Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

The OpenGADGET3 code for cosmological simulations

- An update in preparation of the Key Science Projects -

Milena Valentini

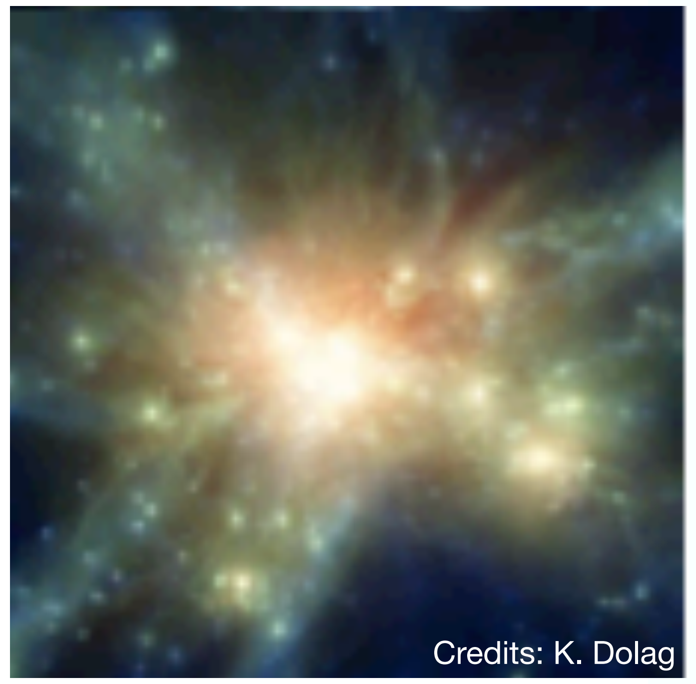
1924 2024
UNIVERSITÀ DEGLI STUDI DI TRIESTE
Dipartimento di **Fisica**
Dipartimento d'Eccellenza 2023-2027

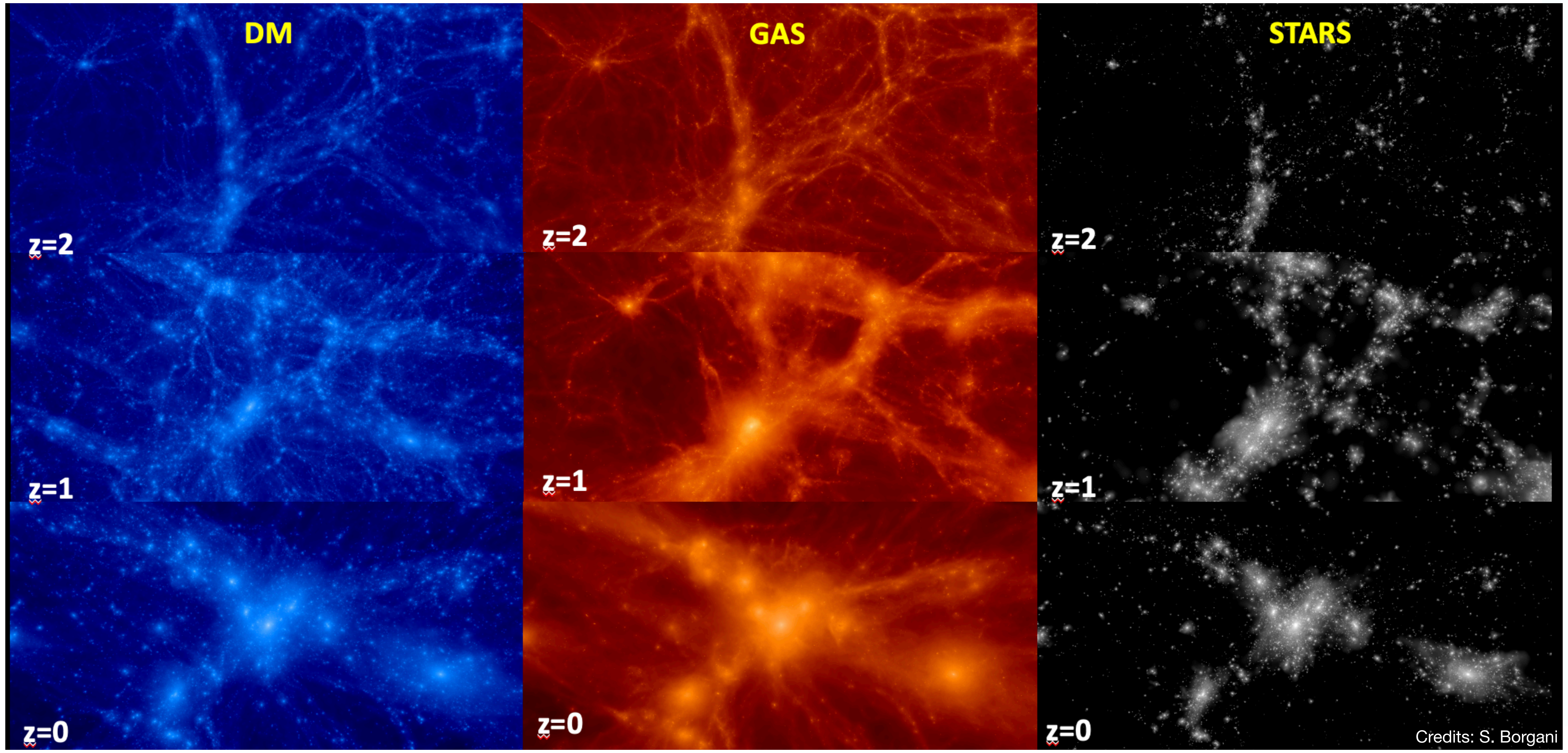
Spoke 3 Technical Workshop, Bologna Dec 17-19, 2024

The *Open GADGET3* code: a state-of-the-art code for HPC

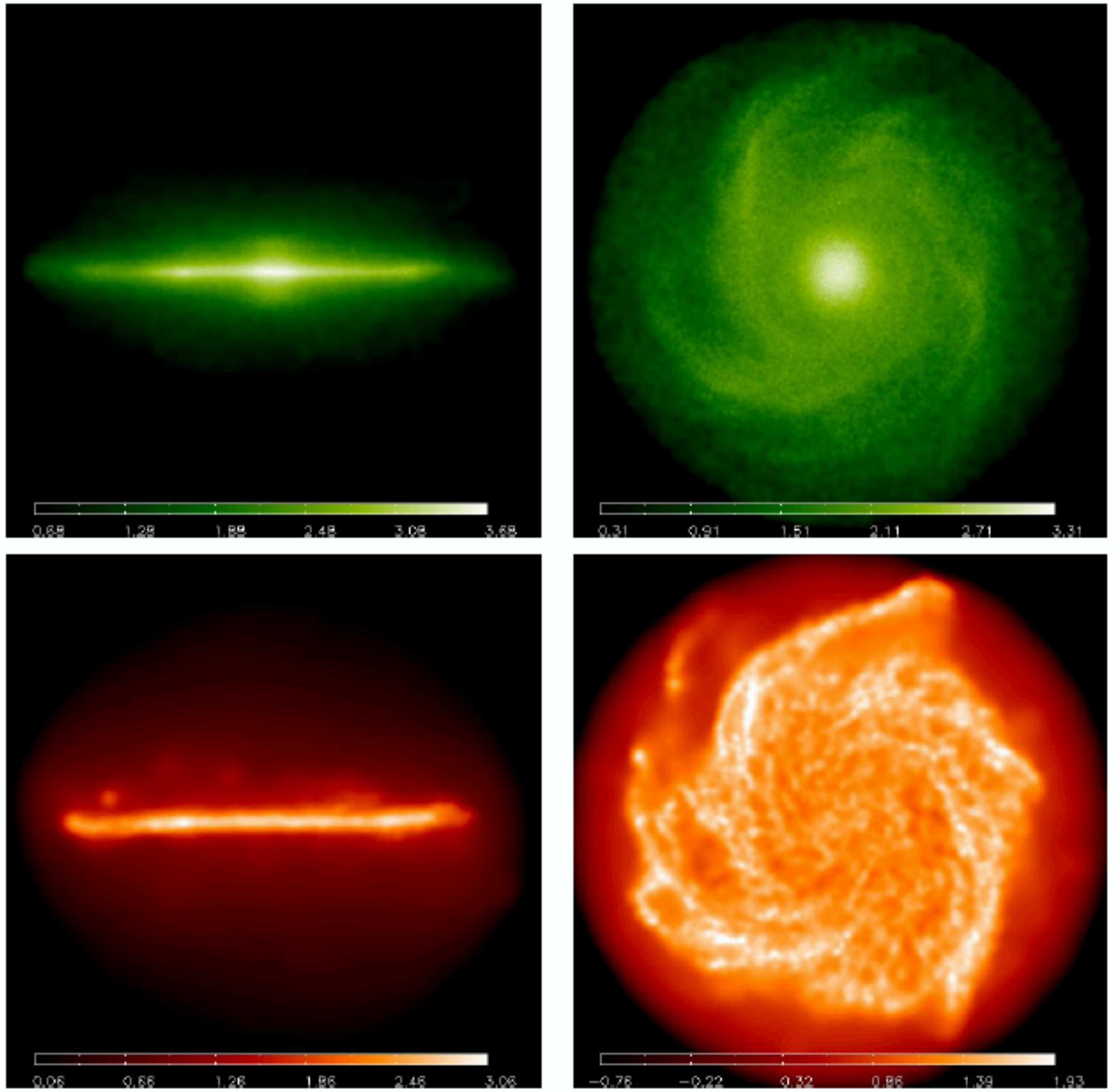
Scientific rationale

- Numerical cosmology
- Structure formation and evolution


Credits: K. Dolag



Credits: S. Borgani



Technical Objectives, Methodologies and Solutions

USM LMU

UNIVERSITÀ DEGLI STUDI DI TRIESTE

The OpenGadget3 code

- **TreePM+SPH code**
- **Highly optimised code:** MPI parallelised + OpenMP
- **Two hydro solvers:** improved SPH formalism or MFM
- **Two sub-grid models** (Muppi, and one based on Springel&Hernquist 2003)
- **Several modules for sub-resolution physics:** star formation, stellar feedback, BH accretion and feedback, chemical enrichment, dust evolution, magnetic fields, cosmic rays
- **Runs on CPUs and GPUs**

MUPPI sub-resolution model

star formation

- description of a multi-phase ISM with H_2-based star formation
- thermal, kinetic, and low-metallicity stellar feedback
- improved cooling table interpolation
- stellar evolution and chemical enrichment

BH

- angular-momentum-dependent gas accretion, dynamical friction, spin evolution
- isotropic, thermal AGN feedback + mechanical AGN feedback

dust

- formation and evolution of dust, and dust-assisted cooling

Main tasks within the WP 2 of Spoke 3

Develop Open-GADGET further:

- **including additional physics modules**
- **enhancing code modularity and readability**
- **improving code performance**

Core teams in Trieste and Munich

Technical Objectives, Methodologies and Solutions

USM LMU

The OpenGadget3 code

- **TreePM+SPH code**
- **Highly optimised code:** MPI parallelised + OpenMP
- **Two hydro solvers:** improved SPH formalism or MFM
- **Two sub-grid models** (Muppi, and one based on Springel&Hernquist 2003)
- **Several modules for sub-resolution physics:** star formation, stellar feedback, BH accretion and feedback, chemical enrichment, dust evolution, magnetic fields, cosmic rays
- **Runs on CPUs and GPUs**

Main tasks within the WP 2 of Spoke 3

Develop Open-GADGET further:

- **including additional physics modules**
- **enhancing code modularity and readability**
- **improving code performance**

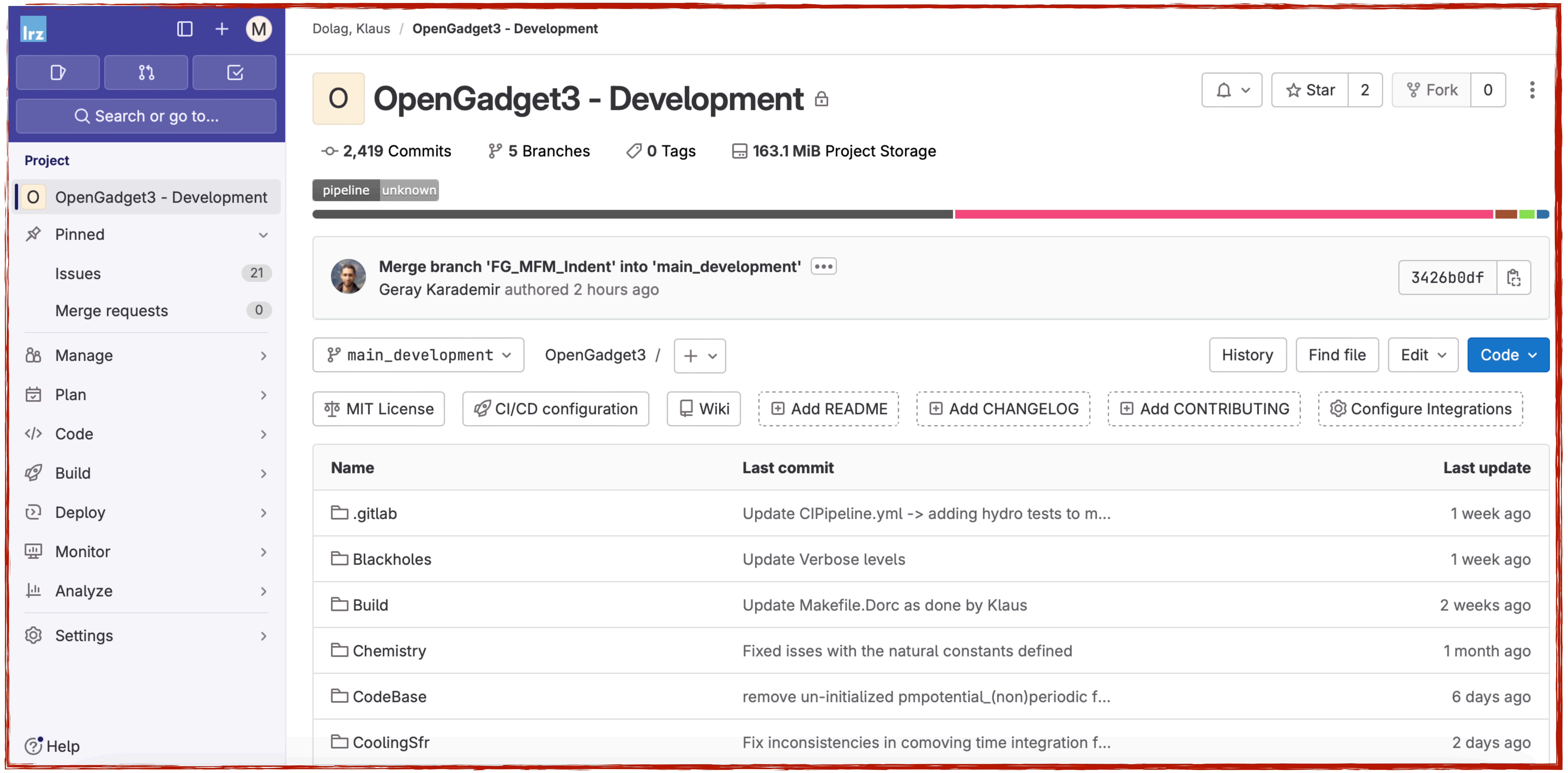
Core teams in Trieste and Munich

Core team in Trieste: S. Borgani, L. Tornatore, G. Murante, M. Valentini, T. Castro, P. Monaco, G. Taffoni, A. Damiano, G. Granato, D. Goz, P. Barai, M. Gitton-R., A. Saro, M. Viel

and collaboration in Munich led by K. Dolag

Accomplished Work, Results

- We moved our code to GitLab
- We defined a more accurate working strategy
- Quite large (> 30 people from different institutes) user community



Accomplished Work, Results

- The OpenGADGET3 project aims at making the use of the many complex physics modules more user friendly.
- Substantial effort in **cleaning** and making **more transparent** the definition of the **code configurations** and of the files setting the many parameters.
- Construction of a **reference structure for the files which configure** several **reference production runs and files of parameters** for the OpenGADGET3 code.

Name	Last modified	Size	Description
Parent Directory		-	
Config_MV_Muppi.sh	2024-03-28 17:27	9.9K	
Config_MV_SH03.sh	2024-03-28 17:28	7.5K	
Files_aux/	2024-02-13 13:08	-	
OpenGadget3.tar	2024-02-02 15:24	14M	
OpenGadget3/	2024-02-02 15:23	-	
dfrogin_25x_Muppi/	2024-02-12 12:08	-	
dfrogin_25x_SH03/	2024-02-12 12:08	-	
dfrogin_250x_Muppi/	2024-02-12 12:08	-	
dfrogin_250x_SH03/	2024-02-12 12:08	-	
dianoga_g15_25x_Muppi/	2024-02-12 12:09	-	
dianoga_g15_25x_SH03/	2024-02-12 12:10	-	
magneticum_box4_25x_Muppi/	2024-02-12 11:57	-	
magneticum_box4_25x_SH03/	2024-02-12 12:09	-	

Accomplished Work, Results

- The OpenGADGET3 project aims at making the use of the many complex physics modules more user friendly.
- Substantial effort in **cleaning** and making **more transparent** the definition of the **code configurations** and of the files setting the many parameters.
- Construction of a **reference structure for the files which configure** several **reference production runs and files of parameters** for the OpenGADGET code.
- **Bug fixing** and tackling subtleties of the sub-grid modelling.

- **Re-structuring** of the code (**modularity**)
- **Cleaning** the code **and documenting** its status

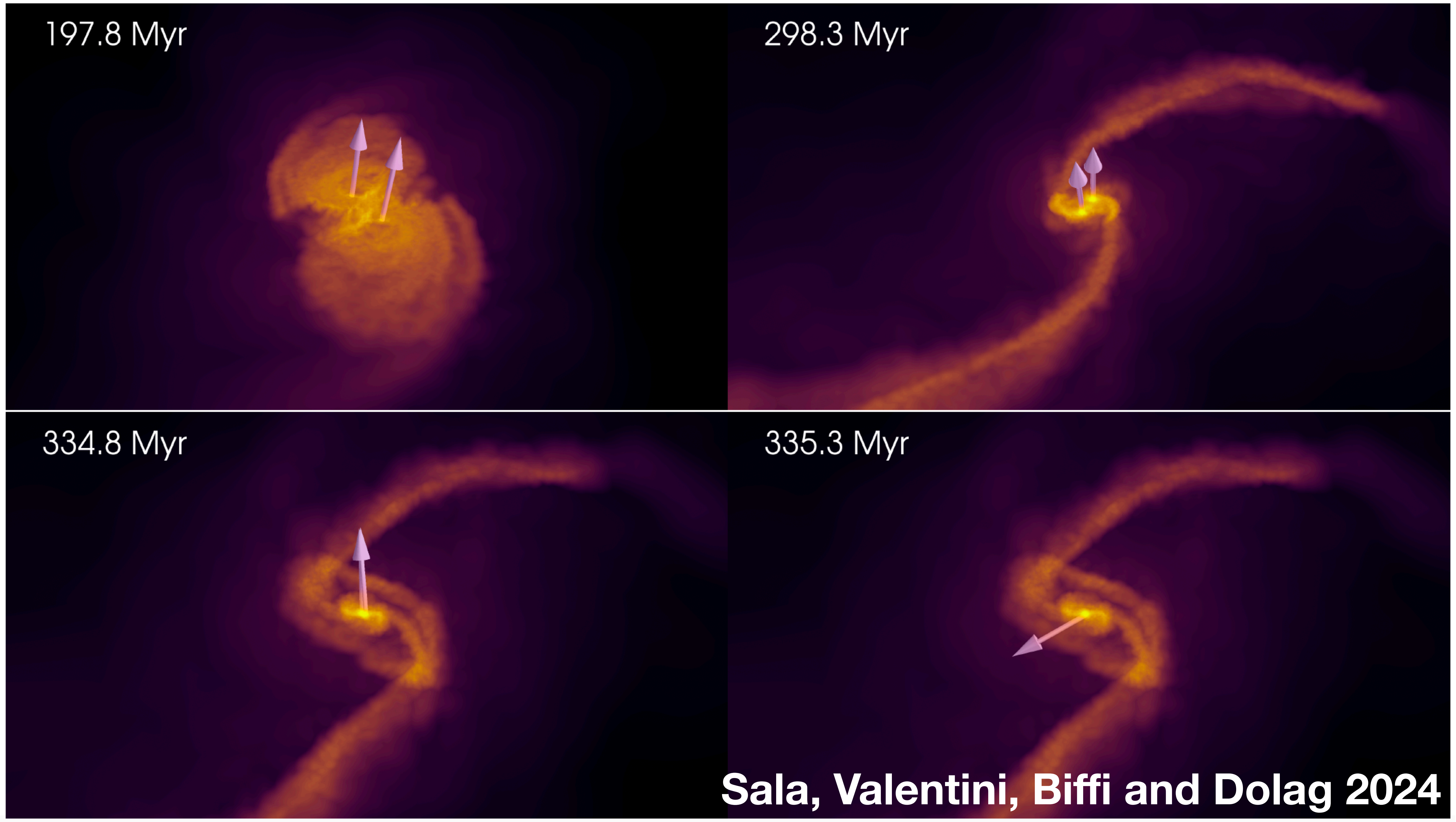
Accomplished Work, Results

Sub-resolution accretion disc links BH accretion on resolved scales to BH spin evolution

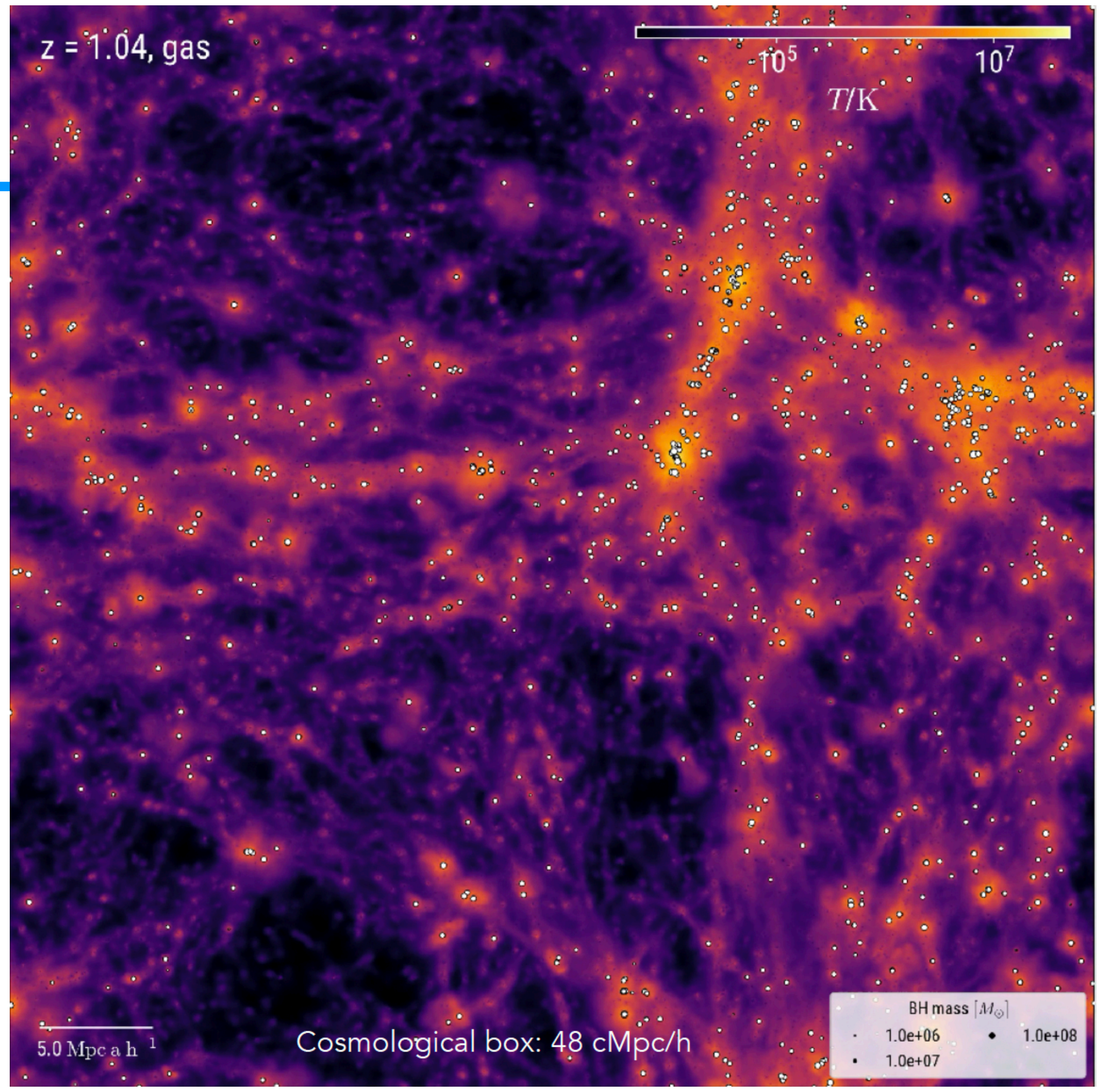
BH accretion rate → disc mass → accretion episodes

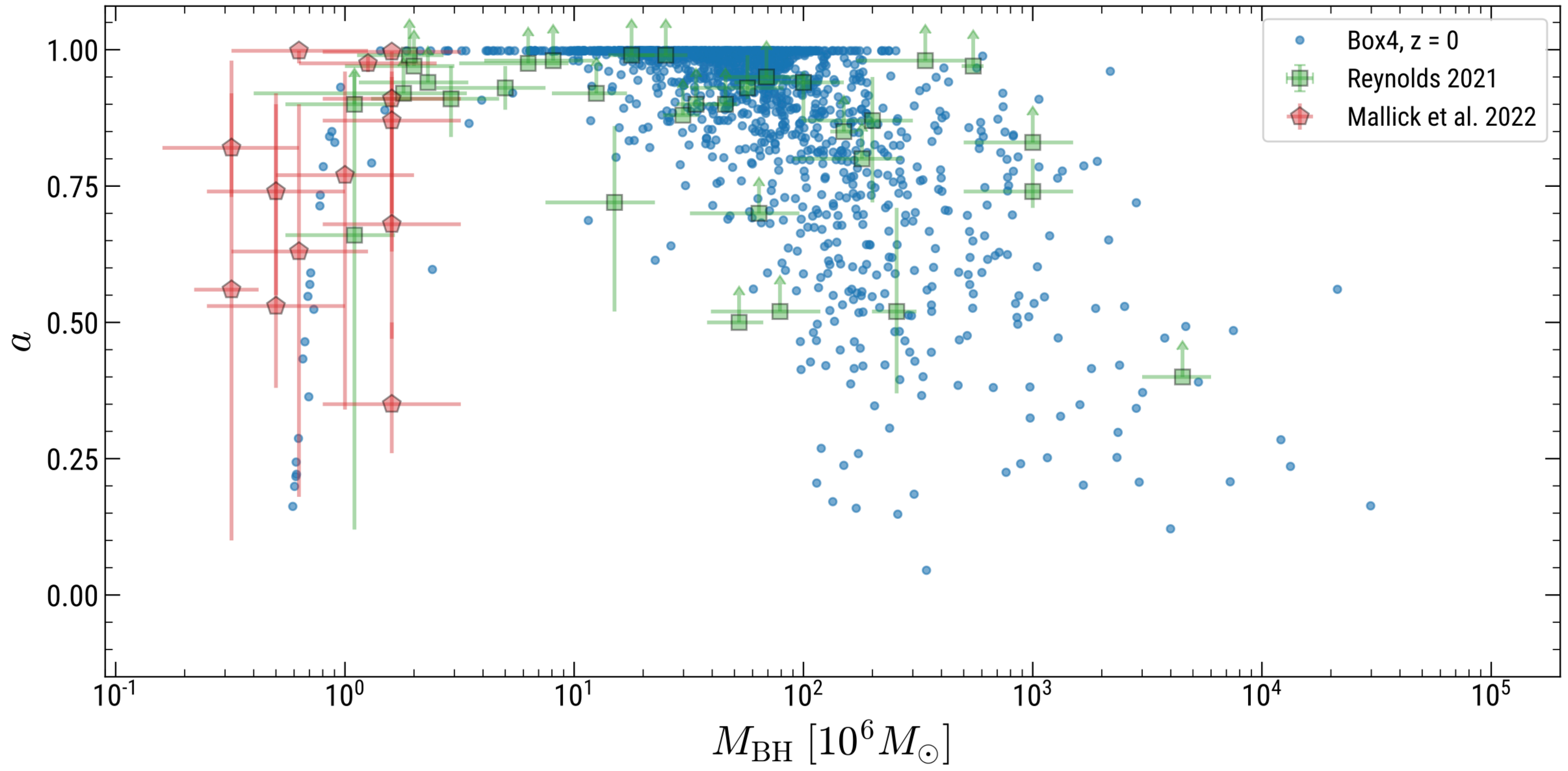
BH spin direction and magnitude defined by angular momentum of accreting gas

Given initial BH spin and mass → post-merger BH spin vector



BH radiative efficiency dependent on BH spin

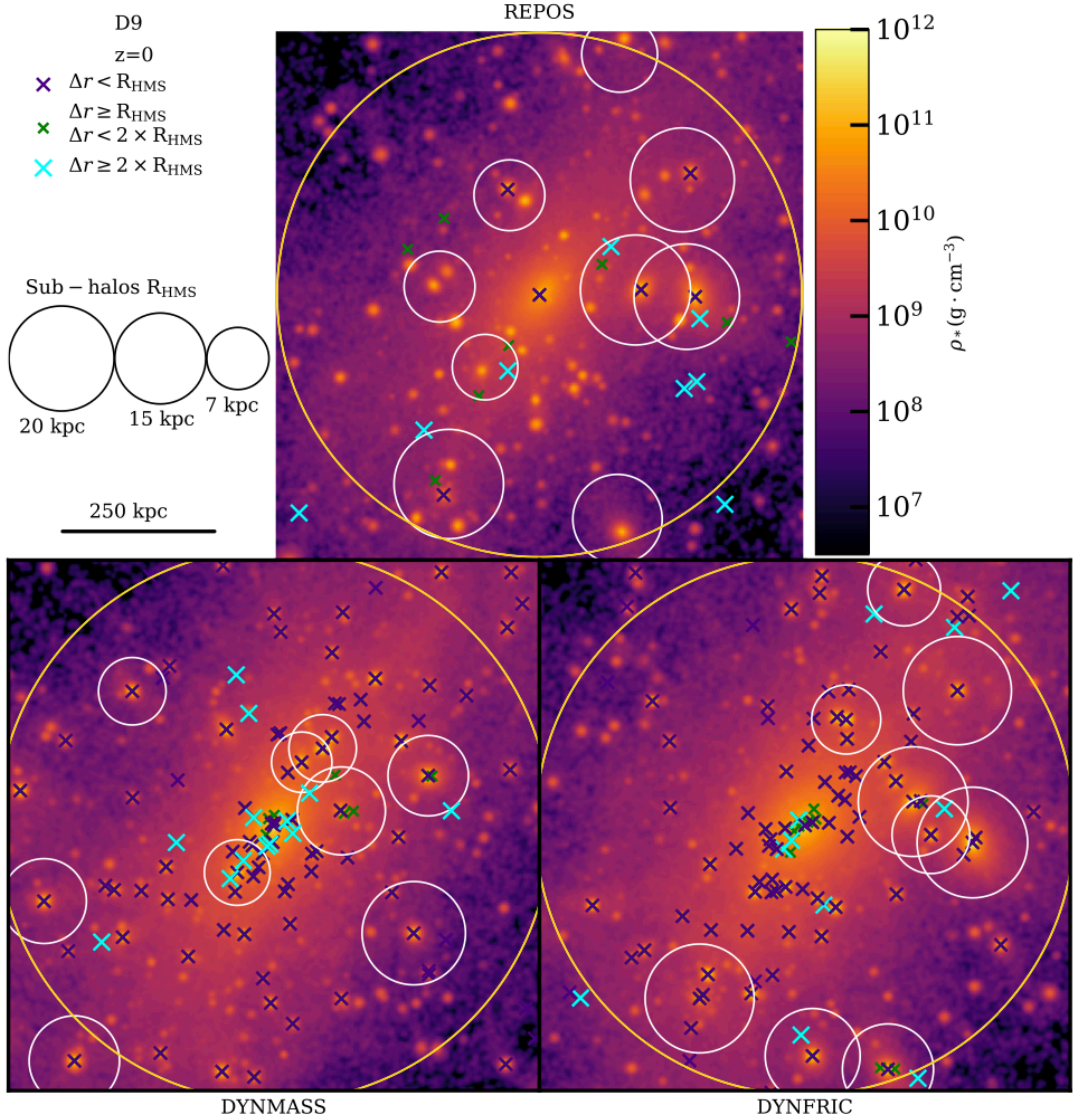


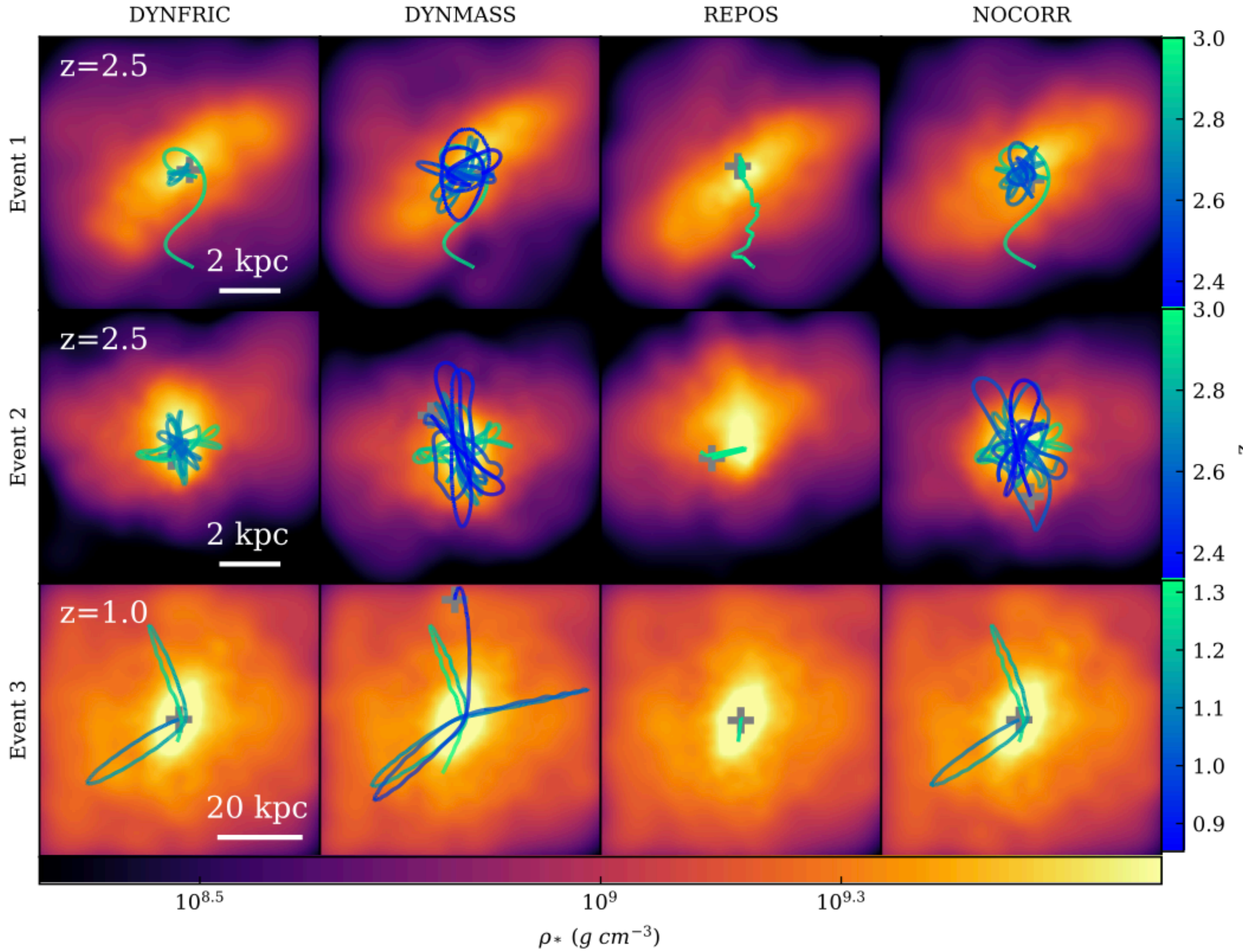


Accomplished Work, Results

Damiano, Valentini, Borgani, Tornatore+ 2024

Adopting different numerical prescriptions
for BH re-positioning has an impact
on **BH dynamics**, AGN feedback, BH-BH mergers





Ongoing: Assessing scalability, targeting performance issues

1. **GPU scalability**

   OpenGadget has most of the modules running on GPUs (thanks to A. Ragagnin).

   We are assessing in detail the scalability of this implementation in order to highlight the blocking factors, mitigate their impact or turn to new strategies with greater parallelism

2. **Performance issues**

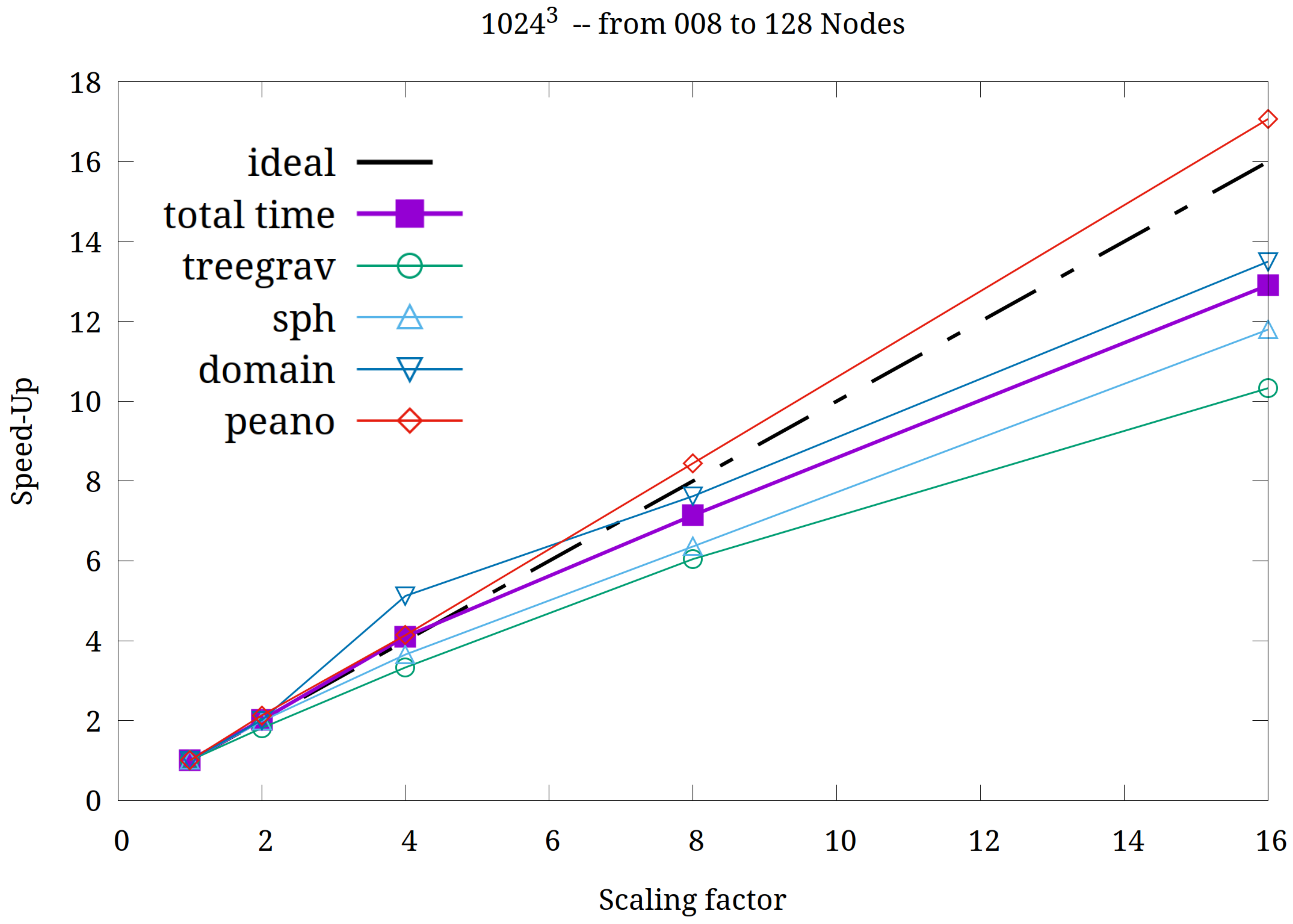
   Detailed profiling with the assistance of POP and SPACE Centers of Excellence

Coordinator of the work: L. Tornatore

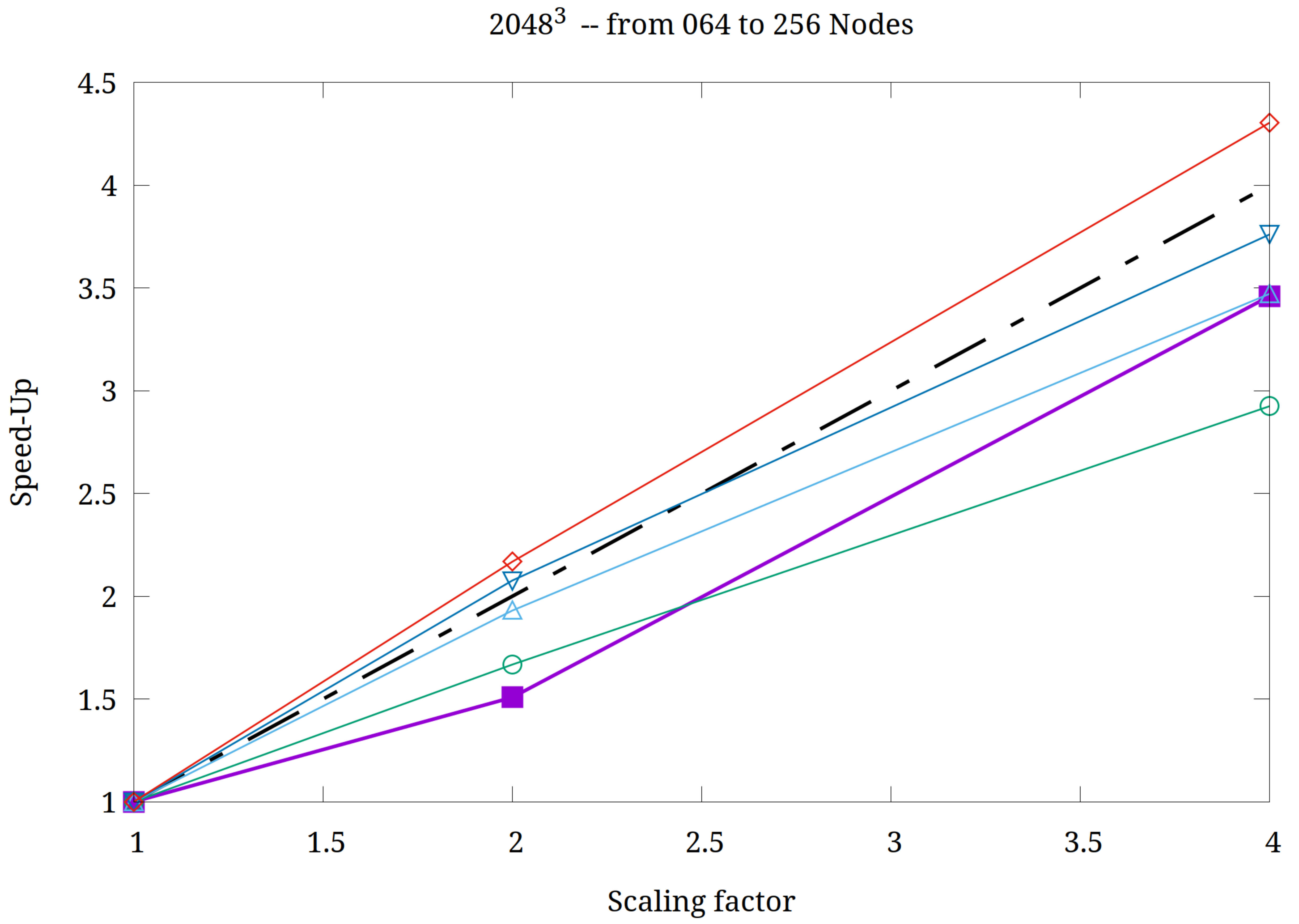
Ongoing: Assessing scalability, targeting performance issues

1) GPU scalability: Speed-Up

2×1024^3, 120 Mpc, up to **512 GPUs**



2×2048^3, 240 Mpc, up to **1024 GPUs**

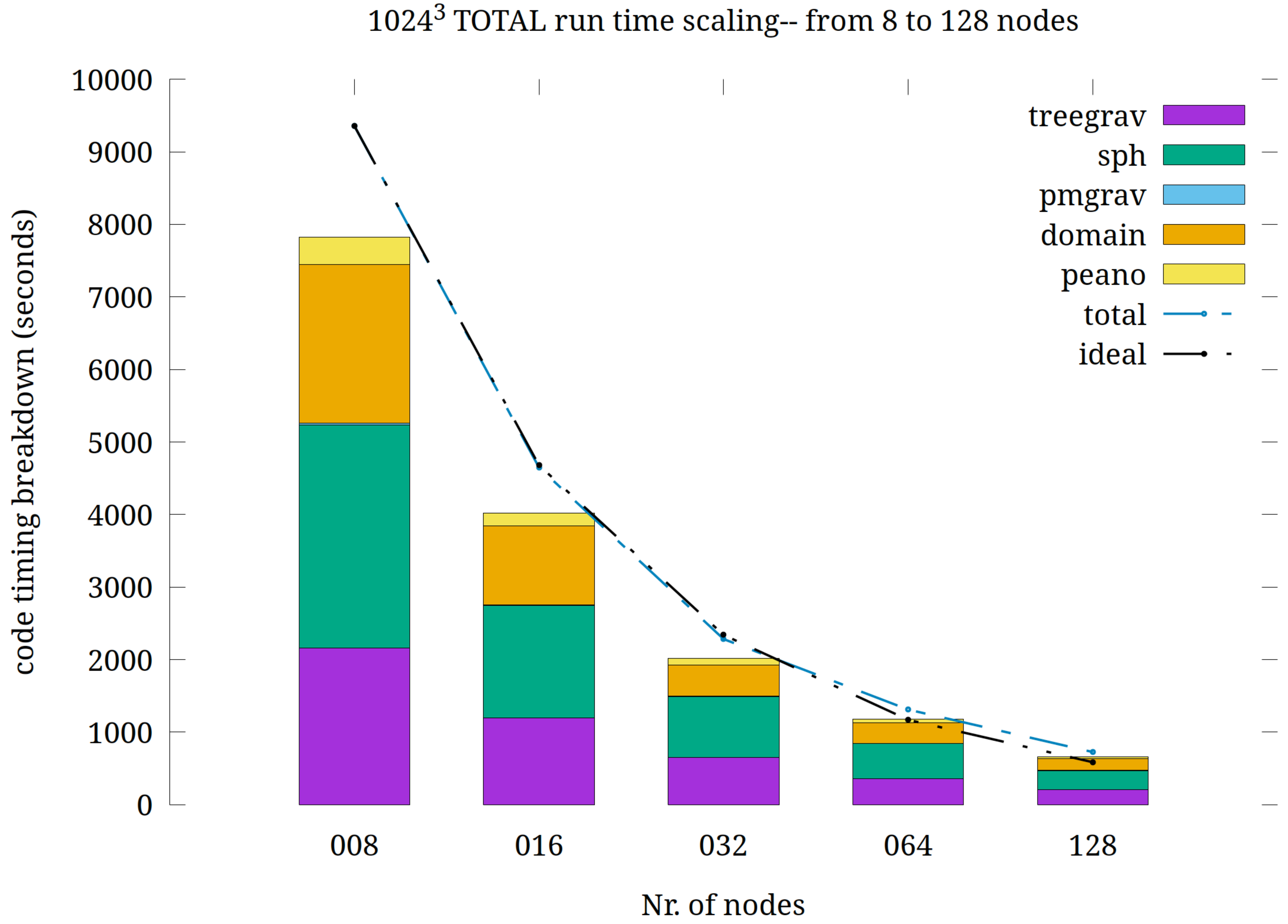


Running a suite of tests, we are assessing in detail the scalability, from 4 nodes up to the entire Leonardo

Ongoing: Assessing scalability, targeting performance issues

1) GPU scalability: more in detail

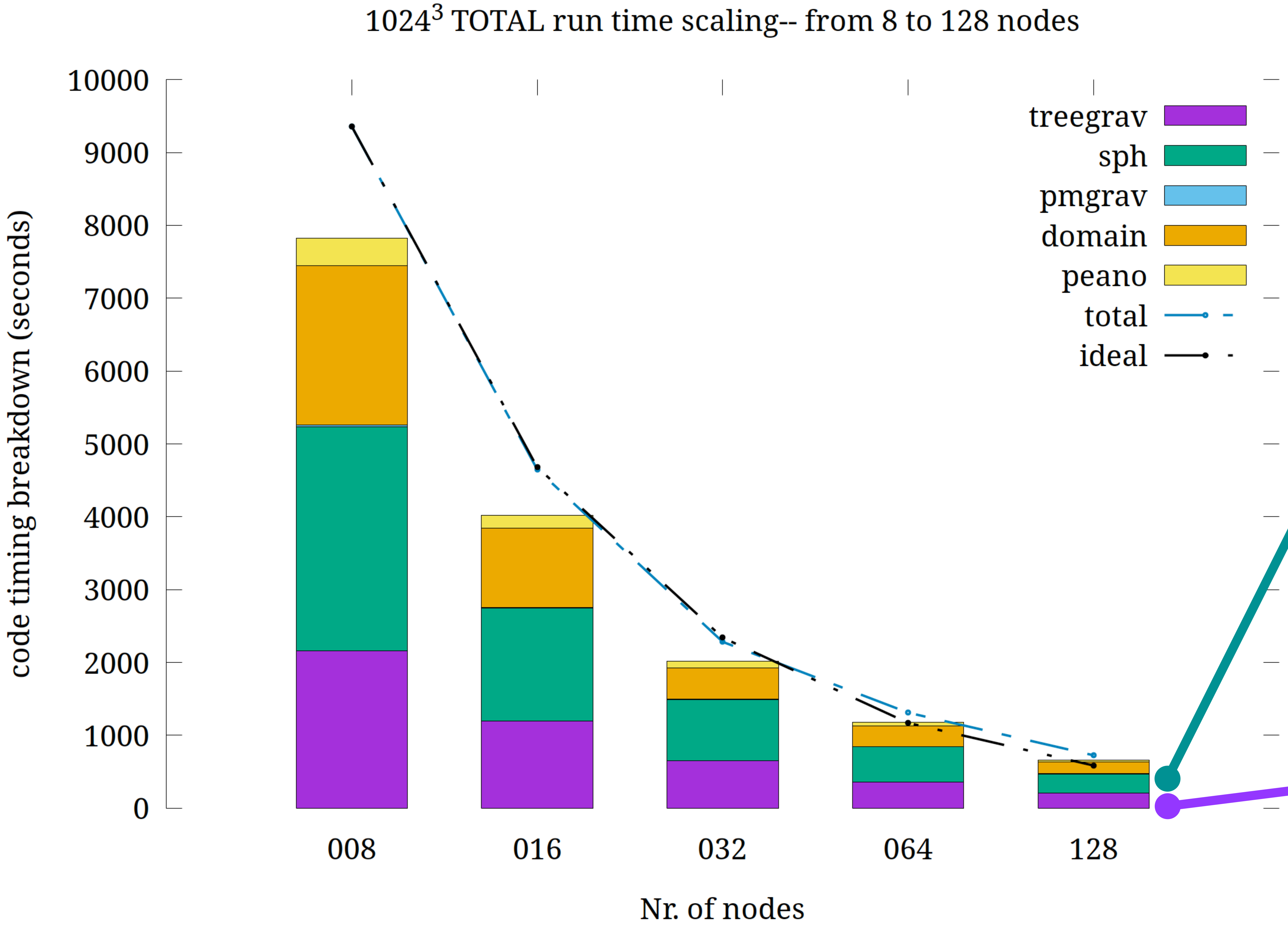
2×1024³, 120 Mpc, up to **512 GPUs**

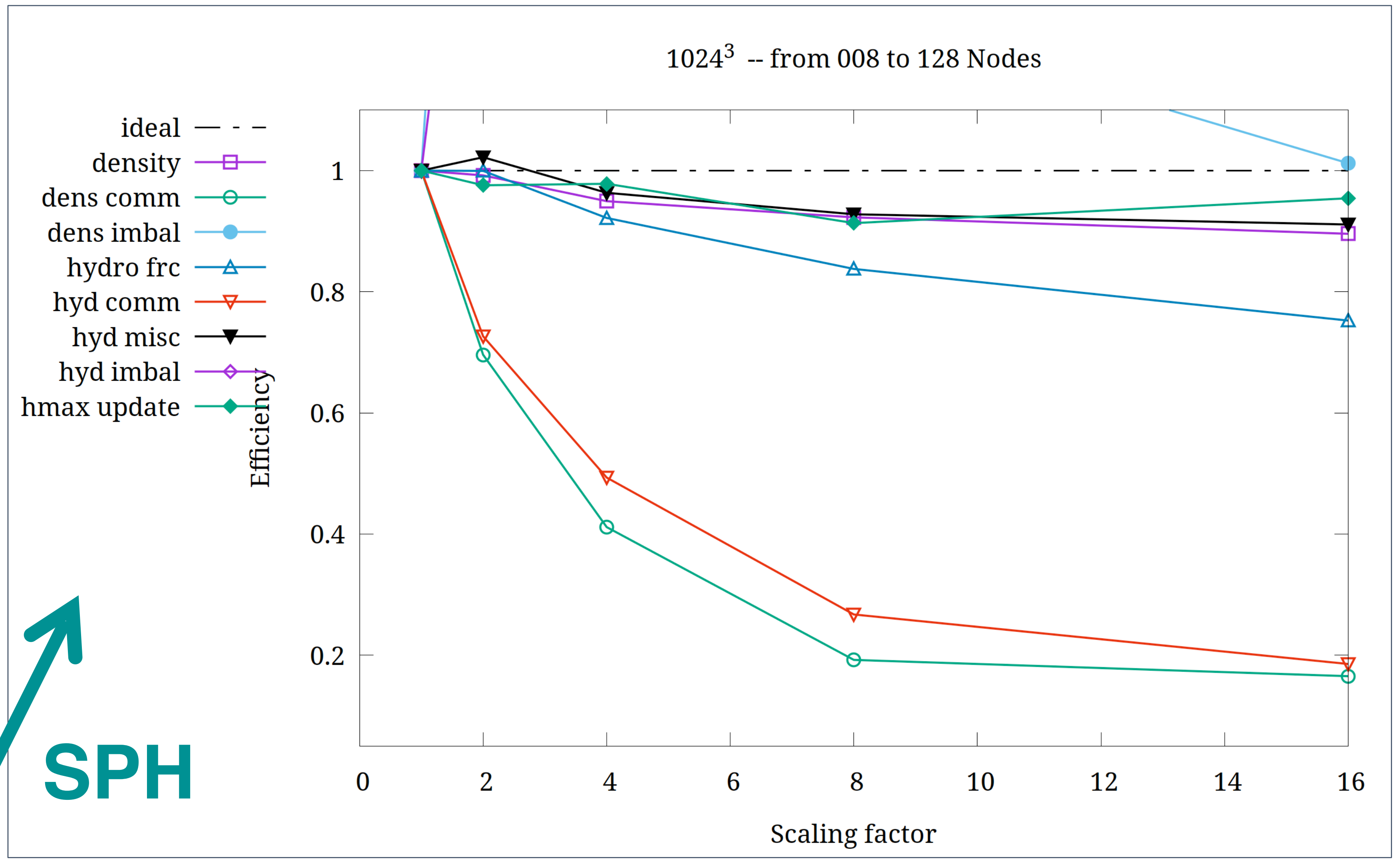


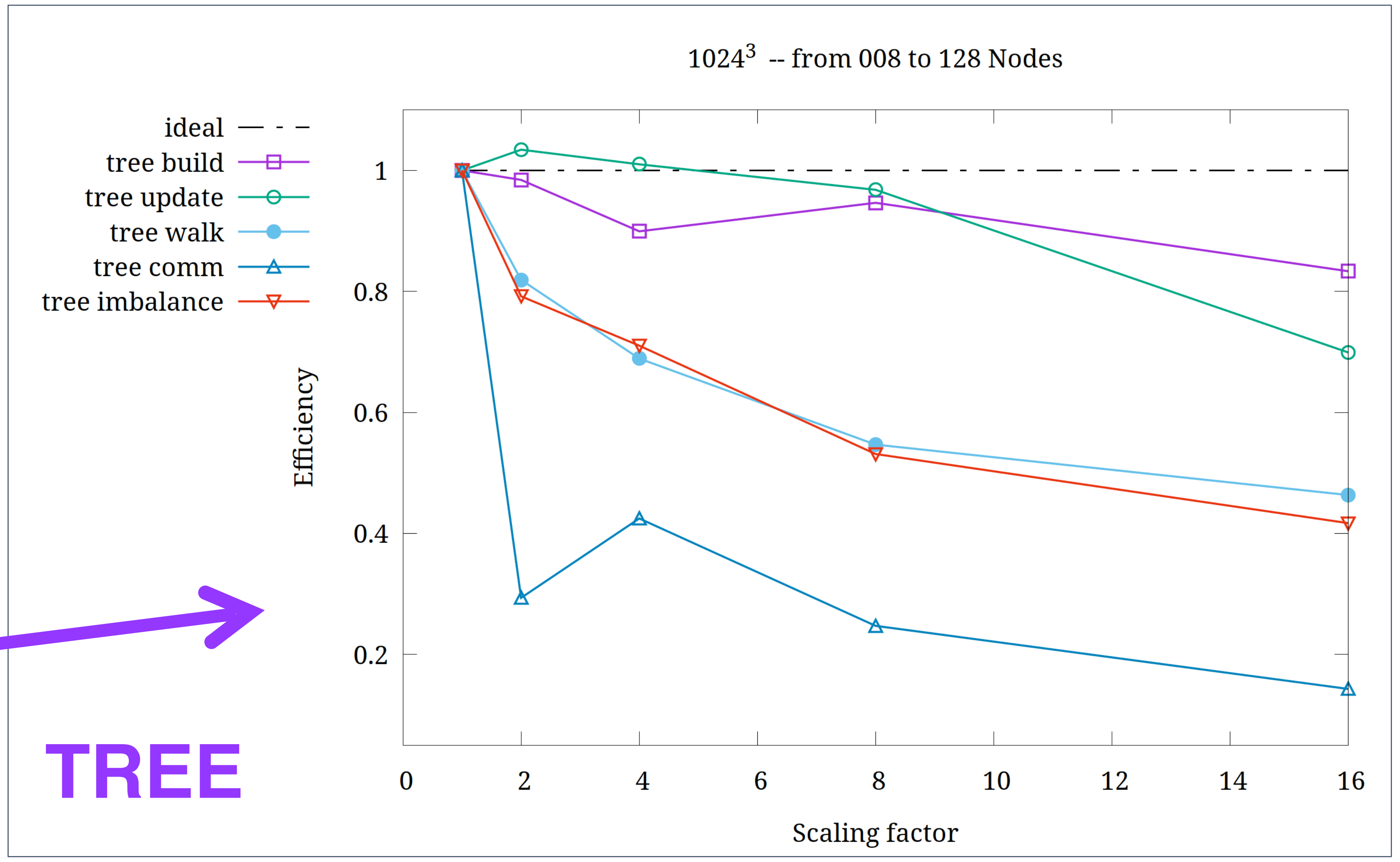
Ongoing: Assessing scalability, targeting performance issues

1) GPU scalability: more in detail

2×1024³, 120 Mpc, up to **512 GPUs**







Ongoing: Assessing scalability, targeting performance issues

1) GPU scalability: more in detail

The gravity tree has some noteworthy performance issues, mostly in

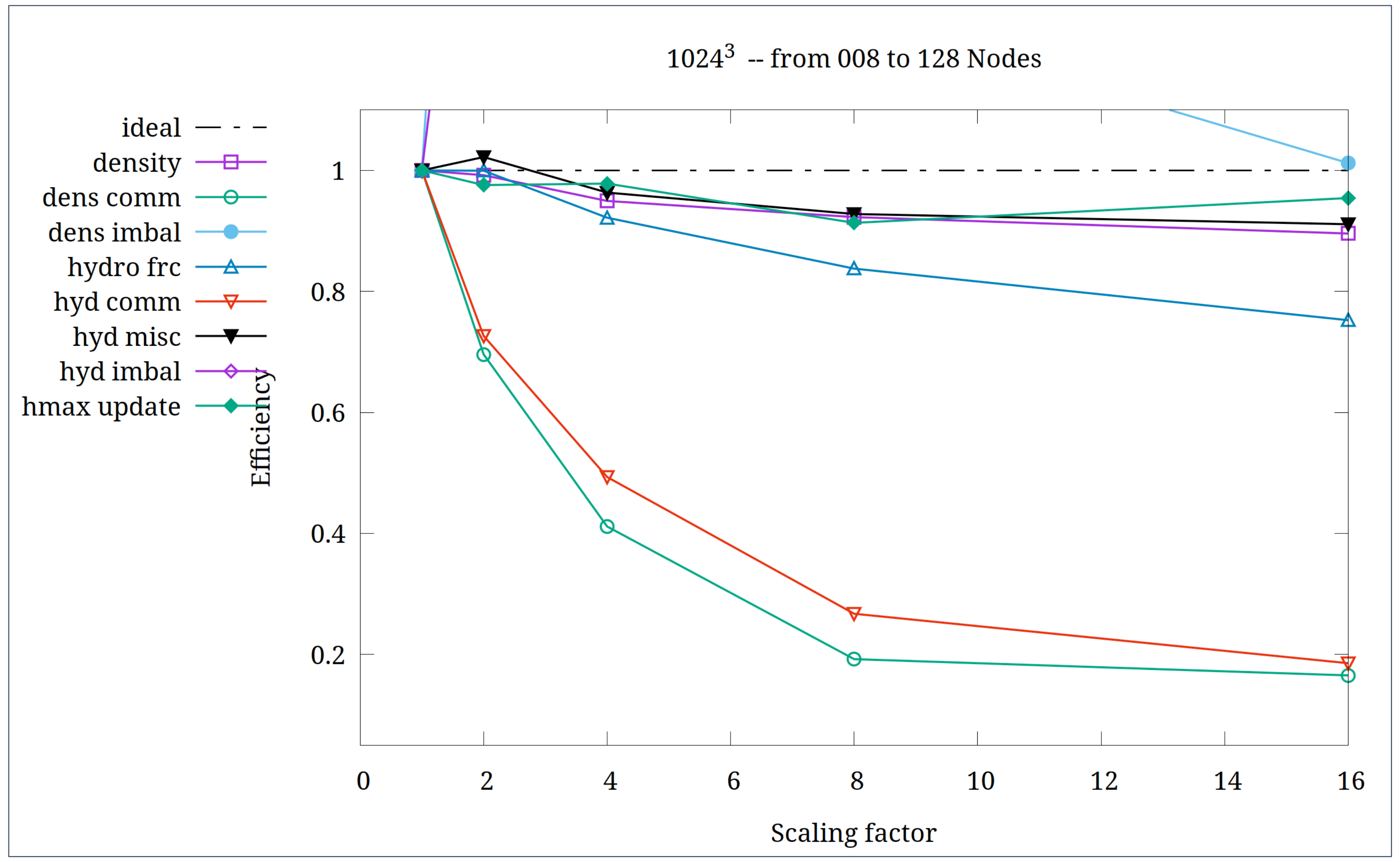
- Tree Walk → Barnes&Hut is not GPU-friendly
- Communication

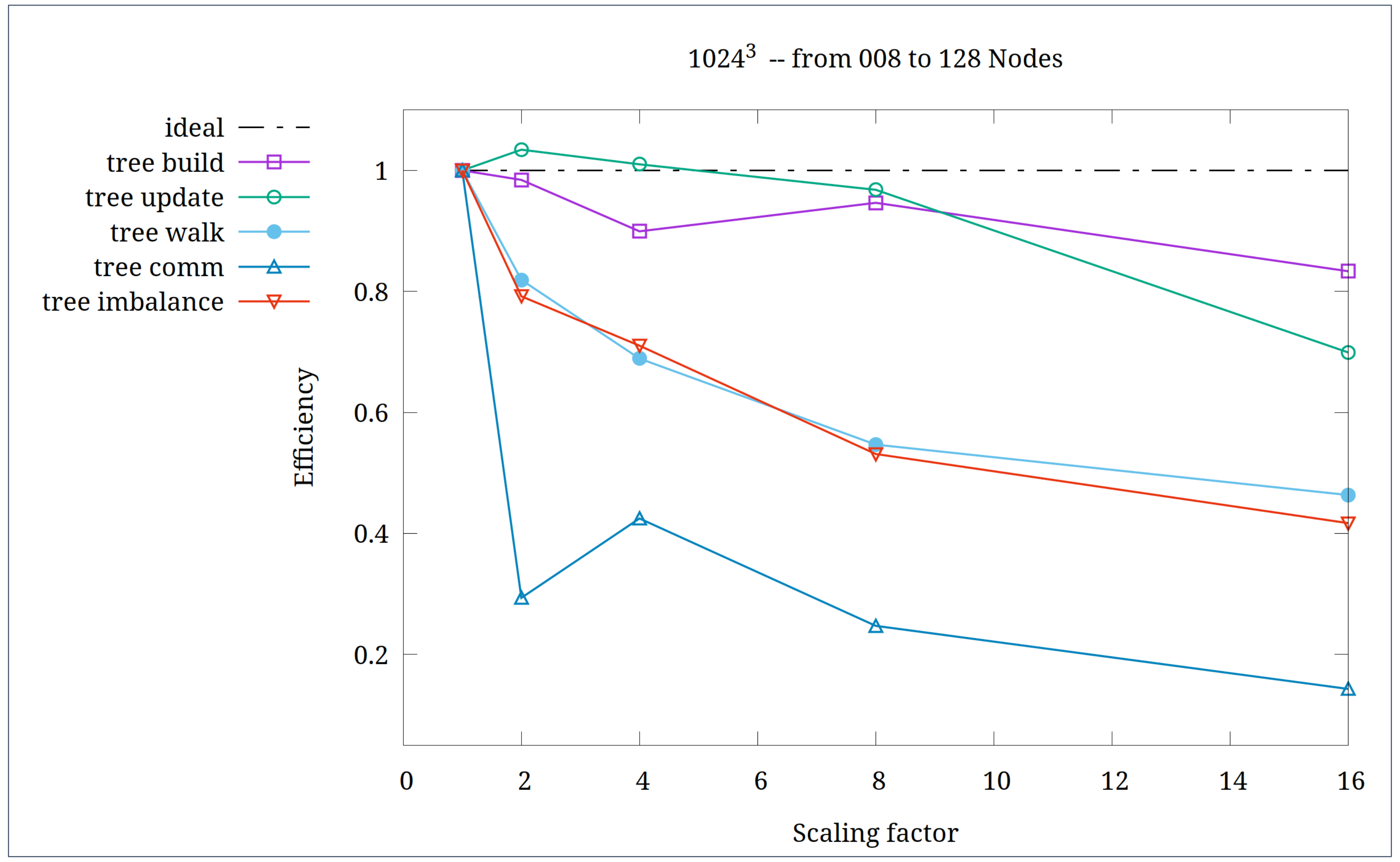
Communication & Nodes update have scalability issues in the SPH part, too.

In the longer term (Dec 2025), we aim for a different implementation:

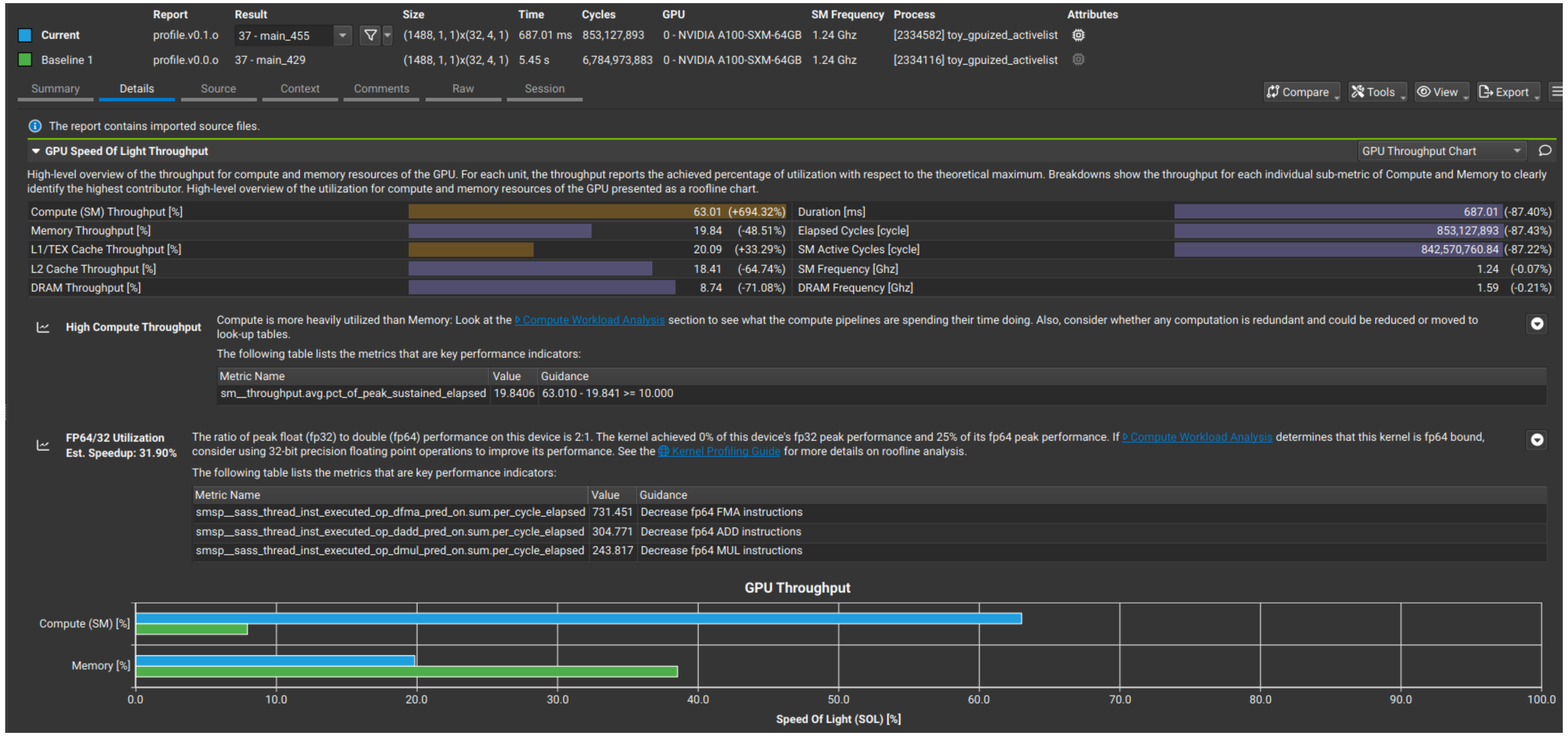
We have extracted a kernel of the code which reproduces the conditions under which gravity is computed in OG3 and which will feature the new, restructured implementation of the tree, where

1. the walk is done for a bunch of particles all together instead of for every single particle, by grouping particles per tree node (they belong to);
2. the Barnes and Hut scheme is not adopted anymore: rather, we opt for a direct computation of the force within a given radius, to avoid to check whether nodes have to be opened and the tree walked further.





Ongoing: Assessing scalability, targeting performance issues



Comparison of two kernels through NVIDIA's NCU profiler

Branch Instructions [inst]	31,905,570,725 (-98.05%)	Branch Efficiency [%]	100.00 (+13.11%)
Branch Instructions Ratio [%]	0.15 (-1.46%)	Avg. Divergent Branches	1,898.30 (-100.00%)

Increased branch efficiency due to a much smaller thread divergence

1) GPU scalability: as for now…

Two strategies have been tested:

kernel 1: reproduces the standard OG3 tree walk strategy —> specific tree walk for each particle (each particle has a specific seed number)

kernel 2: reproduces a modified tree walk strategy, where the geometric centre of the node is considered instead of different tree leaves —> common tree walk for a bunch of particles

Lower threads divergence, higher branch efficiency and better parallelism

Ongoing: Assessing scalability, targeting performance issues

	2048(512x4)[1]	4096(1024x4)[2]	8192(2048x4)[3]
Global efficiency	78.23	73.34	49.75
-- Parallel efficiency	78.23	73.74	55.54
-- Load balance	85.89	83.68	71.74
-- Communication efficiency	91.09	88.11	77.41
-- Computation scalability	100.00	99.47	89.59
-- IPC scalability	100.00	100.87	102.63
-- Instruction scalability	100.00	98.69	86.81
-- Frequency scalability	100.00	99.91	100.56

Percentage(%) (colour scale 0–100)

	2048(512x4)[1]	4096(1024x4)[2]	8192(2048x4)[3]
-- Hybrid Parallel efficiency	78.23	73.74	55.54
-- MPI Parallel efficiency	80.33	76.67	68.64
-- MPI Load balance	88.14	86.91	88.59
-- MPI Communication efficiency	91.14	88.22	77.48
-- Serialization efficiency			
-- Transfer efficiency			
-- OpenMP Parallel efficiency	97.39	96.17	80.91
-- OpenMP Load Balance	97.45	96.29	80.98
-- OpenMP Communication efficiency	99.94	99.88	99.91

Percentage(%) (colour scale 0–100)

2) Performance issues: vectorization

With the assistance of the POP CoE, and within the SPACE CoE, we are profiling in details the code's behaviour.

The results are summarized in tables, as sketched in the figure on the left
(here the example is for the gravity-tree; rows are different metrics, columns refer to the total number of threads)

from which some key indicators can be collected

Number of processes	2048	4096	8192
Elapsed time (sec)	47.714394	25.446344	18.755917
Efficiency	1.0	0.937549	0.635991
Speedup	1.0	1.875098	2.543965
Average IPC	0.961925	0.970340	0.987195
Average frequency (GHz)	3.190112	3.187294	3.207830

Ongoing: Assessing scalability, targeting performance issues

	2048(512x4)[1]	4096(1024x4)[2]	8192(2048x4)[3]
Global efficiency	78.23	73.34	49.75
-- Parallel efficiency	78.23	73.74	55.54
-- Load balance	85.89	83.68	71.74
-- Communication efficiency	91.09	88.11	77.41
-- Computation scalability	100.00	99.47	89.59
-- IPC scalability	100.00	100.87	102.63
-- Instruction scalability	100.00	98.69	86.81
-- Frequency scalability	100.00	99.91	100.56

Percentage(%) scale: 0 – 20 – 40 – 60 – 80 – 100

	2048(512x4)[1]	4096(1024x4)[2]	8192(2048x4)[3]
-- Hybrid Parallel efficiency	78.23	73.74	55.54
-- MPI Parallel efficiency	80.33	76.67	68.64
-- MPI Load balance	88.14	86.91	88.59
-- MPI Communication efficiency	91.14	88.22	77.48
-- Serialization efficiency			
-- Transfer efficiency			
-- OpenMP Parallel efficiency	97.39	96.17	80.91
-- OpenMP Load Balance	97.45	96.29	80.98
-- OpenMP Communication efficiency	99.94	99.88	99.91

Percentage(%) scale: 0 – 20 – 40 – 60 – 80 – 100

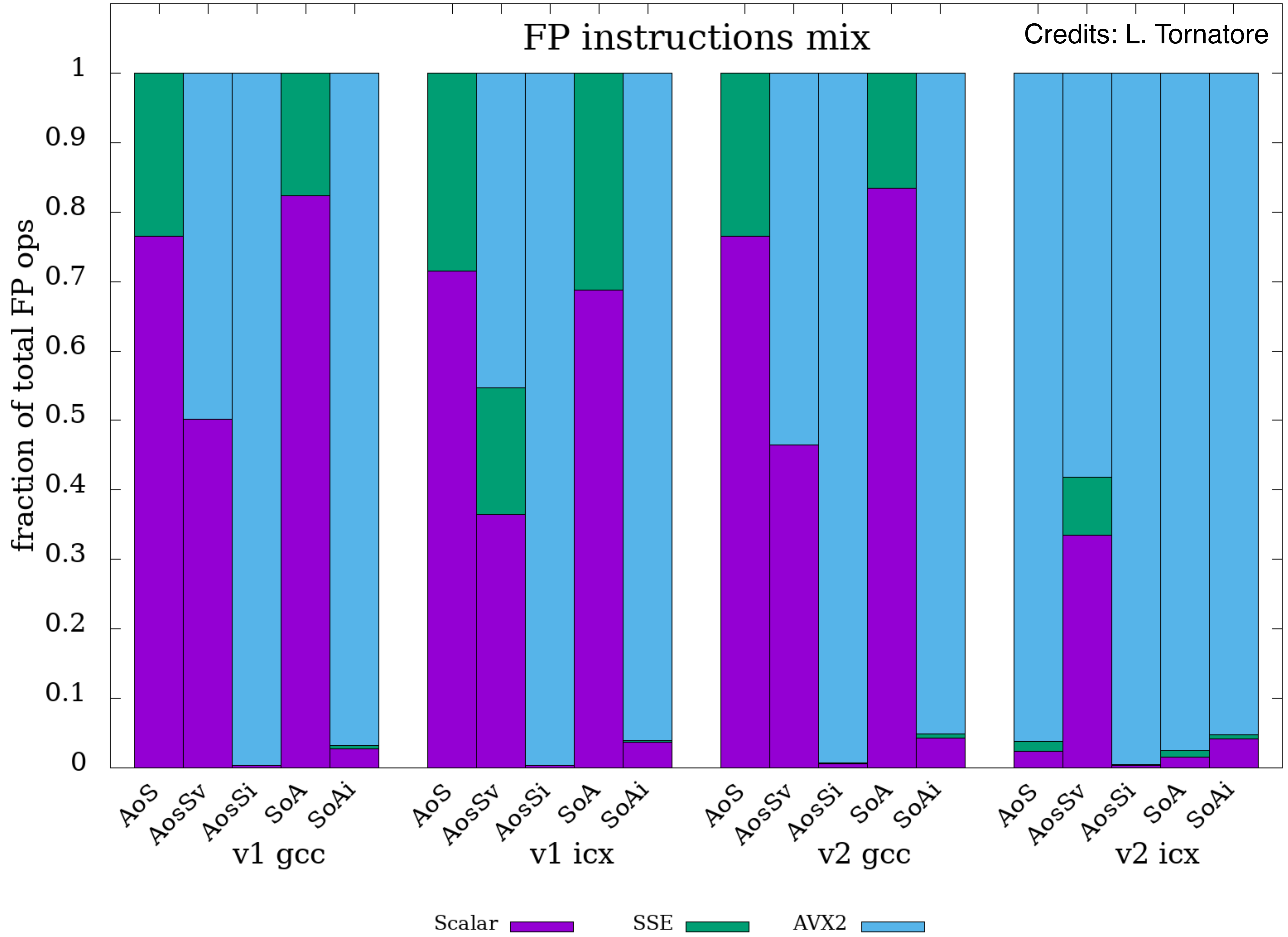
2) Performance issues: vectorization

The low IPC (Instructions Per Cycle), although constant with decreasing workload, indicates that the computational efficiency is not high.
Further inspection returned that in particular the **vectorization ratio is very small** (~10%) and limited to 128bits registers

→ **the main target is to re-formulate the data structures that now consists in Arrays of (large)Structures**

Number of processes	2048	4096	8192
Elapsed time (sec)	47.714394	25.446344	18.755917
Efficiency	1.0	0.937549	0.635991
Speedup	1.0	1.875098	2.543965
Average IPC	0.961925	0.970340	0.987195
Average frequency (GHz)	3.190112	3.187294	3.207830

Ongoing: Assessing scalability, targeting performance issues



Vectorization ratio achieved on average (= fraction of vector floating point (FP) instructions issued to the total number of FP instructions) under different assumptions.

2) Performance issues: vectorization

We have tested the effect of different data layout on the achievable vectorization in a loop that reproduces the N-Body pattern, assuming that:

- A fraction of particle is active
- Every active particle interacts with its neighbours
- Neighbours are not close in memory

We experimented AoS, AosS and SoA with some carefully crafted loops to

- enhance auto-vectorization by the compiler (AoS, SoA)
- test compilers vector extensions (AosSv)
- explicitly use vector intrinsics (AosSi, SoAi)

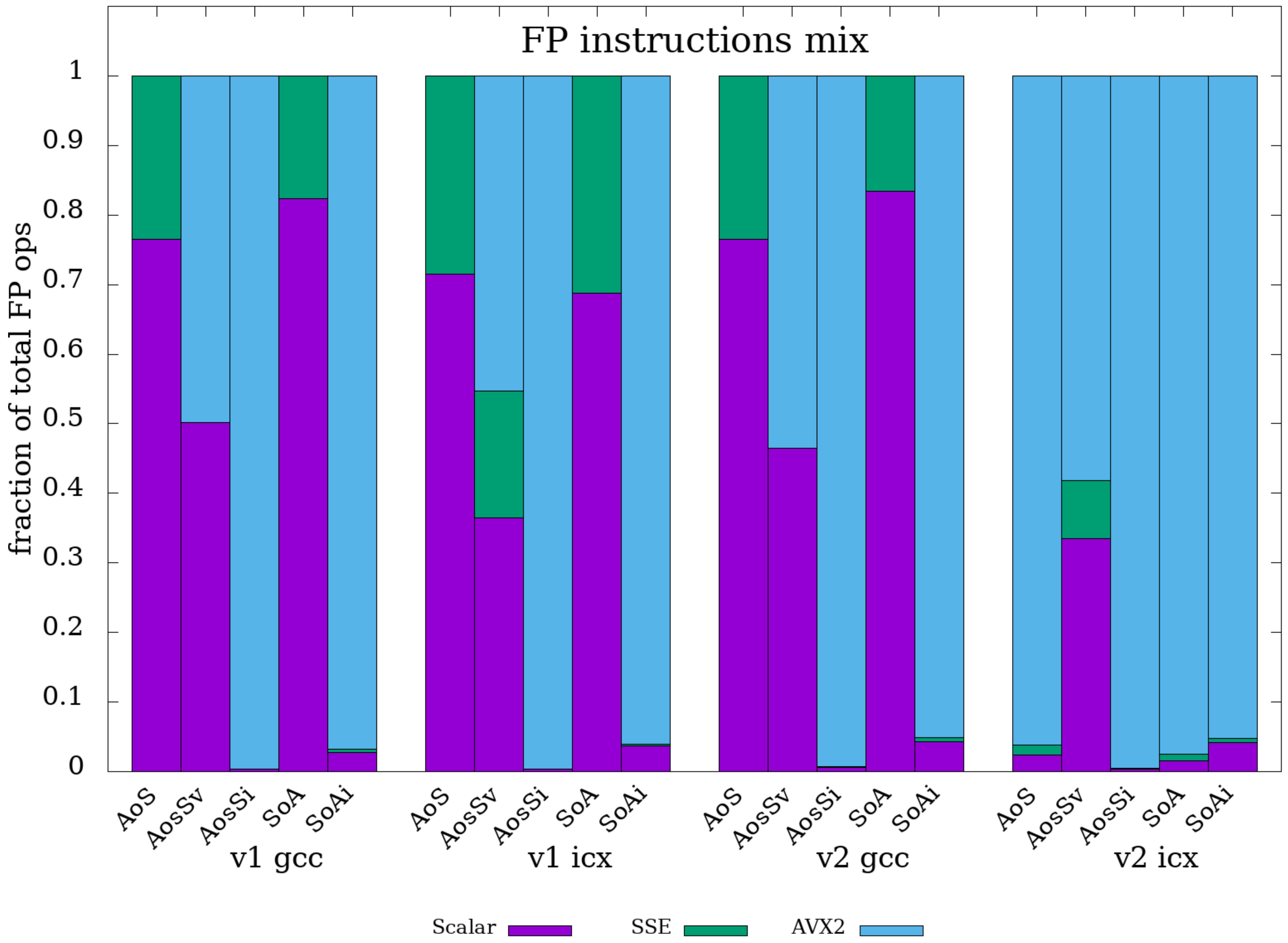
Also, we have tested the effect of enhancing the memory contiguity (**v1** VS **v2**) on different compilers (**gnu** VS **intel**)

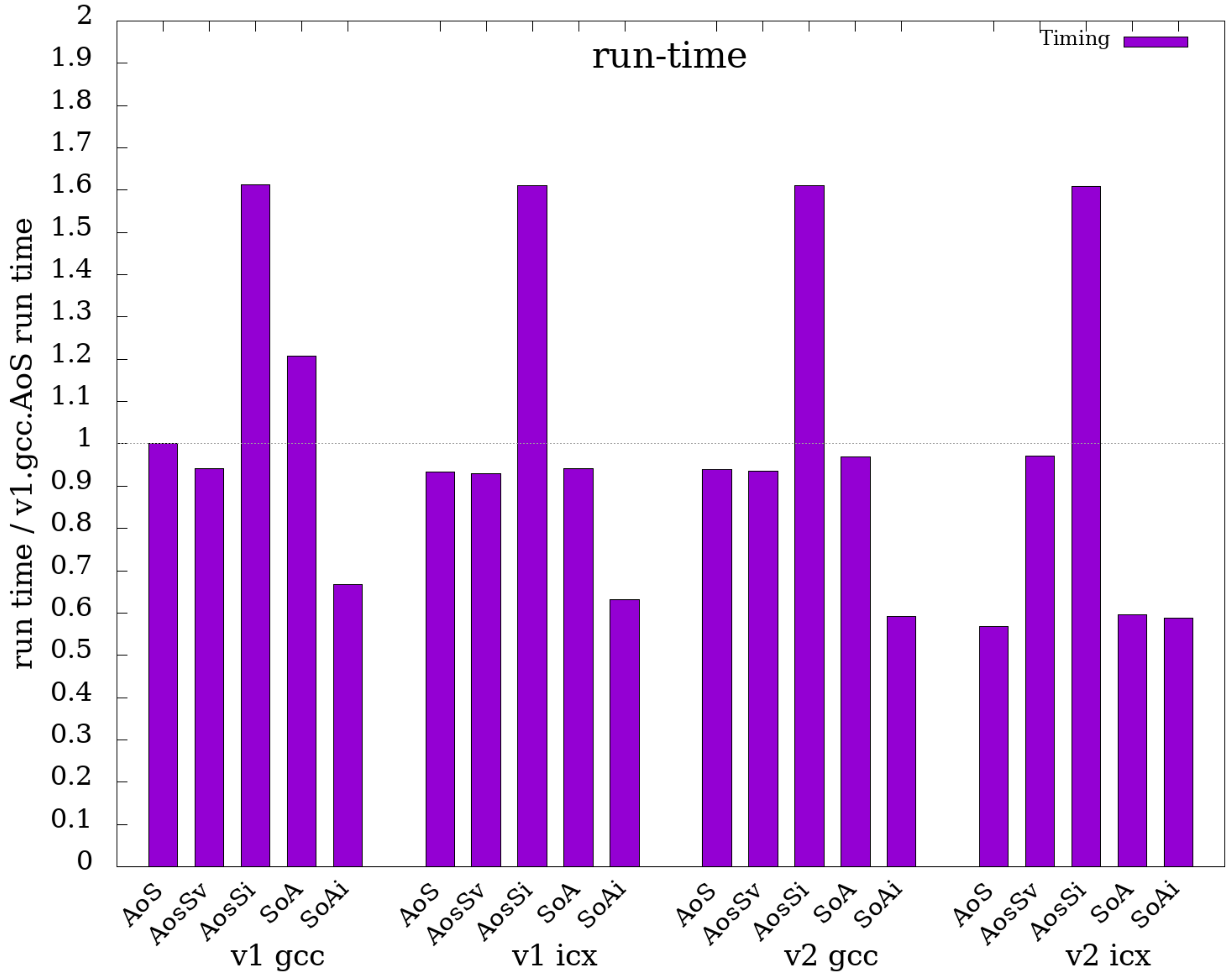
Cons of vector instructions: every instruction requires more CPU cycles, the CPU frequency is generally decreased for an intense vector burst

Ongoing: Assessing scalability, targeting performance issues

2) Performance issues: vectorization

Credits: L. Tornatore

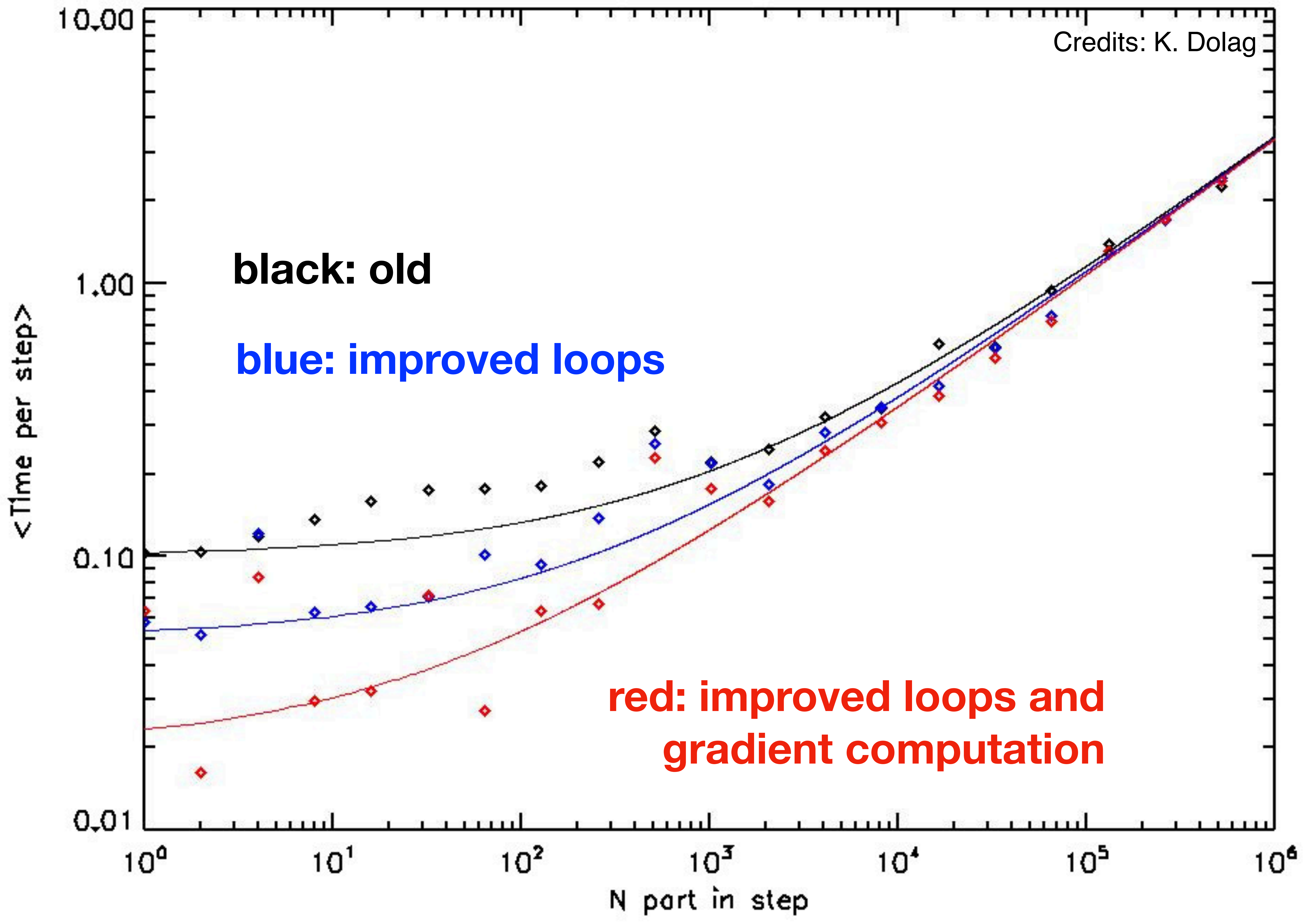




Results from LEONARDO DCGP, obtained by measuring performance counters via PAPI

1. A large vectorization fraction with the wrong data layout is not an advantage (e.g. AosSv) because a larger # of instructions is issued and the cpu frequency is decreased
2. Smaller structures offer ~10% of gain in terms of run-time (e.g. AosSv)
3. Memory contiguity seems to be the most promising trick (go from v1 to v2), especially if the compiler is good in spotting opportunities (see icx vs gcc in v2.AoS)

Ongoing: Assessing scalability, targeting performance issues



Comparison of the required time per time step at different numbers of particles in each time bin.

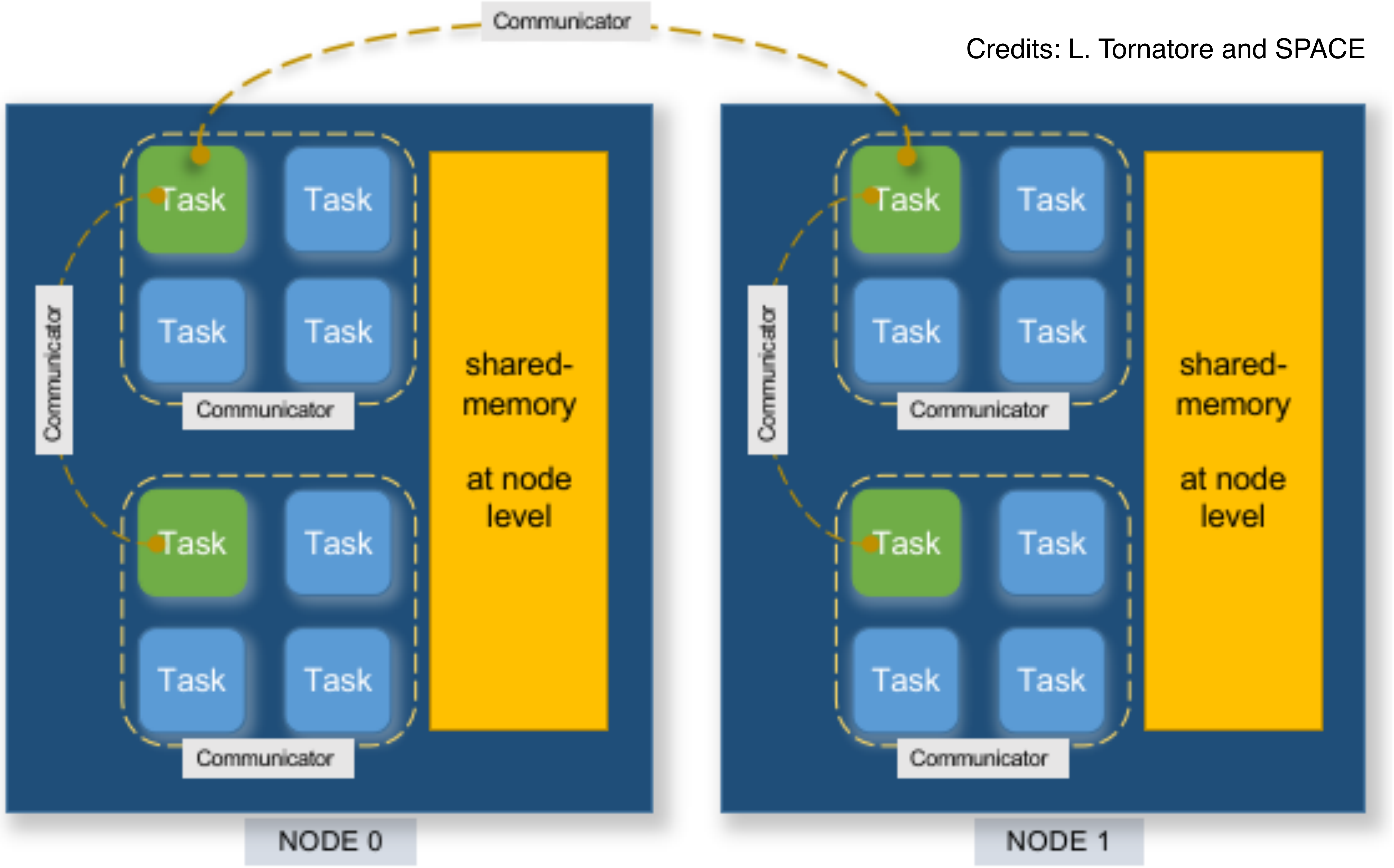
3) CPU optimization

Loop restructuring leads to a 2x performance in timesteps with a small # of particles (blue VS black curves)

Updates on the gradient computation and more precise memory allocation further increase the performance (red VS blue)

In total, these improvements speed up the calculation of the smallest time bins by up a factor of ~5 (red VS black).

Ongoing: Assessing scalability, targeting performance issues



Credits: L. Tornatore and SPACE

Framework (developed within SPACE) to explore the topology of a given infrastructure and build a hierarchy of MPI communicators

4) Topology awareness (= capability of the code to explore the NUMA topology of a machine)

A hierarchy of communicators groups the MPI tasks based on their NUMA affinity.
Every MPI task can understand on what node it is running and which are the other MPI tasks that run on the same node. The tasks running on the same node are grouped in a dedicated communicator and share the node memory via the MPI's shared-memory windows.

Every node has a designated master task that is in charge of MPI communications with other nodes, and the master tasks of all nodes participate in a dedicated MPI communicator.

Final goal: avoid too many communications and develop algorithms that are increasingly communication-free.

Next Steps and Expected Results

So far, results in line with timescale, milestones and KPIs identified.

Key Science Projects

1. → **EAGER: Evolution of gAlaxies and Galaxy clustErs in high-Resolution cosmological simulations**

Stefano Borgani, Milena Valentini, Luca Tornatore, Alice Damiano, Alex Saro, Giuliano Taffoni, Tiago Castro

2. → **SLOTH: Shedding Light On dark matter wiTH cosmological simulations**

Milena Valentini, Stefano Borgani, Tiago Castro, Luca Tornatore, Matteo Viel, Alice Damiano, Pierluigi Monaco, Giuliano Taffoni