The next-generation cosmological code SWIFT

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

This work started as a collaboration between two departments at Durham University (UK):

- The Institute for Computational Cosmology,
- The School of Engineering and Computing Sciences,

with contributions from the astronomy groups at the university of St-Andrews (UK), Perth (Australia), Leiden (Netherlands), Lausanne (Switzerland) as well as the DiRAC and CSCS software teams.

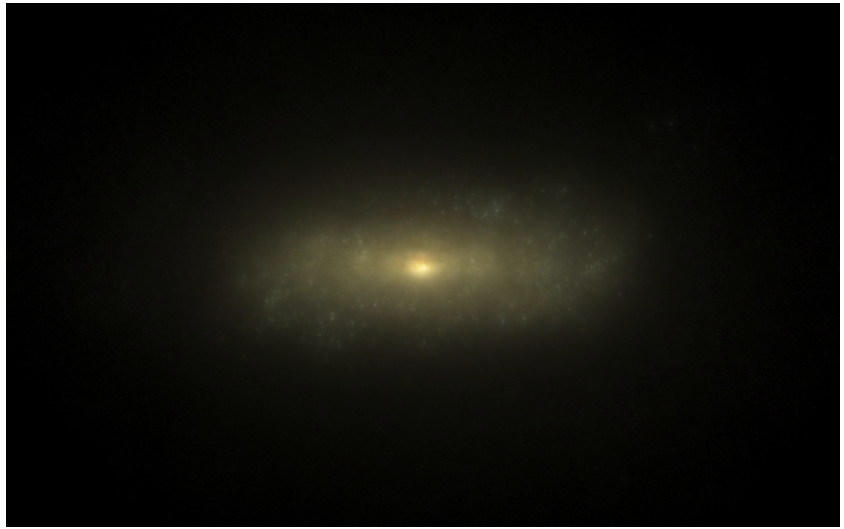
This research is partly funded by an Intel IPCC since January 2015.

Our software is part of the benchmarking challenge for new DiRAC (UK) systems.

What we do and how we do it

- Cosmological simulations of the formation of the Universe and galaxy formation.

- EAGLE project: 48 days of computing on 4096 cores. >500 TBytes of data products (post-processed data is public!).



Trayford+2015

http://www.eaglesim.org/

What we do and how we do it

- Solve coupled equations of gravity and hydrodynamics using particle-based methods (SPH, FVPM, …). → **See poster 06.01 !!**
- Consider the interaction between gas and stars/black holes as part of a large and complex *subgrid* model with free parameters to be calibrated.
- Evolve multiple matter species at the same time.
- Dynamic range of >6 orders of magnitude in “particle size”.
- Typically have >2M time-steps.

Hydrodynamics scheme: The problem to solve

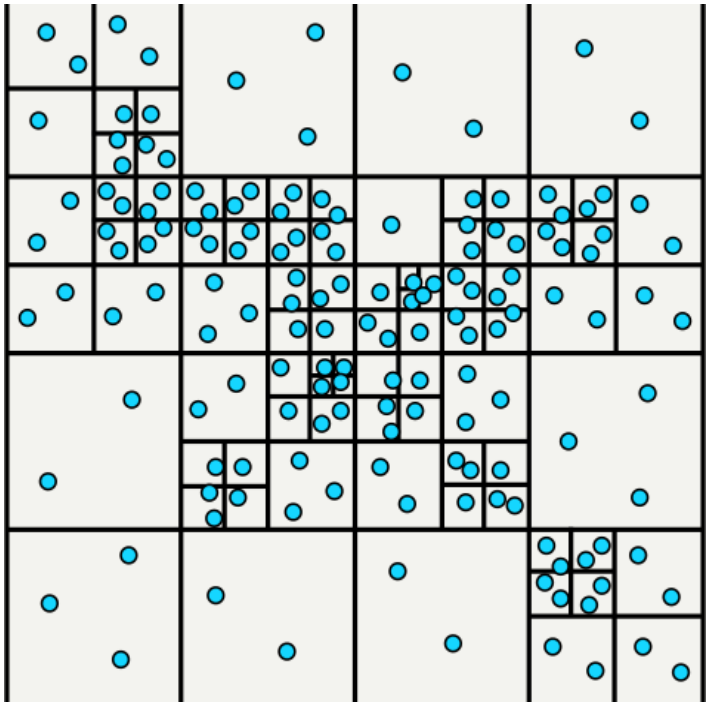
For a set of N (>10^{10}) particles, we want to exchange hydrodynamical forces between all neighbouring particles within a given (time and space variable) search radius. Large density imbalances develop over time.

Challenges:

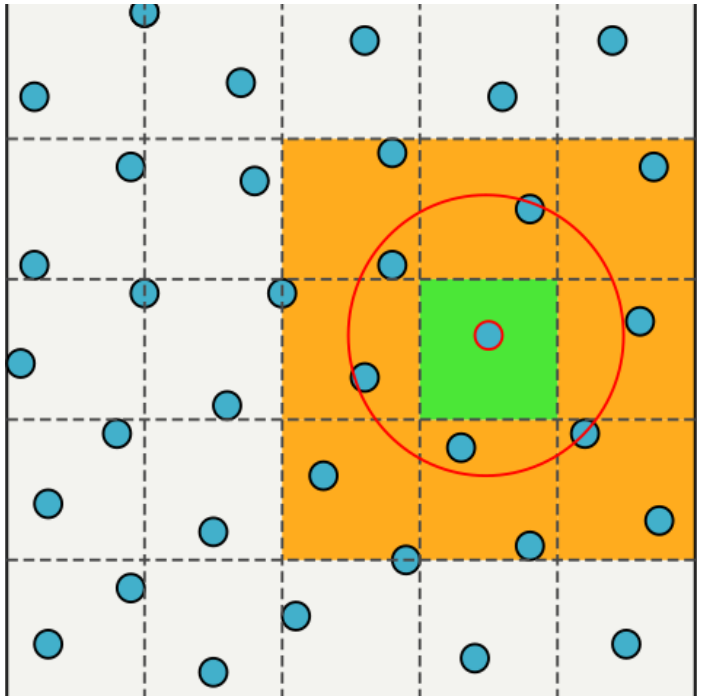
- Particles are unstructured in space, large density variations.
- Particles will move and the neighbour list of each particle evolves over time.
- Interaction between two particles is computationally cheap (low FLOP/byte).
- Individual time-steps for each particle.

SPH scheme: The traditional method

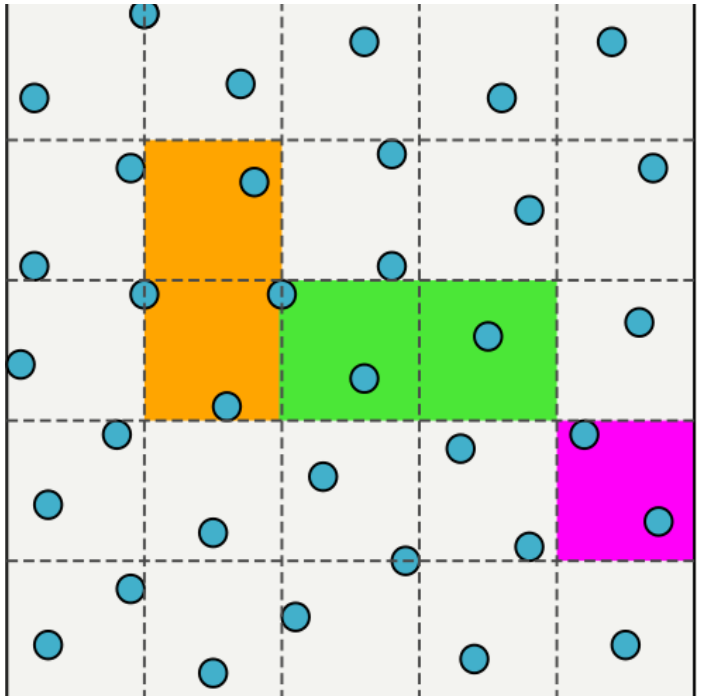
- “Industry standard” code is *Gadget* (Springel+2005).
- MPI-only code.
- Neighbour search based on oct-tree.
- Domain decomposition based on a space-filling curve.



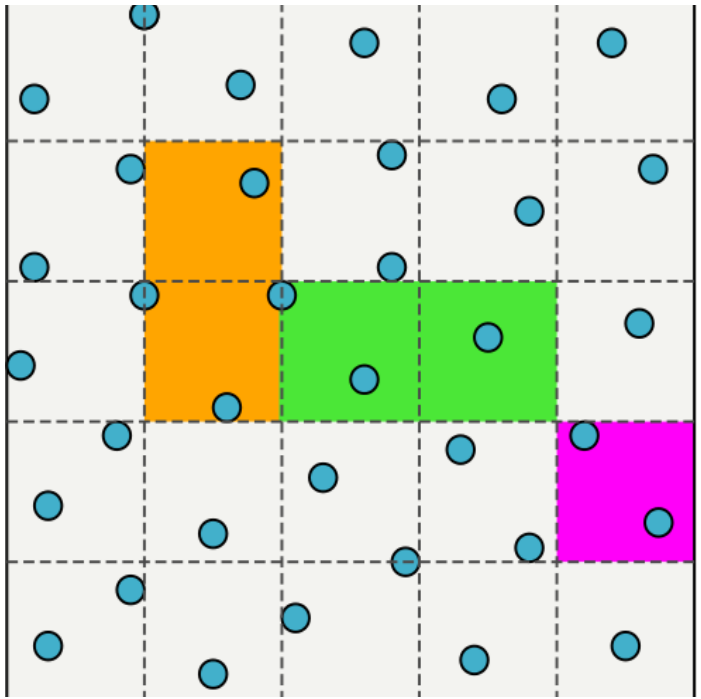
AMR Verlet-list neighbour search



- 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.



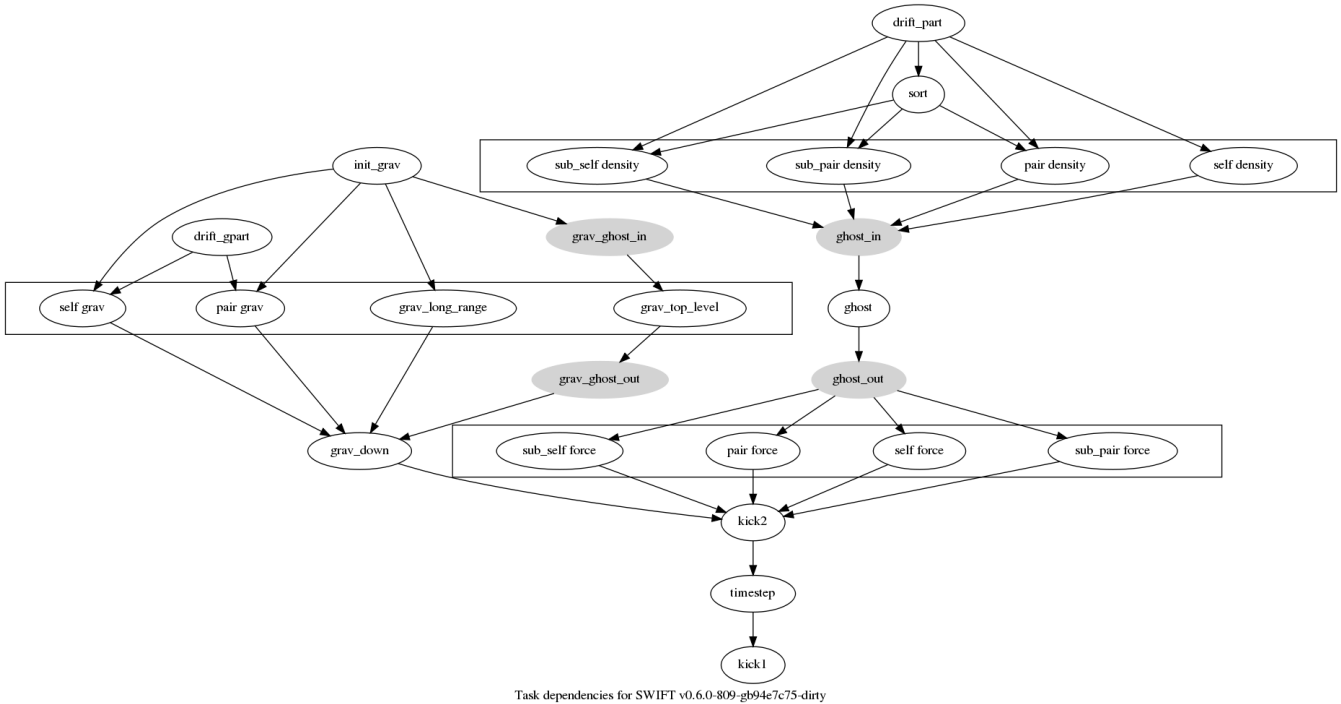
- Cells pairs do not need to be processed in any pre-defined order.
- Only need to make sure two threads do not work on the same cell.
- Cell pairs can have vastly different work-loads.



- 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 SPH



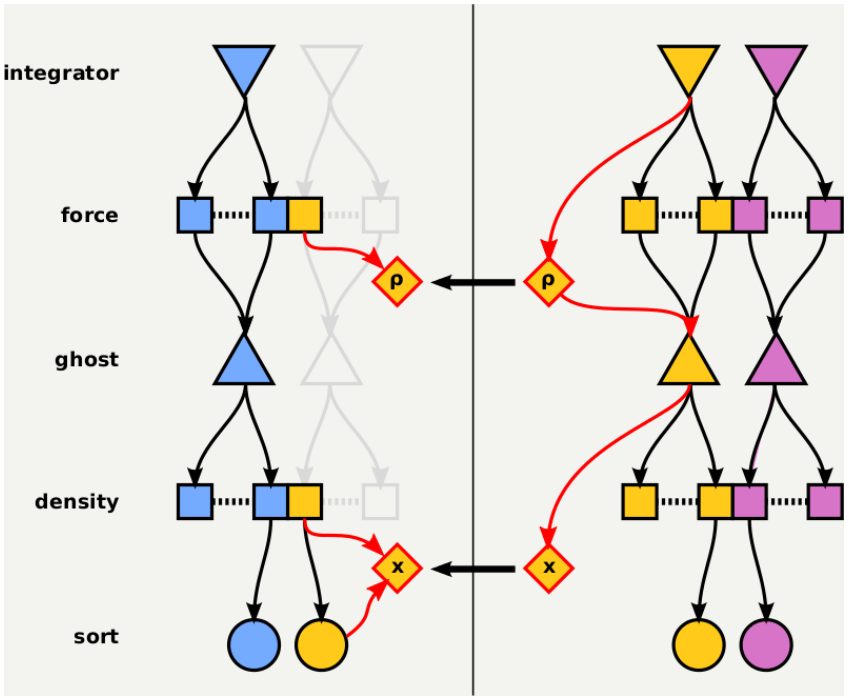
- Task-graph describing the science.
- Arrows depict dependencies.

Task-based parallelism in action

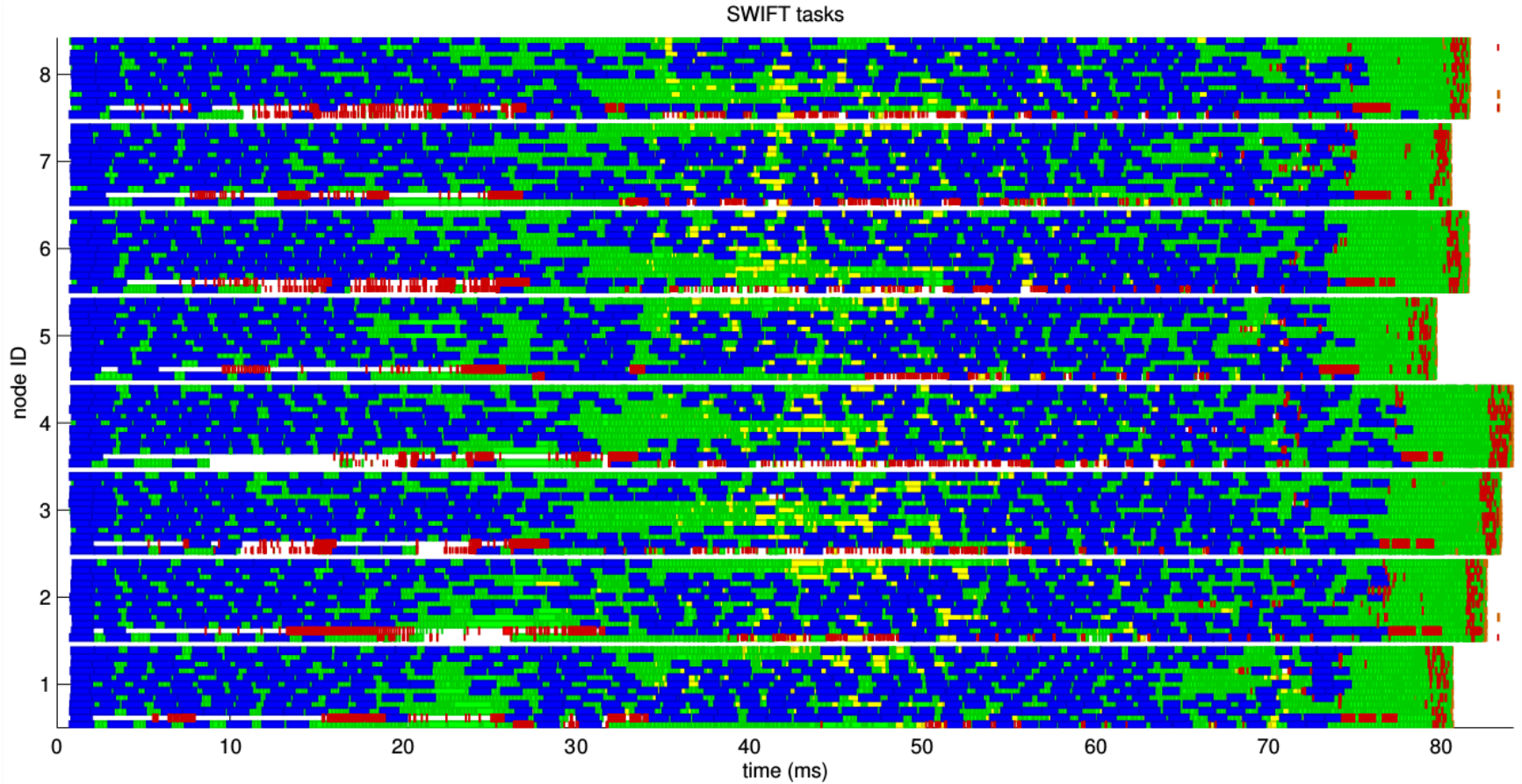
Task-graph for one time-step. Colours correspond to different task types.
Almost perfect load-balance achieved on 16 cores.

Multi-node strategy

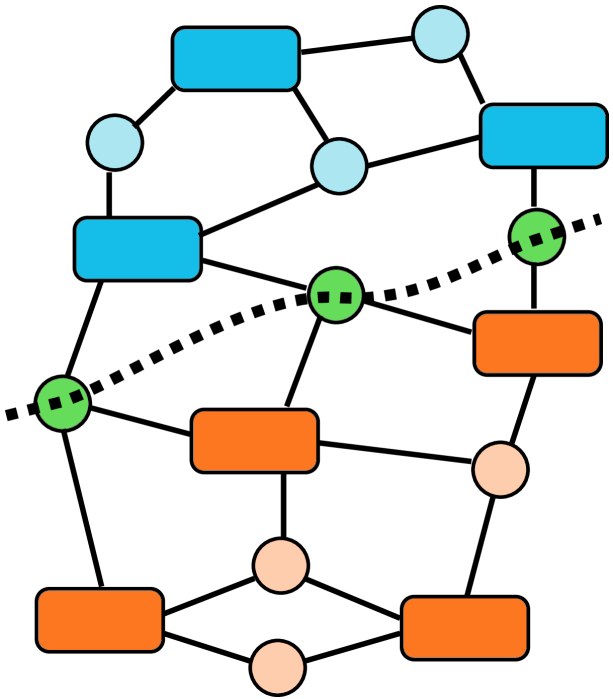
- A given MPI rank only needs cell pairs that are directly adjacent.
- Instead of sending everything and then processing, we send particles on a cell-by-cell basis using asynchronous MPI calls.
- Sending/ receiving is “just” another task within the graph and can be executed at the same time as other “local” operations.
- No global lock or barrier !!



Task plot for multiple nodes

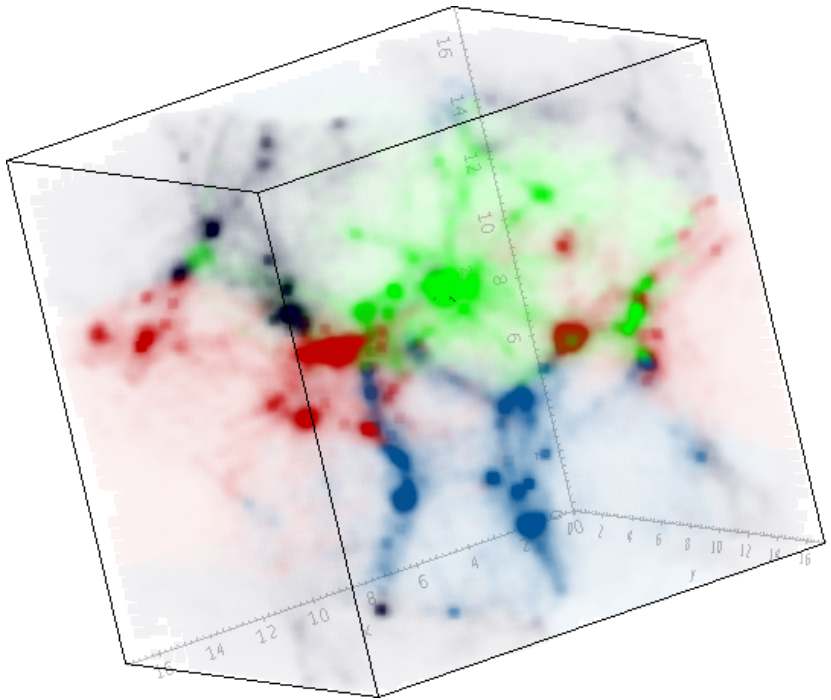


A Graph-based strategy



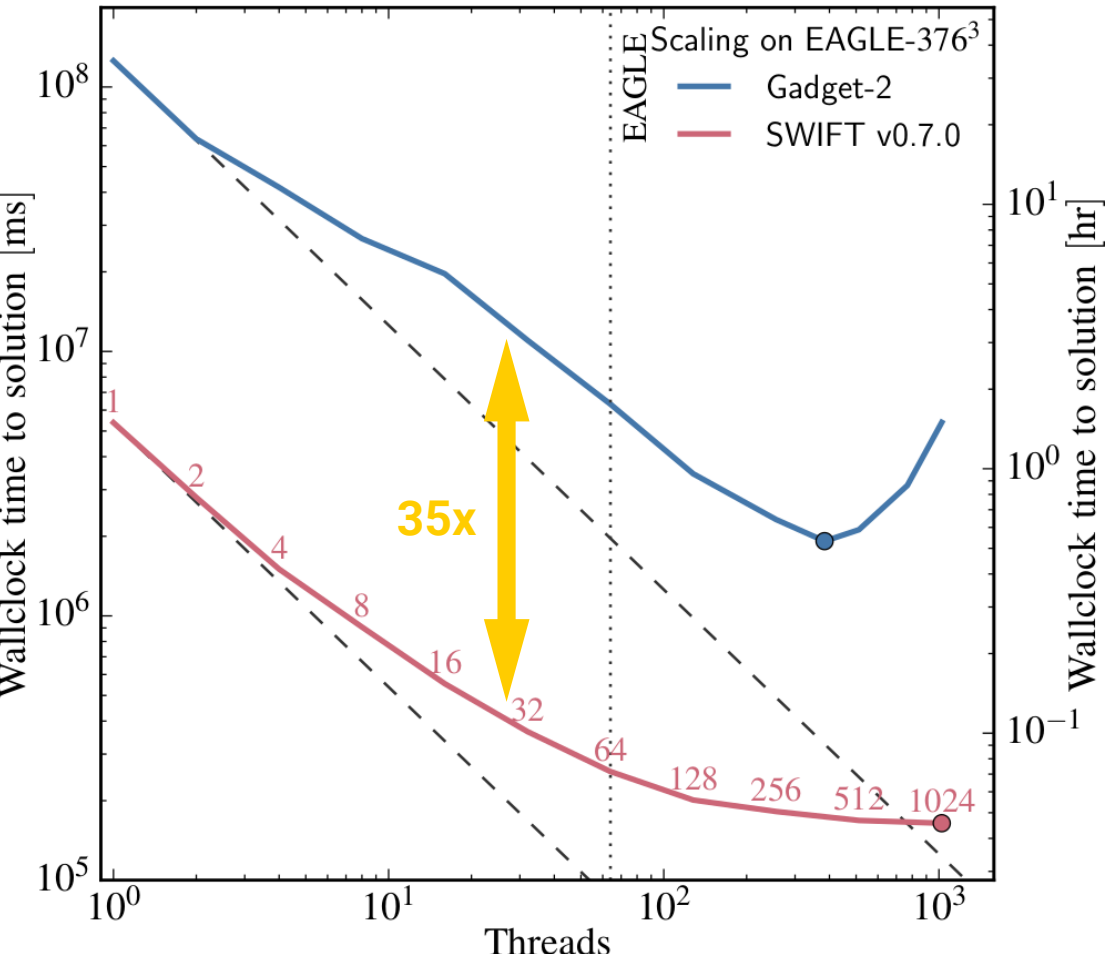
- For each task, we compute the amount of work (=runtime) required.
- We build a graph where the data are nodes and tasks are hyper-edges.
- ***METIS*** is used to split the graph such that the work (not the data!) is balanced.
- Extra cost added for communication tasks to minimise their number.

What does it look like?



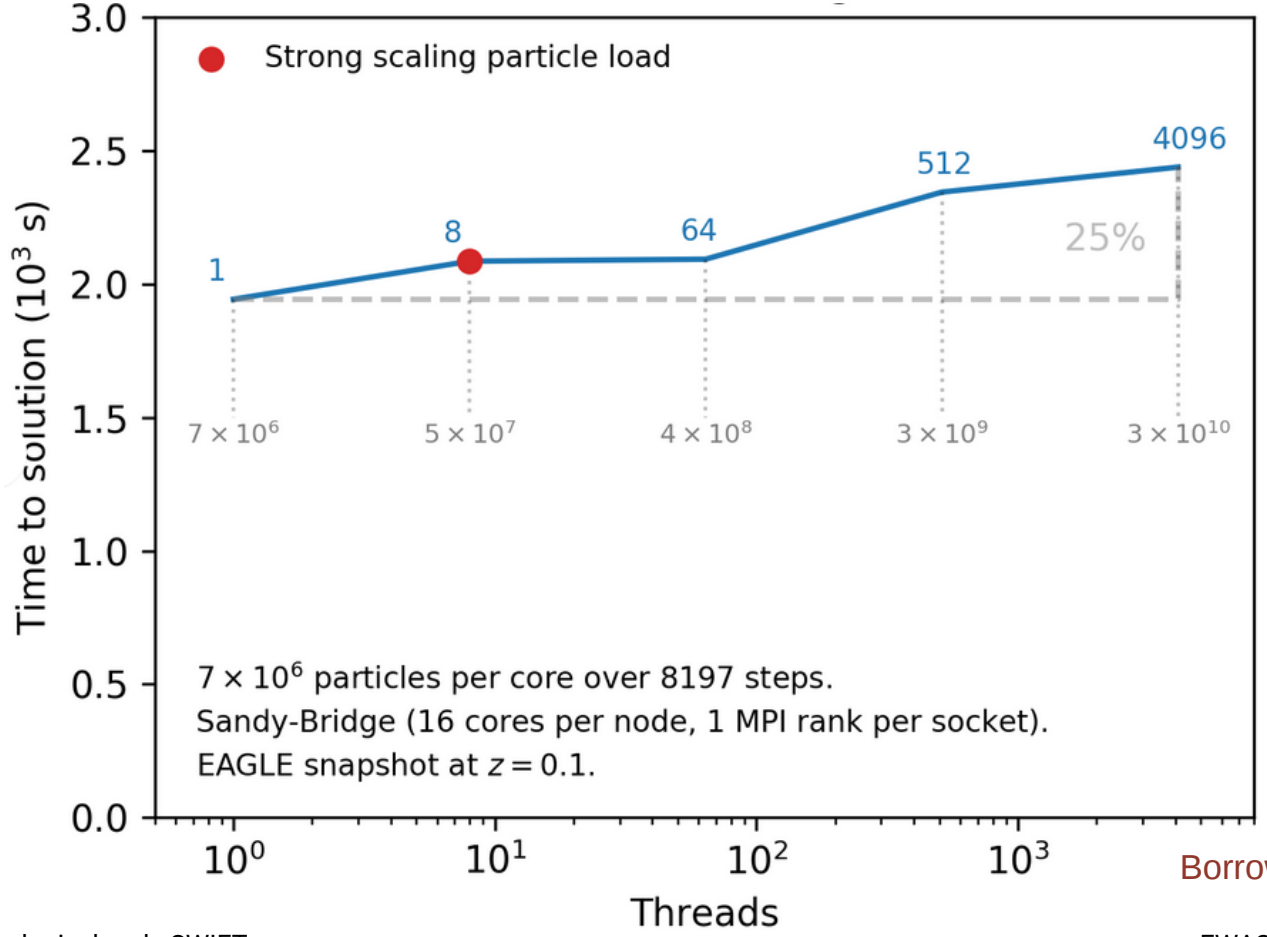
- No regular grid pattern.
- No space-filling curve pattern.
- Good (work) load-balancing by construction.
- The most dense regions are at the centre of their respective domains.

Performance vs. Gadget



- Realistic problem
- Same accuracy.
- Same hardware.
- Same compiler.
- Same solution.

Sustained performance in weak-scaling



Borrow+ (in prep.)

Conclusions

- New algorithms can lead to significant speed-ups over conventional methods. Implicit methods vs. explicit local time-step based solvers.

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

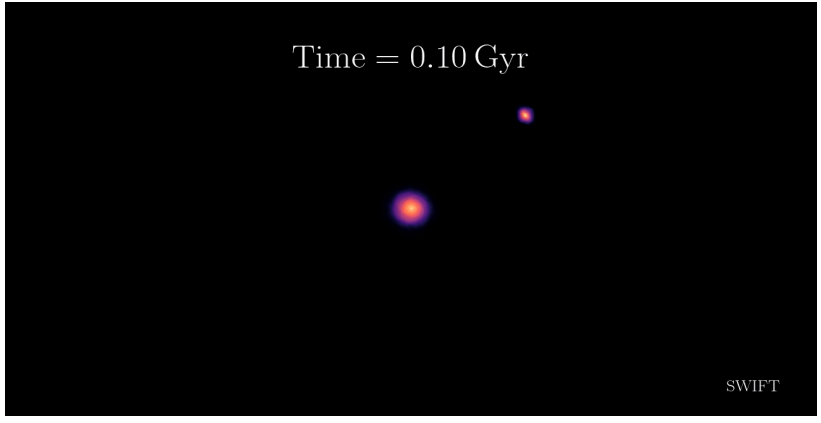
- Task-based parallelism as a viable model for actual scientific applications.

  → Not just a research concept.

- New open-source applications based on an expendable framework, ready for the community to use.

@SwiftSimulation

www.swiftsim.com



See you at the Hack Day on Friday in room 6!

Hausammann, Revaz, Schaller