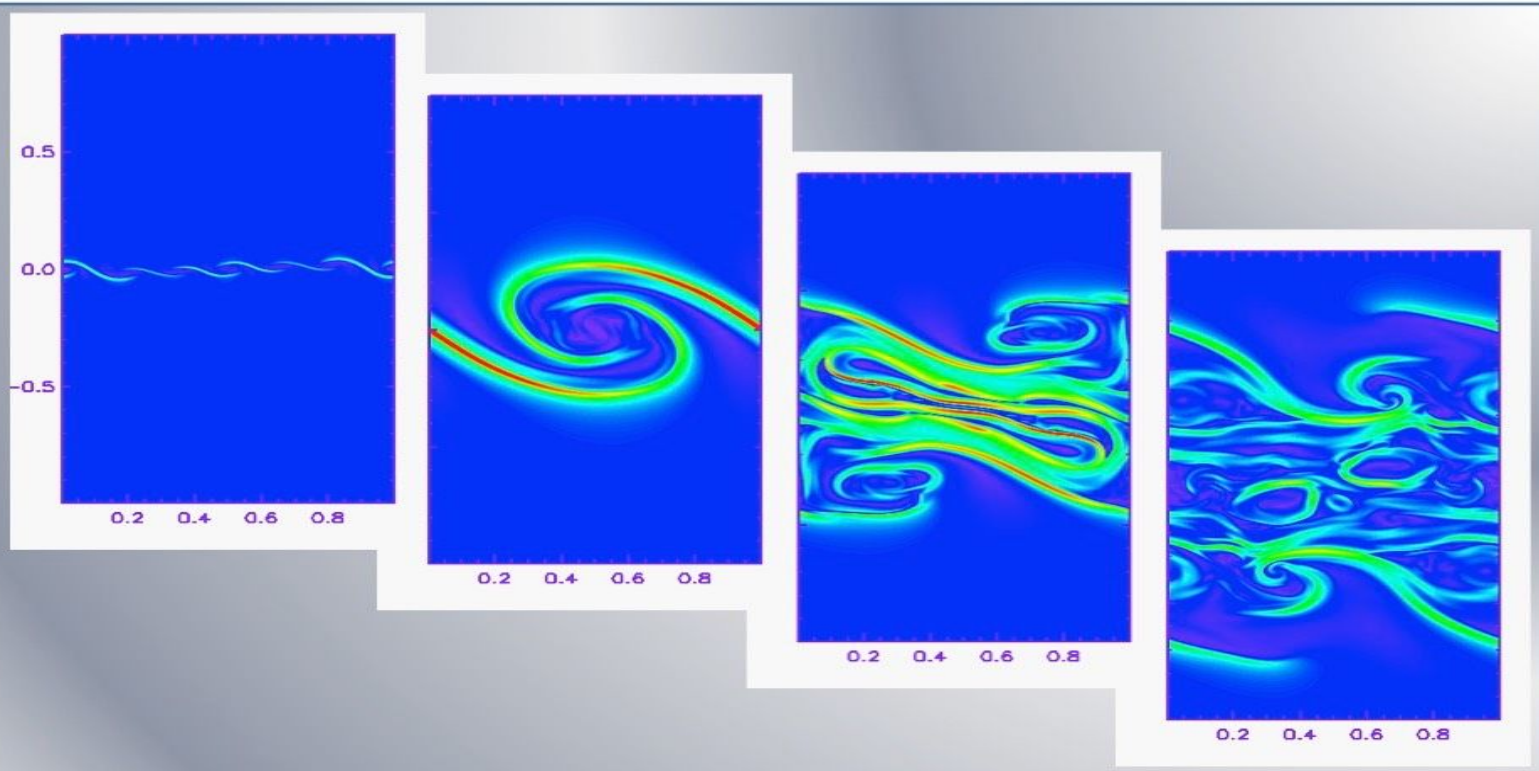
Thermodynamics
- Ideal
- Isothermal
- Non-Constant gamma
- Synge Gas (relativistic)

LEGEND
- Ported
- in progress
- Not ported



An example of a 2D Rayleigh-Taylor instability with density contrast of 2.

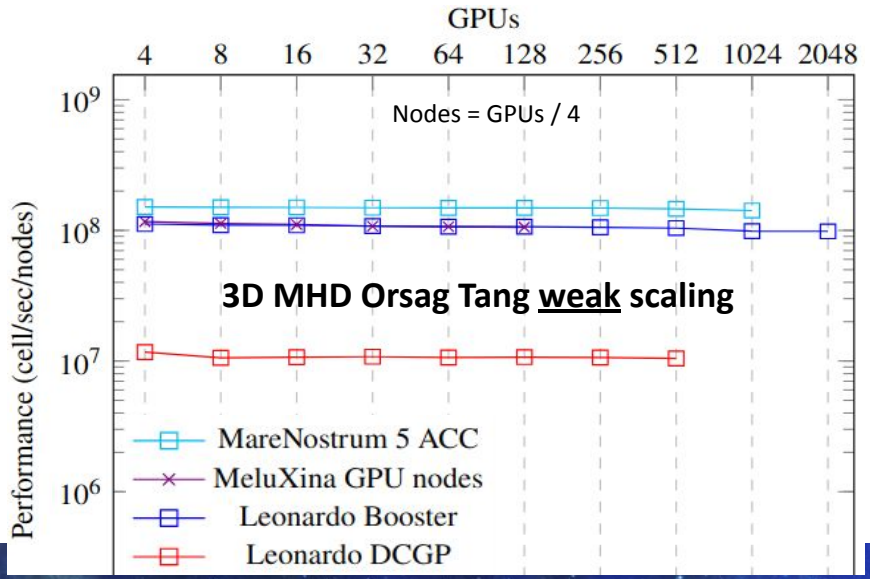
gPLUTO application Example



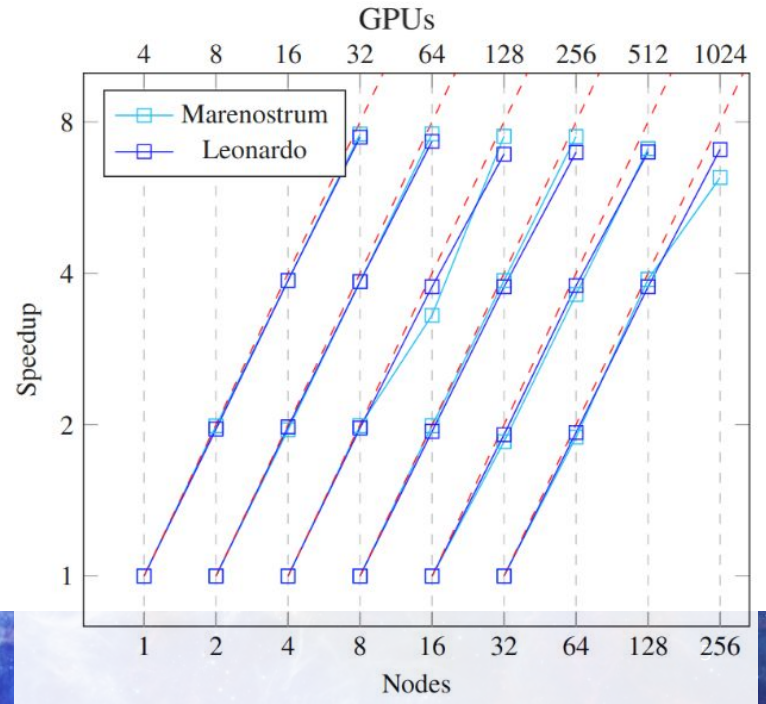
Magnetic pressure evolution in a magnetized Kelvin-Helmholtz instability in a hot relativistic flow, see Mignone et al, MNRAS (2009) 393, 1141.

gPLUTO Benchmark results

- Tested on EUROHPCs NVIDIA GPU HPCs (CINECA - Leonardo, BSC - Marenostrum)
- **speedup** factors (tCPU/tGPU) from **10 to 50** (depends on test)

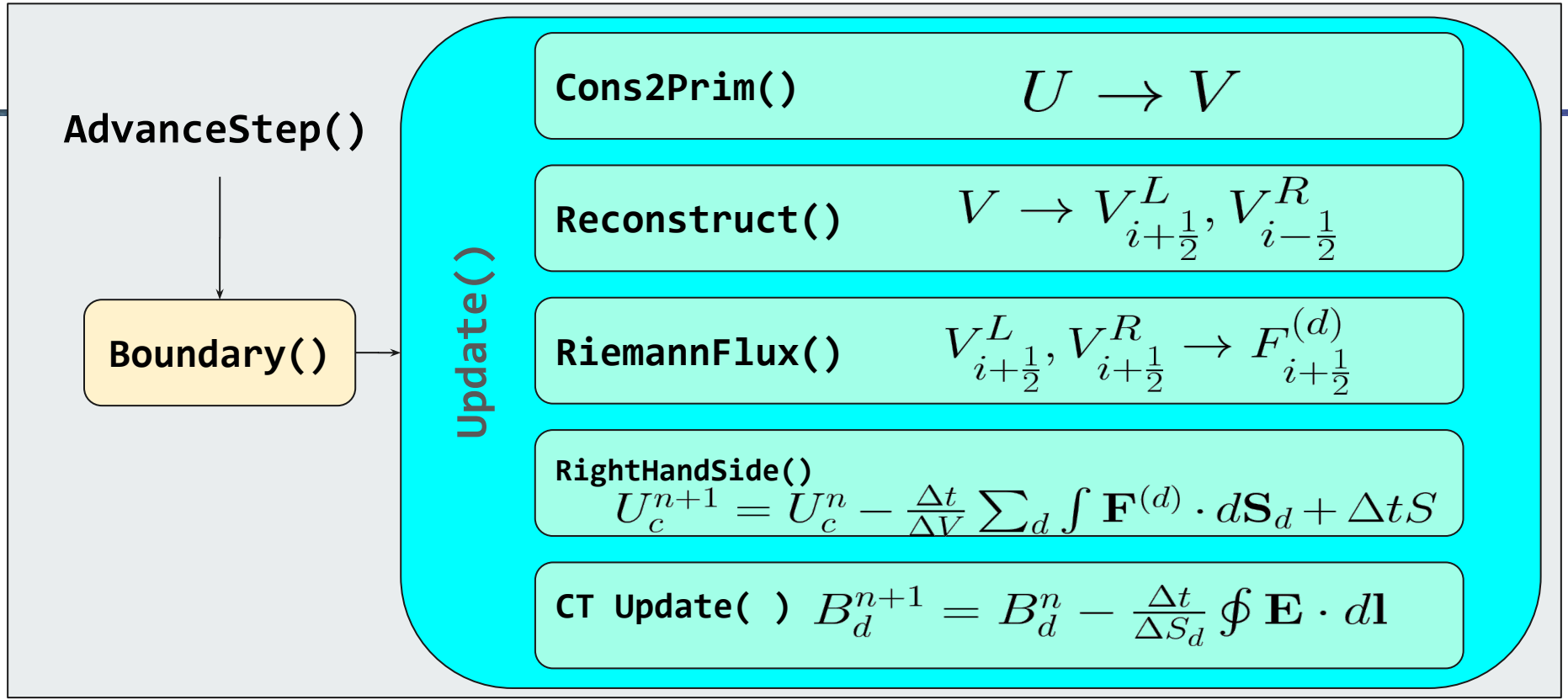


3D MHD Orsag Tang strong scaling



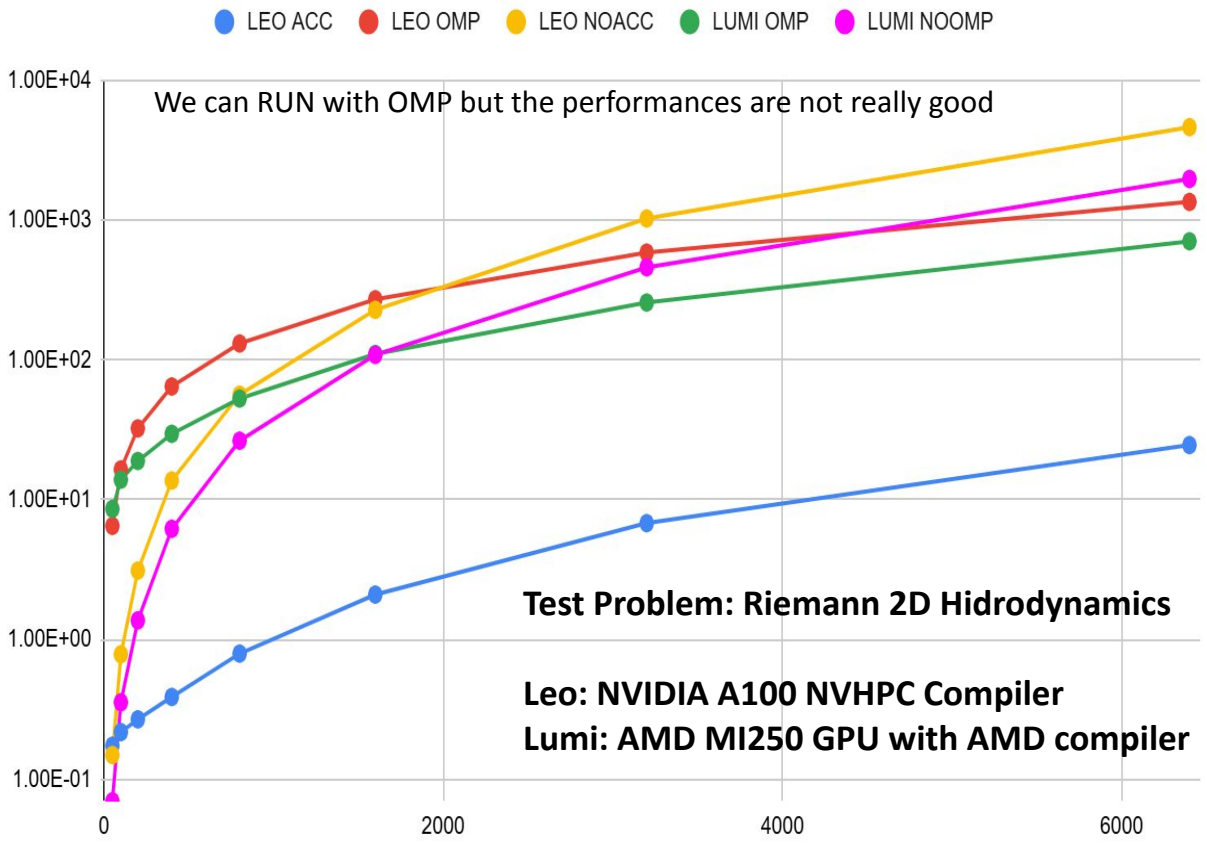
STATUS on LUMI

gPLUTO - (GPU) MAIN KERNELS (from previous experience)



- We select a simple HD test problem: **Riemann2D**
- Code Compile and Run on Leonardo with **NVHPC** Compiler
- **OpenMP** is very slow (comparable to cpu)

- Code Compile and Run on LUMI-G with **CLANG-OMP (**PrgEnv-amd) Compiler and UNIFIED MEMORY
- **similar performance to Leonardo**

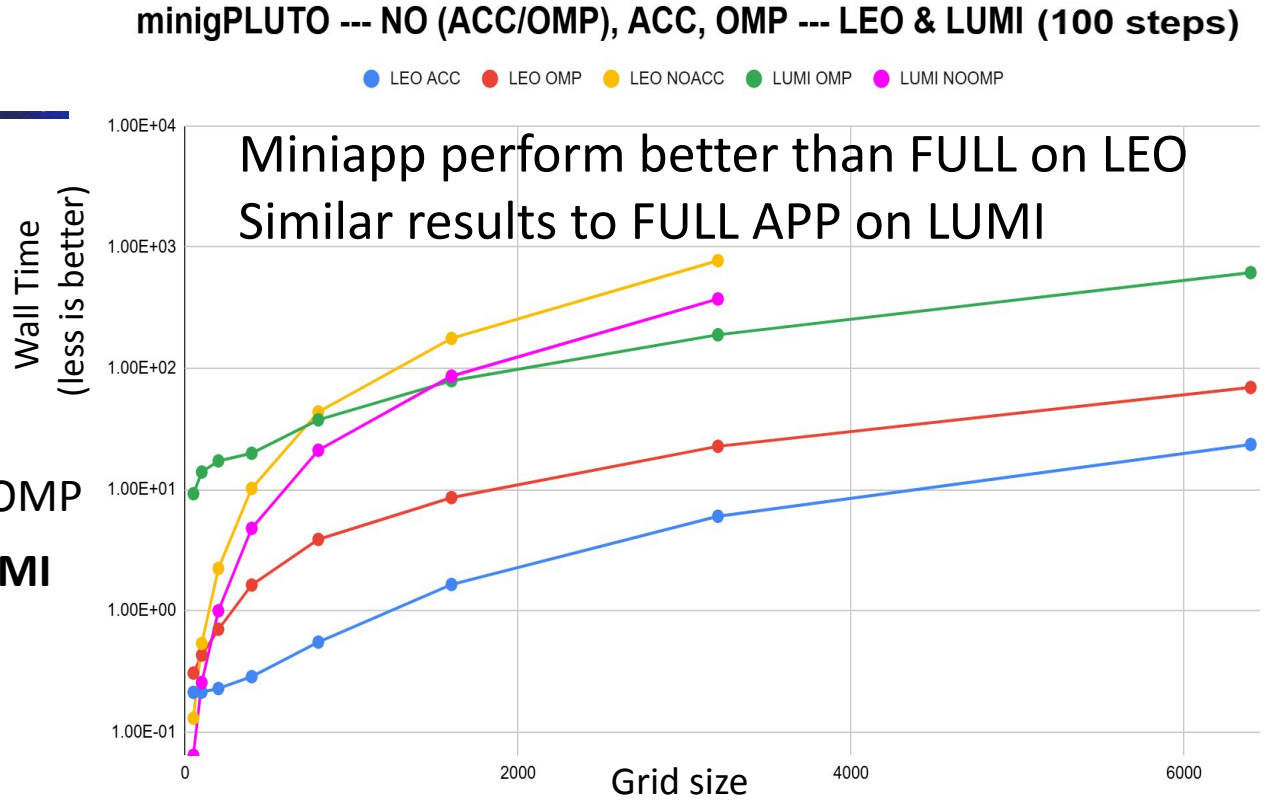


MINI APP

Test Problem: Rieman 2D Hidrodynamics

From previous hackathon experiences:

- **mini-gPLUTO** (HD module only)
- **simplified** data structures
- **tested** mini-gPLUTO on **Leonardo** ncv++ ACC and OMP
- **tested** mini-gPLUTO on **LUMI**



Hackathon Goals

Starting from gPLUTO (or from minigPLUTO)?

Objective: have one single test problem optimized and profiled in order to acquire the know how to parallelize the rest of the code.

Possible roadmap?
In the pre-hackaton day the mentor suggested:

- Work on **miniAPP**
- Use **rocprof** and perfetto to analyze the kernels
- Analyze and optimize kernels one by one with **Omniperf**

Acknowledgement & Disclaimer

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Belgium, Czech Republic, France, Germany, Greece, Italy, Norway, and Spain under grant agreement No 101093441.

Thank You

Backup

Could complex structures represent an ISSUE?

minigPLUTO data structure example:

```
struct Data{
 Ary4D Vc;     /**< The main four-index data */
 Ary4D Uc;
 Ary3D C_dt;
 struct timeStep_   *Dts;
 struct RunConfig_ *run_config;
 void (*fluidRiemannSolver) (struct Data *,
                             timeStep *, Grid *, Box*);
 struct Sweep_ sweep;
};
```

Our experience:

OpenMP requires to be more `*explicit*` than OpenACC

OpenMP copy host to device is `*less cleaver*` than OpenACC

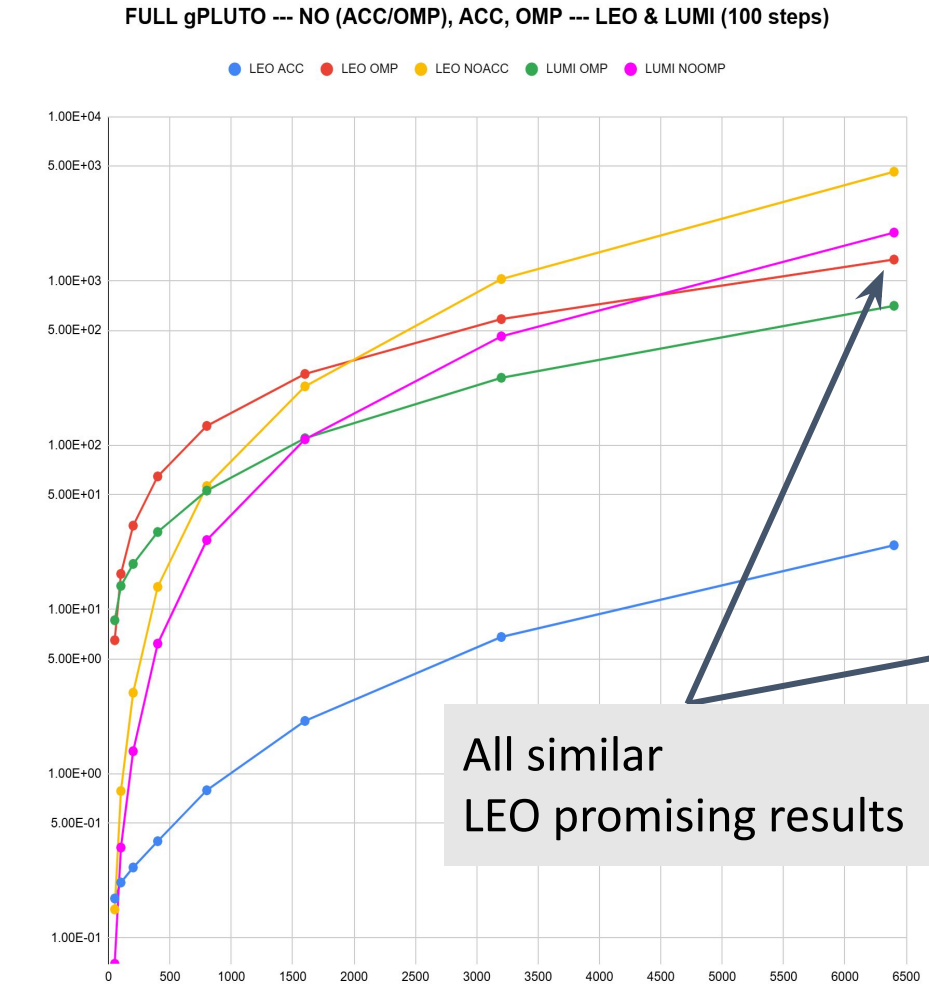
- How does OpenMP copy structures from Host to Device?
- Is there a good practice to do this?
- We found OpenMP **`declare mapper`** but the example does not explain complex object/structures

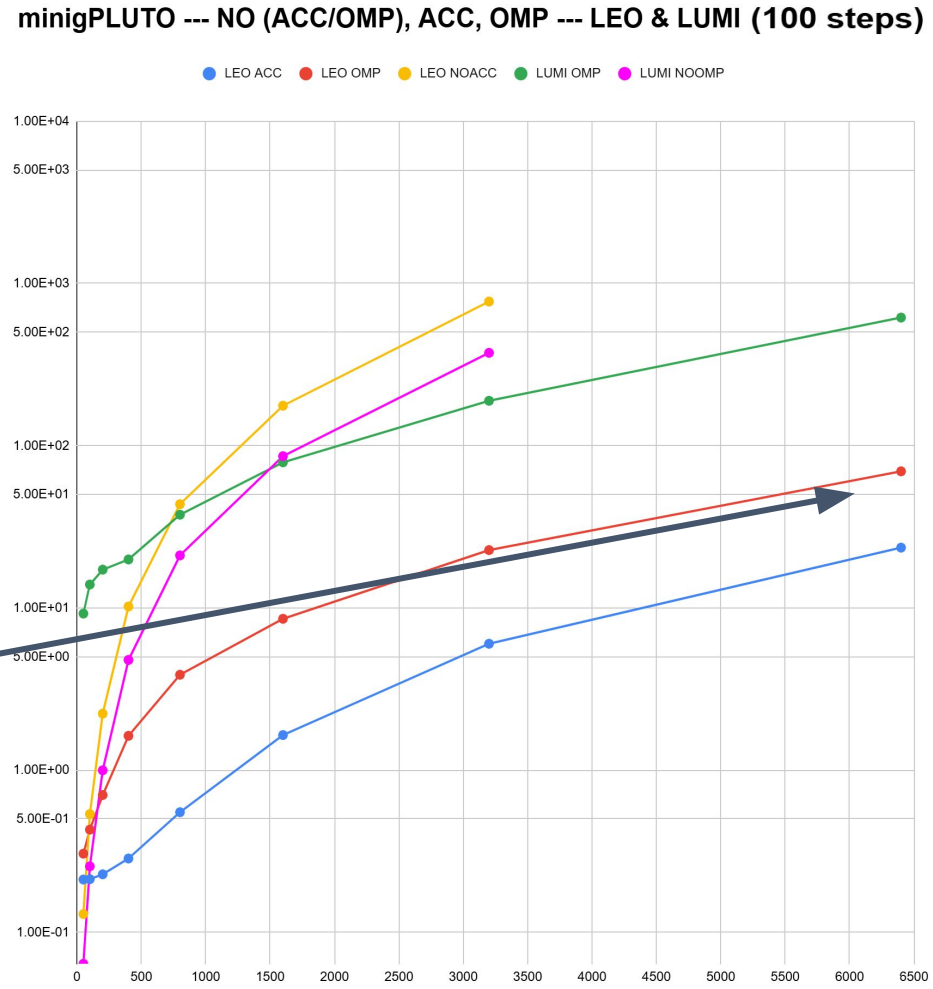
This is a **minigPLUTO** example!

gPLUTO uses more complex structures (e.g. pointers to pointers like `double ***var;`)

Compiling on LEO and LUMI

	LEONARDO BOOSTER	LUMI-G
Modules	• nvhpc/24.3, openmpi/4.1.6--nvhpc--24.3	• craype-x86-trento • PrgEnv-amd • craype-accel-amd-gfx90a • rocm
Compiler	nvc++	• CLANG • CLANG+OpenMP
FLAGs	CFLAGS = -c -g -std=c++17 -O -acc -gpu=managed (only ACC)	CFLAGS = -c -ggdb -O3 -std=c++17 -fopenmp (only OMP)
ENV_VAR		CRAY_ACC_USE_UNIFIED_MEM = 1 && HSA_XNACK = 1

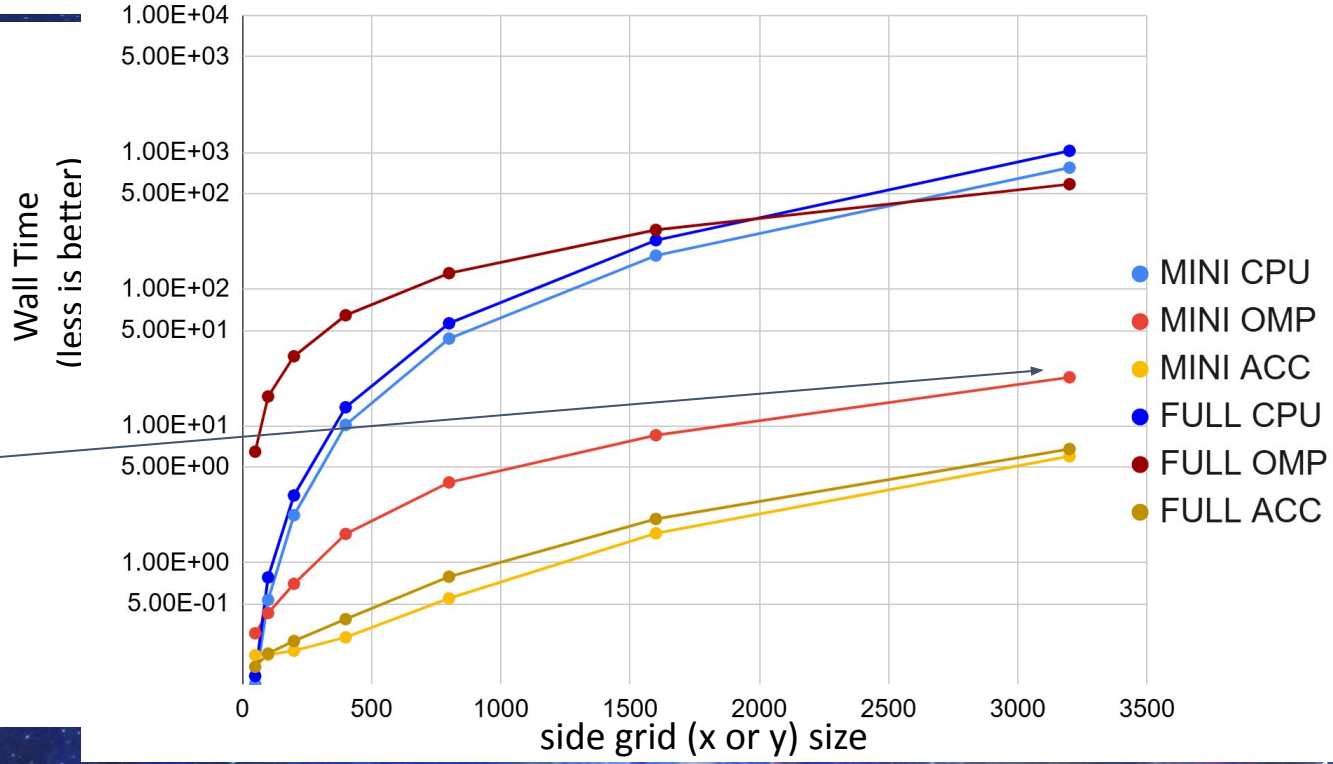




All similar
LEO promising results

LEO Full vs Mini gPLUTO Rieman2D HD (100 steps)

- **mini-gPLUTO** (HD module only)
- **simplified** data structures
- **tested** mini-gPLUTO on **Leonardo** ncv++ ACC and OMP
- minigPLUTO promising results



From OpenACC to OpenMP on Leonardo

From previous hackathon experiences:

- **gPLUTo** Rieman 2D Hidrodynamics
- **Code is compiled and run succesffully on NVIDIA GPU using NVHPC compiler**
- OpenMP is relatively slower than OpenACC