Workshop EXA-DI **Block-structured AMR @Exascale**

Optimizing I/O performance for AMR Code: A case study with RAMSES in Astrophysics

By Loïc STRAFELLA

Table of contents

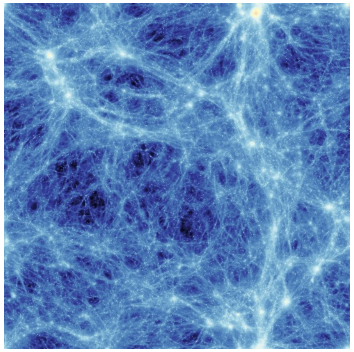
Part I

Why astrophysicists need AMR?

Introduction

The universe (~ giga/mega parsec)**:**

- Filamentary structure : “cosmic web”



Simulation *Extreme-Horizon* [Chabanier et al, 2020]

Galaxy (~ kilo parsec)**:**

- Classification: elliptical, irregular, spiral, ...
- Contains: dark matter, gas, dust
- Different thermodynamic states for gas:
  - Hot ionized (HIM), warm ionized (WIM), war/cold neutral (WNM, CNM), molecular

NGC 1300**,** Hubble, (NASA / ESA)

Molecular cloud (~ tens parsec)**:**

- Made of molecular gas: H_2, CO, …
- Found in spirals arms, central bar, …
- Spot for stellar formation:
  - Kennicutt-Schmidt law [Kennicutt Jr, 1998]

Perseus Molecular Cloud, NASA/JPL-Caltech

1 parsec ~ 3.10^{13} km

Introduction

Multi-scale processes:

- Spatial resolution vary by several order of magnitude:
  - Example: galactic scale ~30 kpc, molecular cloud 50 pc, ...
- Field magnitude can also vary by several order of magnitude:
  - Density 10^{-6} H/cm^3 à 10^{+6} H/cm^3

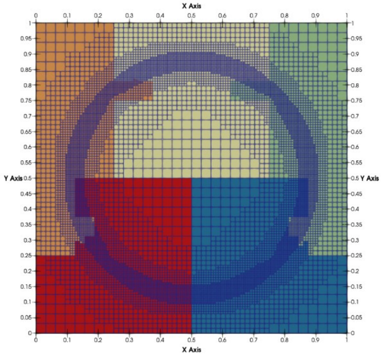
M**ulti-physics:**

- Large scale gravitational/hydrodynamical instabilities
- Thermodynamics, magnetic fields, ...
- Star formation models and its feedback (ionization, supernovae, ...)

→ Need of numerical simulation **with** specific mesh strategy (AMR)

RAMSES [Teyssier, 2002]:

- MPI only, Fortran
- Cell-based
- Fully threaded tree [Khokhlov, 1998]



AMR: Adaptive Mesh Refinement

Introduction : objectives and limitations

State-of-the-art simulations:

- **Sub-galactic region**, e.g. [Brucy et al, 2020], [Colman et al, 2022]
  - Missing large scale turbulence injection mechanism
- **Simplified physics models**
  - No heating/cooling process, [Renaud et al, 2013]
  - No star formation (**SF**), no feedback [Fensch et al. 2023]
- **Short physical time evolution at high resolution**
- **High resolution <u>only</u> in high density region**
  - Missing instabilities that may develop in low density region

<u>Objectives</u>

Full Galaxy (isolated)

Heating/cooling + SF + Supernovae

As long as possible

High resolution in both low and high density region

<u>Requires preliminary work to</u> :

- Reduce the computational cost
- Improve code scalability in terms of consumed memory
- **Improve code scalability regarding I/O**

Introduction : objectives and limitations, DMP

Objective:

Study the regulation of star formation in a Milky Way like galaxy

Data Management Plan (DMP): **FAIR – open science**
« **F**indable, **A**ccessible, **I**nteroperable, **R**eusable »

DMP for the simulation Exa-Milkyway:

- Code **RAMSES** [Teyssier, 2002] (MPI only – no OpenMP)
- **Supercomputer**: CPU **AMD** Epyc 128 cores / node, 256 Go, Irène **TGCC** *
- **Target:** 16.384 – 50.000 MPI processes
- **57 Millions** CPU-hours, **GENCI** **
- 90 To of disk space: 50 To checkpoint + 40 To for analysis
- Open science
  - Web Galactica: http://www.galactica-simulations.eu/db/

* **TGCC**: Très Grand Centre de Calcul du CEA
** **GENCI**: Grand équipement national de calcul intensif

Bottlenecks and critical points

Bottlenecks and critical points:

- ✗ **Low I/O scalability**: ~ 8.000 MPI processes
  - ➢ Time to write/read data (checkpoints): ~ 1h, [Renaud, 2013]
  - ➢ 10 to 20 % of time allocation spent on I/O
- ✗ **High number of files**:
  - ➢ « *file-per-process* » strategy
  - ➢ Limited quota: **2 millions*** (~30 snapshots** max.)
- ✗ **Complex I/O pattern** due to AMR [Wautelet and Kestener, 2011]
- ✗ **Only one data flow**
  - ➢ Data used for both analysis and checkpoint
- ✗ **High data volume production**
  - ➢ Limited quota: **100 To*** (~30 snaphots max.)

I/O & data

- ✗ **Low scalability memory footprint**

Memory

- ✗ Not possible to change the number of MPI processes at restart

MPI

* *Quota per project at* TGCC

** *Snapshot : directory / files associated to a given time*

After I/O optimization

Solutions:

- **Improved I/O scalability up to 50,000 MPI**
  - Time to write or read data (16.384): **~ 3 minutes** (4,7 To) (vs ~ 1h)
- **Reduced number of files**:
  - ~200,000 files : ~ **150 snapshots** (vs ~ 30)
- Separation of data flow, new data model, data compression
  - **90 To** disk usage: **140 snapshots for analysis, 10 snapshots checkpoints**
  - **200 Go / snapshot** (analysis) vs **4,7 To / snapshot** (checkpoint)

I/O & Data

- Optimization of **memory footprint*** :
  - **Reduction of 64 Go / node**** (25 % of available memory per node)

Memory

- New feature: changing the number of MPI processes on restart

MPI

Tested, **validated** and **used** in production run Exa-MilkyWay

** With 16384 MPI on the ExaMilkyWay simulation*

*** Patch "LIGHT_MPI_COMM" open sourced for the community*

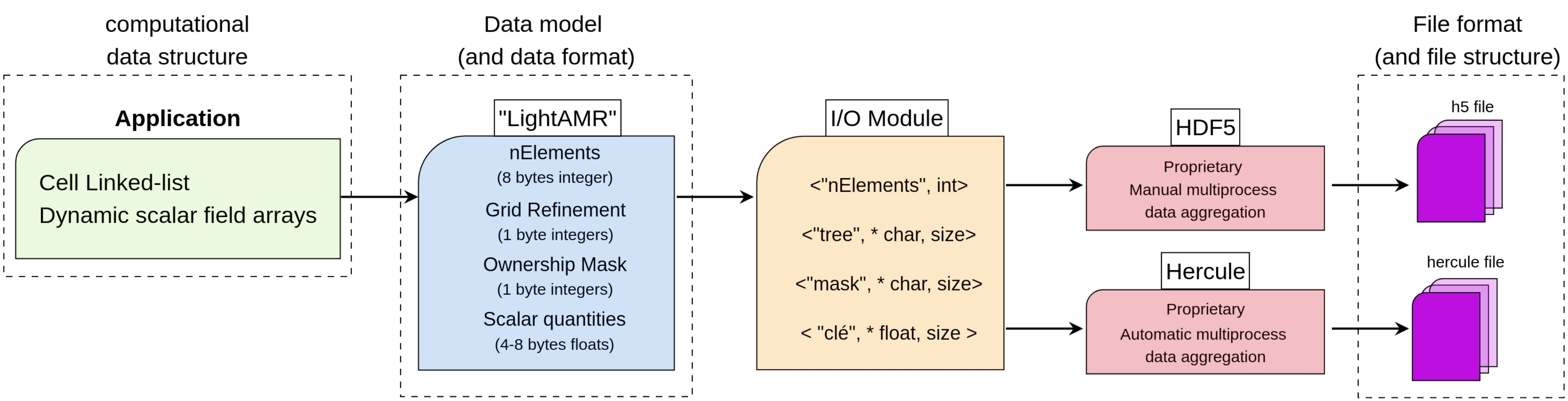
Part II

I/O Optimization

I/O: Some definitions

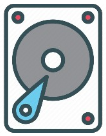
<u>Different structures</u>:

- ***Computational data structure***: linked list, hash map, ….
- ***Data model*** *(for I/O)*: “unstructured”, “custom”*, “lightAMR”, …
- ***Data format***: compressed or not, specific encoding strategy, ...
- ***File format*** *(and file structure): linked to the I/O library*
  - Different file format and file structure possible for a given data model



* “custom” = specific to a given code

I/O: RAMSES I/O strategy



<u>Initial state</u>:

- **« file-per-process » strategy**
  - *Lot of files:* does not take advantage of parallel filesystem
- **Custom data model**
  - *Per level data write/read* : lot of small I/O calls
- **Custom file format with Posix interface**
  - Nightmare for post-processing tools
- **Single data flow for multiple purposes**: checkpoint & analysis
  - Too much data for analysis
  - Unnecessary computation for checkpoint/restart
  - Use of temporary arrays

How to optimize the I/O ?

- ~~Try to optimize file system or user parameters~~
- Use a specific library (made by expert in the field)
- Small number of I/O calls with large amount of data
- Reduce the data volume (data model, data compression, reduce precision, …)

I/O: Optimization strategy

Objectives:

- Change I/O strategy ("file-per-process") → Hercule parallel I/O library
- Use of specific parallel I/O library
- Separation of data flow based on purpose → Split current data flow
- Change I/O pattern → New data models

*: Developed at CEA-DAM, [Bressand et al, 2012]

I/O: Optimization strategy

Objectives:
- Change I/O strategy (“file-per-process”) → Hercule parallel I/O library
- Use of specific parallel I/O library
- Separation of data flow based on purpose → Split current data flow
- Change I/O pattern → New data models

→ **Parallel I/O libraries**: HDF5, NetCDF, MPI-IO, **Hercule** *, ...

Advantages of Hercule:
- Native parallel I/O based on application parallelism (MPI)
- Simple **API** (compare to HDF5 for example)
- Database like strategy
- “Key – value”: easy access to data

I/O strategy, data centric, data management

→ Separation of data flow:
- One specific for checkpoint/restart
- One specific for analysis

Reduce data volume, optimize data model, Reduce memory footprint, Optimize I/O pattern

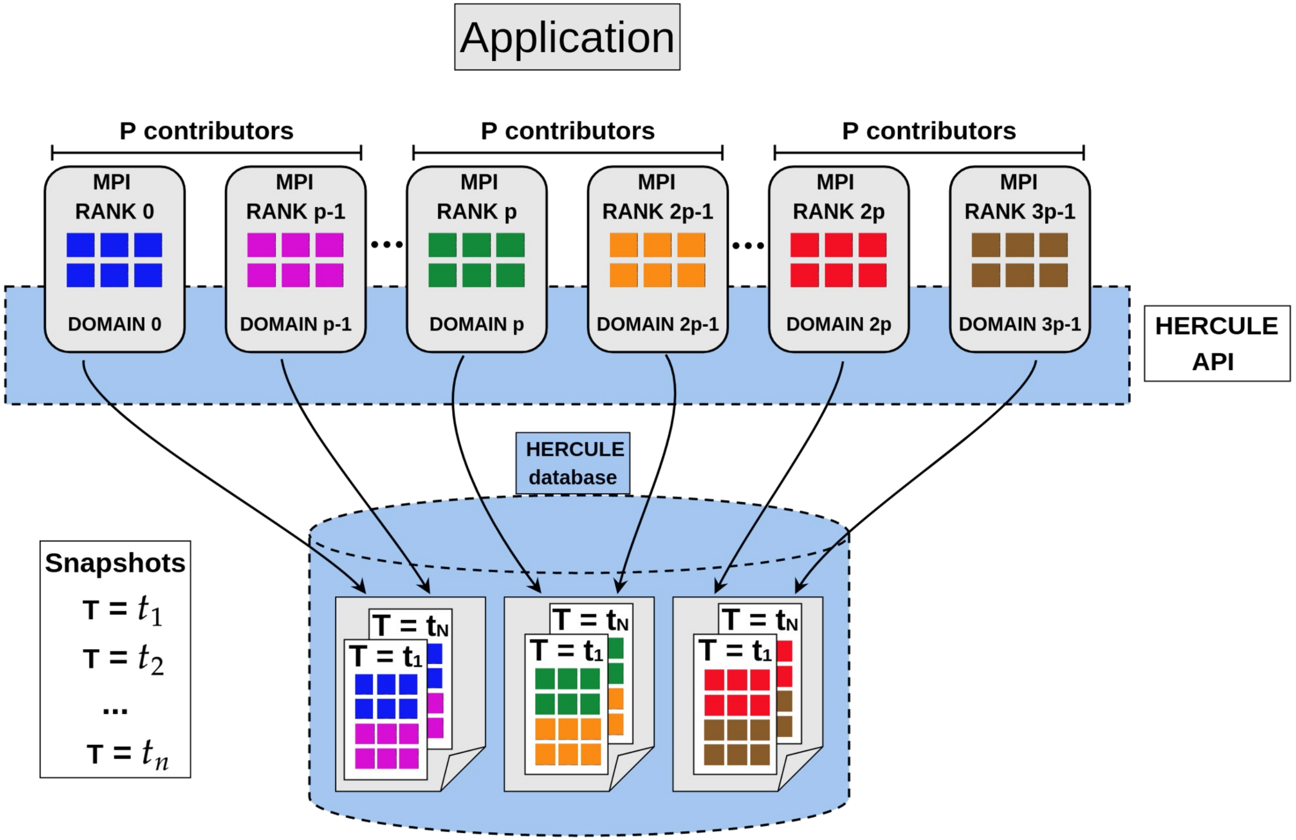
*: Developed at CEA-DAM, [Bressand et al, 2012]

I/O: Hercule

Key points:

- I/O strategy: « multiple-shared-file »
- Database like access: based on time, on MPI rank (domain), ...
- Data accumulation in the same file ("append")

→ Transparent for the user

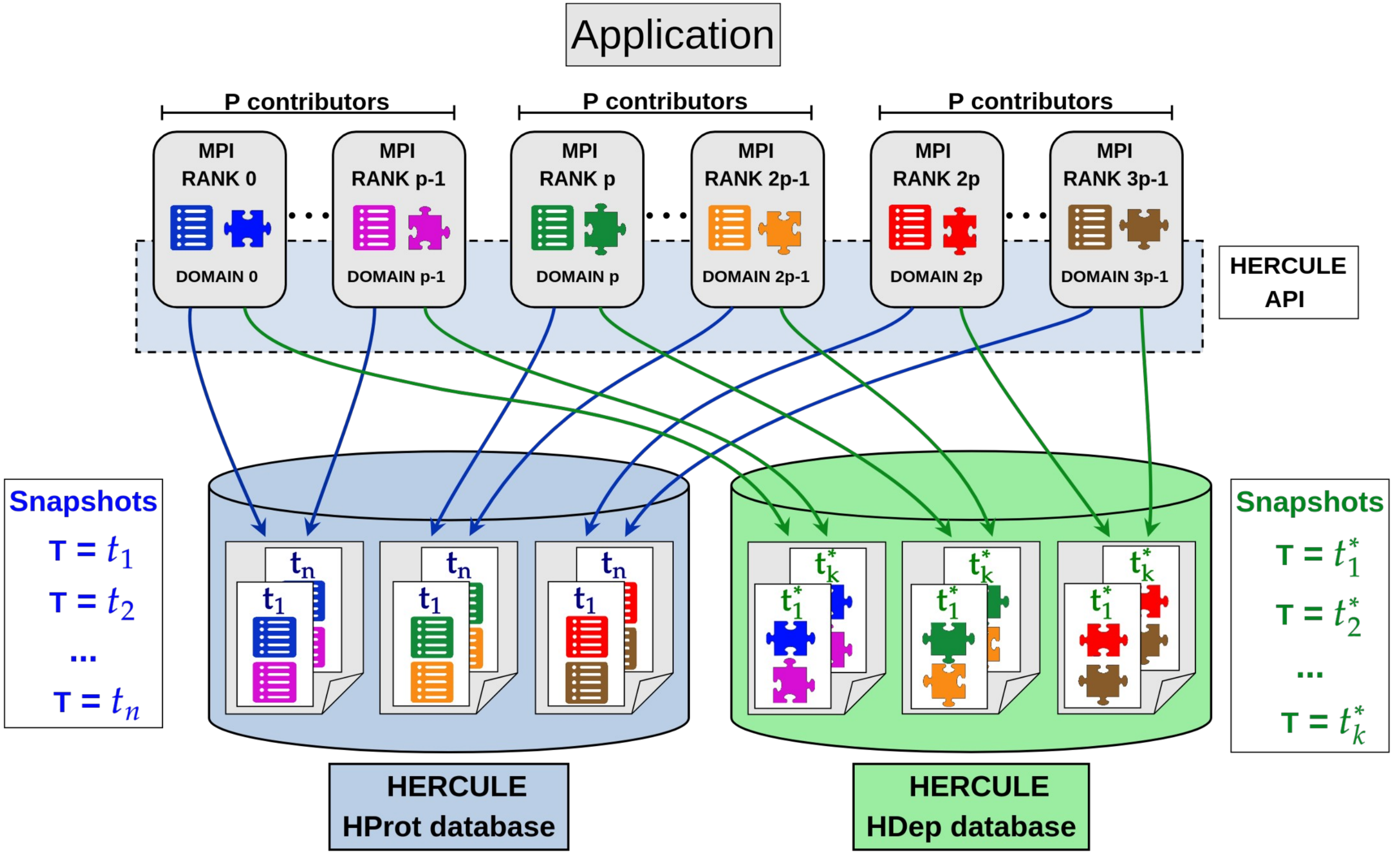


I/O: Hercule database

Two different Hercule databases:

HProt ➡ For checkpoint/restart, no specific semantic, “basic” write/read operation

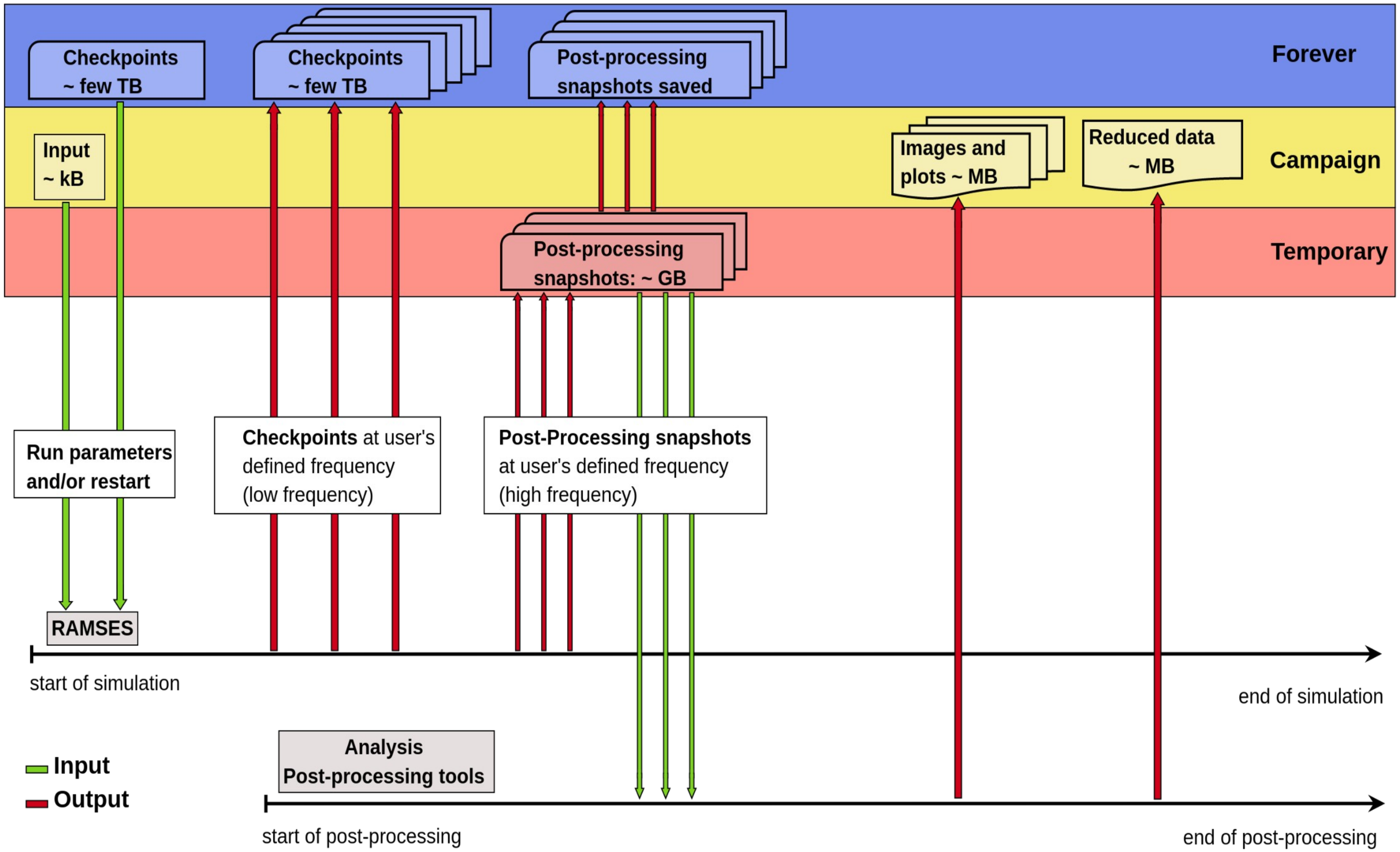
HDep ➡ For analysis and post-processing, specific data model and semantic API



I/O: New data flow diagram

New RAMSES data flow with Hercule's databases

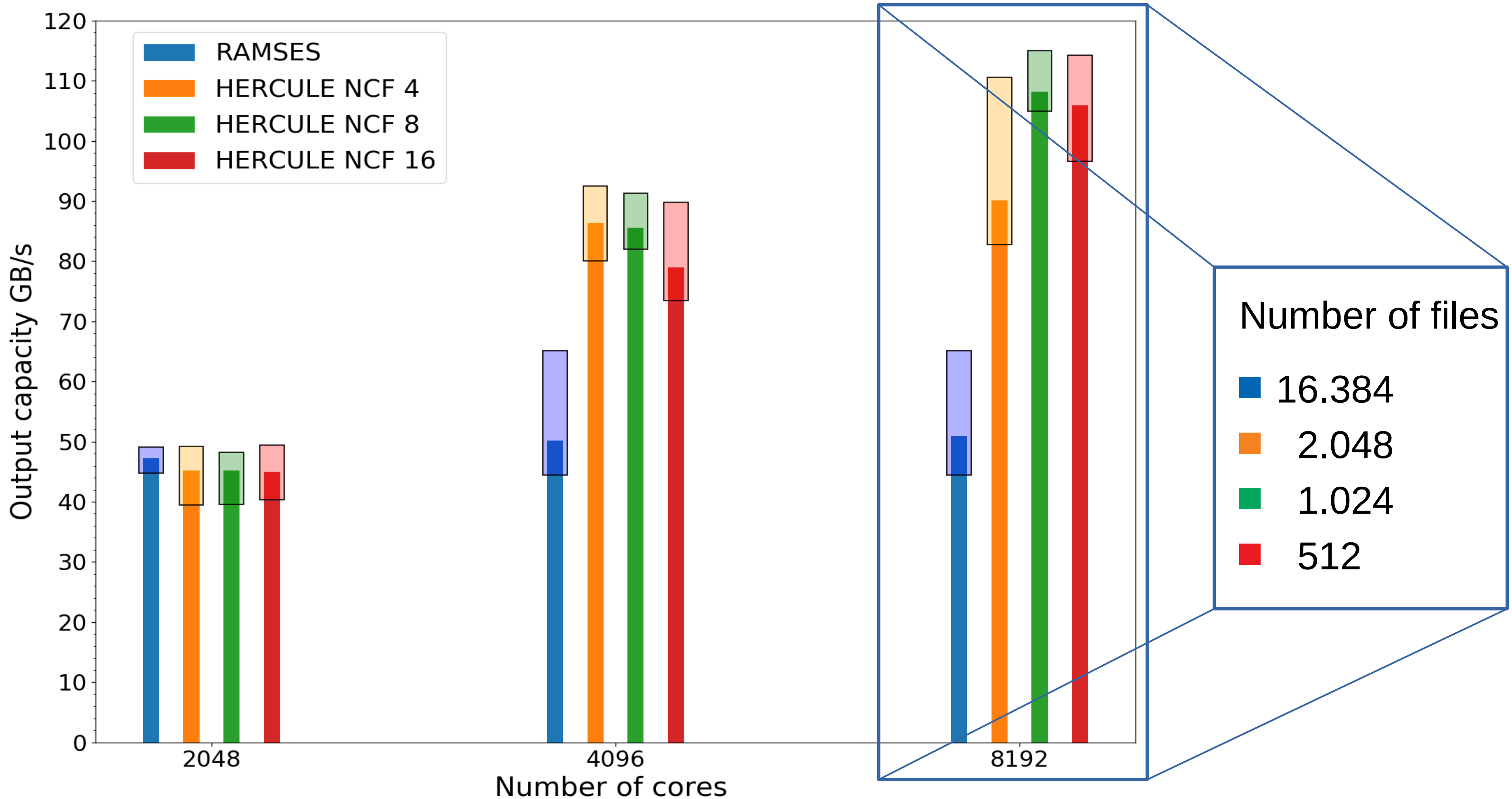
- **HDep**: High "dump" frequency, low volume
- **HProt**: Low "dump" frequency, high volume



Benchmarks Write: RAMSES + HERCULE

Benchmarks:

- **Checkpoint/restart** Sedov 3D use case – 2048^3 cells – I/O only (no time integration) – 500 Go
- **Hercule**: Number of Contributor Per File (« NCF »)

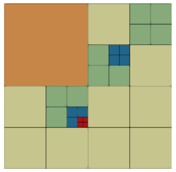


CEA TGCC - partition « scratch »

Part III

A new data model for TB-AMR

LightAMR data model



Why we need a new data model ?

- No standard data model for tree-based AMR
- RAMSES data model not specific for post-processing
- High data volume produced with other data model
  - « unstructured » not adapted for AMR data
- Need to keep the hierarchy of cells

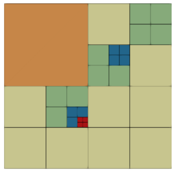
→ New LightAMR post-processing specific data model

How to make it a “standard” ? (Open science)

- Provide clear documentation (scientific publication, “readthedocs”, ...)
- Independent from the code that produce the data
- Self-consistent
- Example of usage / extraction of information (~~Optionnal~~)

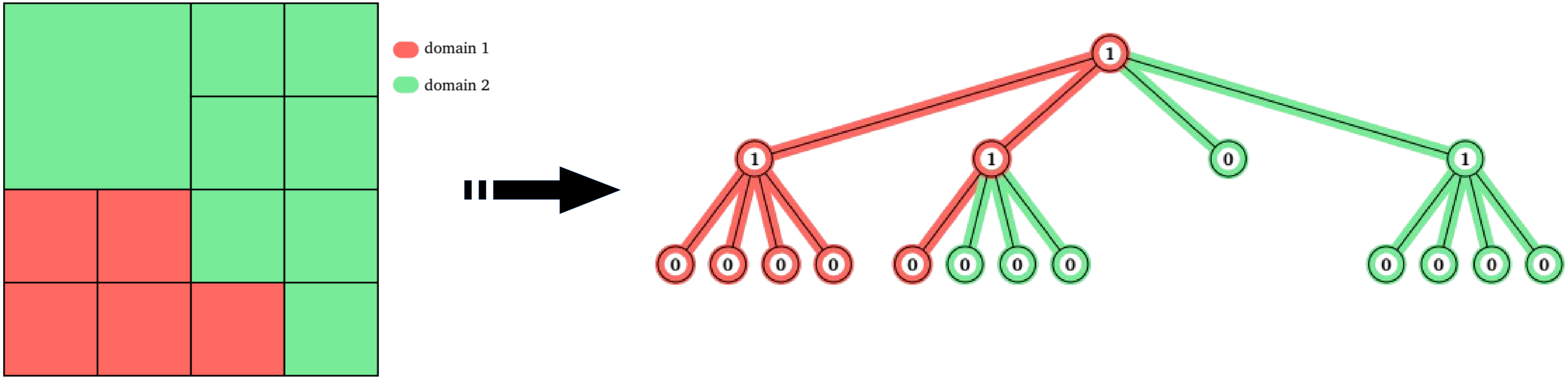
* *“LightAMR format standard and lossless compression algorithms for AMR: RAMSES use case”*, Strafella & Chapon, 2022

LightAMR data model



Requirements for the new data model:

- Compact representation of the mesh
  - Reduce the cost of storage or transfer
  - Keep implicit as much as info. as possible, e.g. : cell centers
- Keep the hierarchy of cells
  - Allow optimization in post-processing: level-of-details, region of interest, ...



LightAMR data model & additional algorithms

3 additional algorithms:

- **<u>Redundancy removal</u>** (« Tree pruning ») :
  - Remove ghost cells and nodes
  - Small computational and memory cost
  - **High impact on data volume:** 8 - 45 % (depending on use case)

- **<u>Mesh data compression</u>**:
  - Use of « run-length-encoding »
  - High data compression ratio
  - Mesh description negligible: **< 1 % of total volume**

- **<u>Floating point data compression</u>** (32 or 64 bits):
  - “delta” compression, “parent-child-predictor” (**PCP**)
  - Small memory cost, fast (~ Go/s)
  - ***Lossless*** *:* no loss of information (bit to bit), **low ratio**
  - ***Lossy*** *:* max. abs. Point-wise error + *centered around 0*, **high ratio**

LightAMR : data volume reduction

- Test dataset (real simulation) : FRIGG[1], ORION[2], Extreme-Horizon[3], Cosmic-Dawn III[4]

	RAMSES		LightAMR ($E_{rel} \leq$ 2e-3)		
	AMR	**HYDRO (db)**	**AMR***	**HYDRO (sp)****	**Gain**
ORION	9 Go	47 Go	6 Mo	2 Go	x28
FRIGG	30 Go	113 Go	10 Mo	8 Go	x18
Extreme-Horizon	500 Go	2330 Go	800 Mo	176 Go	x16
CoDa III	16 To	52 To	20 Mo	2,7 To	x25

- From production Exa-MilkyWay (RAMSES + LightAMR + Hercule)

	Protection/reprise	LightAMR ($E_{rel} \leq$ 2e-4)	Gain
Exa-MilkyWay	4,7 To	190 Go	x24

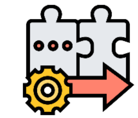
1. [P. Hennebelle. 2018], conversion to LightAMR
2. [E. Ntormousi and P. Hennebelle., 2019], conversion to LightAMR
3. [S. Chabanier et al., 2020], conversion to LightAMR
4. [L. Joseph et al., 2022], conversion to LightAMR

* lossless compression
** lossy compression

Part IV

Post-processing LightAMR data

Méthodologie de post-traitement



Data for analysis:

- Used data model: **LightAMR**
- Number of domains
  - ➔ 512 (**Orion**) up to 131.072 (**CosmicDawn III**)
- Number of cells
  - ➔ ~500 millions (**Orion**) to ~500 billions (**CosmicDawn III**)
- Data volume
  - ➔ From Gigabytes to Terabytes (**LightAMR** compressed or not)

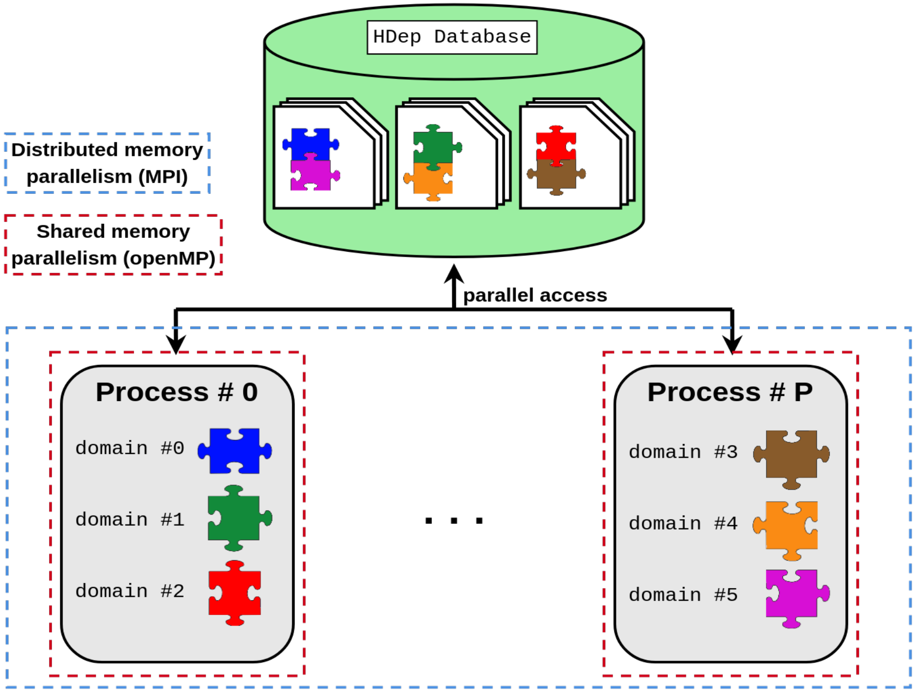
Requirements for a post-processing tool:

- Good I/O scalability
  - ➔ Compatible with parallel I/O library: Hercule, HDF5
  - ➔ Need of **MPI**
- Good computational scalability
- Compatible with multicores architectures
  - ➔ Shared memory parallelism: **OpenMP**

Strategy for post-processing

Take advantage of Hercule “per domain” data access :

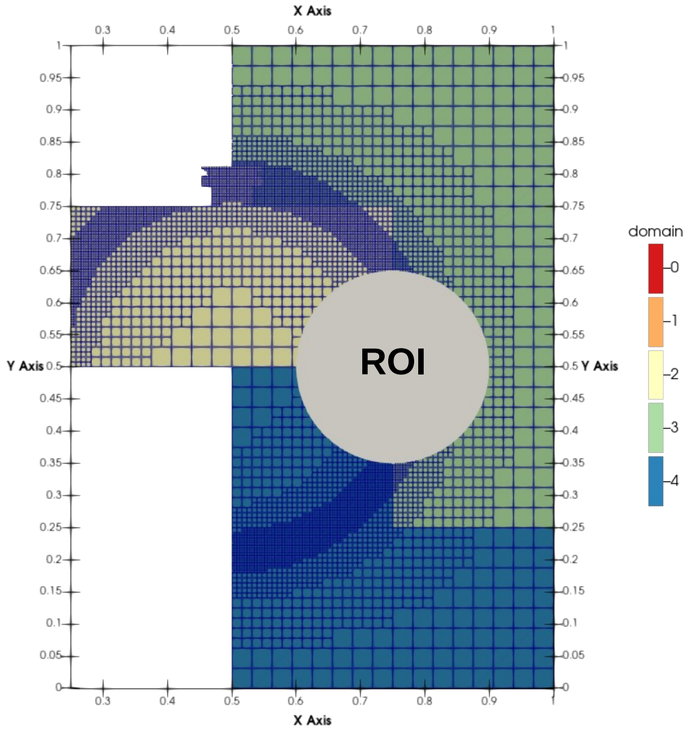
- Hybrid parallelism: distributed memory (**MPI**) **and** shared memory (**OpenMP**)
  - 1 MPI ➡ a list of domains
  - 1 openMP thread ➡ one domain

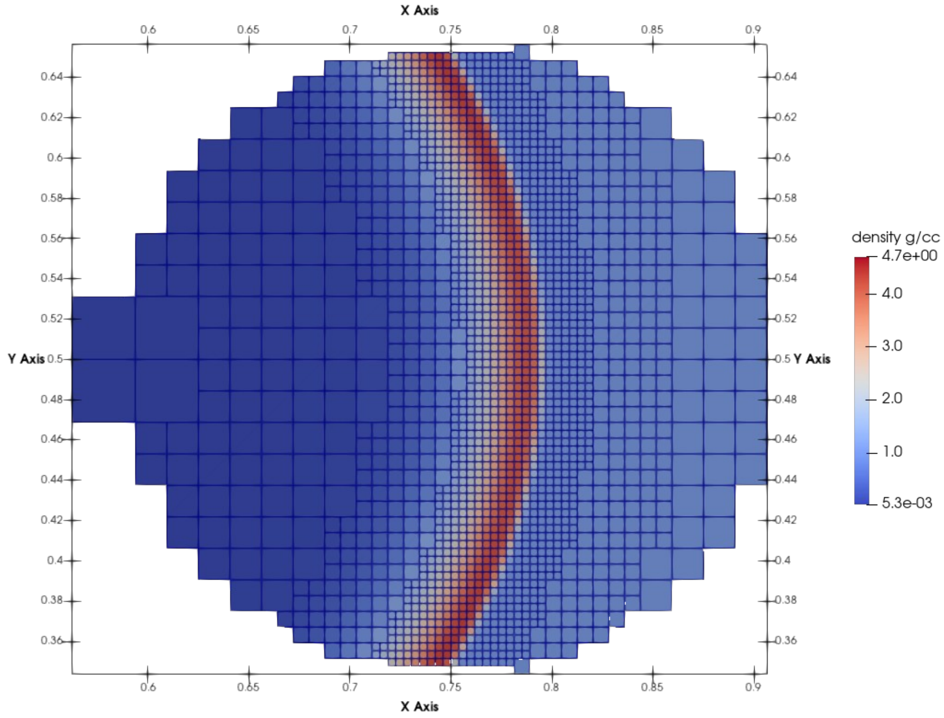


Strategy for post-processing

Different optimization strategy :

- *"Region of Interest"* **(ROI)**
  - Select domains of interest: **optimize I/O and memory**
  - Select cells to analyze: **optimize computation and memory**
- *"Level-of-details"* **(LOD)**





Conclusion

Strategy for post-processing

More work to do:

- More I/O benchmarks: checkpoints vs analysis, impact of compression, …
- Extension of LightAMR for other AMR strategy

Recommendation for codes with I/O bottleneck and/or high data volume

- Check the data model
- Use of data compression ?
- Double precision ? Single precision ?
- Field selection ?
- Which I/O library ?

Workshop EXA-DI **Block-structured AMR @Exascale**

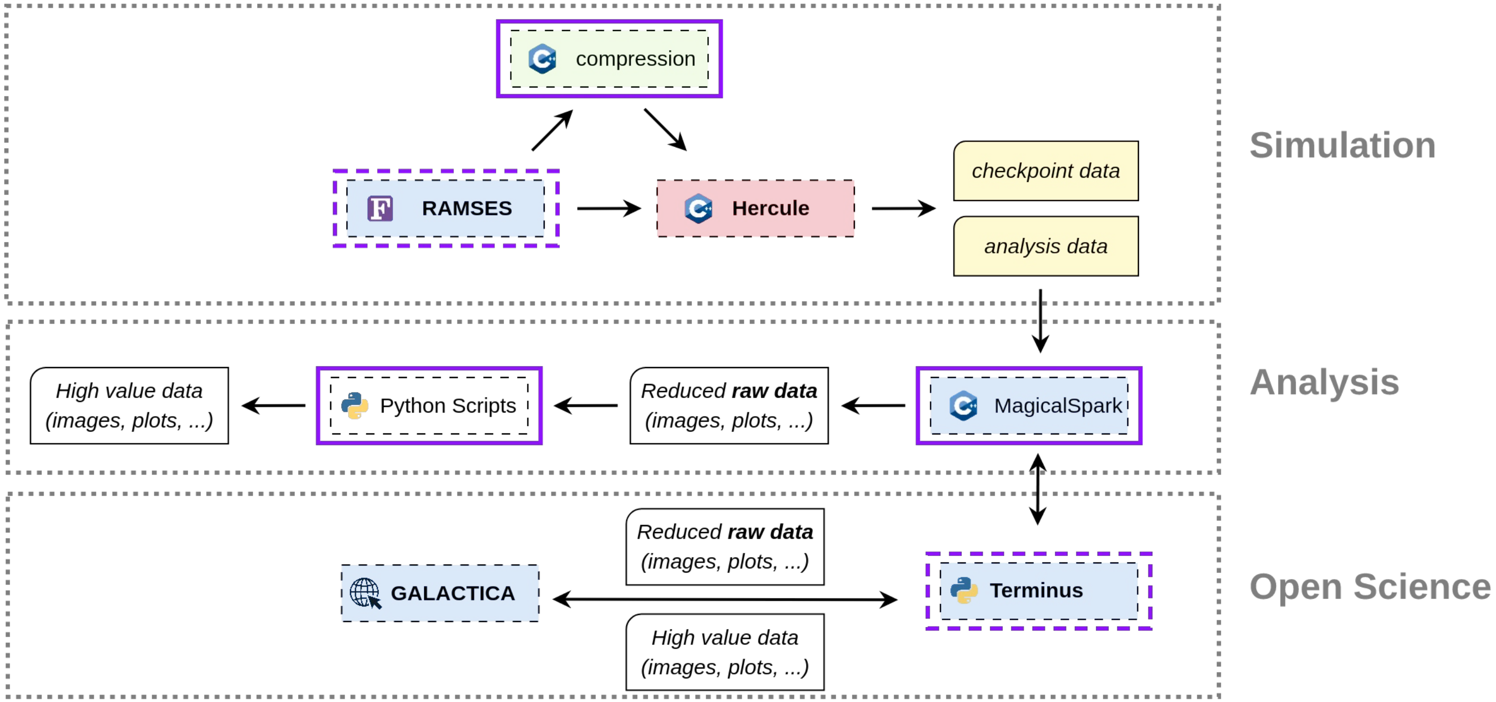
Questions ?

Loic Strafella

Data pipeline

Development and optimization of the data pipeline

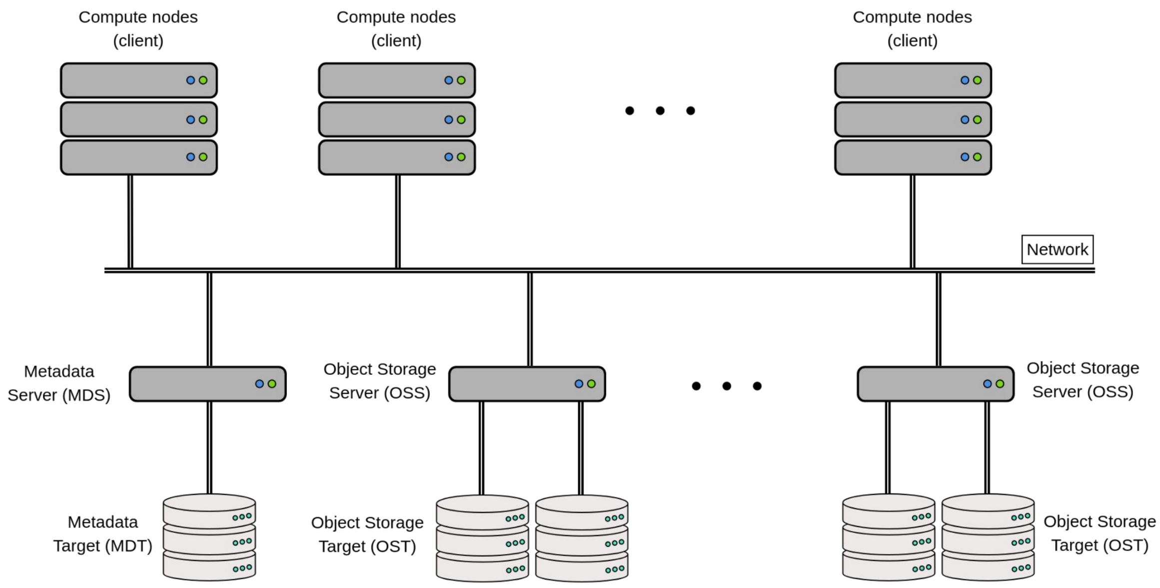
- **Parallel I/O**: Ramses + Hercule, HDF5 extension
- **Data model**: LightAMR
- **Data compression**: algorithms CPS52, PCP
- **HPDA** : post-processing tool, python interface



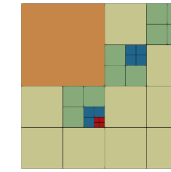
Lustre parallel filesytem

Schéma d'un système **Lustre**

- **Noeuds de calculs :** application
- **Serveur de méta-données :** information sur les dossiers et fichiers
- **Serveur de stockage :** stockage des données brutes
- **Réseaux**



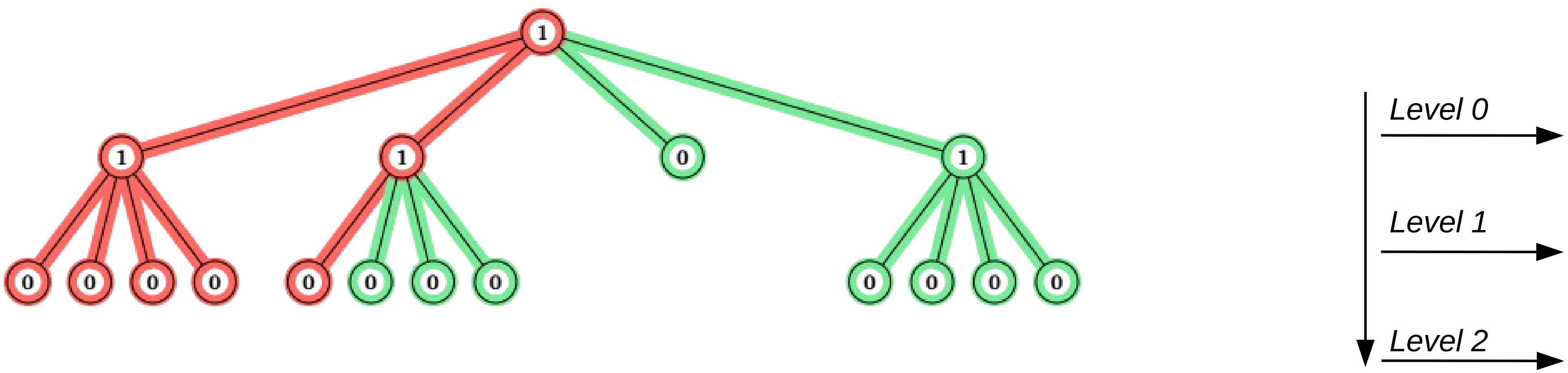
LightAMR data model



Mesh description:

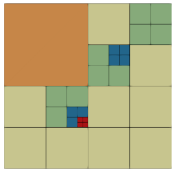
Refinement { **1** : refined (node) / **0** : not refined (leaf)

Ownership { **0** : owned / **1** : not owned



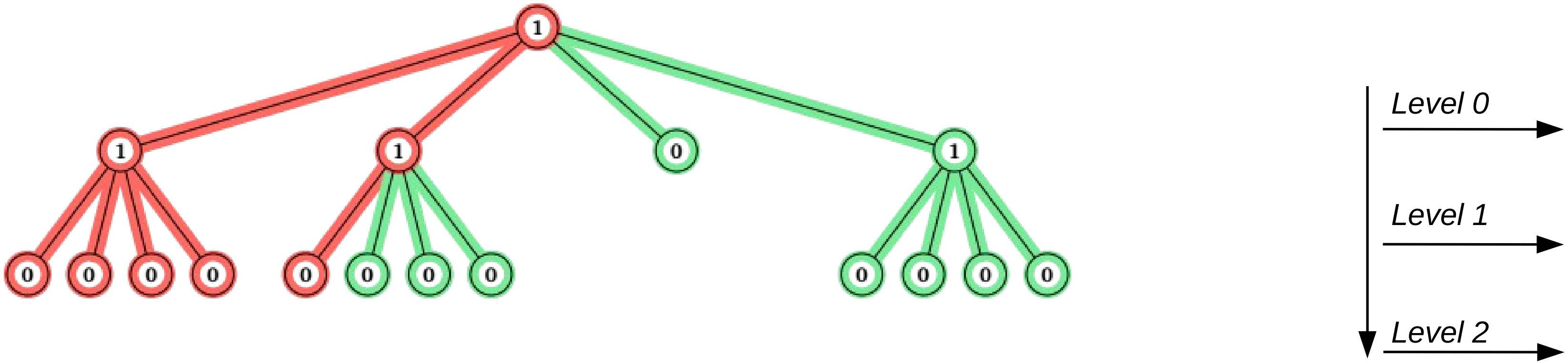
Grid refinement domain 1	1	1	1	0	1	0	0	0	0 .	0	0	0	0 .	0	0	0	0
Ownership mask domain 1	0	0	0	1	1	0	0	0	0 .	0	1	1	1 .	1	1	1	1
Grid refinement domain 2	1	1	1	0	1	0	0	0	0 .	0	0	0	0 .	0	0	0	0
Ownership mask domain 2	1	1	1	0	0	1	1	1	1 .	1	0	0	0 .	0	0	0	0

LightAMR data model



Cell values:

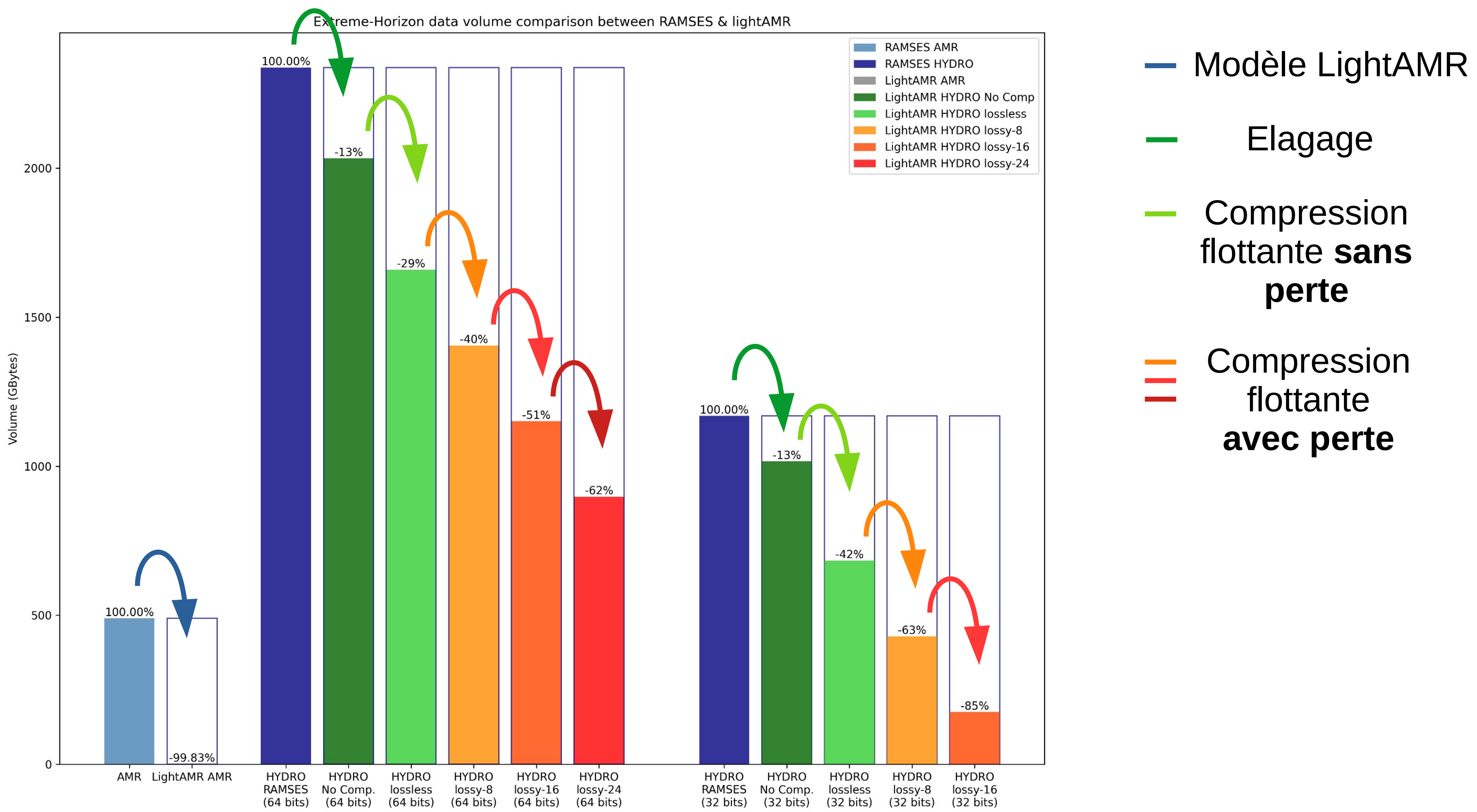
- **Scalar field**: same order as node/leaf description (1:1 « mapping »)
  - Expecting a value on each **node**
- **Vector field**: split in **n** scalar fields (optimization in post-processing)



Grid refinement domain 1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Ownership mask domain 1	0	0	0	1	1	0	0	0	0	0	1	1	1	1	1	1	1
Scalar field data domain 1	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	d_{12}	d_{13}	d_{14}	d_{15}	d_{16}

LightAMR: réduction de volume de données

Extreme-Horizon: 2.3 To **Hydro.** data + 490 Go of **AMR** (RAMSES)



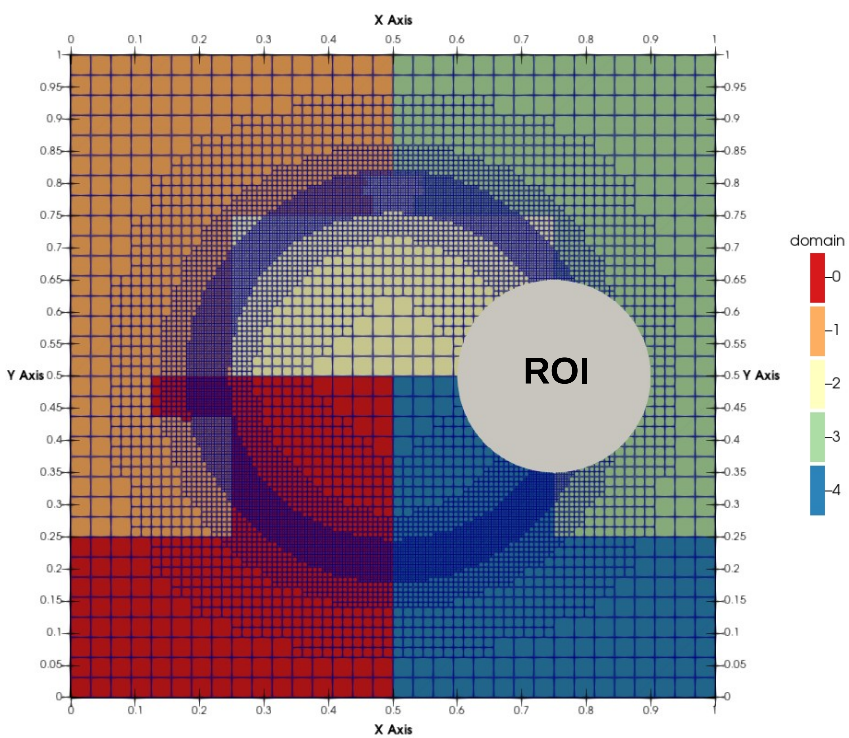
Méthodologie de post-traitement

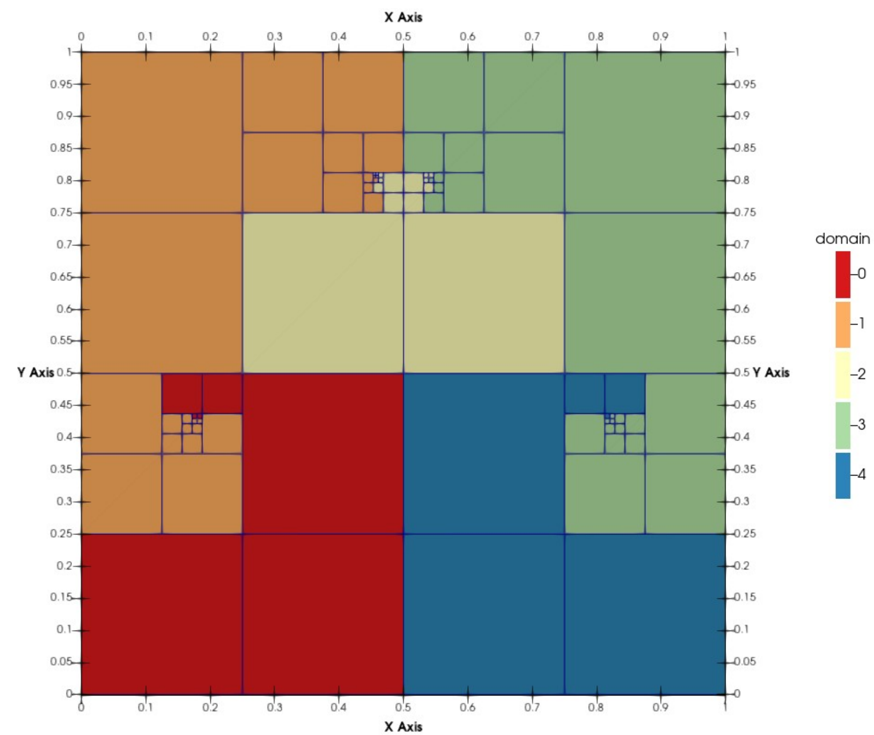
Utilisation de Région d'intérêt (**ROI**, *« Region of Interest »*)

- Sélection des domaines à traiter : **optimisation des E/S + mémoire**
- Sélection des cellules à traiter : **optimisation des calculs + mémoire**

Pavage Cubique Minimal

- Calculé à partir des méta-informations (clés de Hilbert) ou de la description du maillage
- Utilisé comme structure d'accélération pour le lancer de rayons

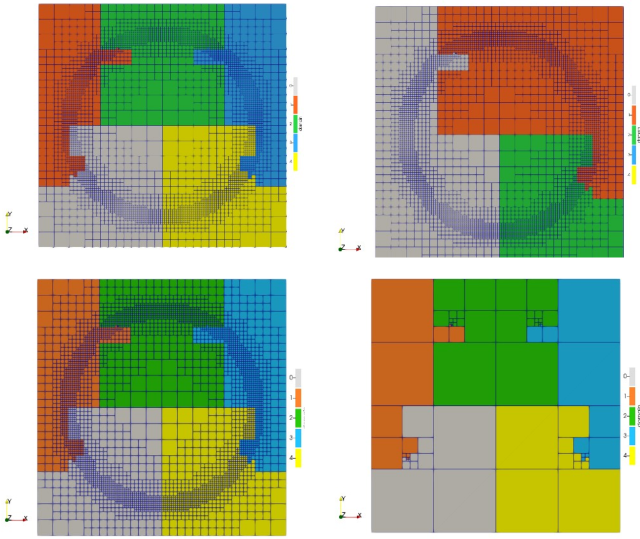




Fonctionnalité de l’outil d’analyse

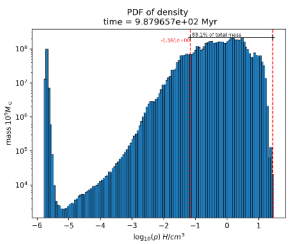
<u>Traitements et algorithmes disponibles</u> :

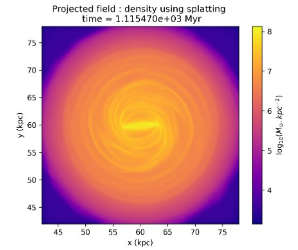
- Agglomération de domaines
- Conversion: vtkHyperTreeGrid, non structuré, ...
- Analyse par secteurs, calcul de grandeurs dérivées, particules, ...
- Production de carte 2D (flou gaussien, « raytracing »)
- Calcul d’histogramme 1D, 2D, …
- « Batch processing » en cas de ressource mémoire faible



<u>Quelques chiffres</u>:

- 1 Carte 2D par lancer de rayons (100 Mpixels) : **~ 30 secondes**
  - 72 MPI + 40 threads / MPI : 2880 coeurs
  - ~ 30 milliards de cellules



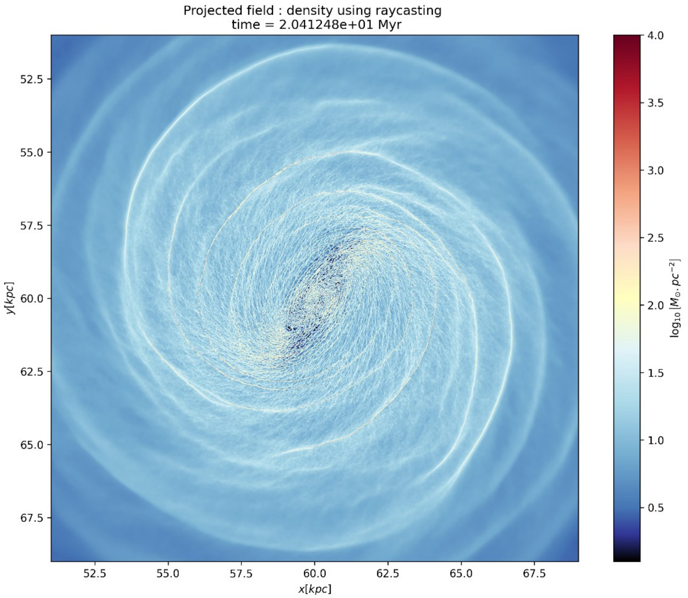


→ Stratégie efficace avec les données testées:
- Scalable et “memory friendly”
- Tire profit des E/S parallèle (Hercule)
- Dédié au **LightAMR**

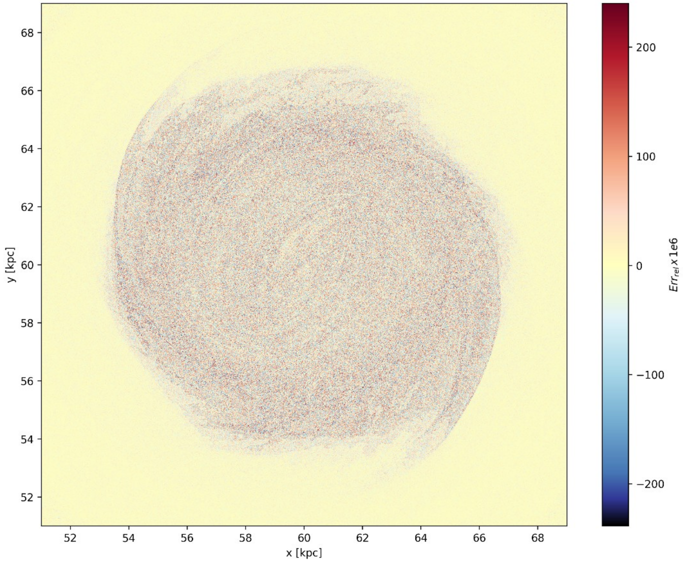
Quel Impact de la compression avec perte ?

Estimate the impact on results is essential !, e.g. climate data [Baker et al, 2016]

→ Define a data compression strategy adapted to the needs (DMP)



Relative Error x 10^6



→ No impact on analysis or scientific results

2D Map, power spectra, histogram, etc.