CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

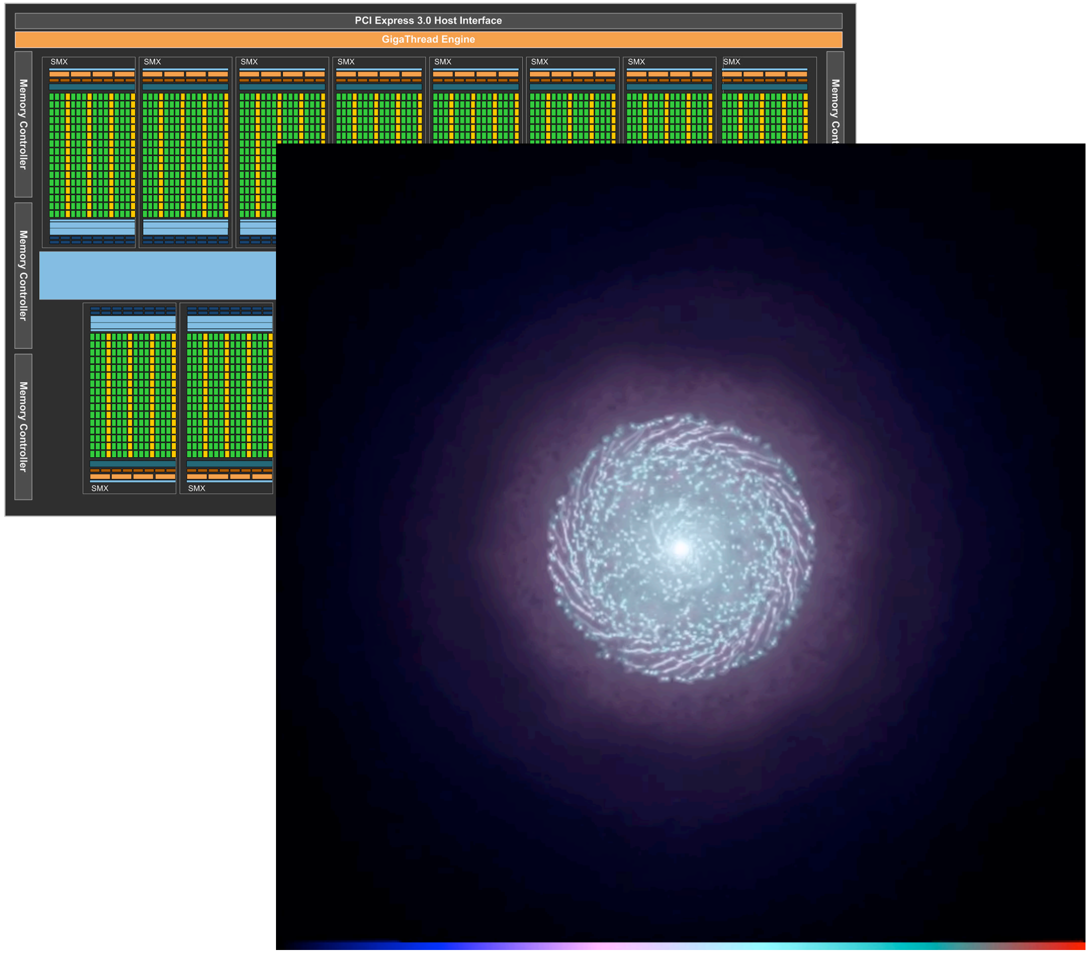
The Ramses Code for Numerical Astrophysics: Toward Full GPU Enabling

Claudio Gheller
(ETH Zurich - CSCS)

Giacomo Rosilho de Souza
(EPF Lausanne)

Marco Sutti
(EPF Lausanne)

Romain Teyssier
(University of Zurich)



Simulations in astrophyisics

- **Numerical simulations represent an extraordinary tool to study and solve astrophysical problems**
- **They are actual virtual laboratories, where numerical experiments can run**
- **Sophisticated codes are used to run the simulations on the most powerful HPC systems**

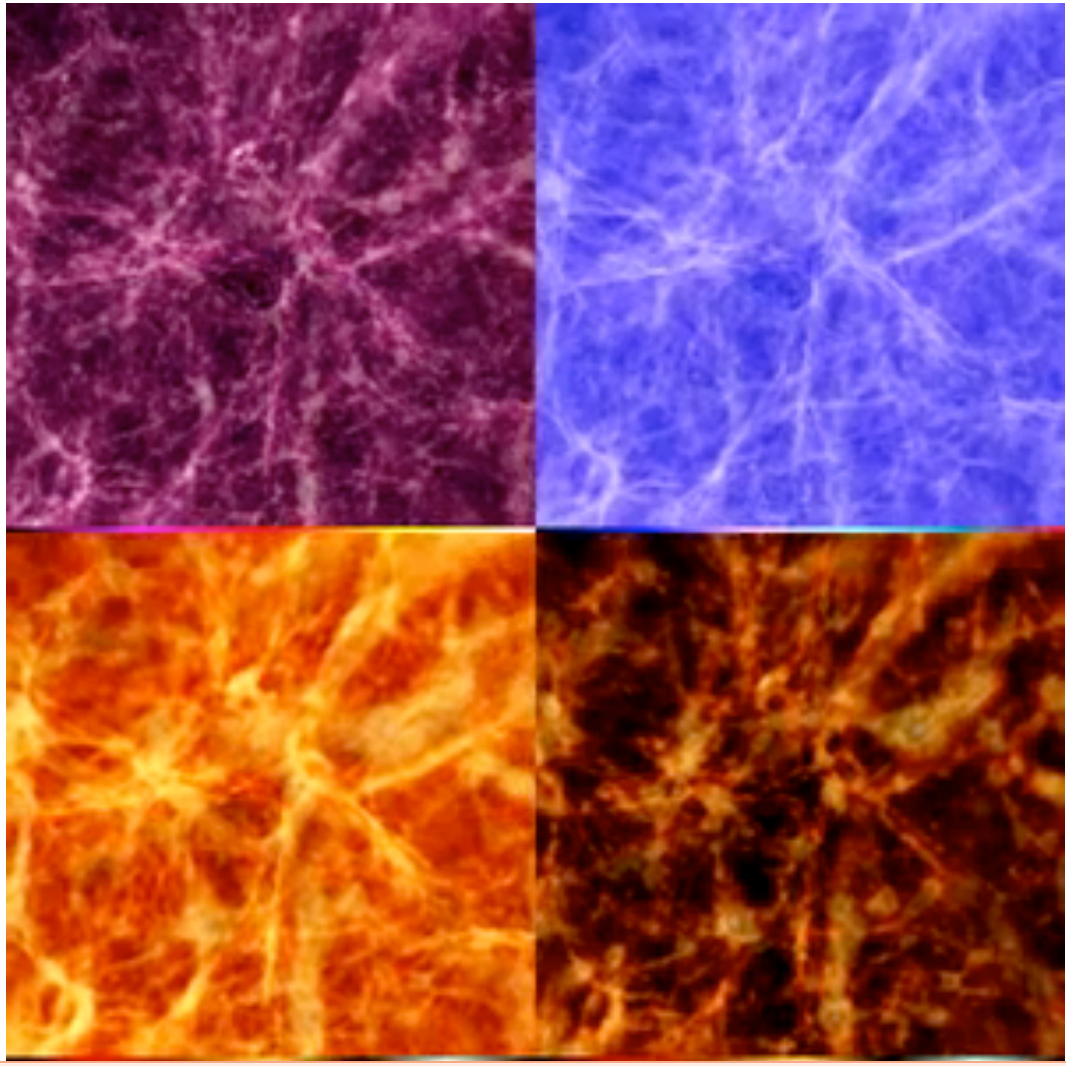
Evolution of the Large Scale Structure of the Universe

Visualization made with Splotch (https://github.com/splotchviz/splotch)

Magneticum Simulation, K.Dolag et al., http://www.magneticum.org

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Multi-species/quantities physics



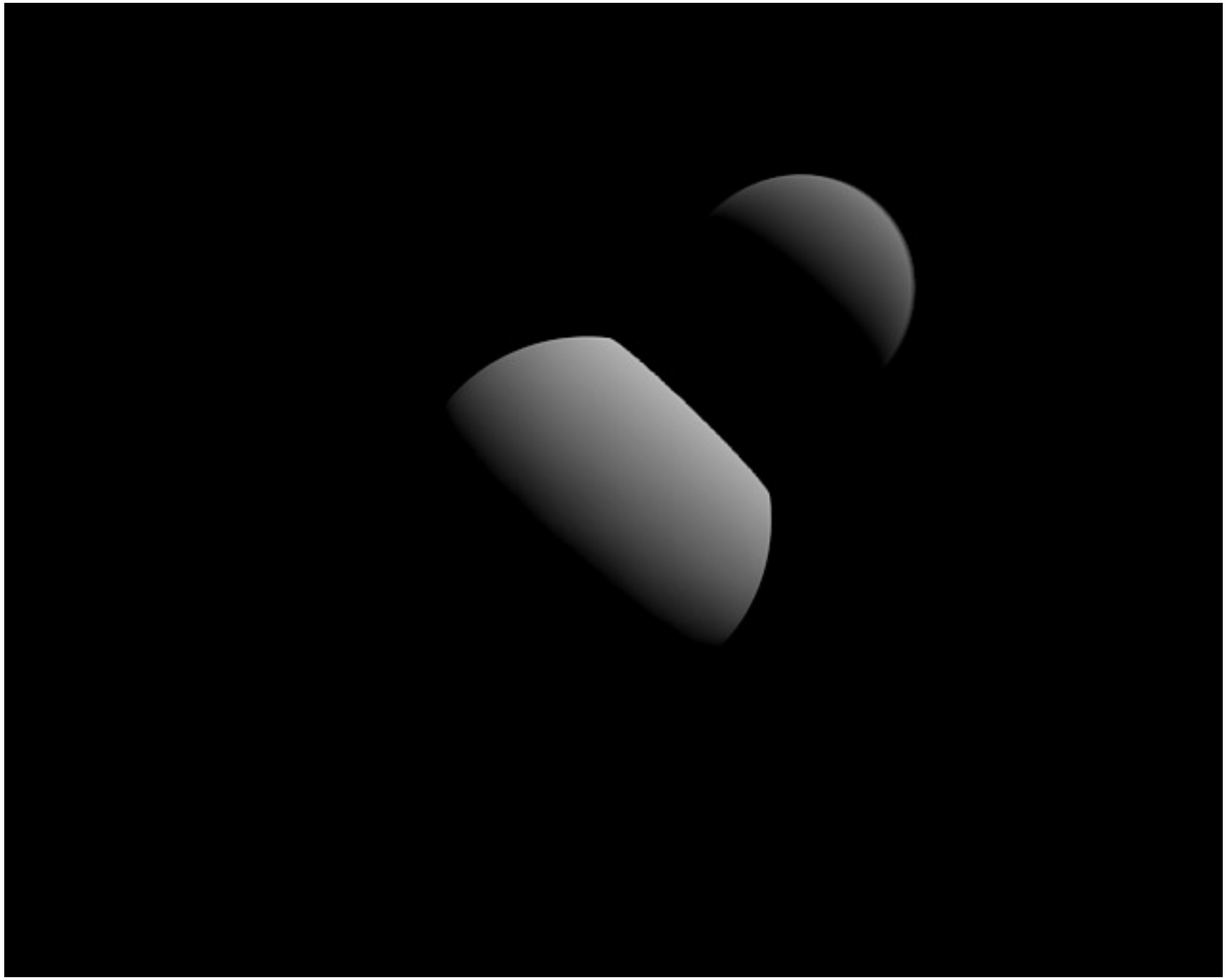
Visualization made with Splotch (https://github.com/splotchviz/splotch)

F.Vazza et al, Hamburg Observatory, CSCS, PRACE

Galaxy formation

IRIS simulation, L.Mayer et al., University of Zurich, CSCS

Formation of the moon



R.Canup et al., https://www.boulder.swri.edu/~robin/

Codes: RAMSES

- RAMSES (R.Teyssier, A&A, 385, 2002): code to study of astrophysical problems
- various components (dark energy, dark matter, baryonic matter, photons) treated
- Includes a variety of physical processes (gravity, magnetohydrodynamics, chemical reactions, star formation, supernova and AGN feedback, etc.)
- Adaptive Mesh Refinement adopted to provide high spatial resolution ONLY where this is strictly necessary

- Open Source
- Fortran 90
- Code size: about 70000 lines
- MPI parallel (public version)
- OpenMP support (restricted access)
- OpenACC under development

HPC power: Piz Daint

"Piz Daint" CRAY XC30 system @ CSCS (N.6 in Top500)

Nodes:

5272 CPUs 8-core Intel SandyBridge equipped with:

- 32 GB DDR3 memory
- One **NVIDIA Tesla K20X GPU** with 6 GB of GDDR5 memory

Overall system

- 42176 cores and 5272 GPUs
- 170+32 TB
- Interconnect: Aries routing and communications ASIC, and dragonfly network topology
- Peak performance: 7.787 Petaflops

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Scope

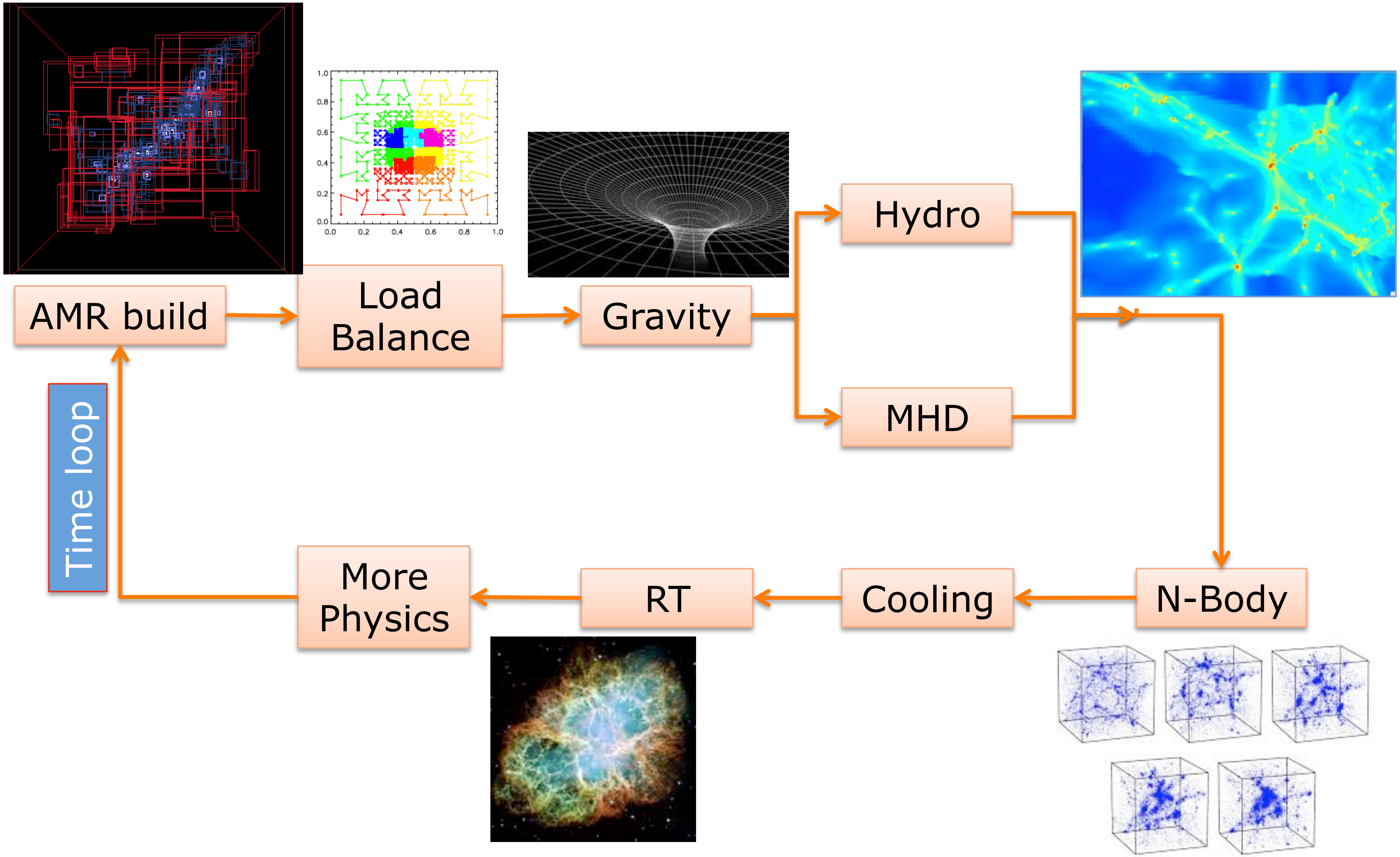
Overall goal: Enable the RAMSES code to exploit hybrid, accelerated architectures

Adopted programming model: OpenACC (http://www.openacc-standard.org/)

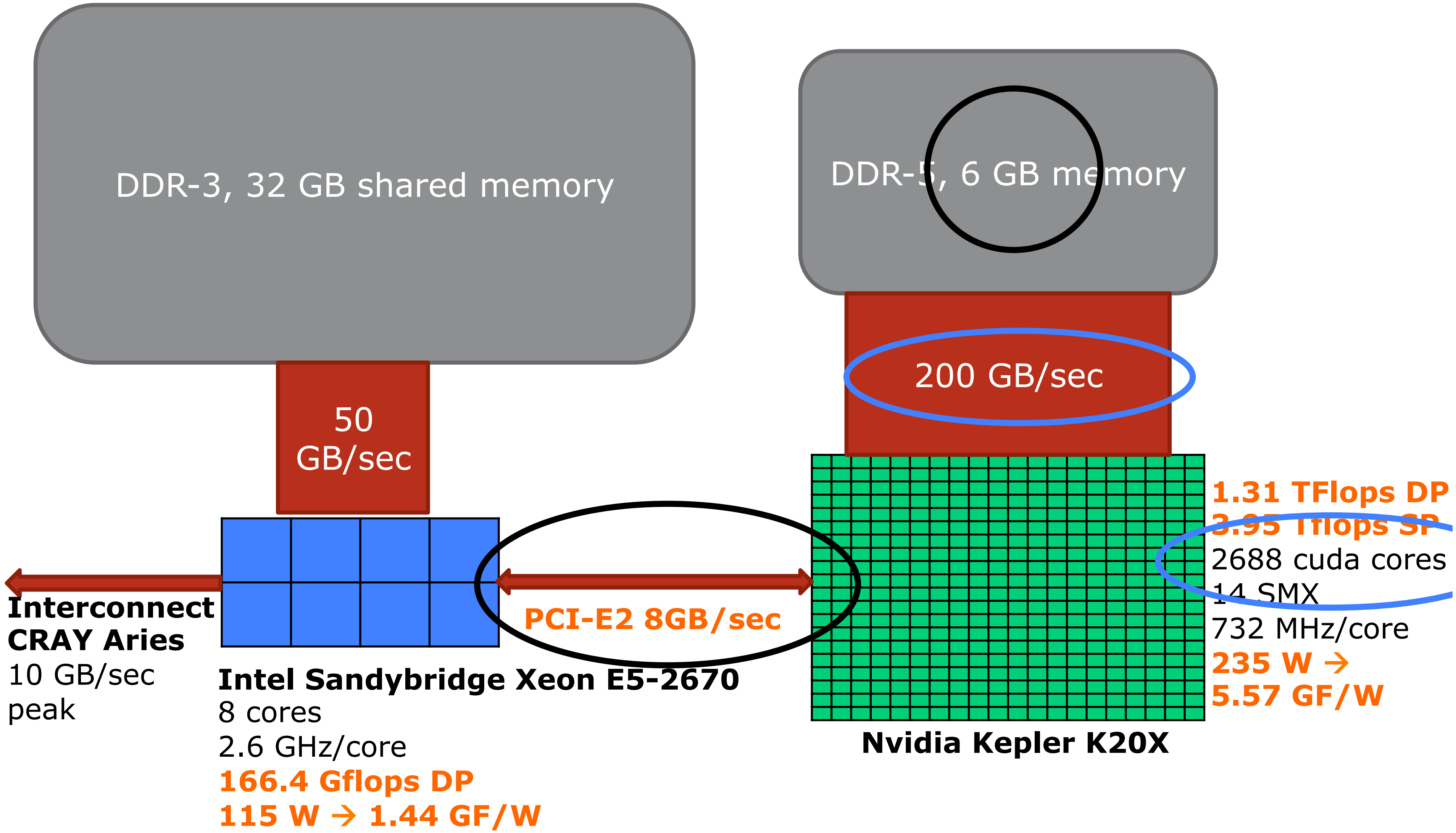
Development follows an incremental "bottom-up" approach

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

RAMSES: modular physics

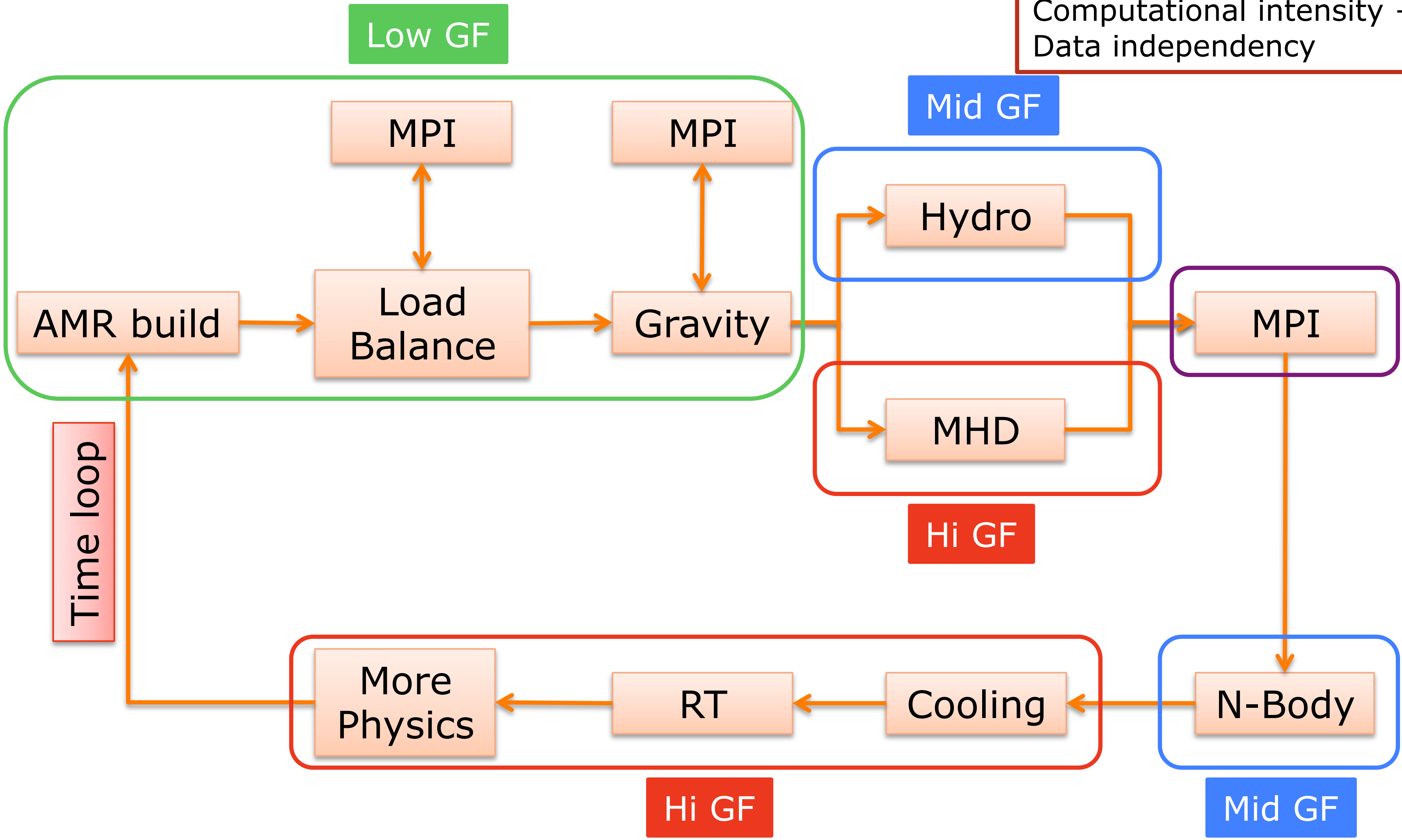


Processor architecture (Piz Daint)



RAMSES: Modular, incremental GPU implementation

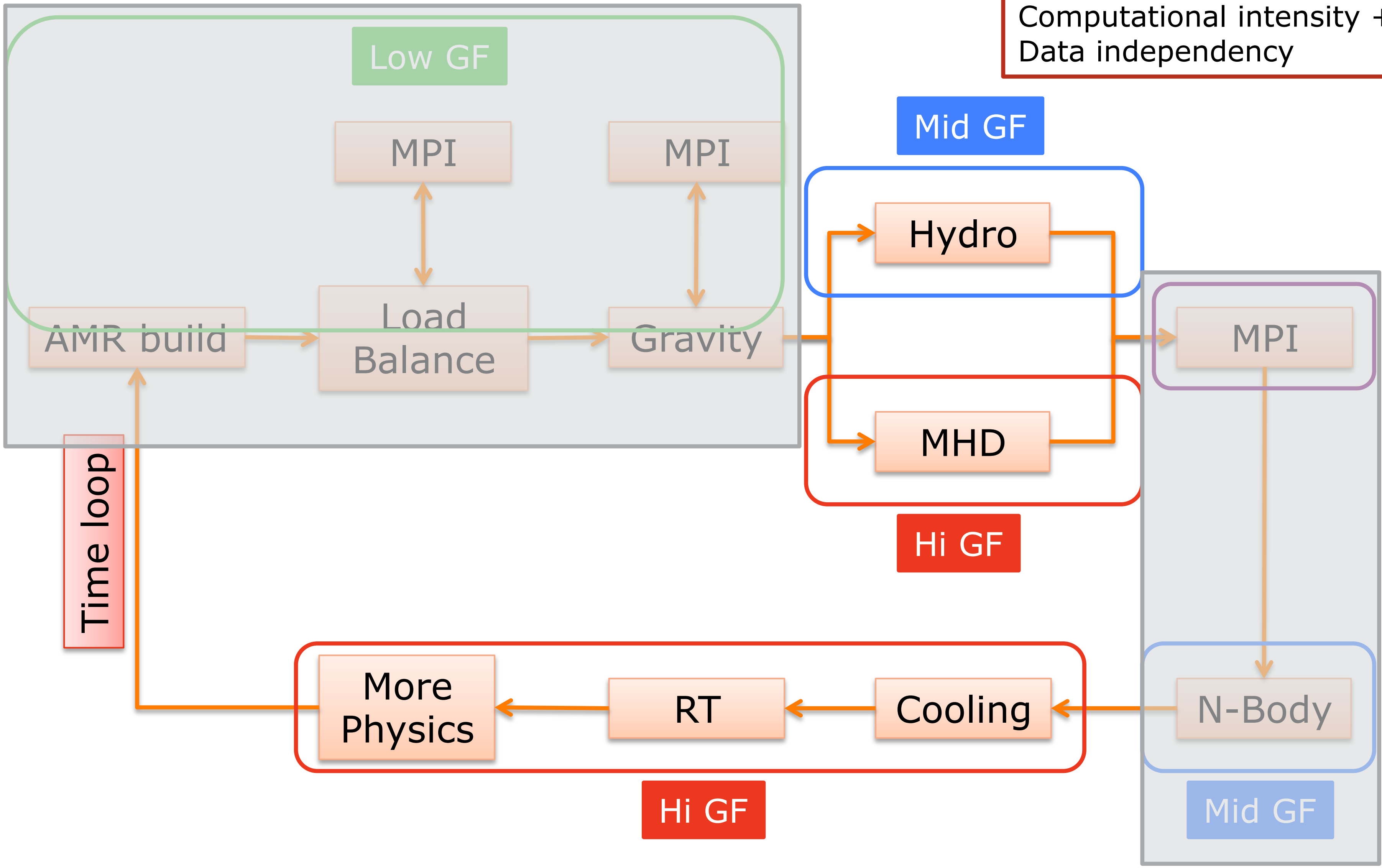
GF = "GPU FRIENDLY"
Computational intensity +
Data independency



CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

First steps toward the GPU

GF = "GPU FRIENDLY"
Computational intensity + Data independency



CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

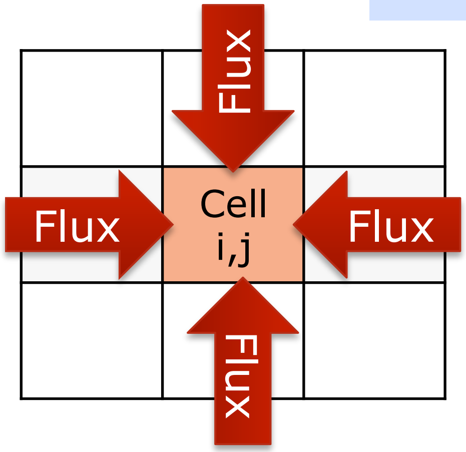
Step 1: solving fluid dynamics

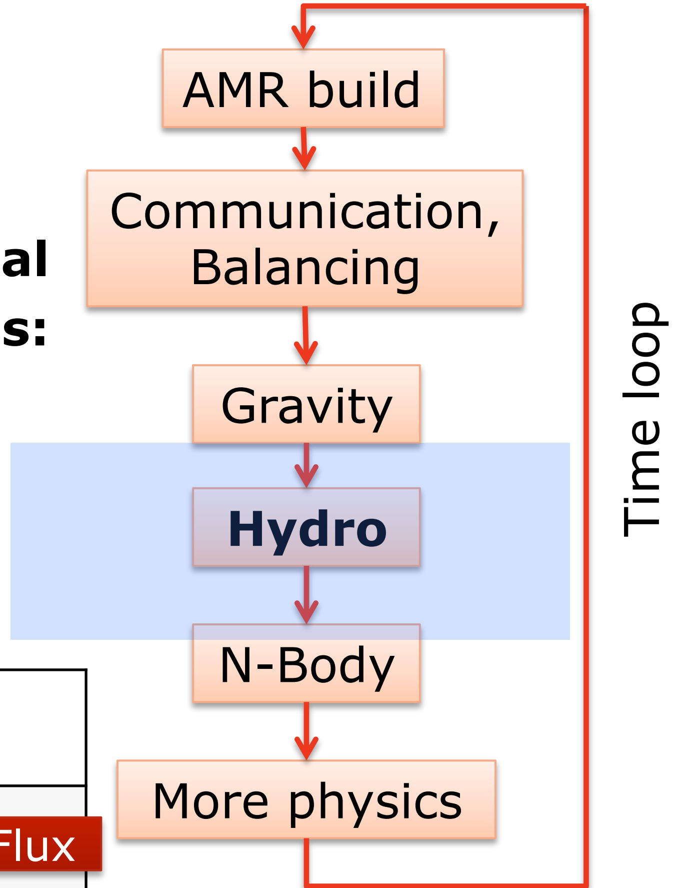
- **Fluid dynamics is one of the key kernels;**
- **It is also among the most computational demanding;**
- **It is a local problem;**
- **fluid dynamics is solved on a computational mesh solving three conservation equations: mass, momentum and energy:**

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0$$

$$\frac{\partial}{\partial t}(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p = -\rho \nabla \phi$$

$$\frac{\partial}{\partial t}(\rho e) + \nabla \cdot [\rho \mathbf{u}\,(e + p/\rho)] = -\rho \mathbf{u} \cdot \nabla \phi$$

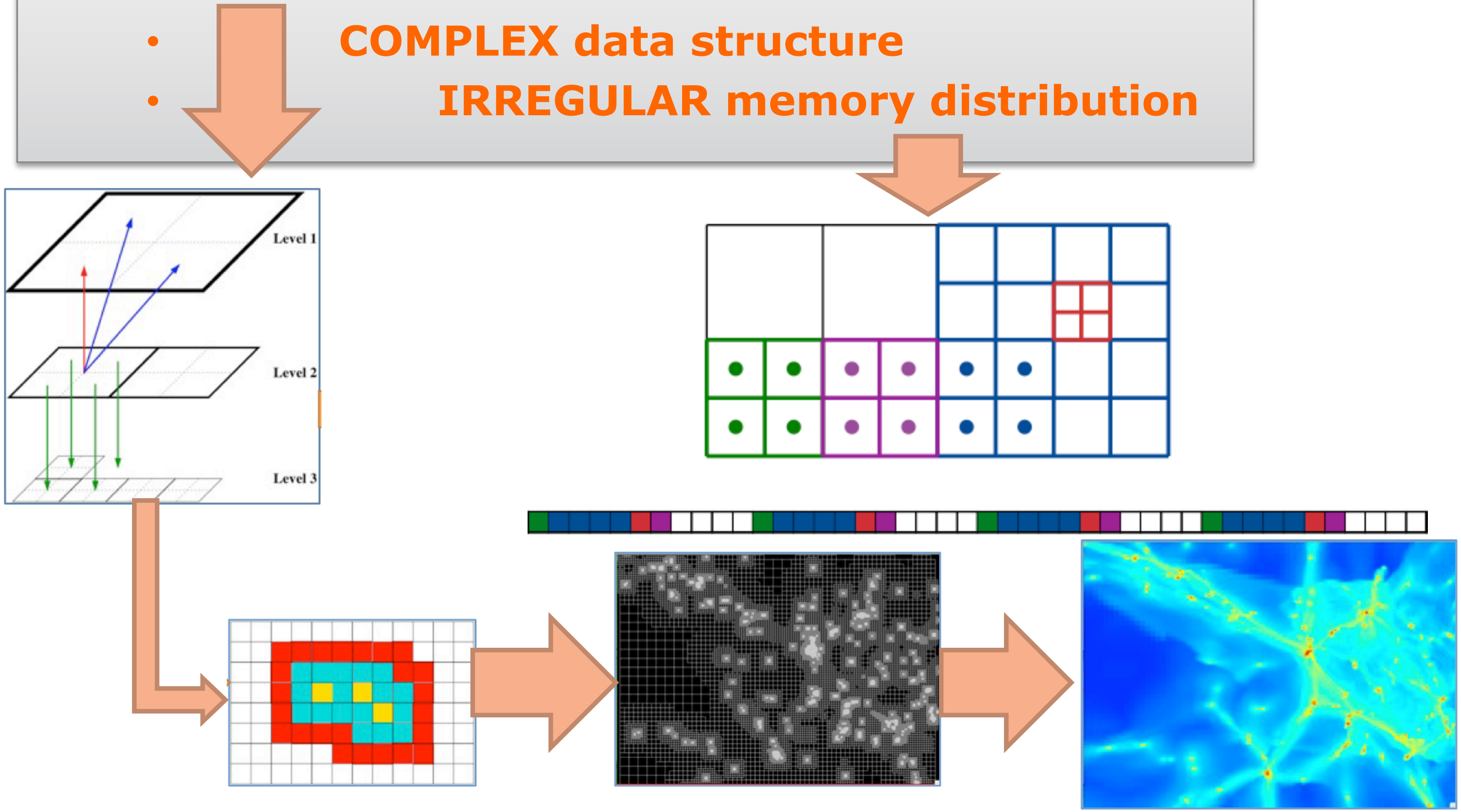




The challenge: RAMSES AMR Mesh

Fully Threaded Tree with Cartesian mesh

- **CELL BY CELL refinement**
- **COMPLEX data structure**
- **IRREGULAR memory distribution**



CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

GPU implementation of the Hydro kernel

1. Memory Bandwidth:

1. reorganization of memory in spatially (and memory) contiguous large patches, so that work can be easily split in blocks with efficient memory access
2. Further grouping of patches to increase data locality

2. Parallelism:

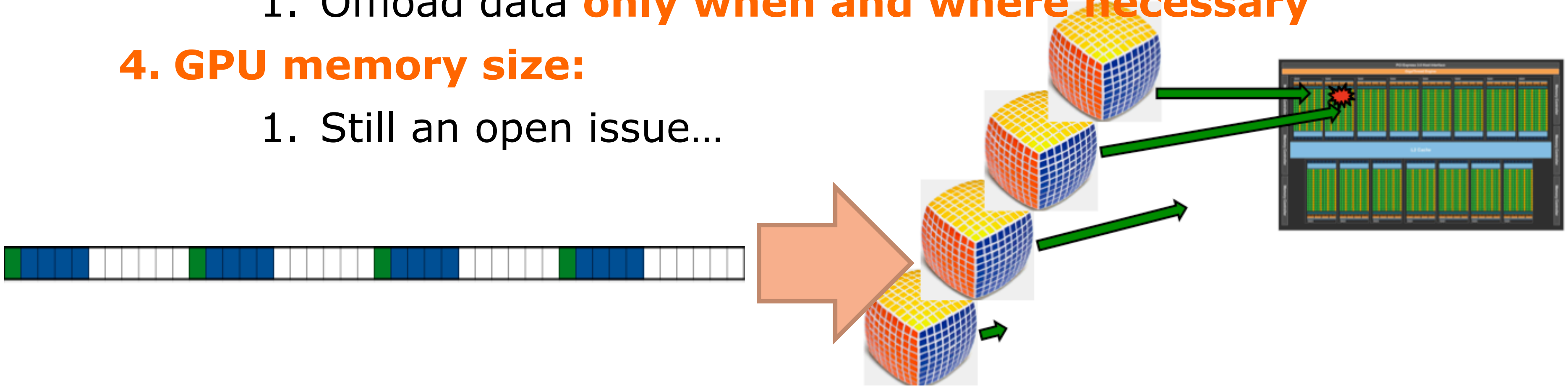
1. patches to blocks assignment,
2. one cell per thread integration

3. Data transfer:

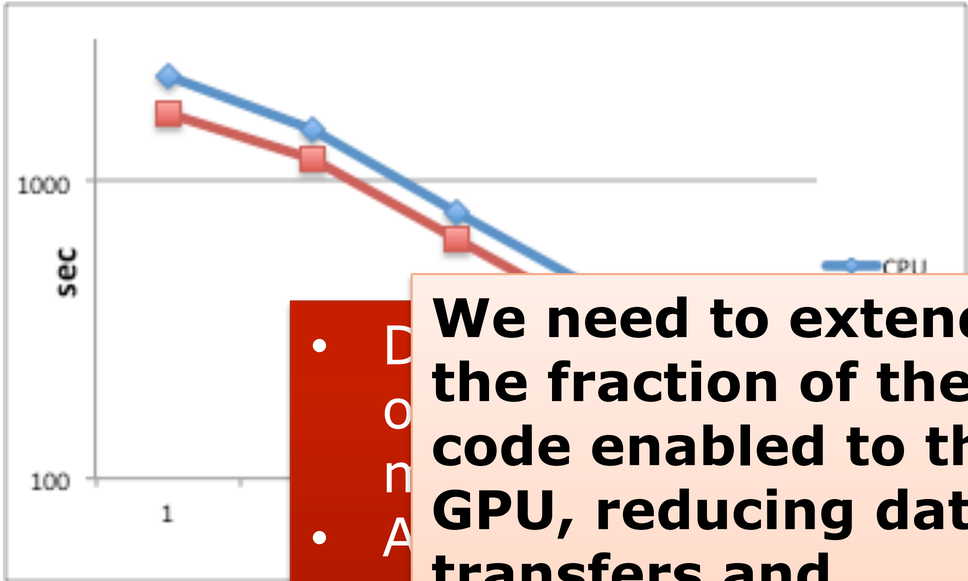
1. Offload data **only when and where necessary**

4. GPU memory size:

1. Still an open issue…



Some Results: hydro only



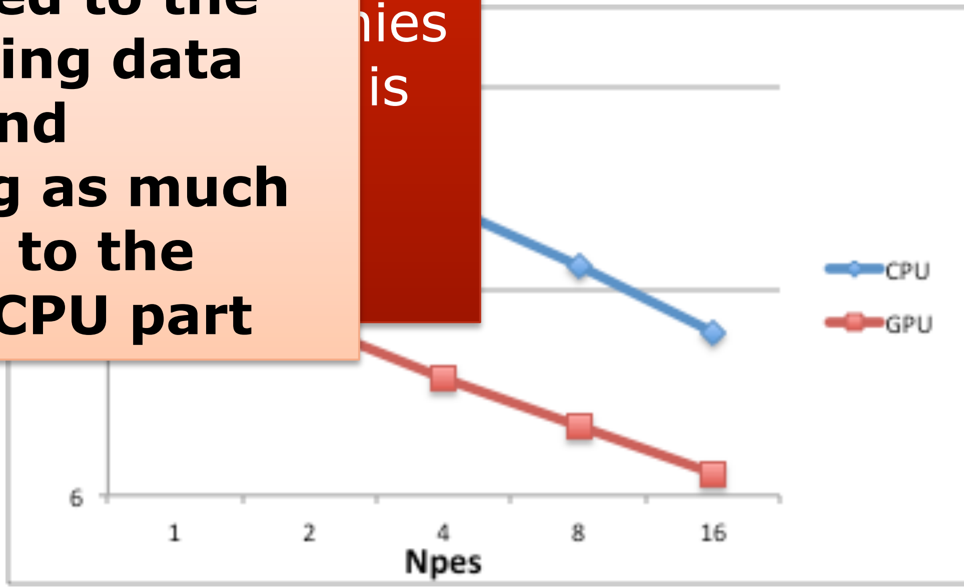
Scalability of
and GPU vers
(Total time)

Fraction of time saved using the GPU

0.2431415
0.20057766
13076

- D … 0%
  o … h
  n … ies
- A … is
  s
- N
  c

We need to extend the fraction of the code enabled to the GPU, reducing data transfers and overlapping as much as possible to the remaining CPU part

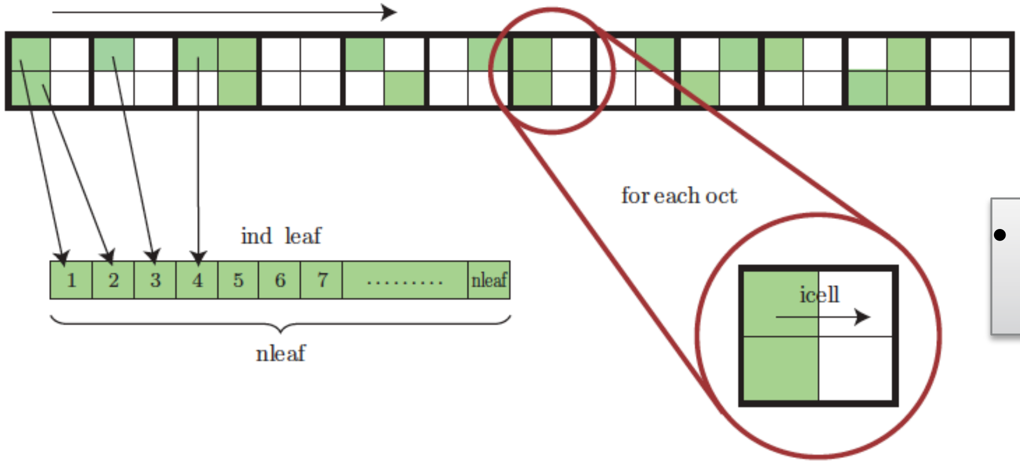


Scalability of the CPU and GPU versions (Hydro time)

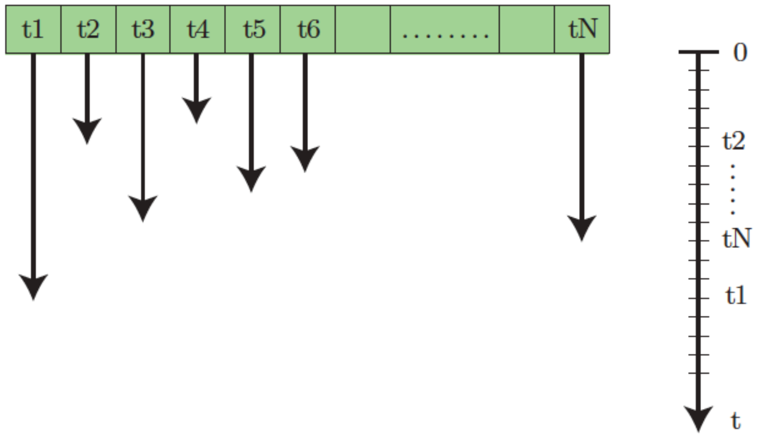
Step 2: Adding the cooling module

$$\frac{\partial}{\partial t}(\rho e) + \nabla \cdot \left[\rho \mathbf{u}\left(e + \frac{p}{\rho}\right)\right] = -\rho \mathbf{u} \cdot \nabla \phi + \sum_{i=1}^{n} \mathcal{L}_i$$

- **Energy is corrected only on leaf cells independently**
- **GPU implementation requires minimization of data transfer…**

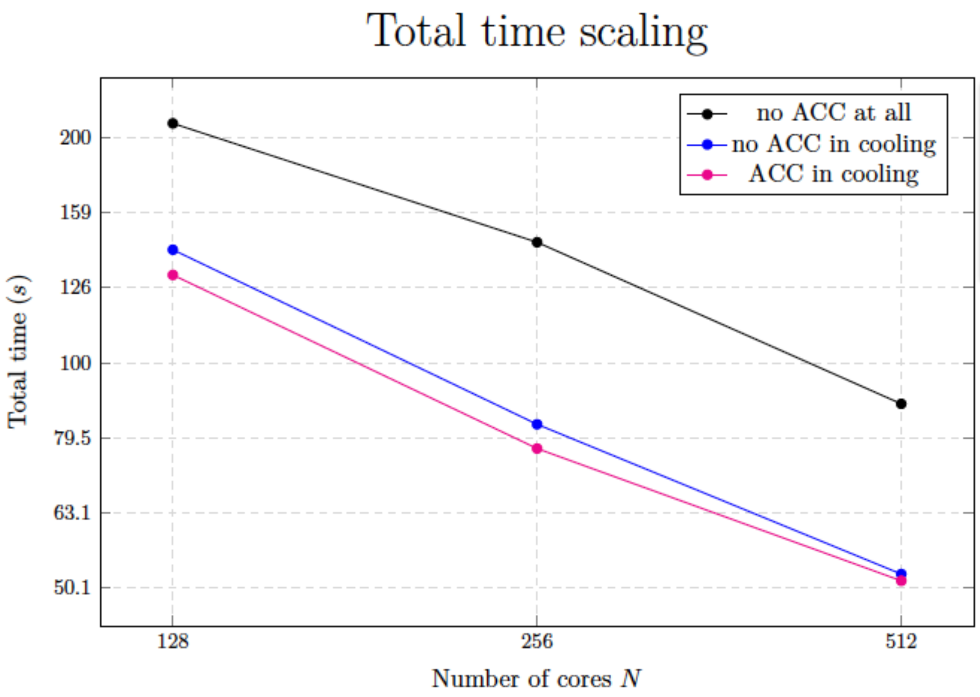


- **Iterative procedure with a cell-by-cell timestep**

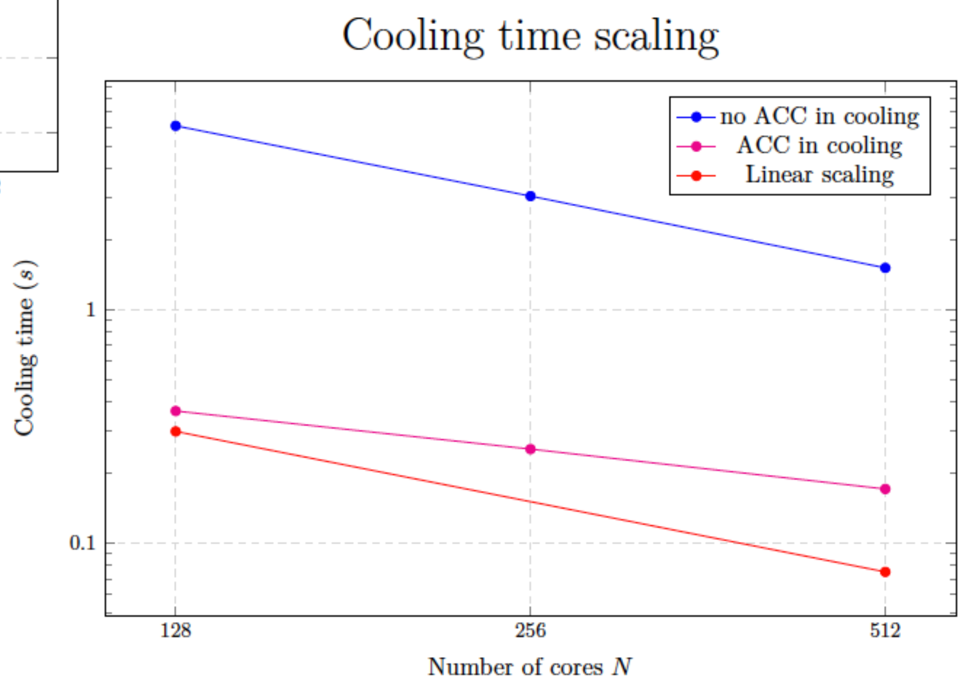


- **exploitation of the high degree of parallelism with "automatic" load balancing:**

Adding the cooling

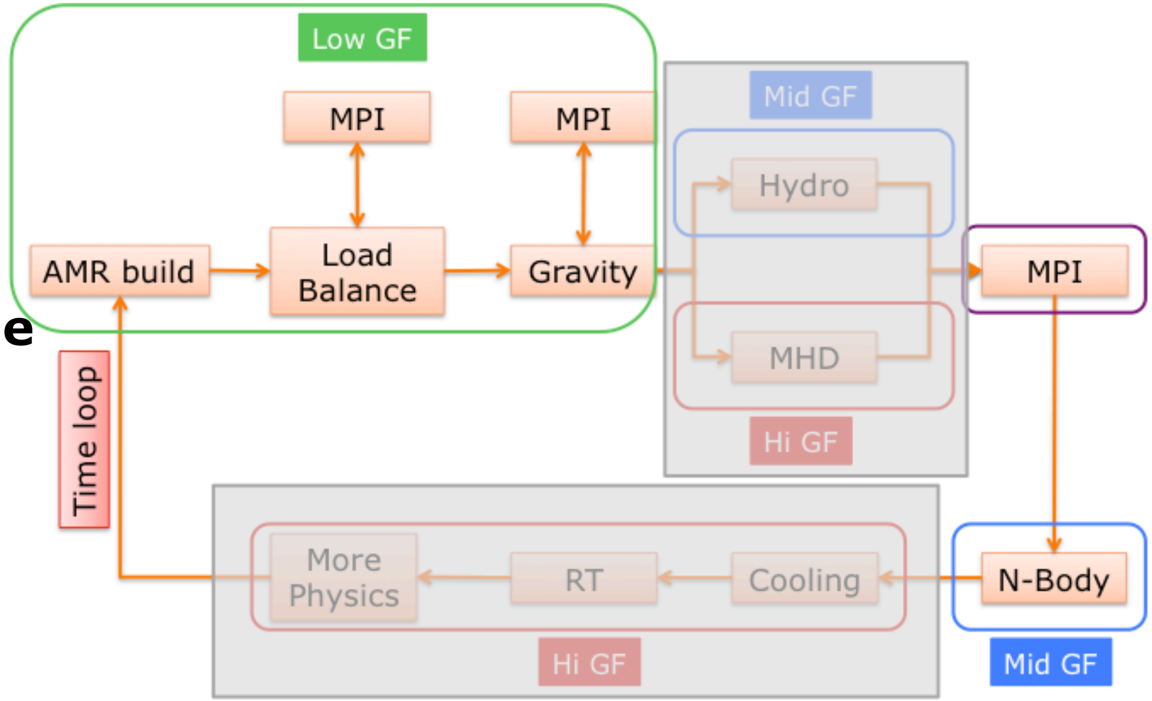


- **Comparing 64 GPUs to 64 CPUs: Speed-up = 2.55**



Toward a full GPU enabling

- **Gravity is being moved to the GPU**
- **ALL MPI communication is being moved to the GPU using the GPUDirect MPI implementation**



- **N-body will stay on the CPU**
  - Low computational intensity
  - Can easily **overlap** to the GPU
  - No need of transferring all particle data, **saving time but especially GPU memory**

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Summary

Objective:

Enable the RAMSES code to the GPU

Methodology

Incremental approach exploiting RAMSES'modular architecture and OpenACC programming mode

Current achievement:

Hydro and Cooling kernels ported on GPU; MHD kernel almost done

On-going work:

- **Move all MPI stuff to the GPU**
- **Enable gravity to the GPU**
- **Data transfer minimization**

Thanks for your attention