First steps towards cosmological simulations with full EAGLE physics

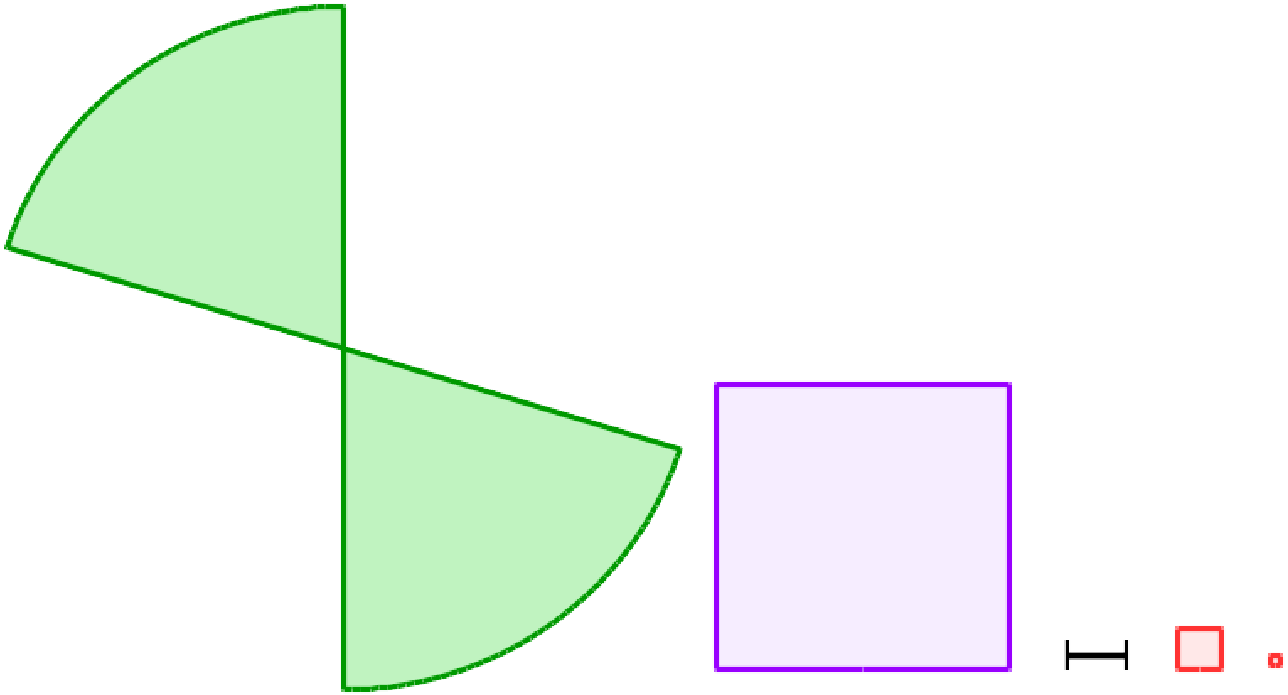
Matthieu Schaller

Leiden Observatory, Netherlands

with

Stefan Arridge, Josh Borrow, Alexei Borissov (DiRAC), Richard Bower, Aidan Chalk (Hartree Centre), Peter Draper, Pascal Elahi (Perth), Pedro Gonnet (Google), Loic Hausamman (EPFL), Yves Revaz (EPFL), Bert Vandenbroucke (St. Andrews), James Willis

Cosmological scales?



Any needs for more?

- Thinking of future weak-lensing surveys:
  - Measure some cosmological information on scales down to ~ 1-30 Mpc. Clearly “baryon effects” seen on these scales.
  - “Common wisdom” asks for volume in excess of 300 Mpc.
  - That asks for particles counts in excess of 4500^3 ~ 100 billion.
  - With EAGLE code that would be >300M CPU hours and 1.3PB of RAM

SWIFT 101

- Aimed at replacing Gadget for EAGLE-like runs.
- Use task-based parallelism, modern algorithms, better parallelisation and domain decomposition.
- Leaner memory footprint, faster i/o, more modular, multiple hydro schemes.
- Collaboration with computer scientists and industry.

SWIFT 101

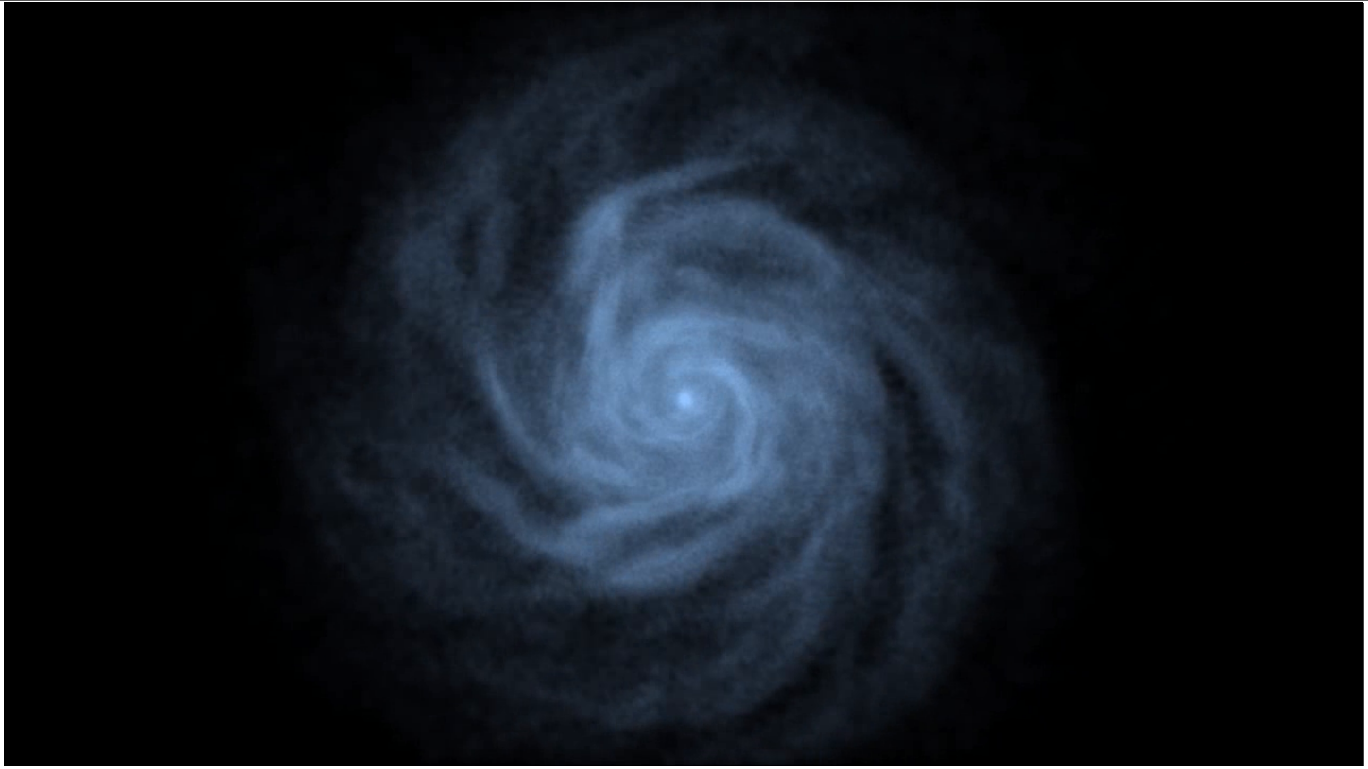
- Hydro neighbour finding based on regular AMR cell structure. Many flavours of SPH + “GIZMO”.

- FMM for gravity with a multipole-mesh method for periodic gravity.

- Particles sorted to enhance the vectorization of the code.

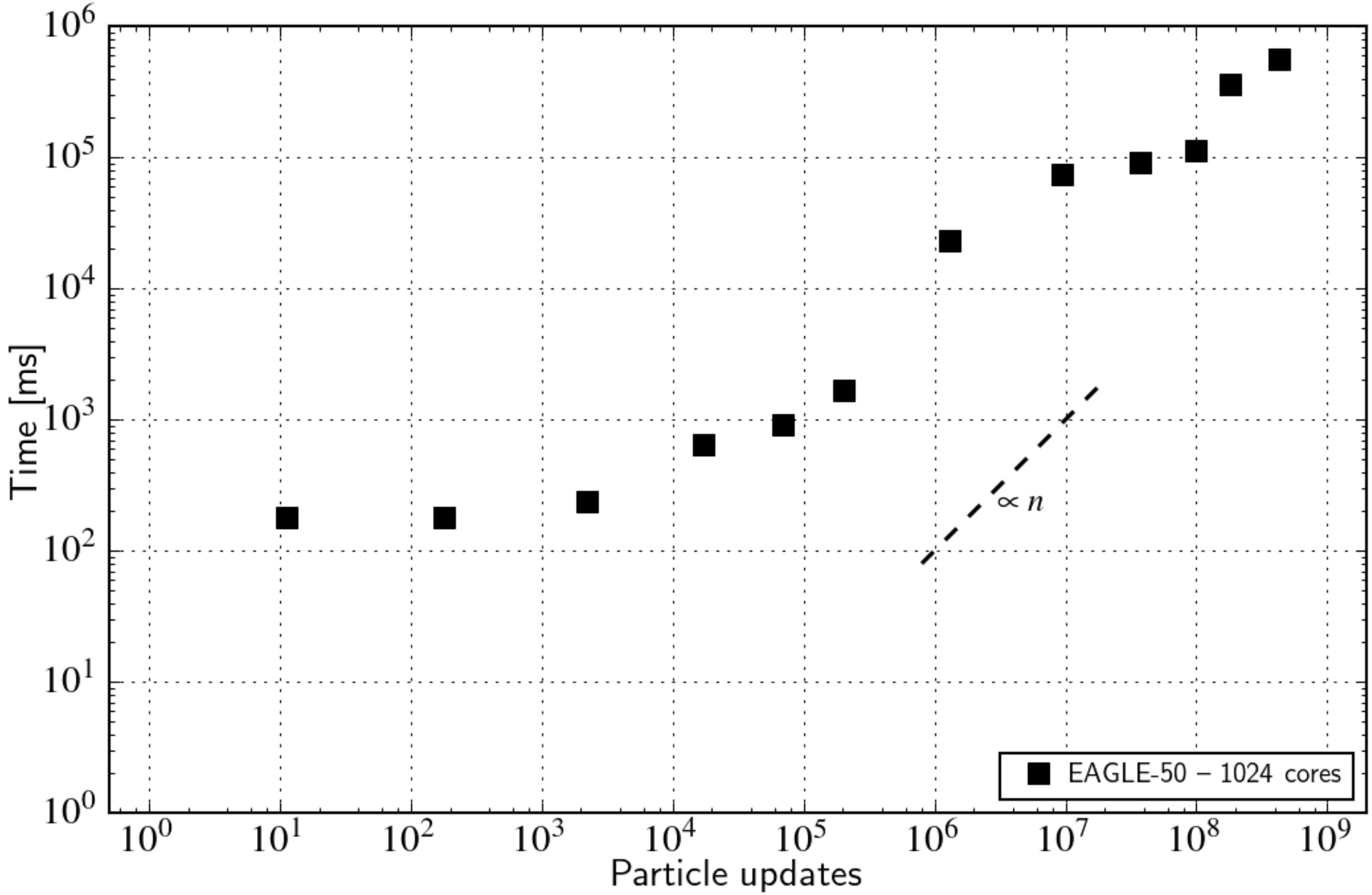
- Activation of work only in the “active” parts of the tree.

SWIFT interlude - AGORA galaxy

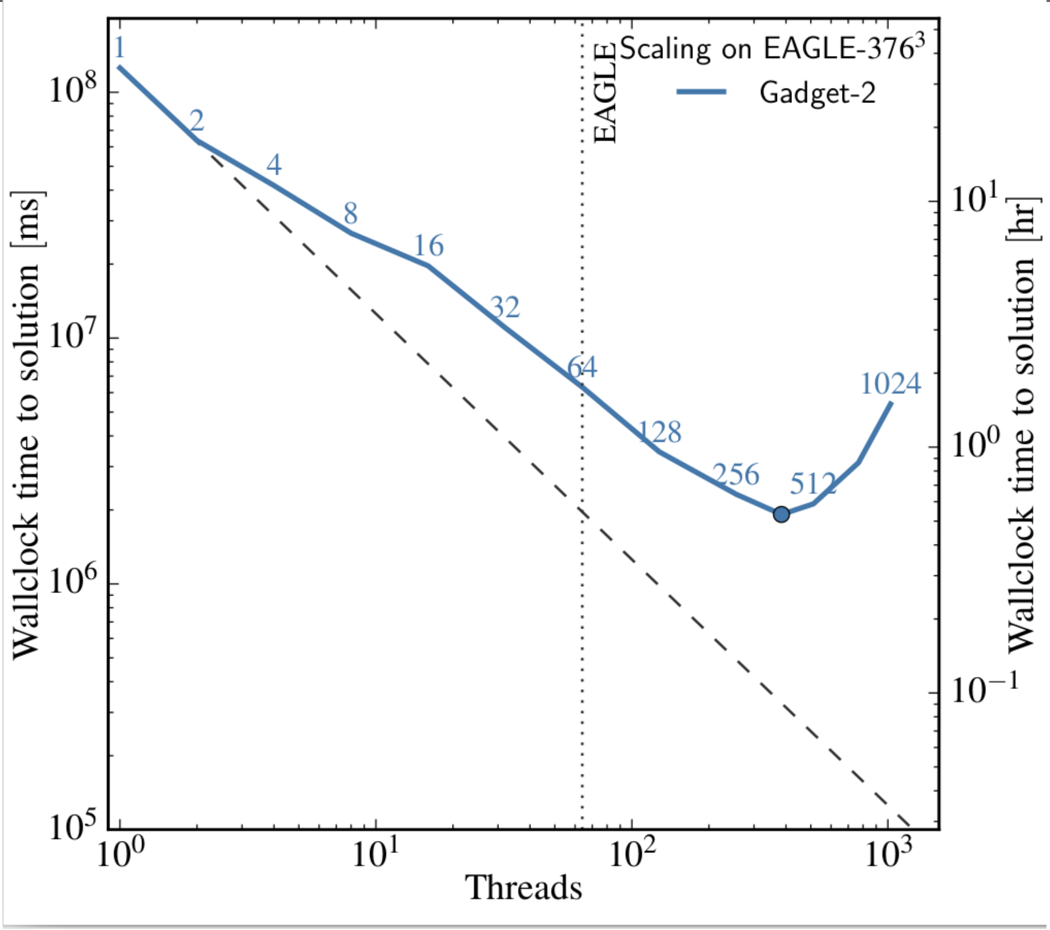


L. Hausammann, Y. Revaz, MS

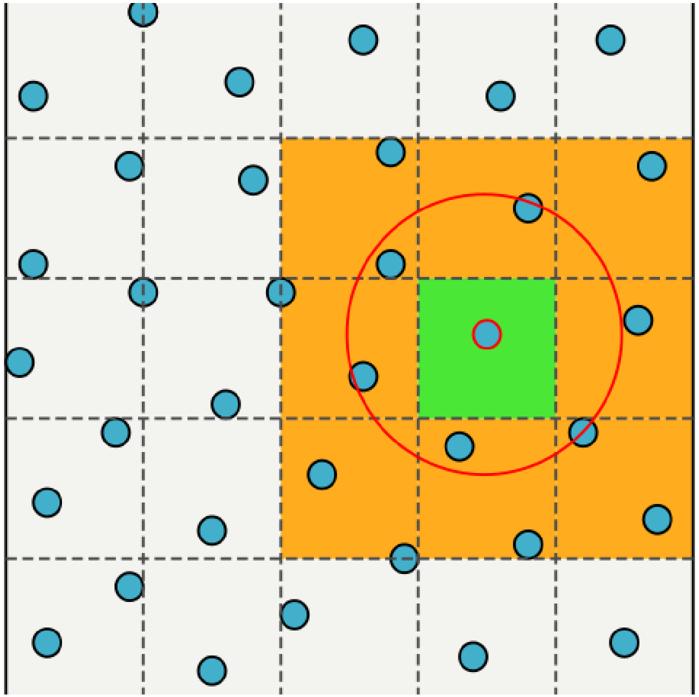
EAGLE code performance



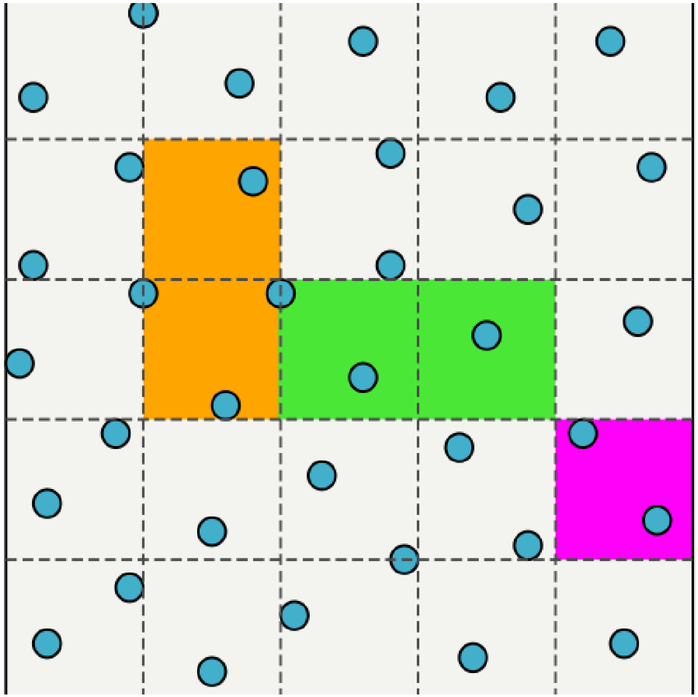
Strong scaling behaviour



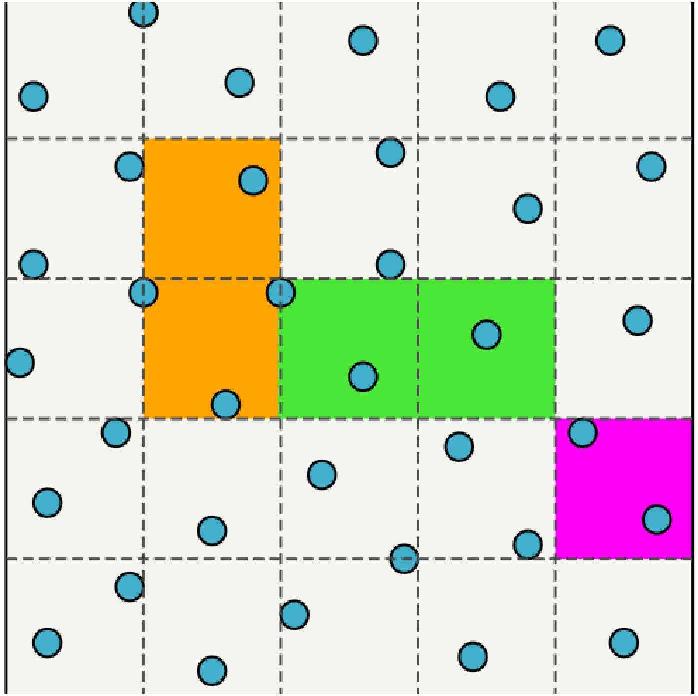
AMR Verlet-list neighbour search



- Target ~500 particles per cell via adaptive mesh refinement.
- Cell size naturally matches particle neighbour search radius.
- Particles only interact with particles in the same cell or any direct neighbouring cell.



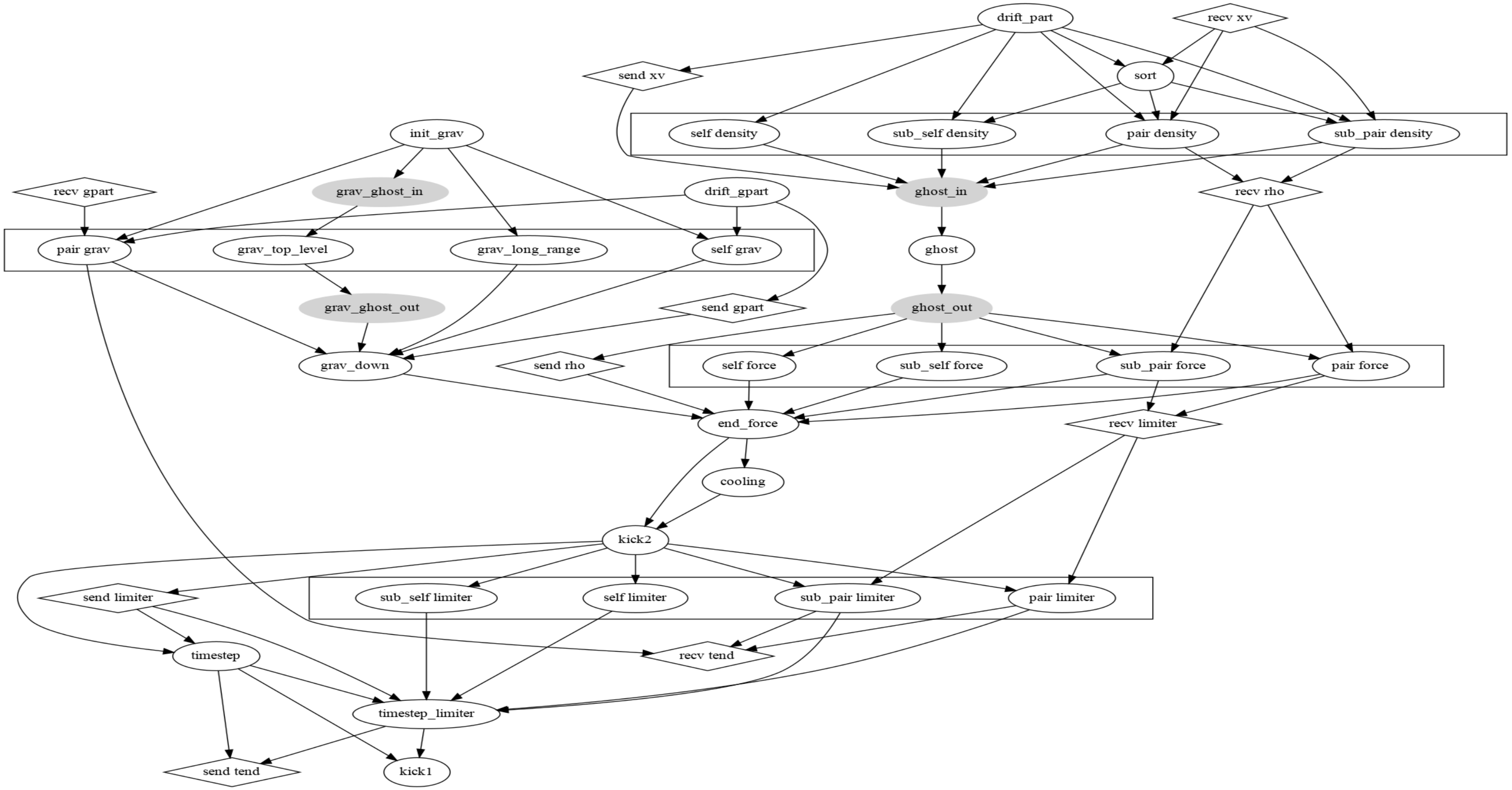
- Cells pairs do not need to be processed in any pre-defined order.
- Only need to make sure two threads do not work on the same cell.
- Cell pairs can have vastly different work-loads.



- Cells pairs do not need to be processed in any pre-defined order.
- Only need to make sure two threads do not work on the same cell.
- Cell pairs can have vastly different work-loads.

→ Need runtime dynamic scheduling

Task-based parallelism for science case



Task dependencies for SWIFT v0.6.0-1090-ga8e380a5-dirty

Task-based parallelism in action

Task-graph for one time-step. Colours correspond to different task types.
Almost perfect load-balance achieved on 16 cores.

Local time-stepping

Our particle methods for gravity and hydro-dynamics have a linear cost.

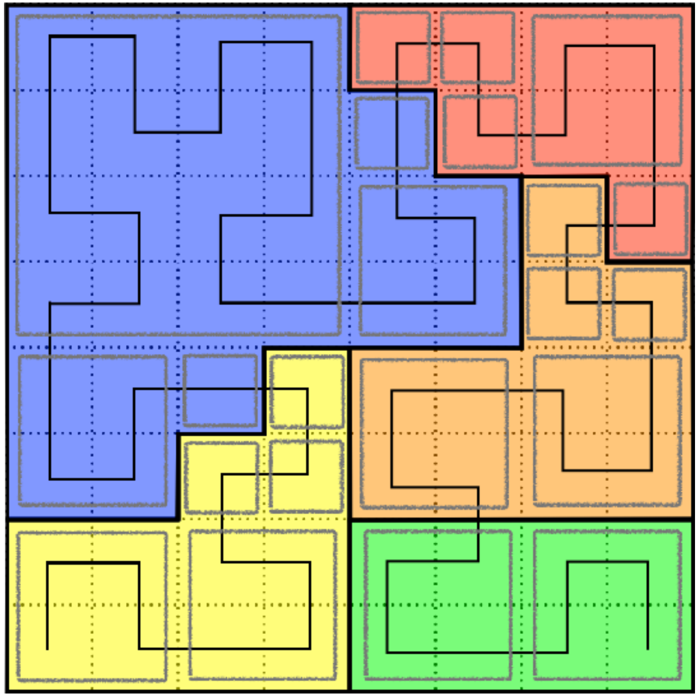
→ By only updating the particles on a short time-step we can gain orders of magnitude in run time.

→ You also kill your scaling.

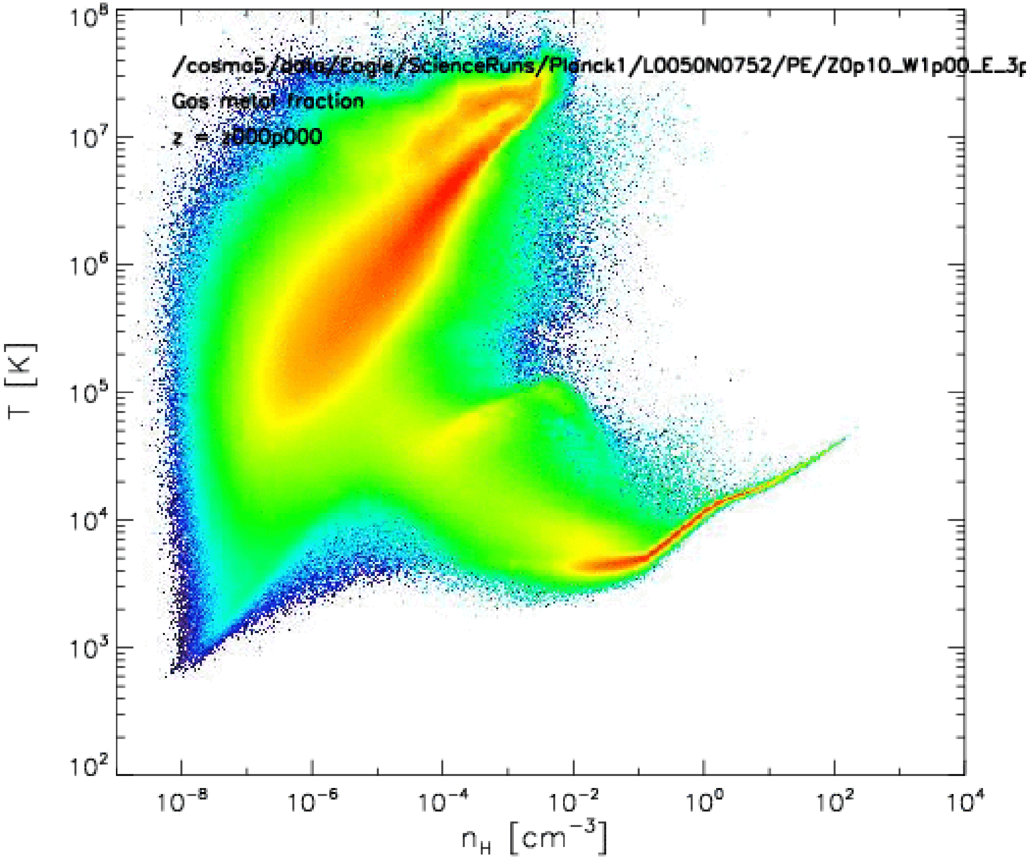
The question is then how to load-balance and parallelize this.
How do you update ~10 particles efficiently on 1000+ nodes?

Domain Decomposition

1. Sort the data along a space-filling curve.
2. Split the curve such that each node gets the same number of particles.
3. Do this at the top-level or close to the top.



Smallest time-steps?



A hand-written solution

- Avoid MPI for small updates.
- The particles with the smallest time-steps should be at the centre of their domains and not require any “halo” particles.
- Small time-steps are at the centre of galaxies.

- Identify galaxies → Grow domains organically around them
→ Get efficient code.

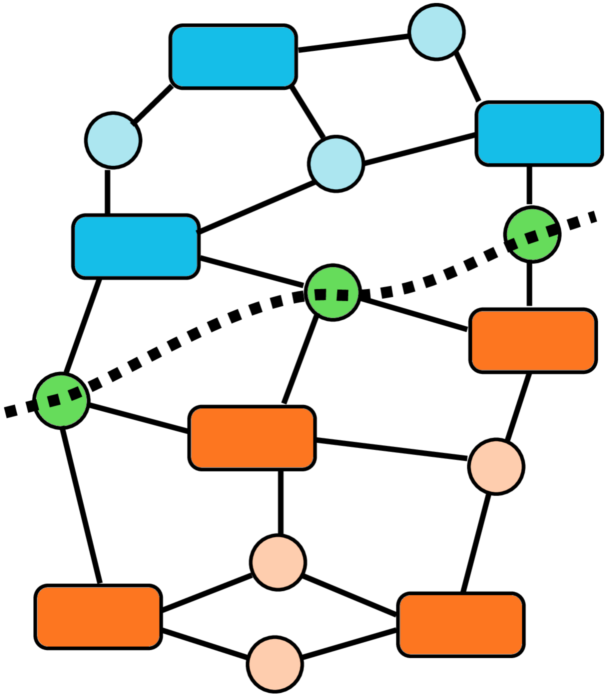
A hand-written solution

- Avoid MPI for small updates.
- The particles with the smallest time-steps should be at the centre of their domains and not require any “halo” particles.
- Small time-steps are at the centre of galaxies.

- Identify galaxies → Grow domains organically around them
  → Get efficient code.

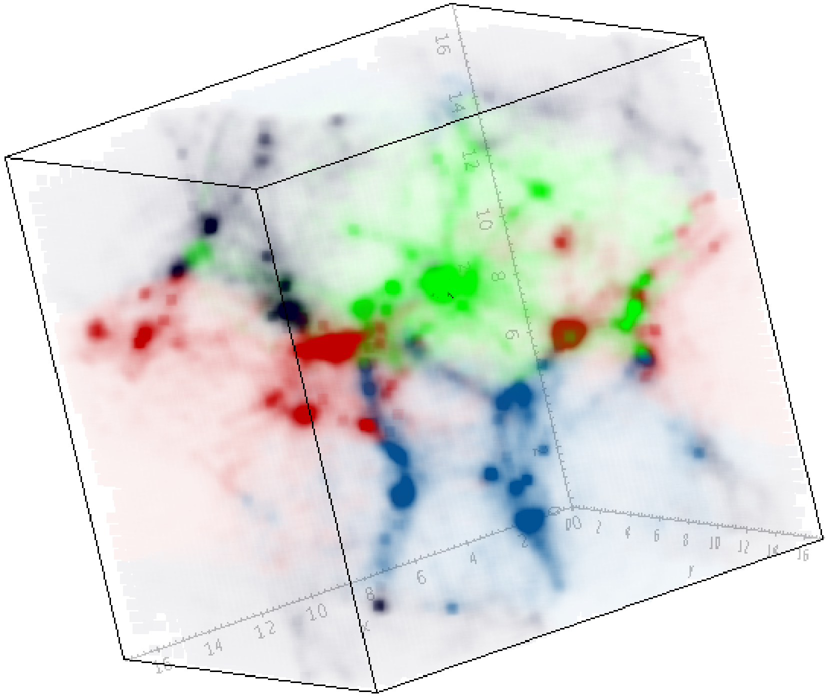
Drawbacks: Not generic. Identifying galaxies is not trivial.

A Graph-based strategy



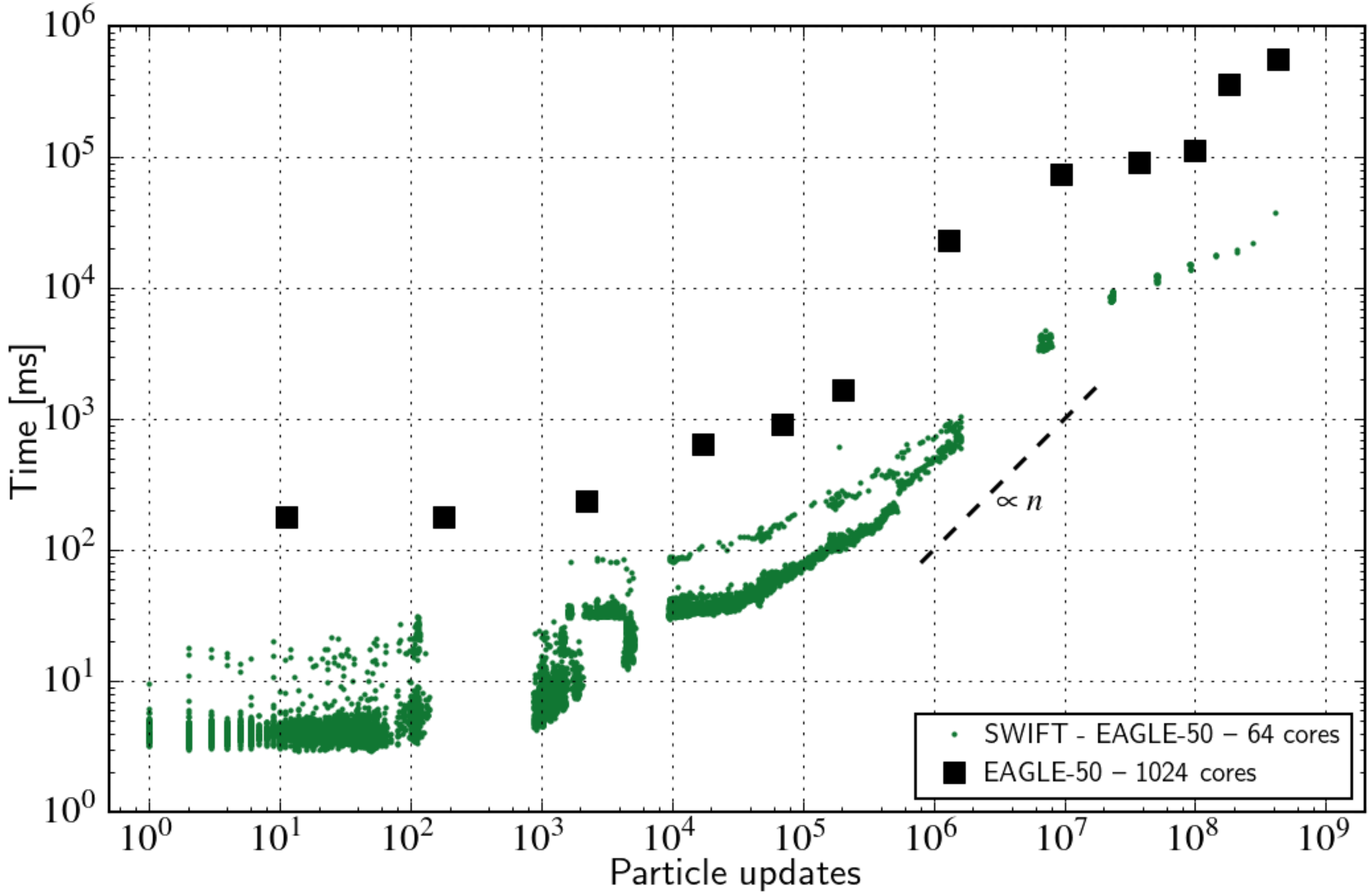
- For each task, we compute the amount of work (=runtime) required.
- We build a graph where the data are nodes and tasks are hyper-edges.
- METIS is used to split the graph such that the work (not the data!) is balanced.
- Extra cost added for communication tasks to minimise them.

What does it look like?

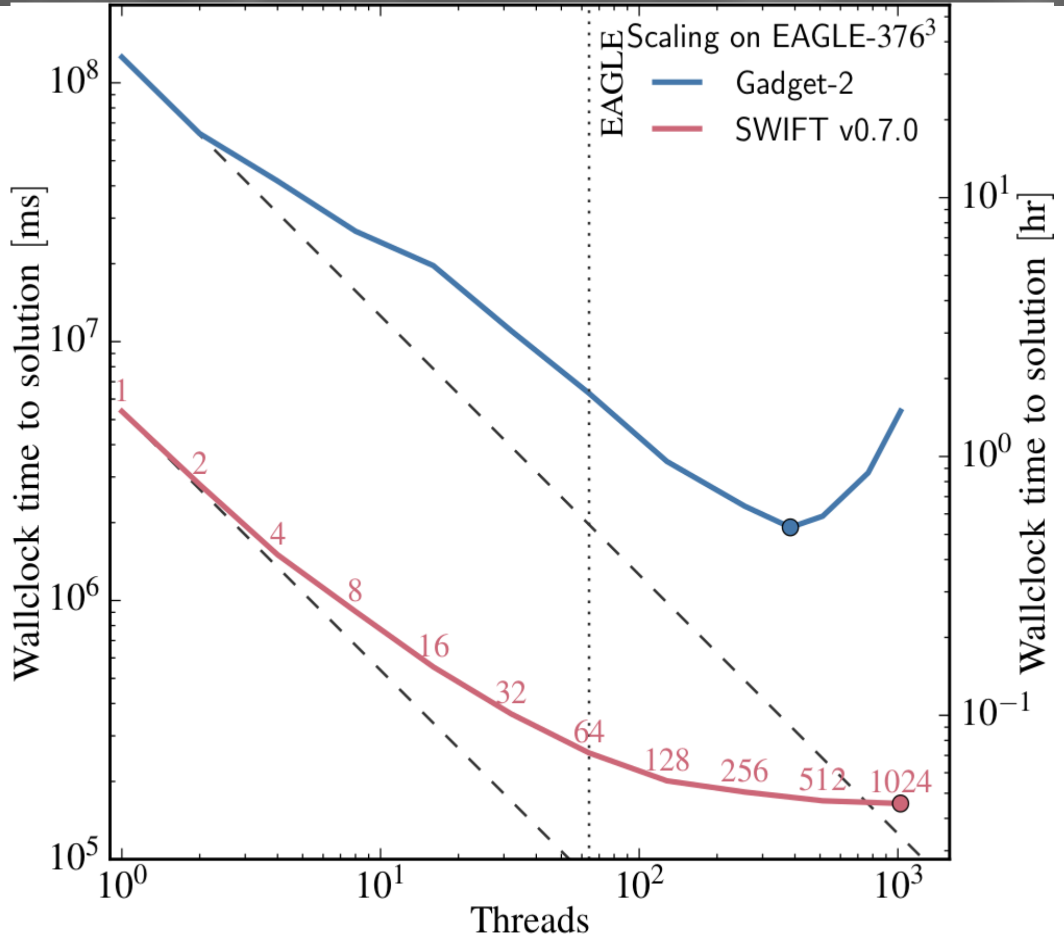


- No regular grid pattern.
- No space-filling curve pattern.
- Good (work) load-balancing by construction.
- The most dense regions are at the centre of their respective domains.

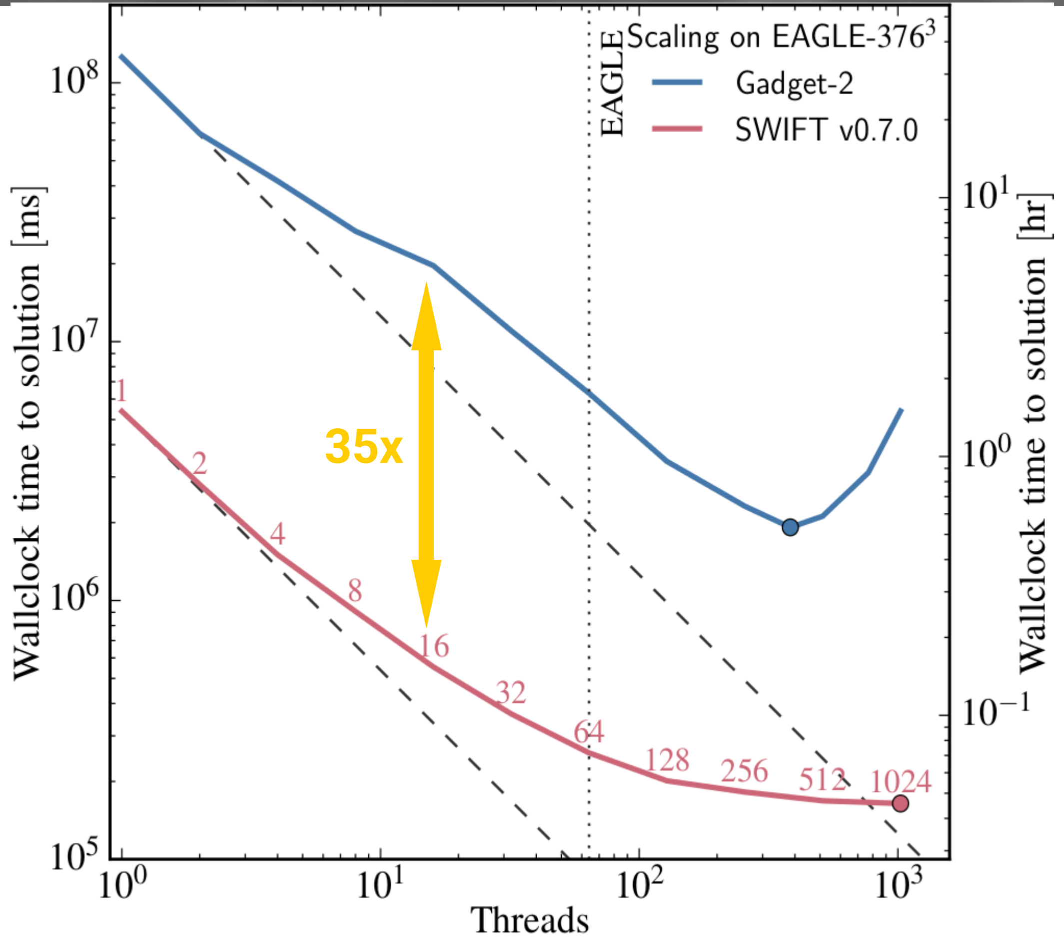
SWIFT code performance



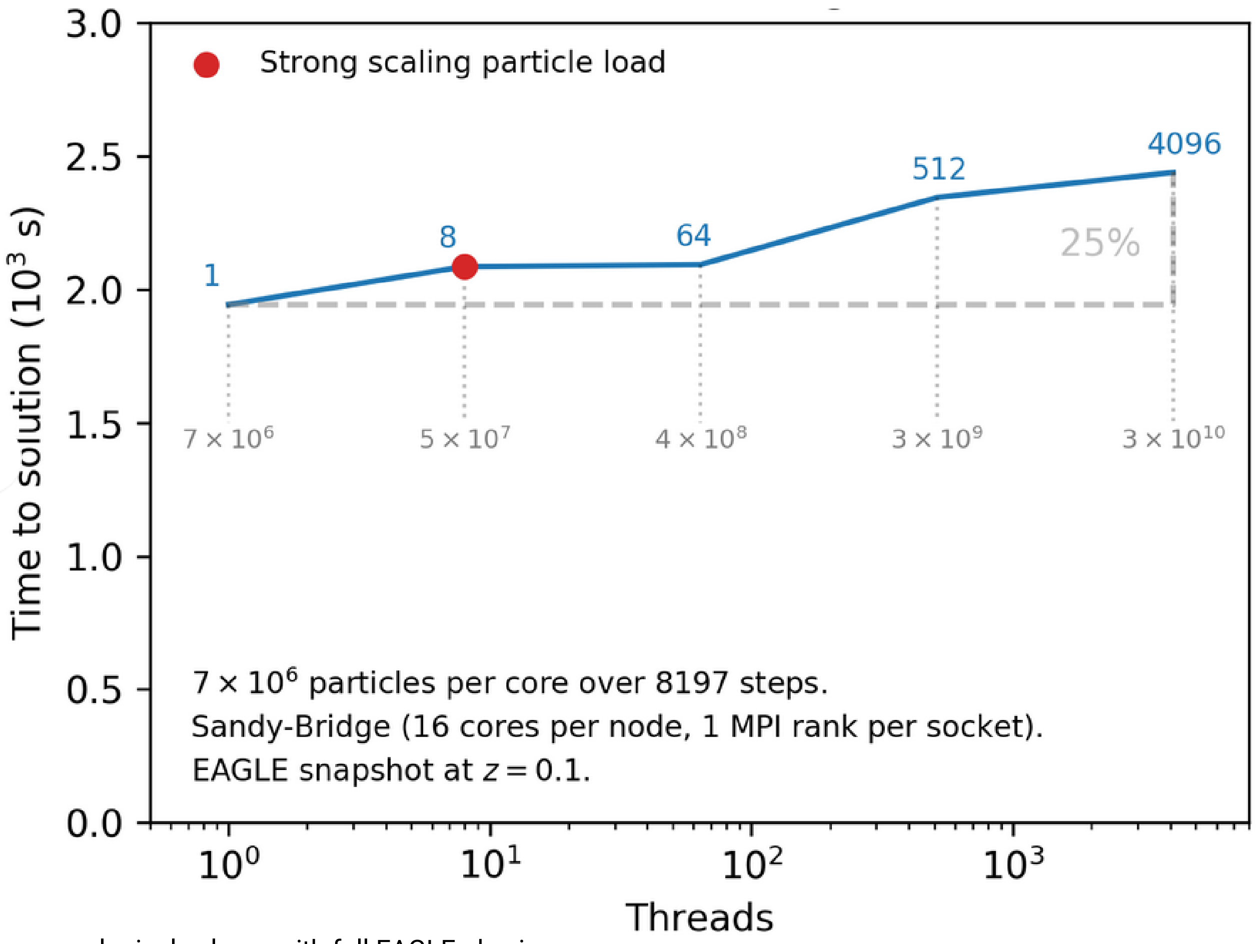
Strong scaling behaviour



Strong scaling behaviour



Sustained performance in weak-scaling



Conclusions

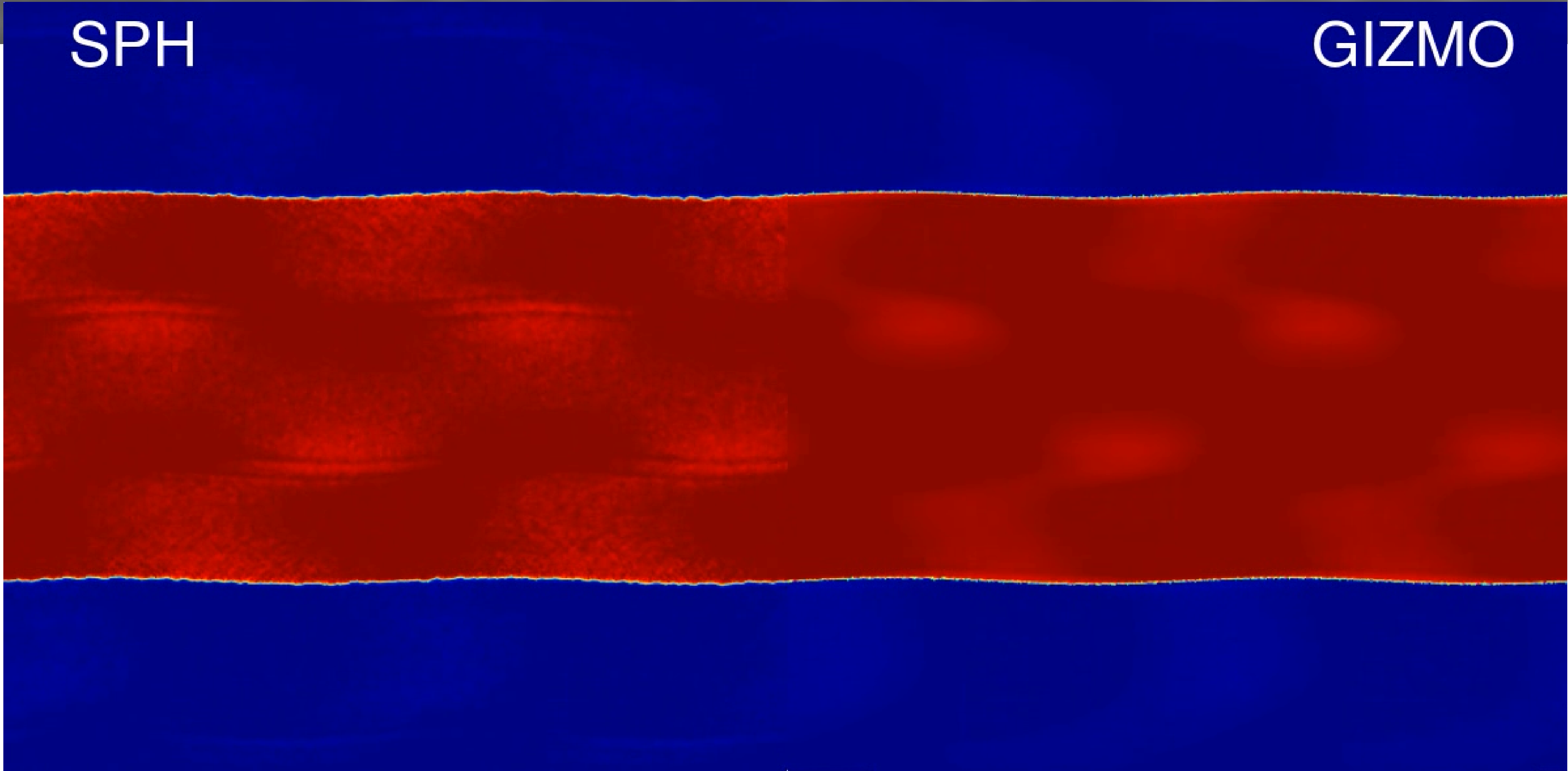
- New algorithms can lead to significant speed-ups over more conventional established methods.

  >30x over Gadget → "half way from peta-scale to exa-scale via algorithms"

- Task-based parallelism as a viable model for actual scientific applications.

  → Not just a research concept.

- The challenge of deep time-step hierarchies requires a rethinking of the domain decomposition algorithms.


SPH
GIZMO