SWIFT, EAGLE-XL, and the future of Computational Cosmology

Josh Borrow

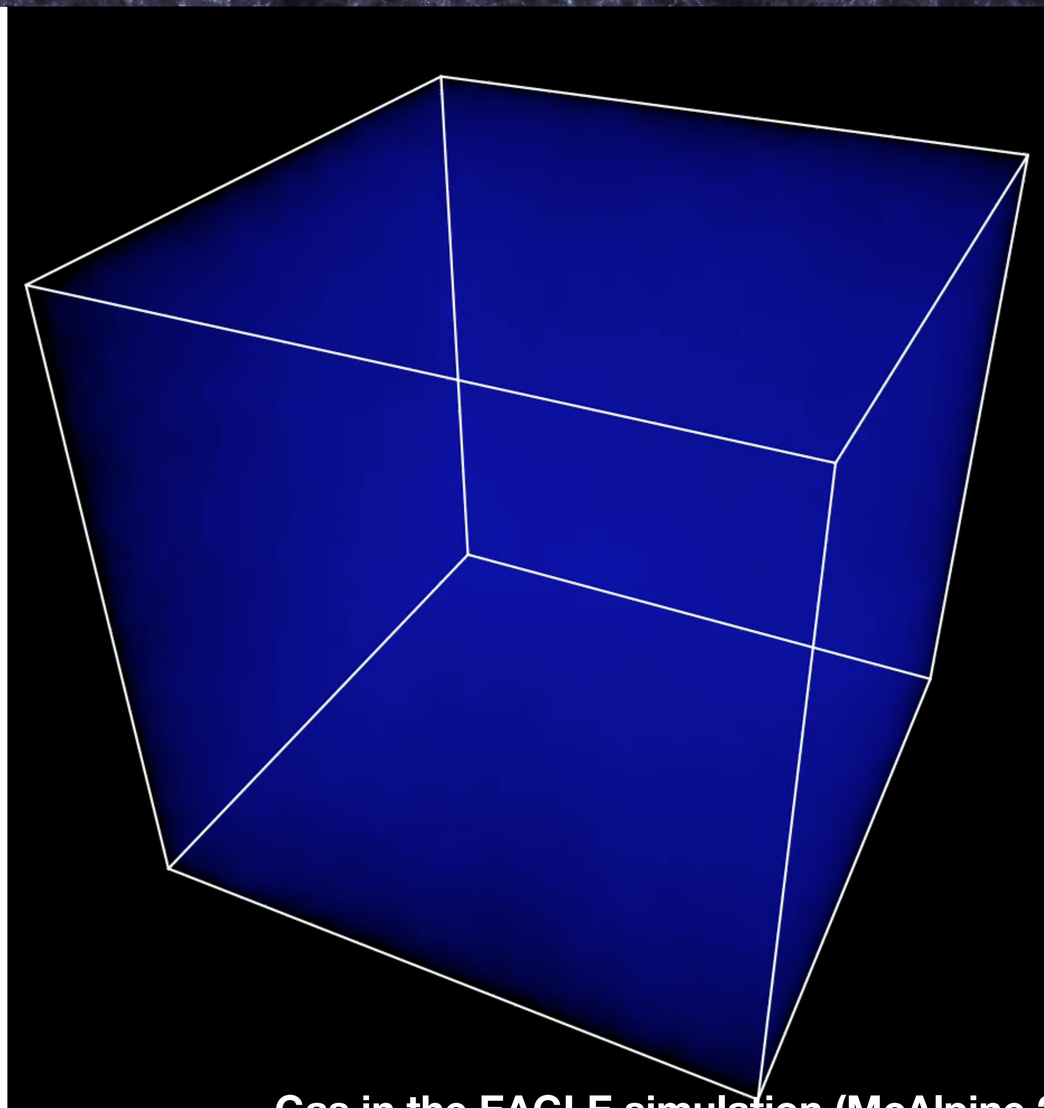
ICC, Durham

& Matthieu Schaller, Richard Bower, Peter Draper, Pedro Gonnet, James Willis, Jacob Keggeris, Alexei Borrisov, …

LJMU ARI | 23rd November 2018

Overview

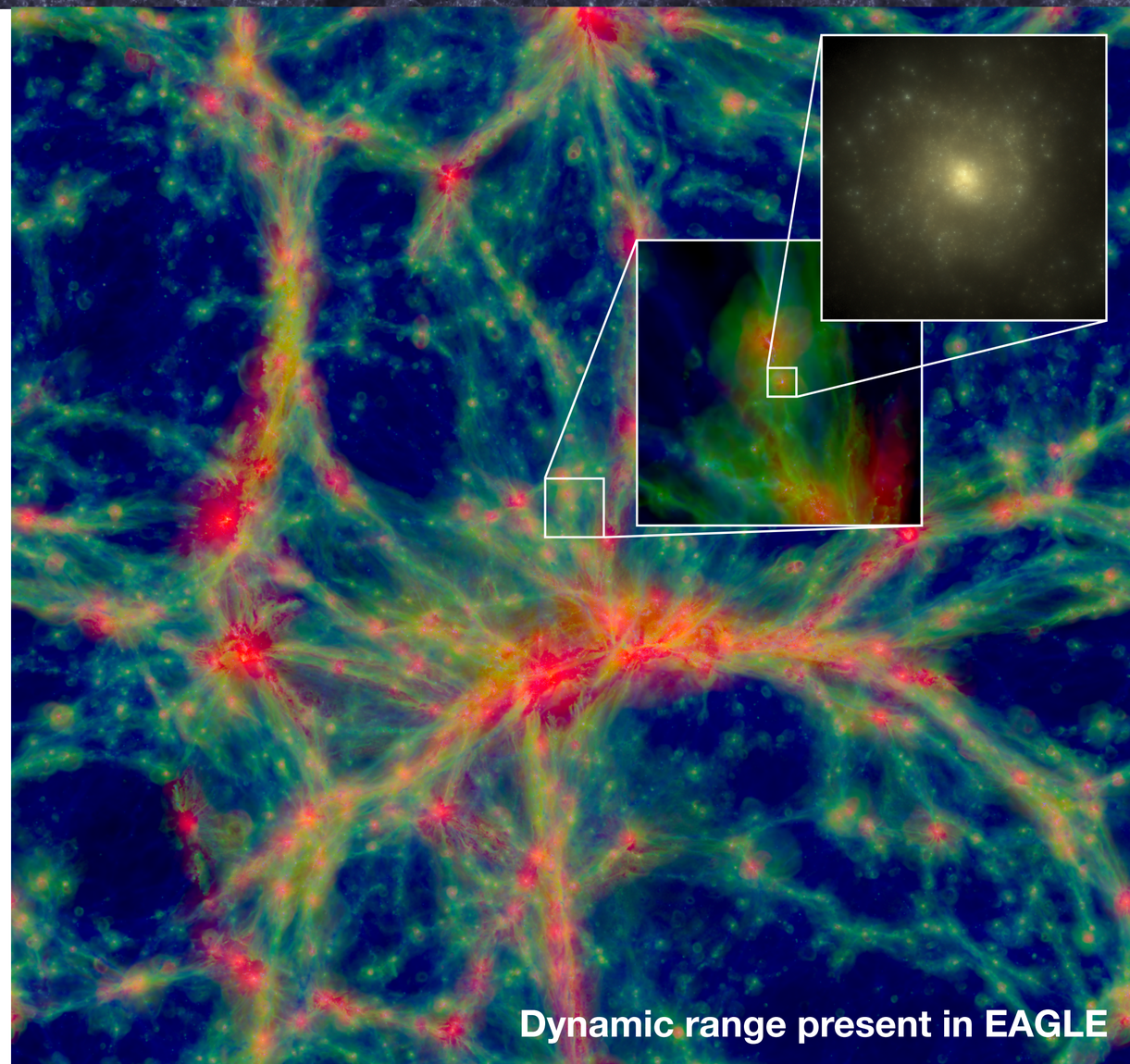
- What is EAGLE?
- Our motivation for SWIFT: EAGLE-XL
- What is SWIFT?
- Performance enhancements and where to find them
- Current status of the project



Gas in the EAGLE simulation (McAlpine 2017)

What is (was) EAGLE?

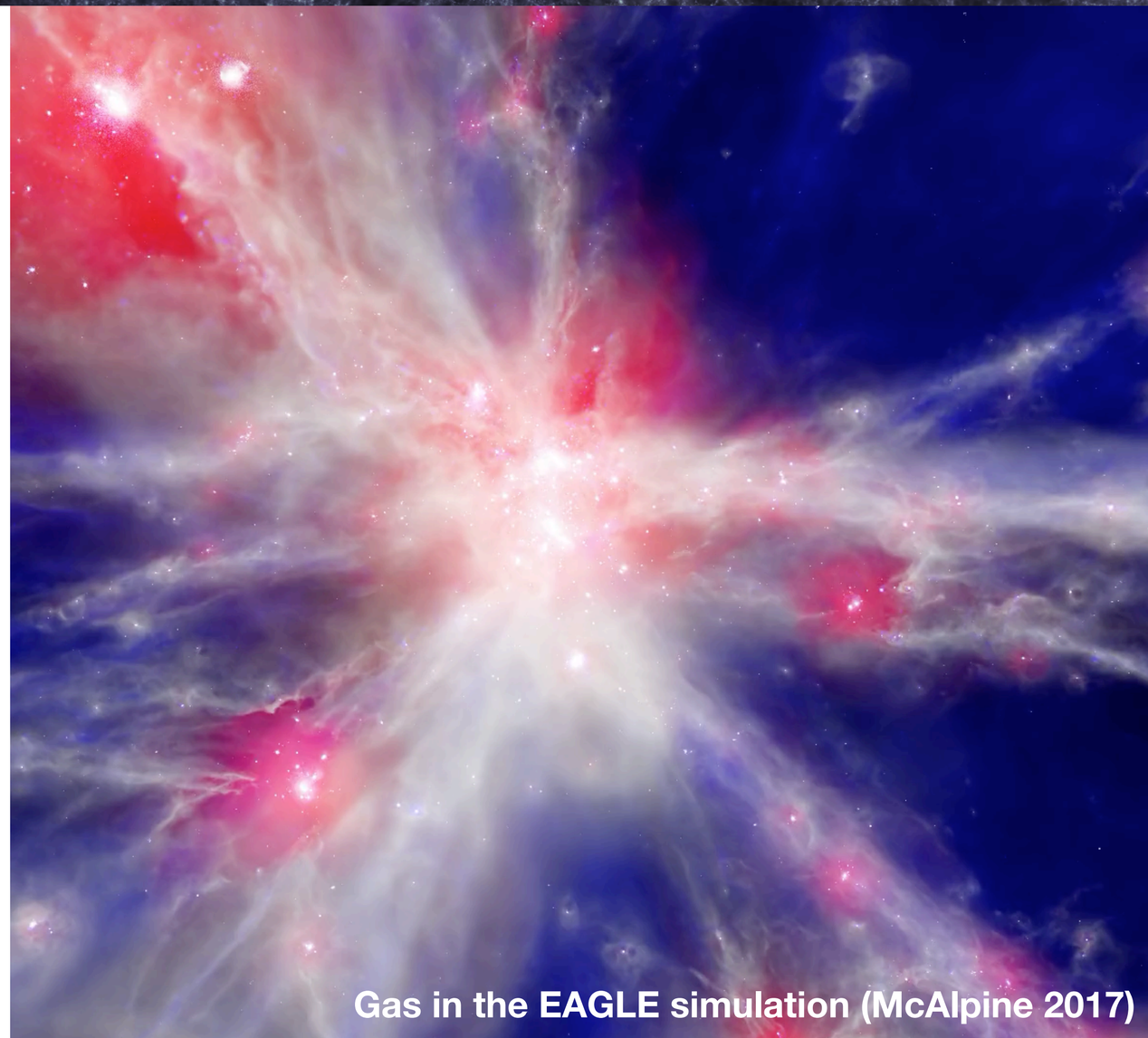
- Cosmological simulation
- Mass resolution 10^6 solar masses
- Spatial resolution ~kpc
- Allows us to study how galaxies form and evolve



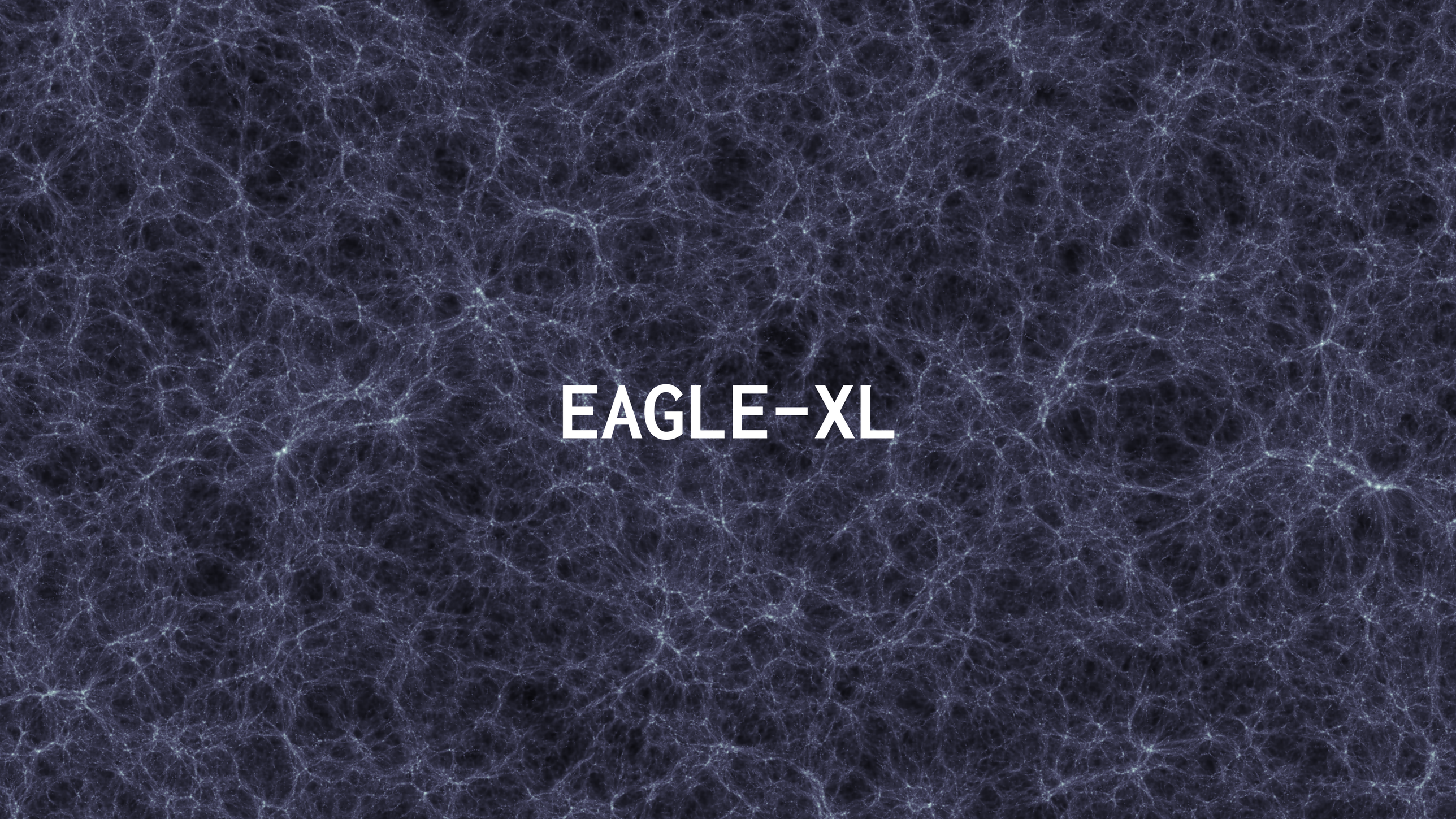
Dynamic range present in EAGLE

What is (was) EAGLE?

- Include sub-grid physics:
- Cooling
- Star formation and feedback
- Black holes and AGN

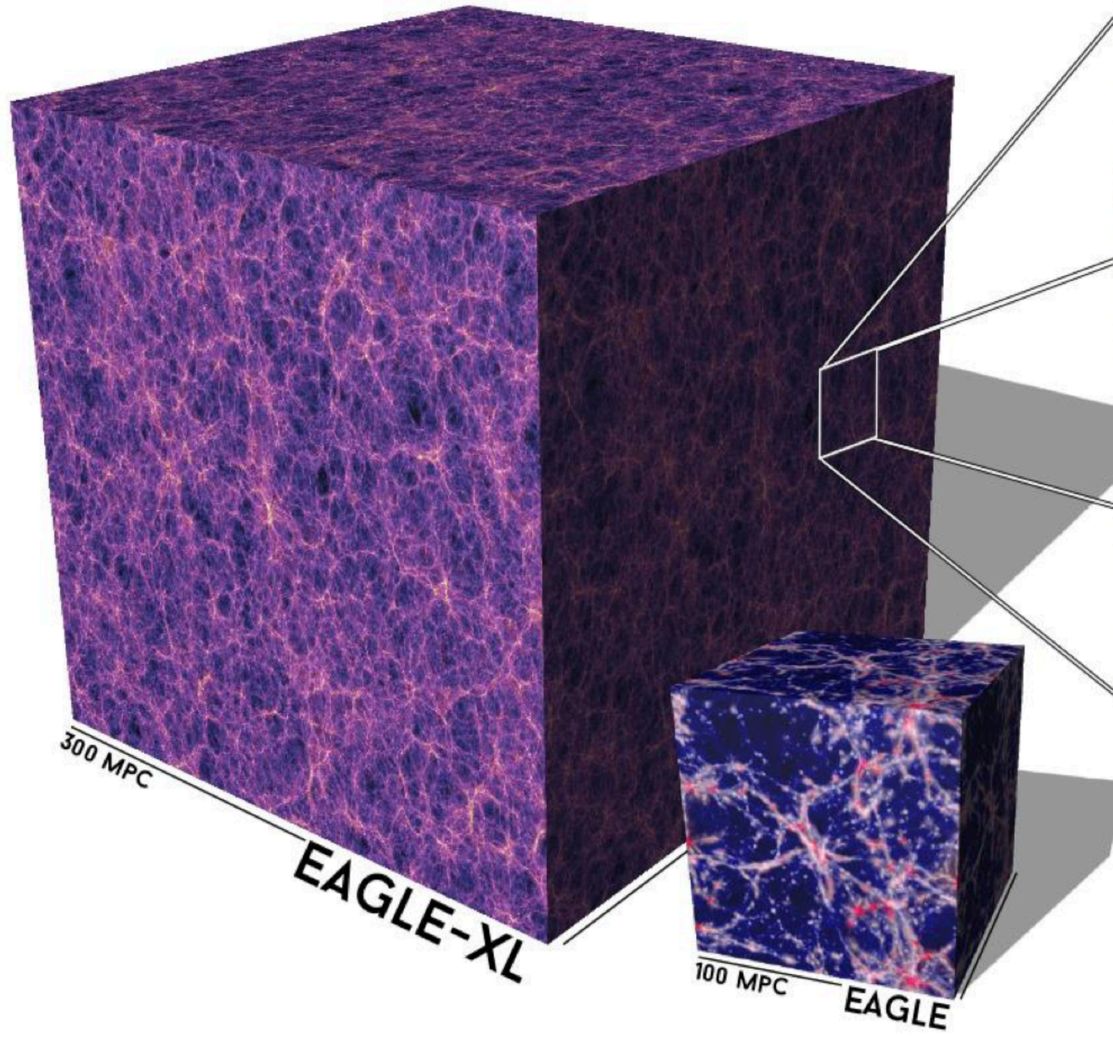


Gas in the EAGLE simulation (McAlpine 2017)


EAGLE-XL

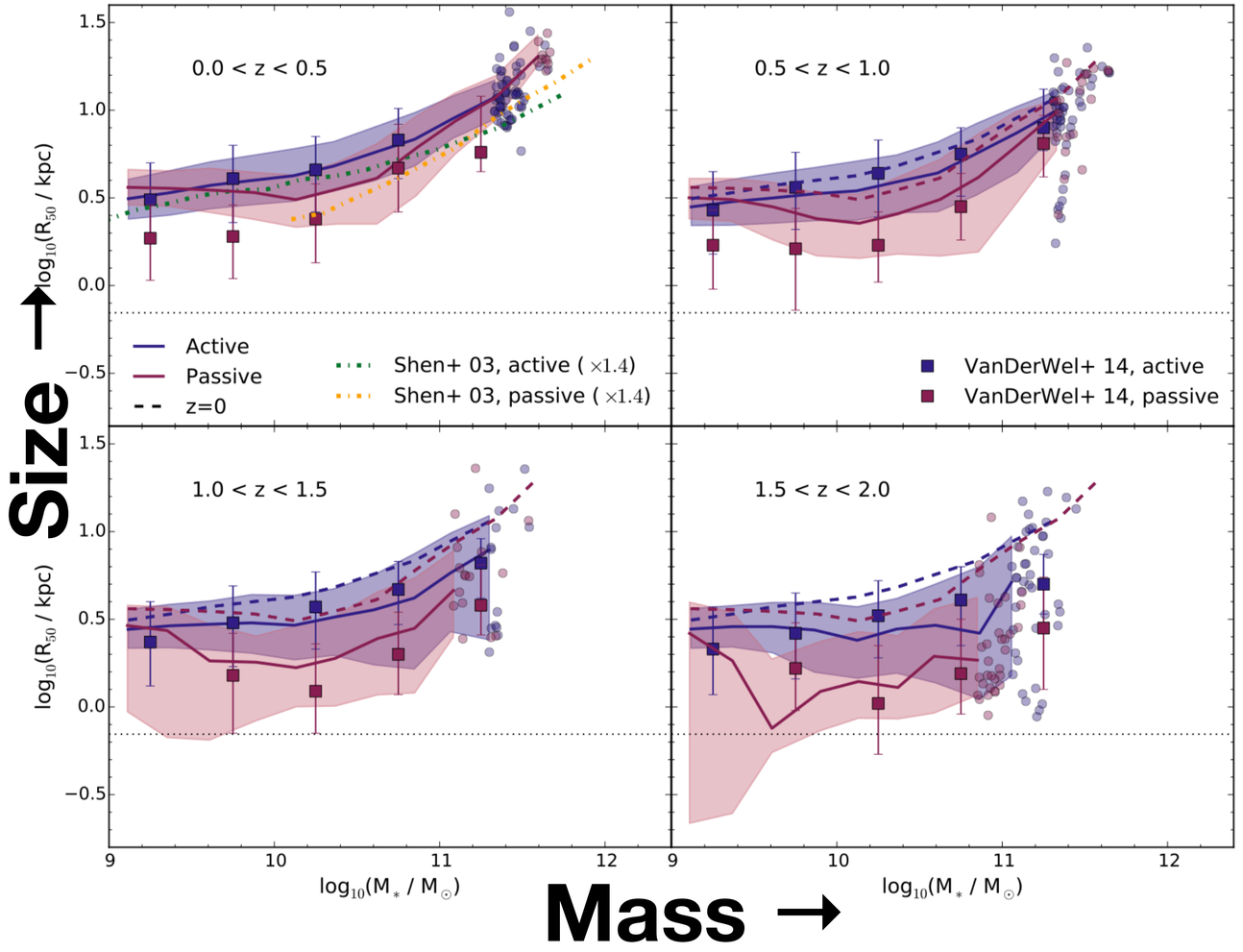
EAGLE-XL

- Same resolution as original EAGLE
- Baryonic particles with mass 10^6 solar masses
- 3^3 times the volume
- ~91 billion gas particles.



Why?

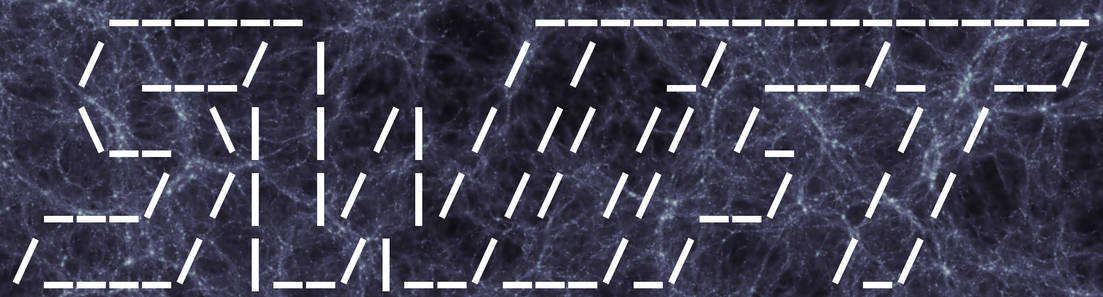
- For me? Statistics!
- 27x larger volume, 27x more objects.
- 10 SMGs in EAGLE-100
- ~300 SMGs in EAGLE-XL!



Mass-Size relation in EAGLE, Furlong+, 2016

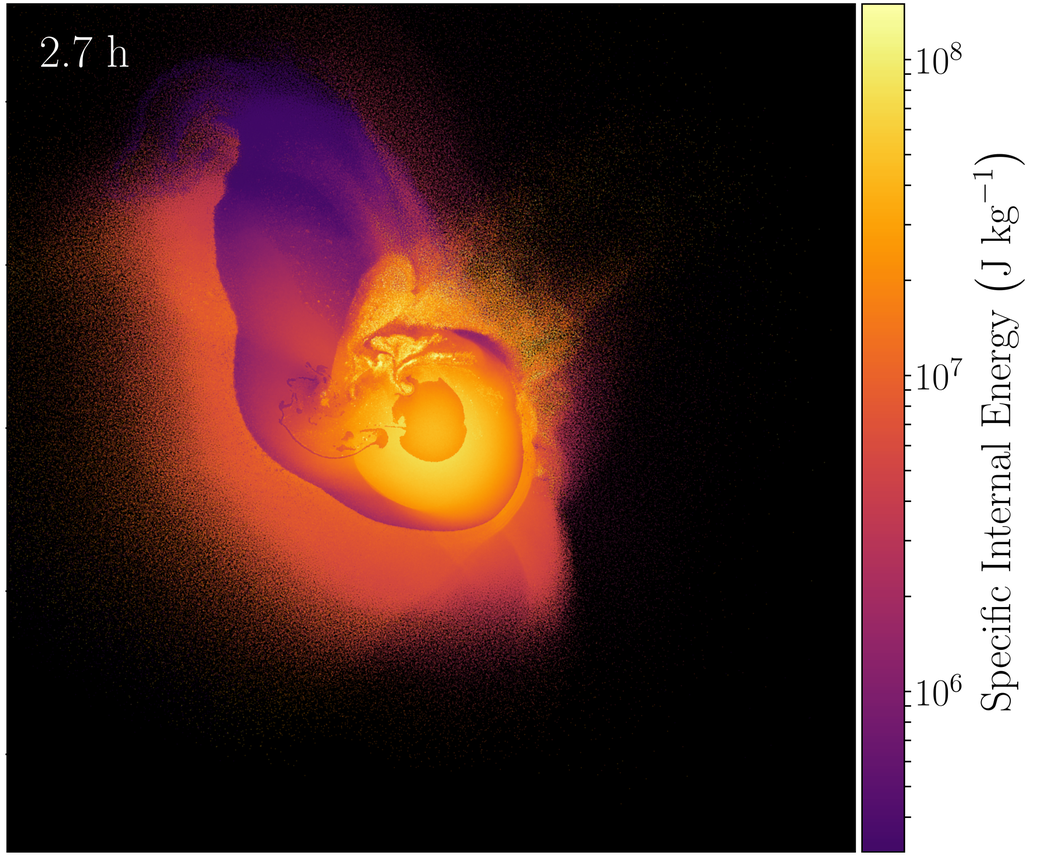
How?

- Original EAGLE took 2 months on 4096 cores
- GADGET-3 does not scale further
  - MPI requires too much memory overhead
  - Imbalance of communication
- This is *impossible* with the GADGET-based code



What is SWIFT?

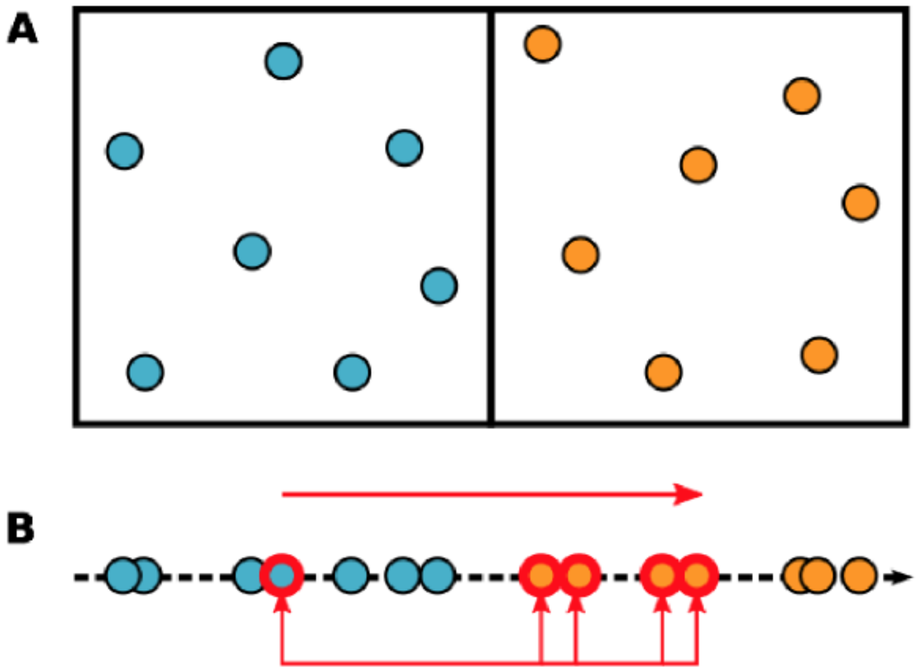
- SWIFT is a new cosmological simulation code
- Particle-based hydrodynamics (SPH, GIZMO)
- Fast Multipole Method (FMM) for gravity (O(N) v.s. O(NlogN) for Barnes-Hut)
- Streaming i/o



The highest resolution planetary impacts simulation ever performed, made possible with SWIFT
Kegerreis+, 2019

Start again

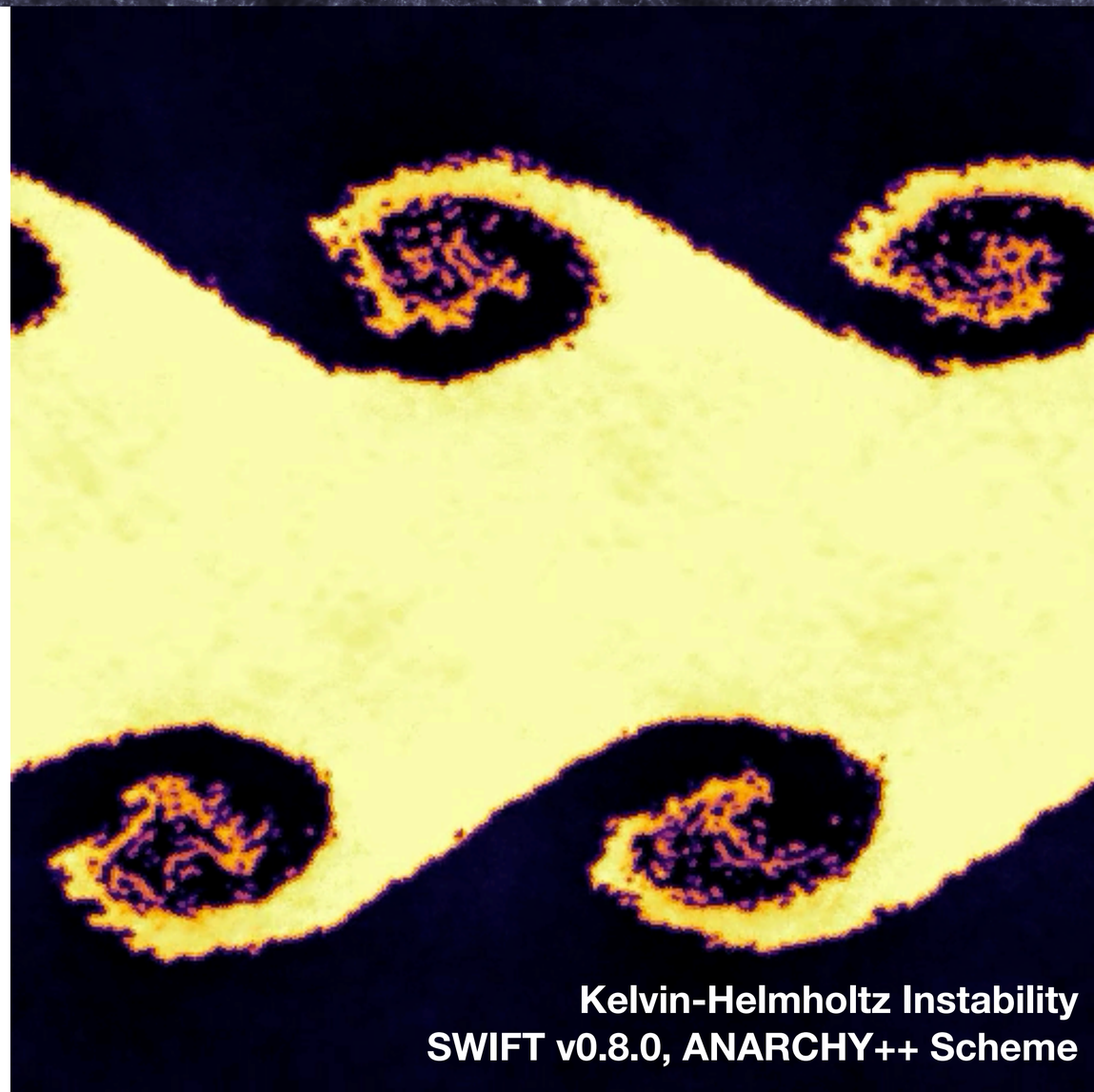
- Throw out *everything* and start from scratch
- 6 years ago, Matthieu Schaller (ICC) & Pedro Gonnet (CS) started work on SWIFT
- New algorithms, new data structures, and new interface



Sorted cell interaction in SWIFT
Gonnet+, SPHERIC, 2013

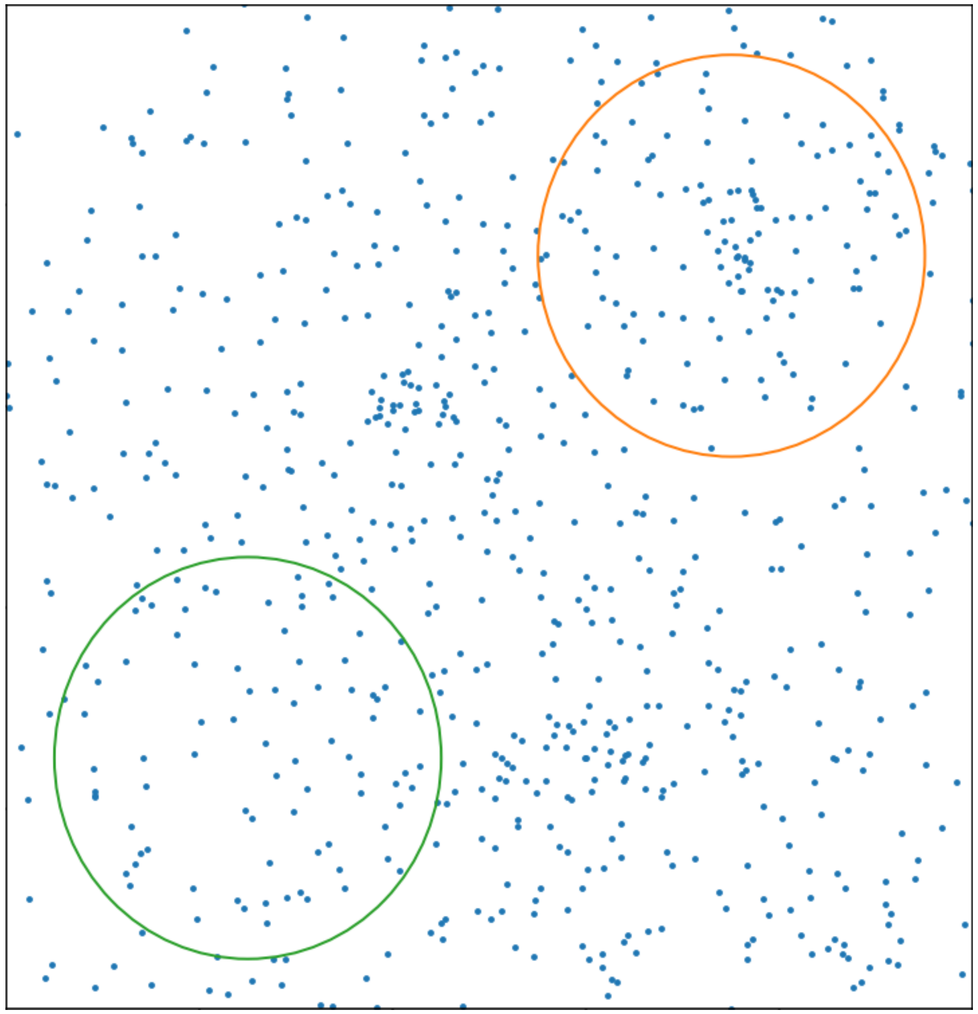
Start with hydrodynamics

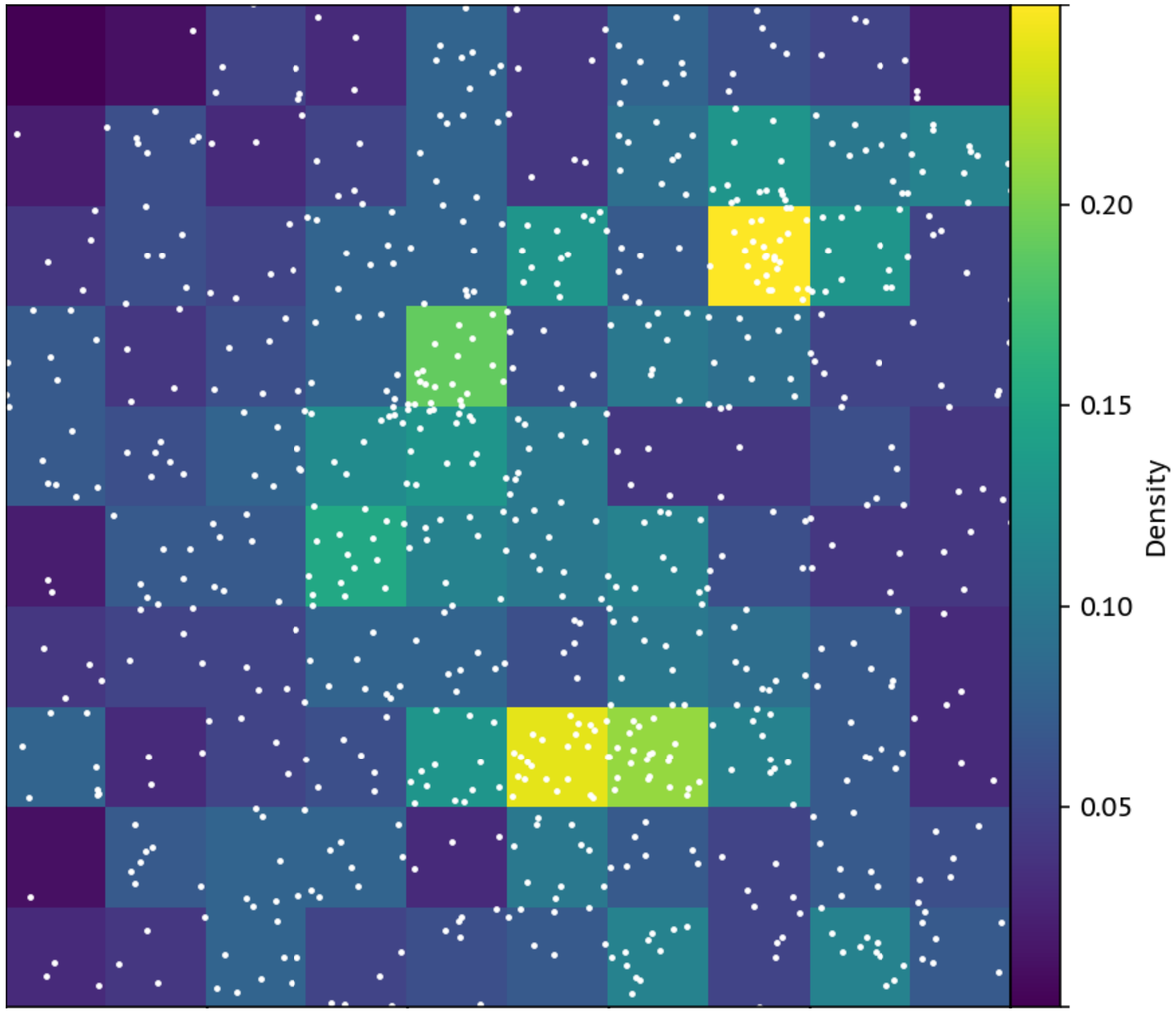
- GADGET: hydro added onto gravity code
- Hydrodynamics typically takes up the majority of computation time
- Particle-based code - Smoothed Particle Hydrodyamics (SPH)
- Allows fluid elements to be tracked over time



Kelvin-Helmholtz Instability
SWIFT v0.8.0, ANARCHY++ Scheme

Short introduction to SPH

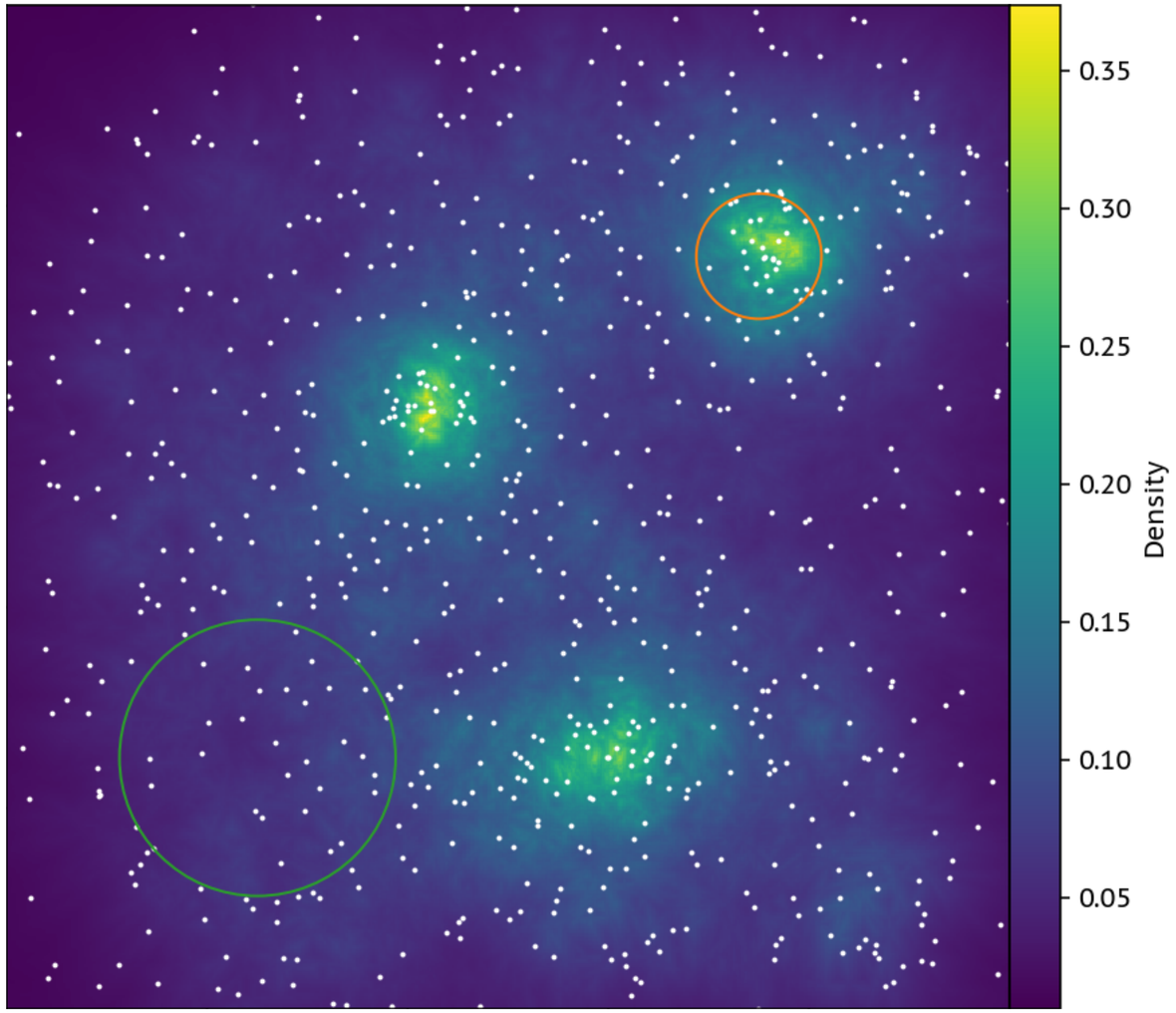




Short introduction to SPH

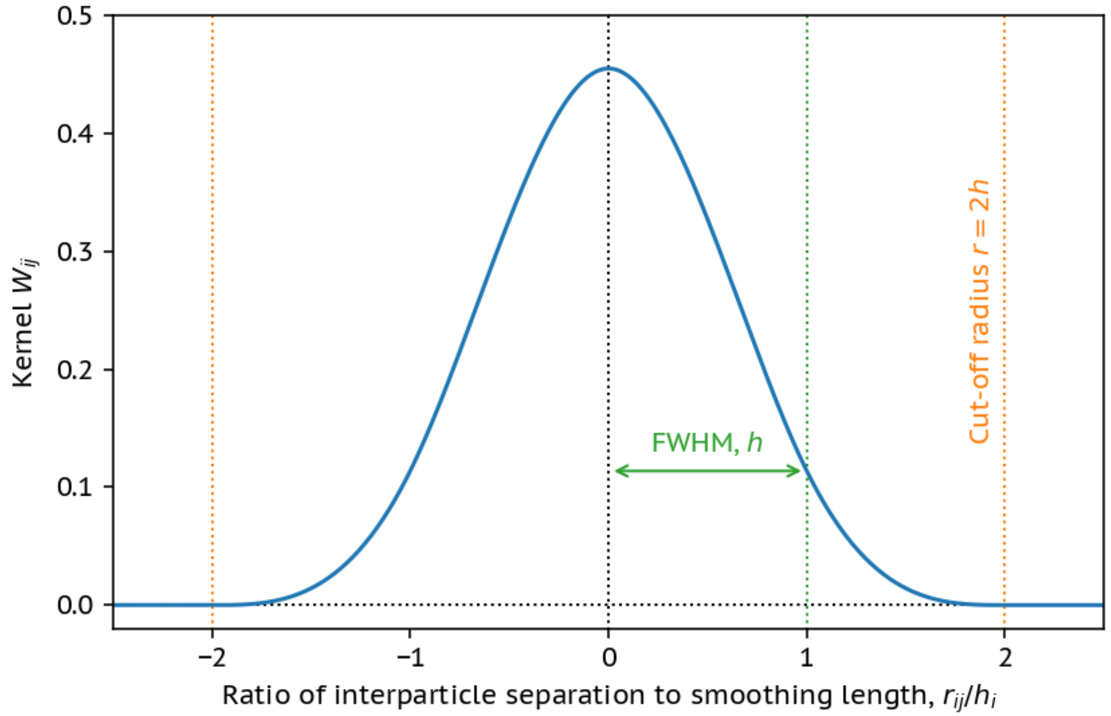
- What if we change the volume we consider for each particle?
- Set *R* such that our volume encloses 30 particles.
- Density for each particle is then

$$\rho_i = \frac{30 m_{\mathrm{part}}}{\frac{4}{3}\pi R_i^3}$$



Short introduction to SPH

- We can do better! We care less about particles that are further away.
- Weight the contribution from each particle with a “kernel”.
- Size kernel to enclose 30 particles within cut-off radius.

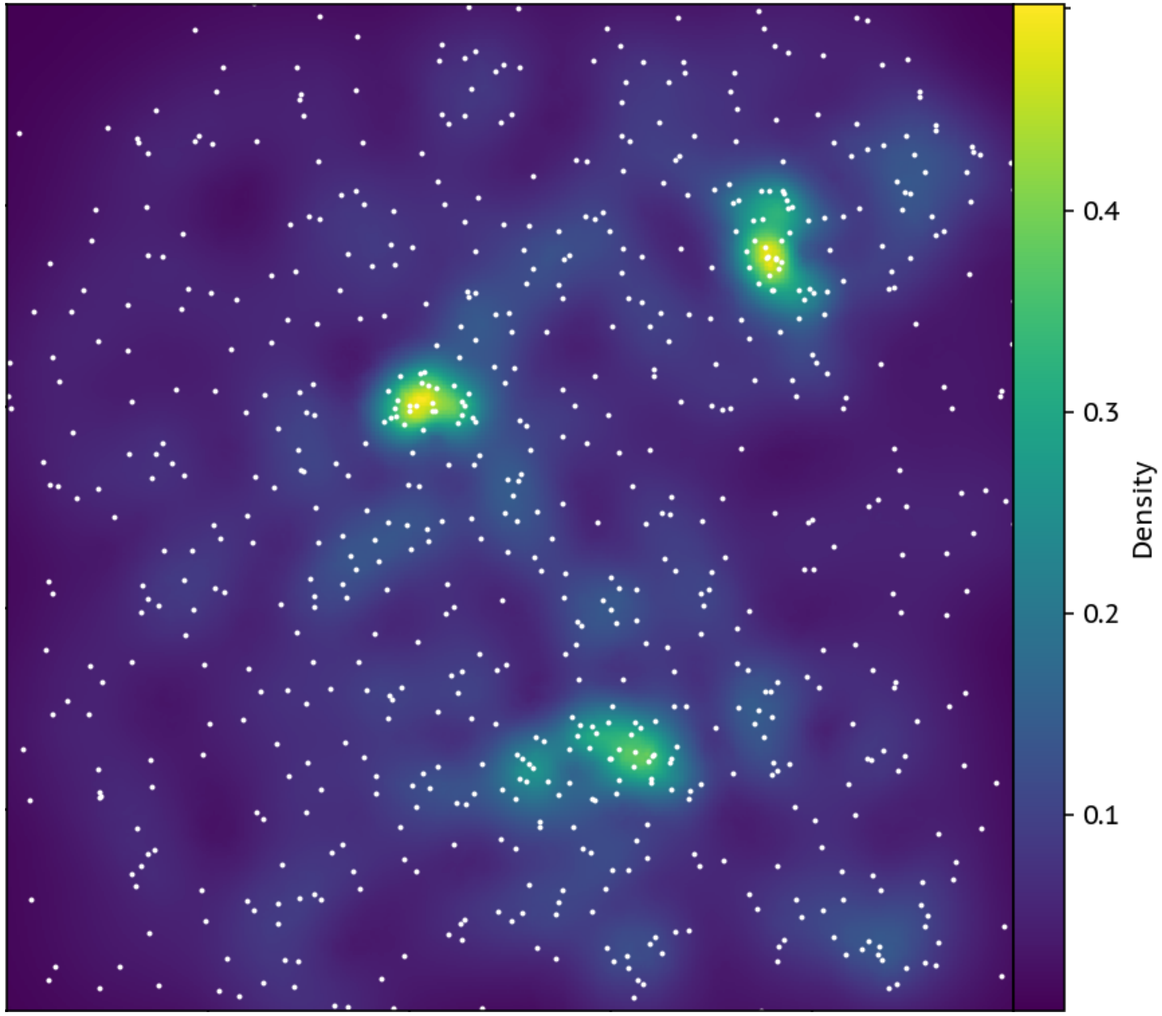


Short introduction to SPH

- Density for each particle now given by:

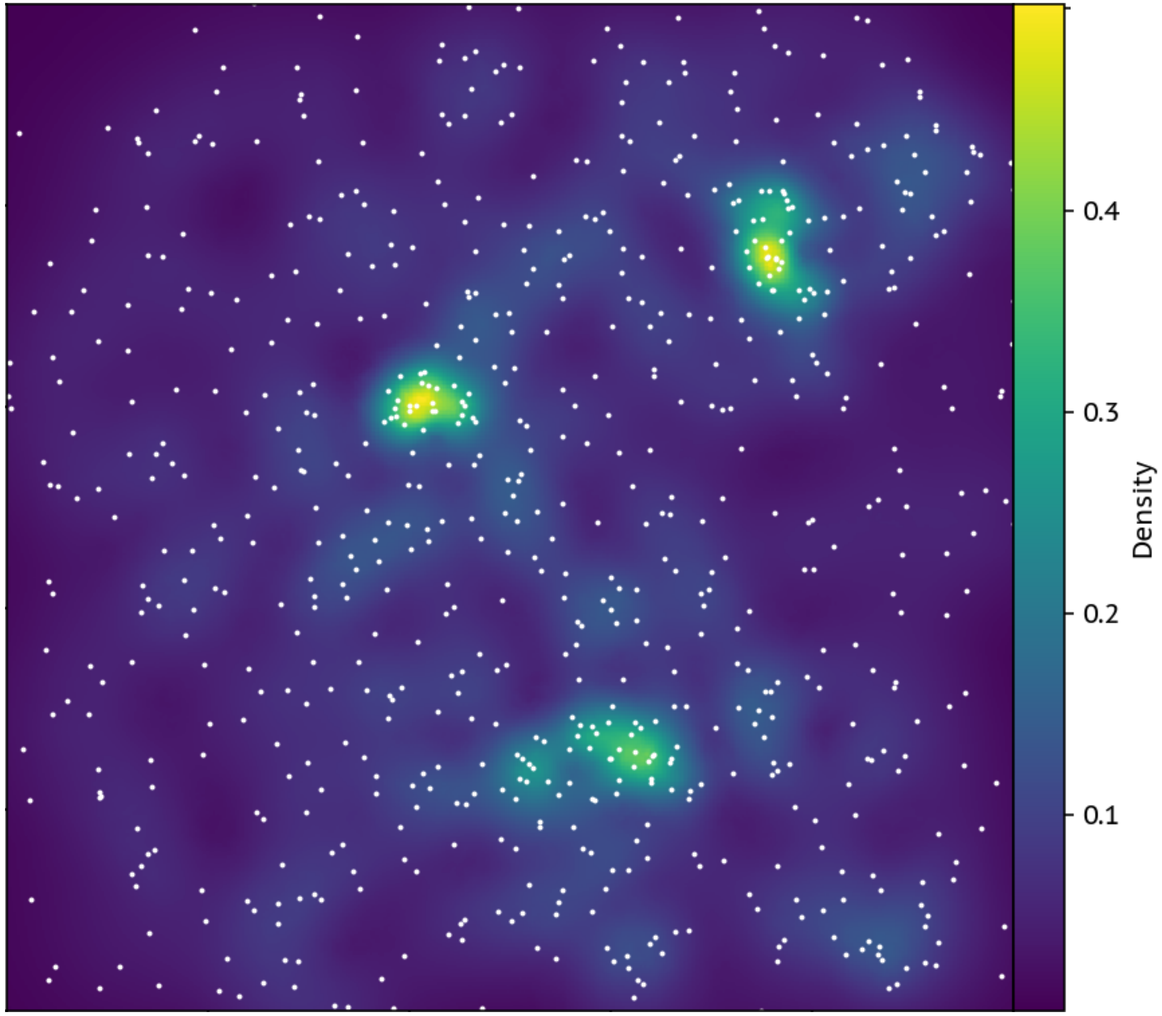
$$\rho_i = \sum_{j=1}^{30} m_{\mathrm{part}} W(r_{ij}, h_i)$$

- Density (+ temperature, tracked by particles) gives pressure, which gives dynamics



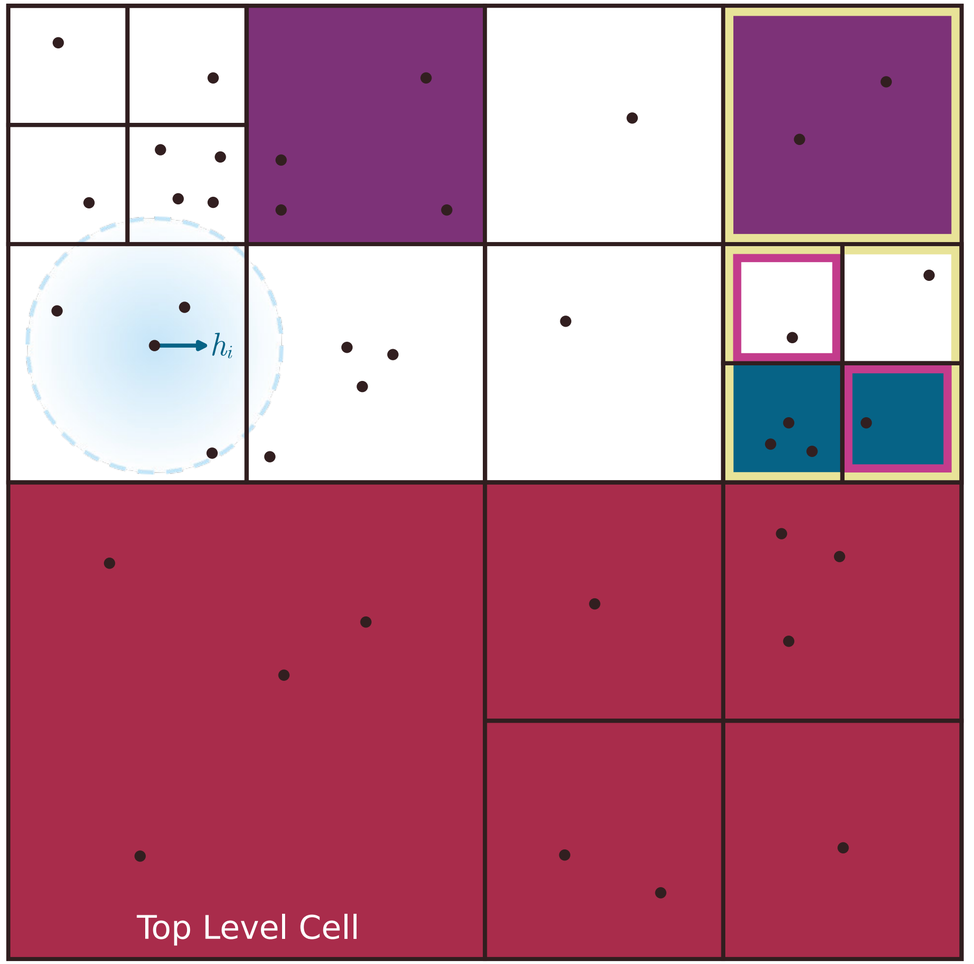
What do we need to know?

- Need to know the neighbours of all particles at each time-step
- Need to be able to compute the interactions between particles
- Need to be able to effectively load-balance such an operation



Enter the pseudo-verlet list

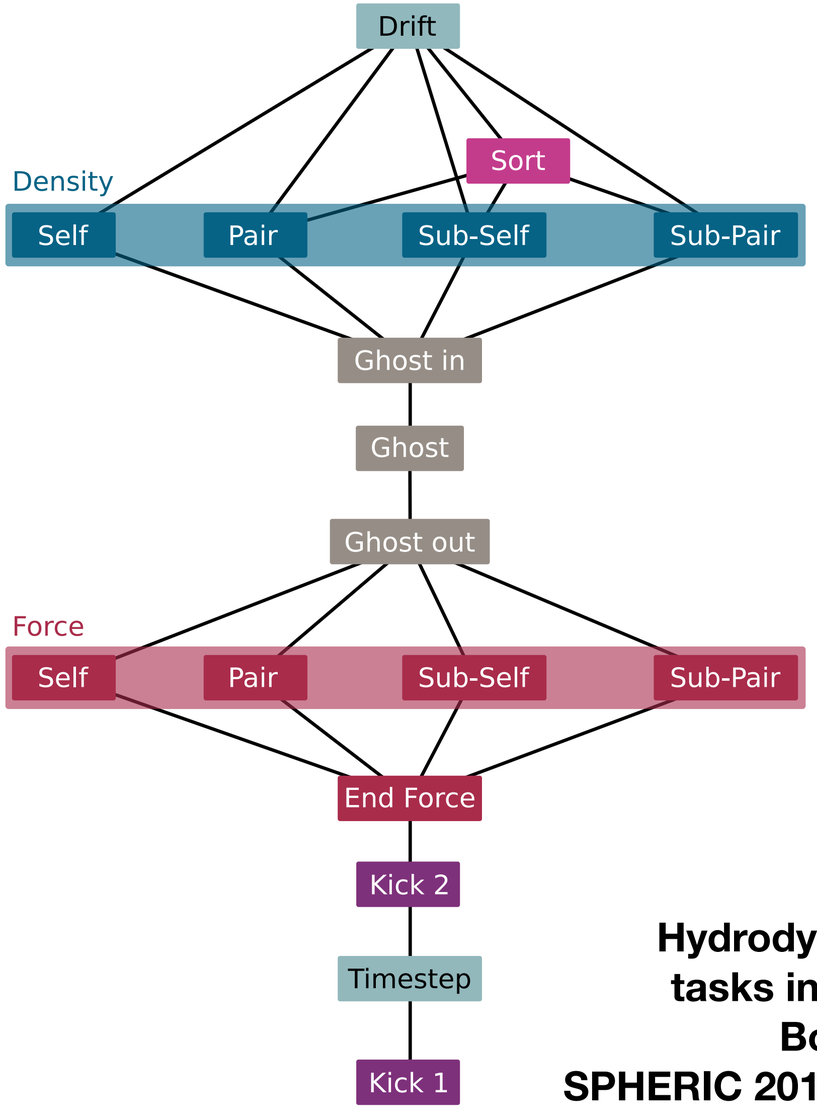
- Construct an oct-tree, as in other codes
- This time, though, group particles into cells
- Cells contain up to 400 particles
- Refine the tree until this criterion is satisfied
- We now know “everything” - no “tree walk”



Cell mesh construction in SWIFT
Borrow+, SPHERIC 2018, 2018

Spliting work into discrete chunks

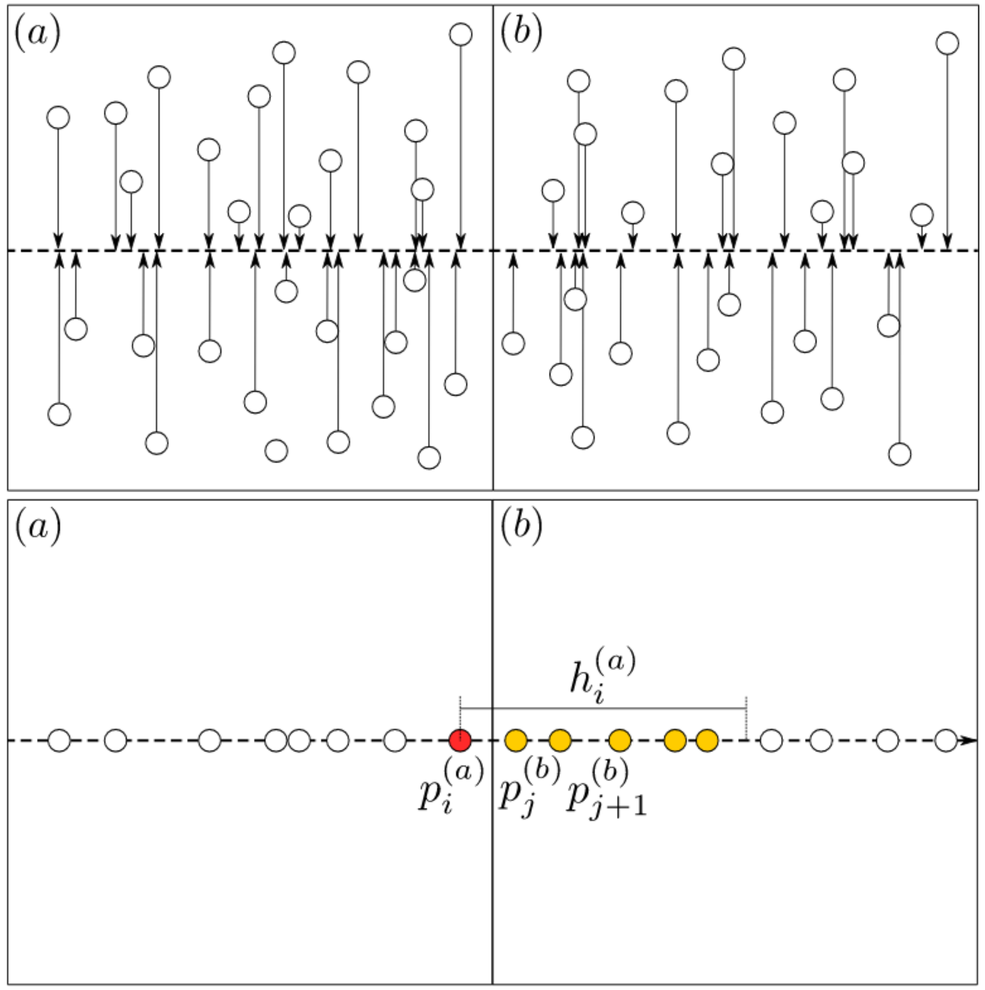
- SWIFT uses task-based parallelism
- Each thread runs something different
- Computation is put onto a queue that is picked up by threads that are free
- Dependencies and conflicts are imposed to ensure tasks are performed globally in the correct order



Hydrodynamics tasks in SWIFT
Borrow+, SPHERIC 2018, 2018

Interacting particles

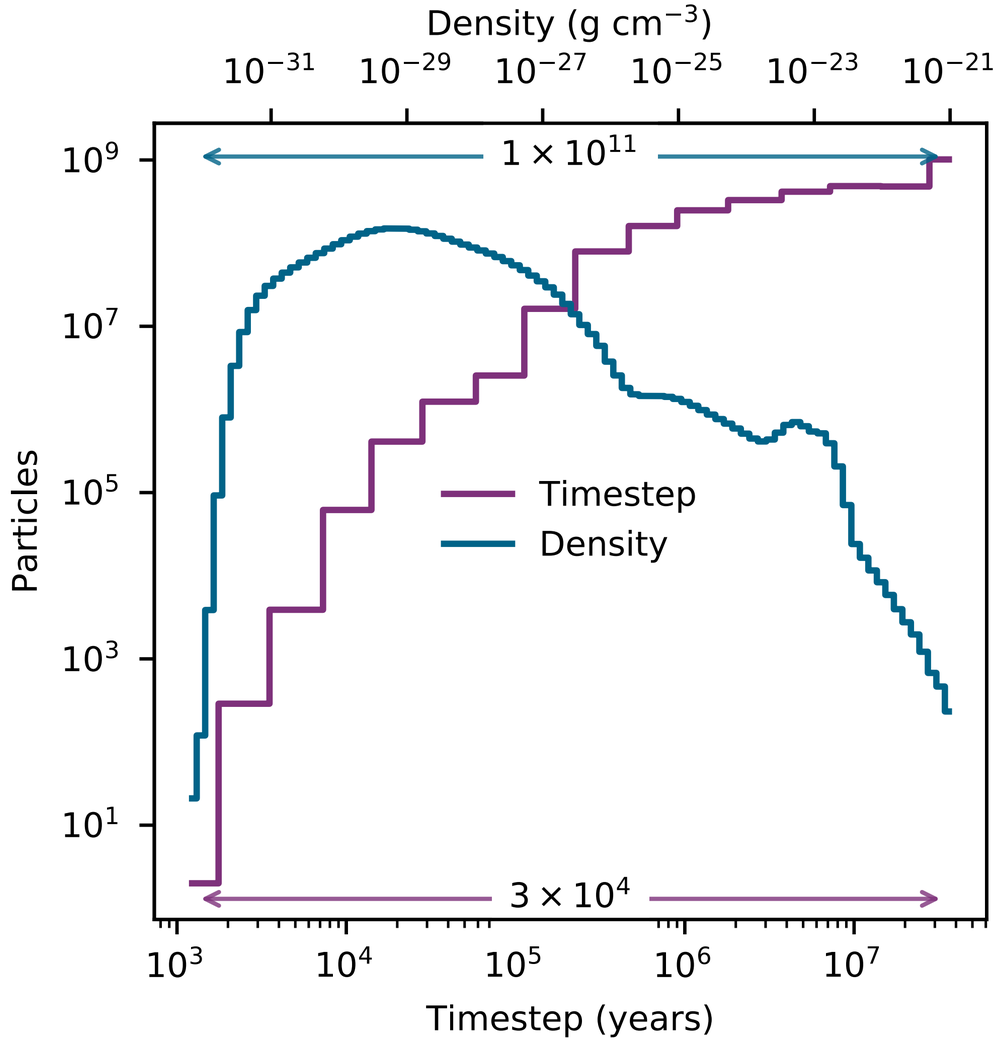
- New processors have a bunch of features
- These “SIMD” units allow you to do a bunch of operations all at once
- Writing code for these is hard: employ computer scientists



SIMD vectorised interaction between two cells
Willis+, ParCo 2017, 2017

Multiple Timesteps

- Hydrodynamics in cosmological simulations is highly adaptive
- 10^{11} range in density
- $10^{4.5}$ range in time-step!
- This is necessary; only very few particles are on those short time-steps - updating the whole volume is very costly

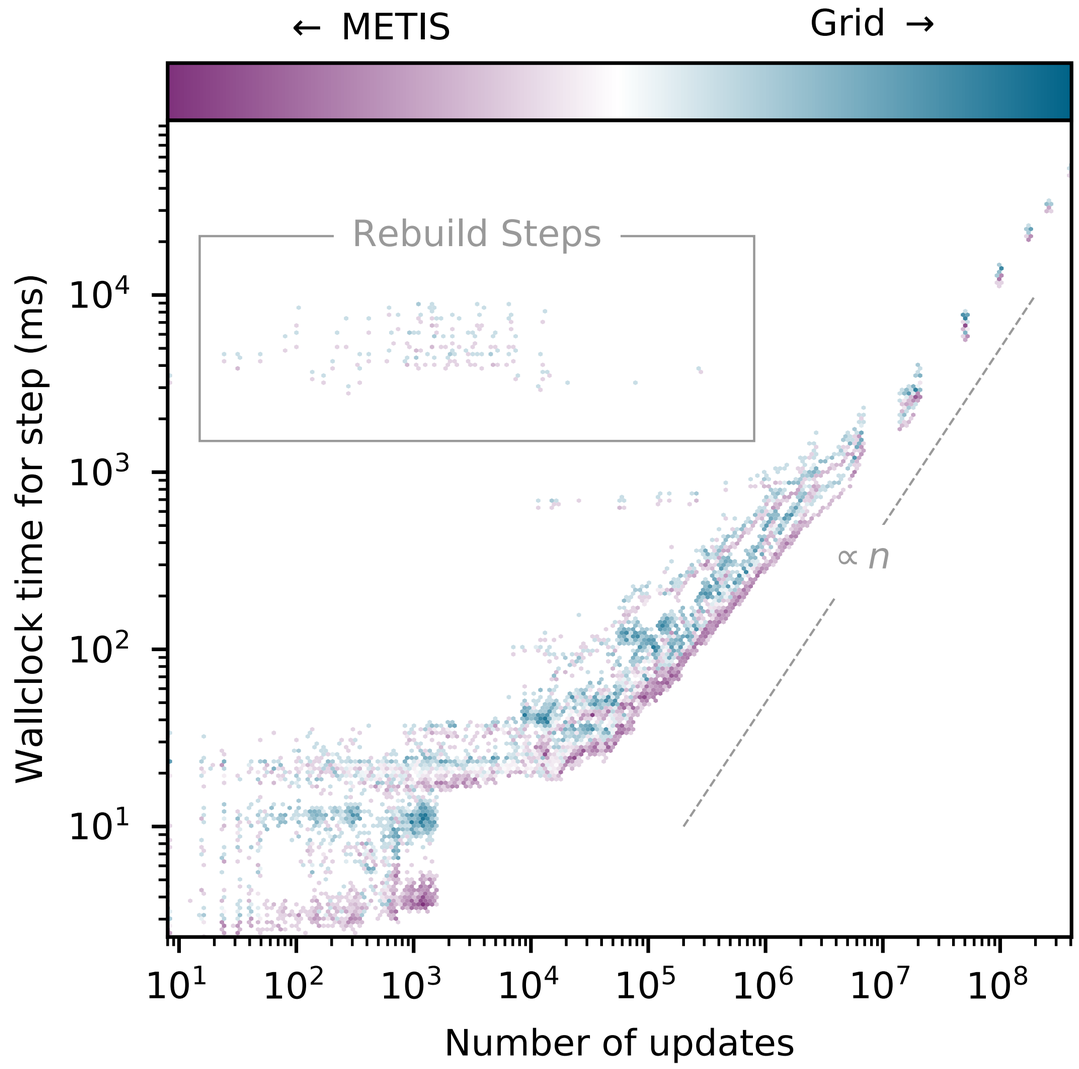


Time-step variation in the EAGLE-50 volume. Borrow+, SPHERIC 2018, 2018

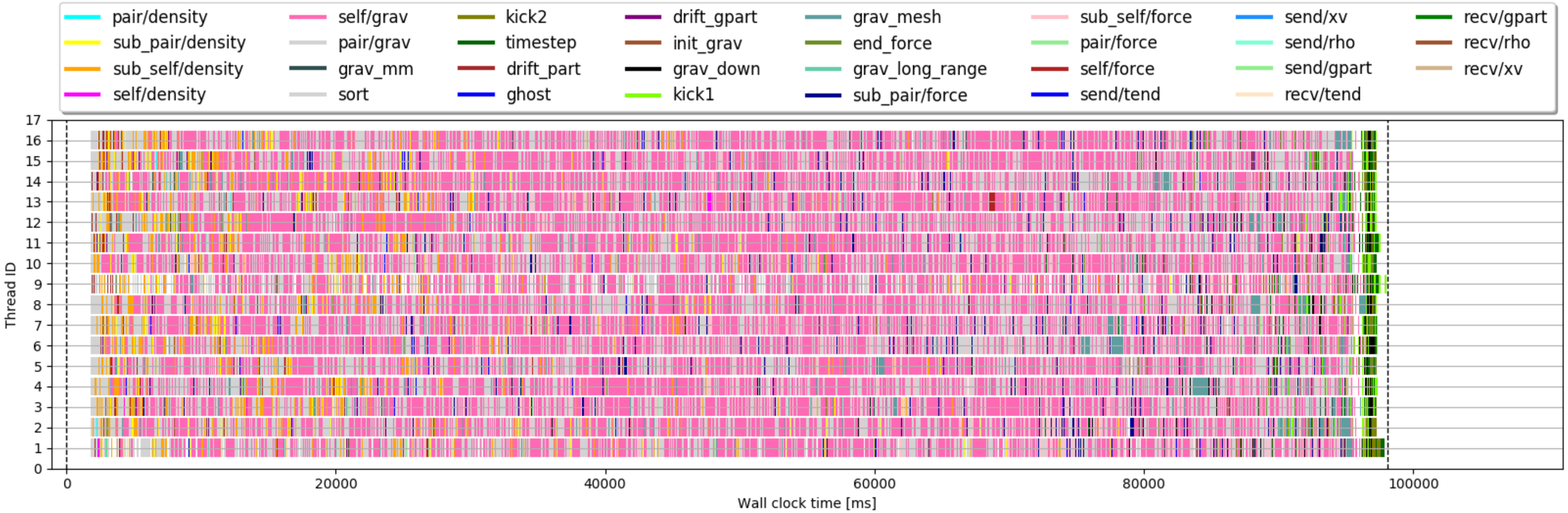
Load balancing

- All of that is great, but if you can’t balance it across many, many cores it’s useless

- The multiple time-steps makes this even harder; I don’t want to do communication for the short time-steps, but at the same time I want the big steps (majority of runtime) to be well balanced.



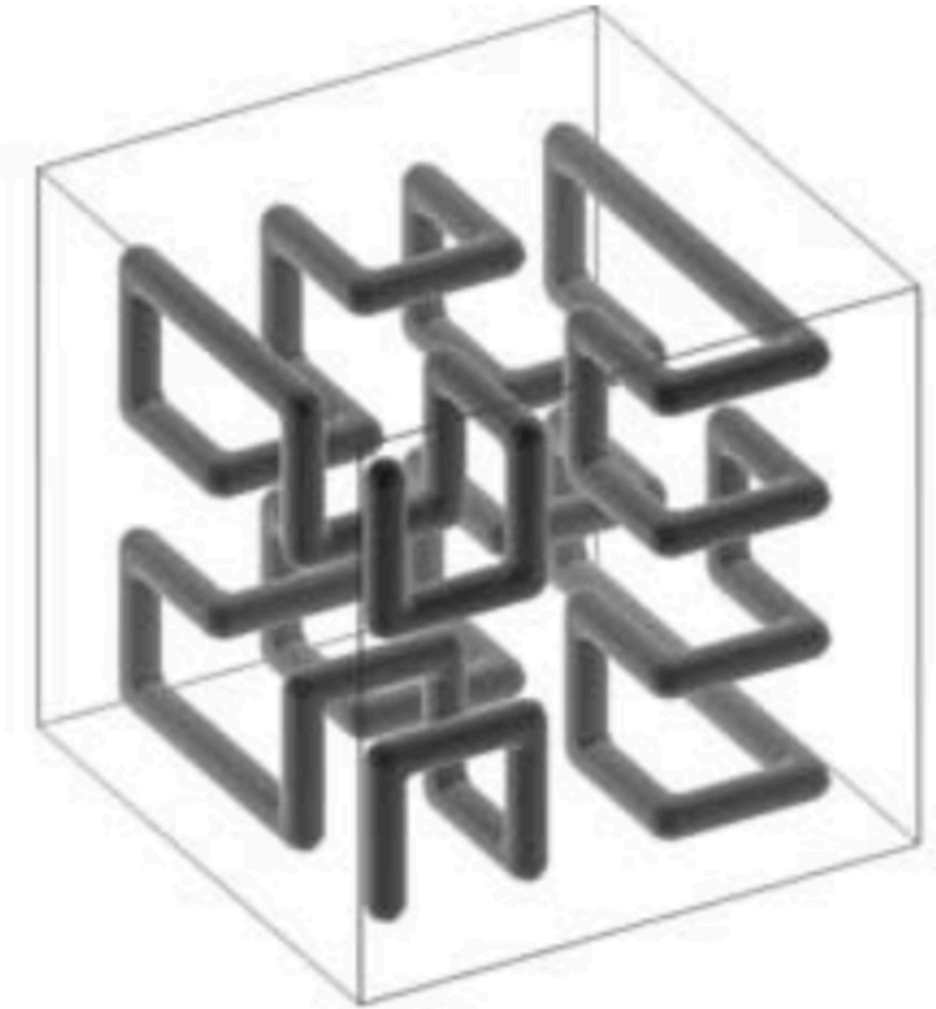
What does that look like?



Task plot in SWIFT. Draper 2018

Domain Decomposition

- Other codes use a space-filling curve
- Chop up that curve into equal length sections
- Stick each section onto a rank
- This achieves memory, not computation, balance



Peano-Hilbert space-filling curve used by GADGET-2, Springel 2005

Can we do this better?

- Using the tasking system, we can get a handle on the amount of work each top-level cell contains
- Can figure out how much communication it needs to do in advance
- Bisect the task graph, not the spatial distribution of particles!

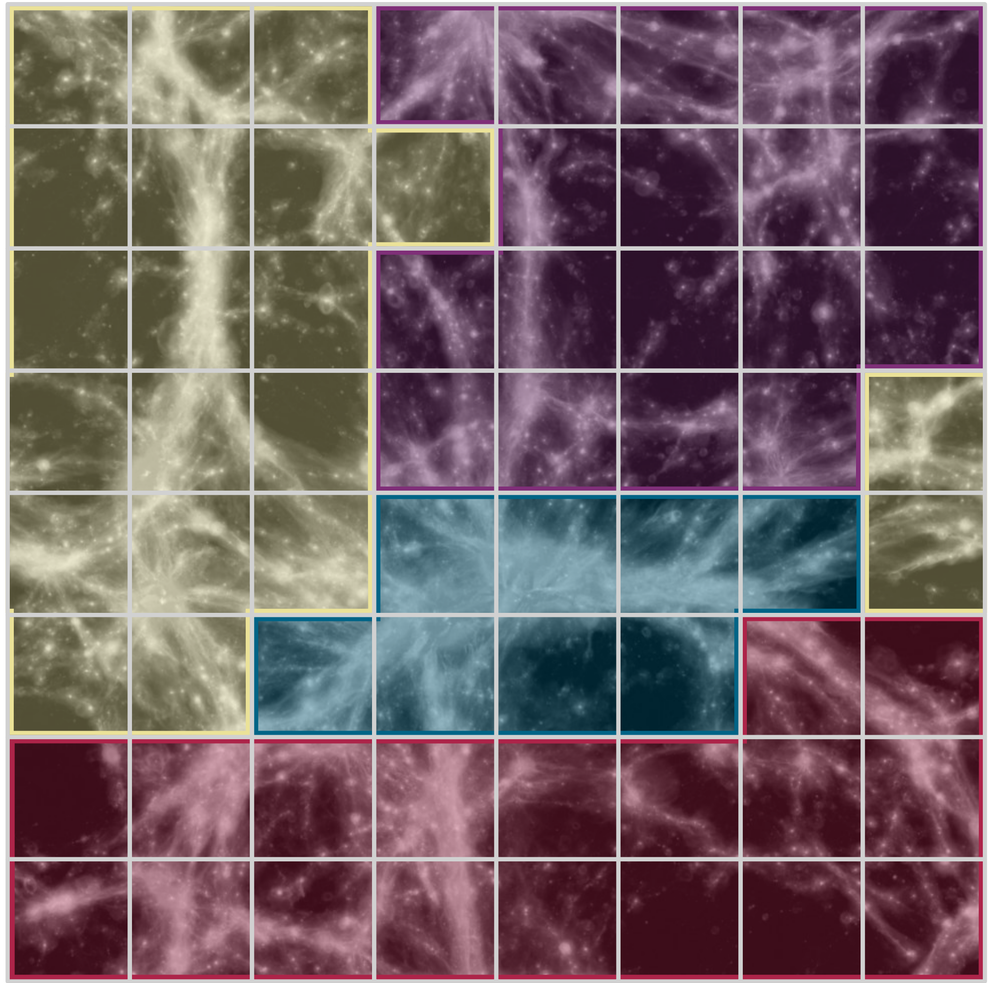
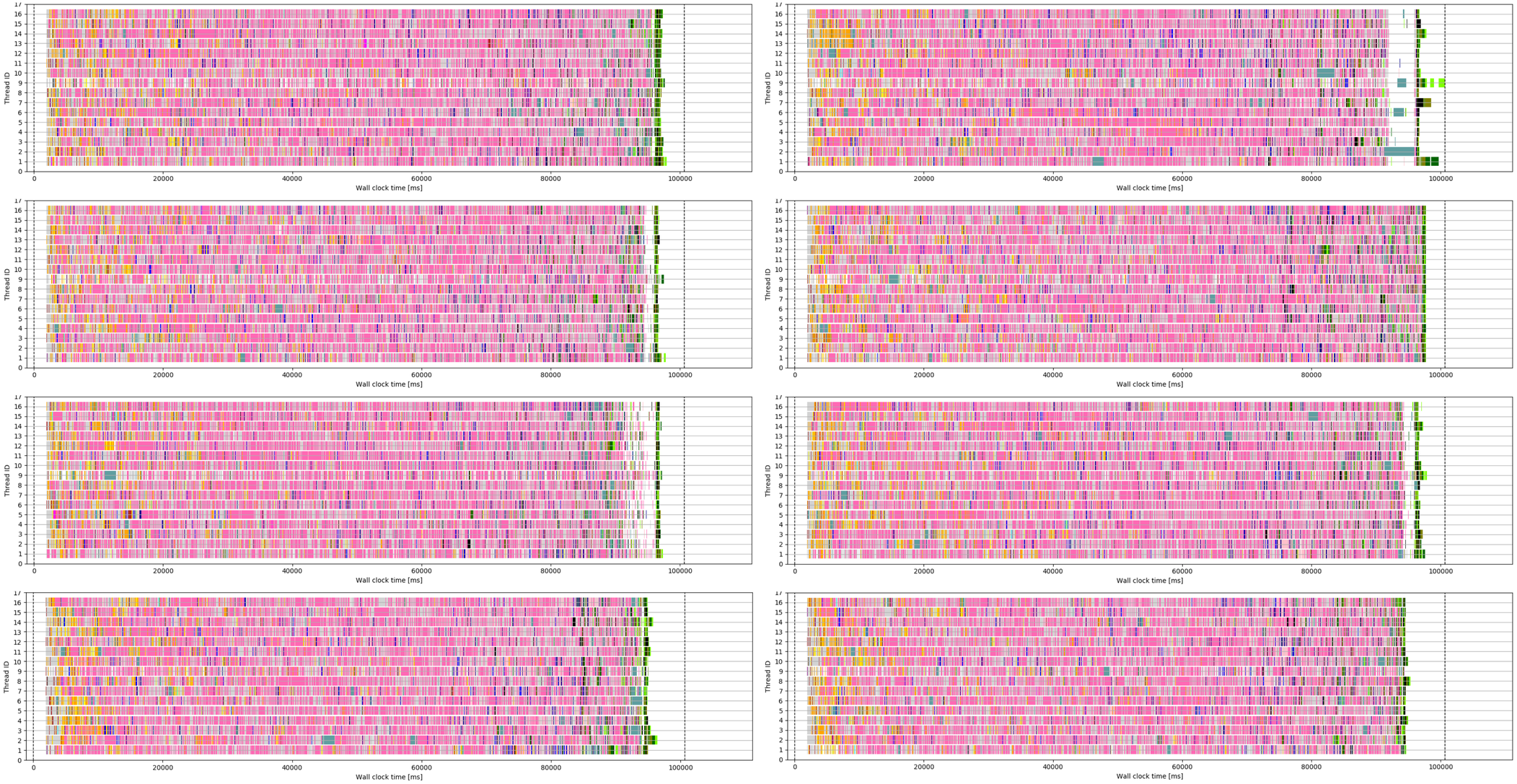


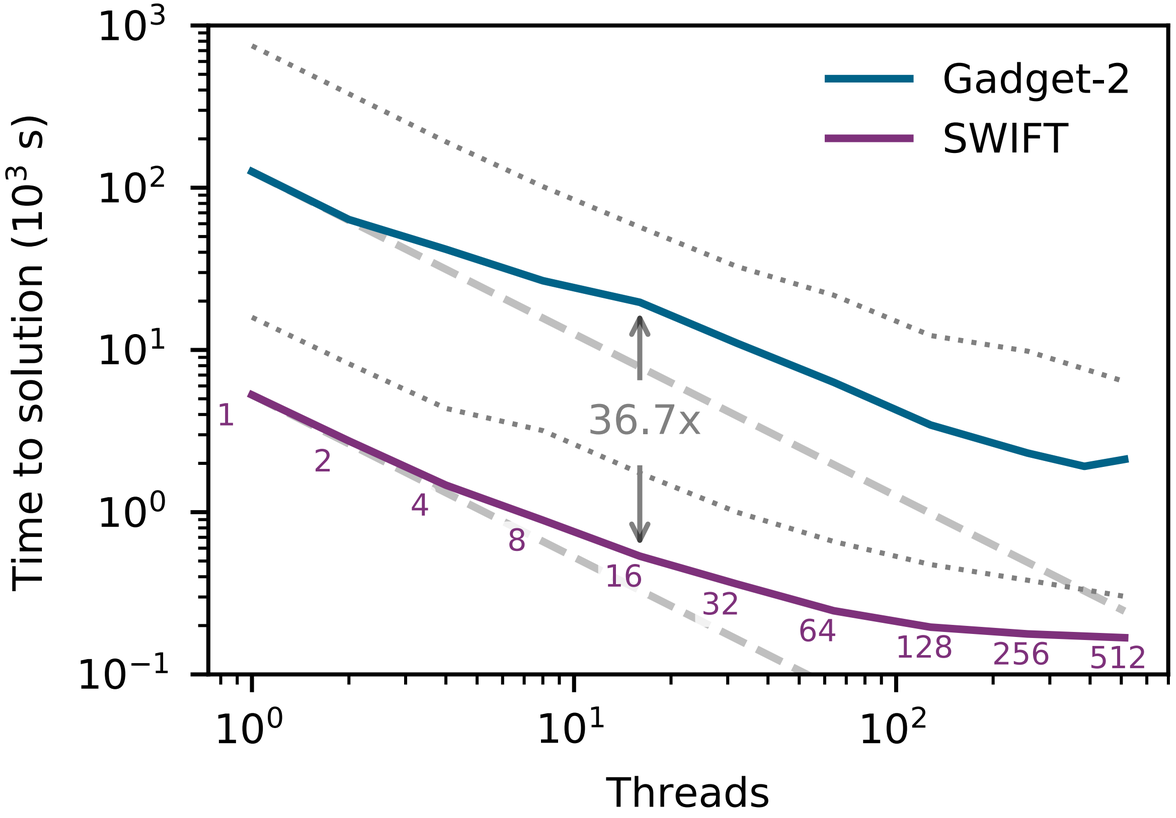
Illustration of top-level domain decomposition in SWIFT
Borrow+, SPHERIC 2018, 2018

Multi-Node Tasking



Results: Strong scaling

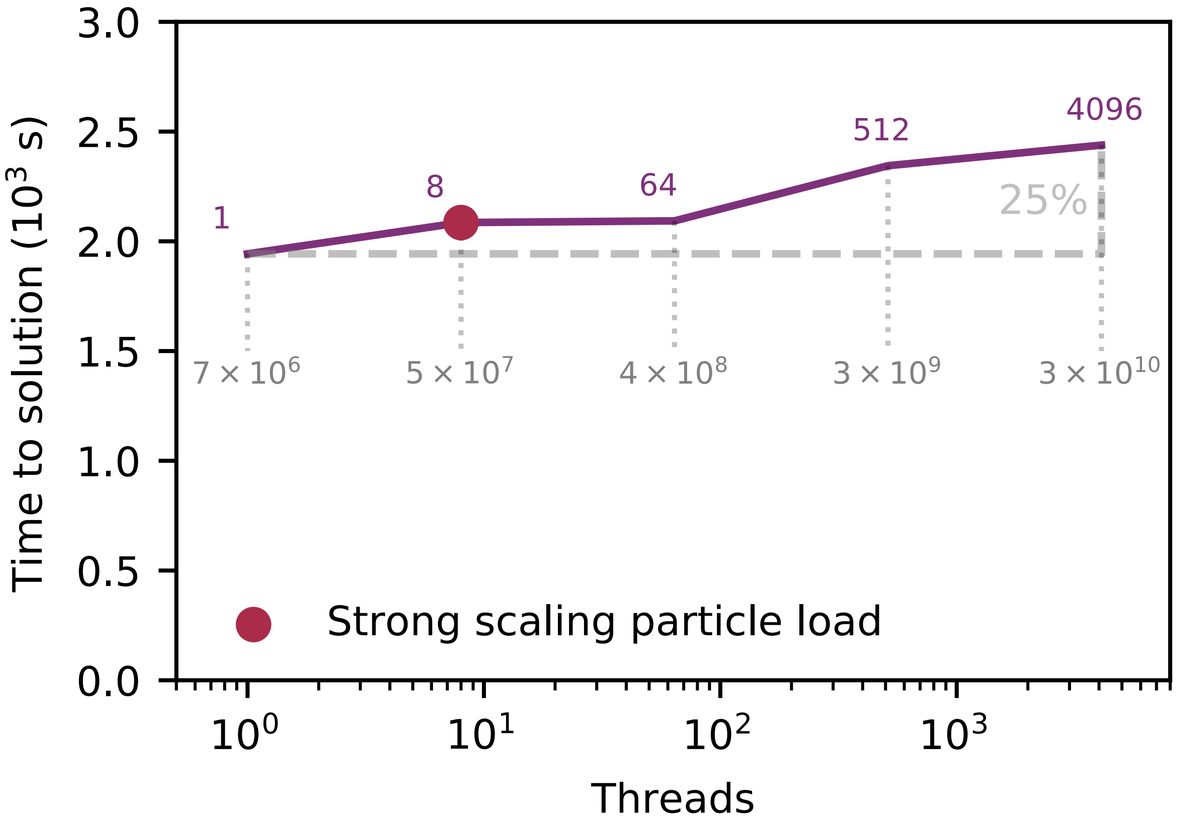
- Strong-scaling: for a given problem, throw more cores at it and see how much faster you go
- SWIFT shows a 37x improvement over GADGET-2 (hydro-only), and about 20x with hydro+gravity.



SWIFT strong scaling on the EAGLE-25 z=0.1 box
Borrow+, SPHERIC 2018, 2018

Results: Weak scaling

- Weak-scaling: for a given *load*, copy it again and again and see how much speed you lose
- SWIFT shows *near perfect* weak scaling (this is extremely rare).
- Shown at the worst-case position: z=0.1 after all structure has formed

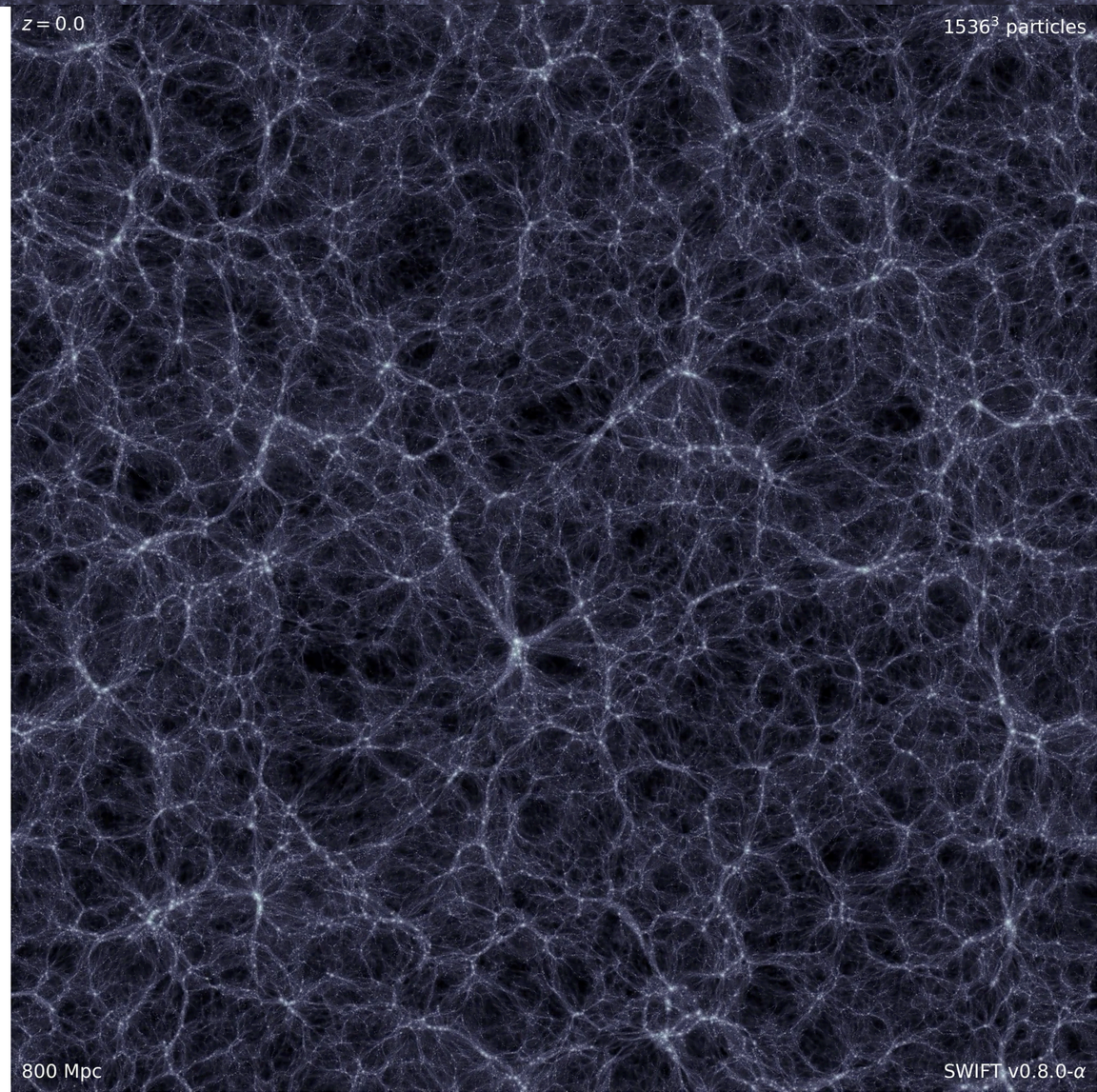


SWIFT weak scaling on the replicated EAGLE-25 z=0.1 box
Borrow+, SPHERIC 2018, 2018

Recent Work

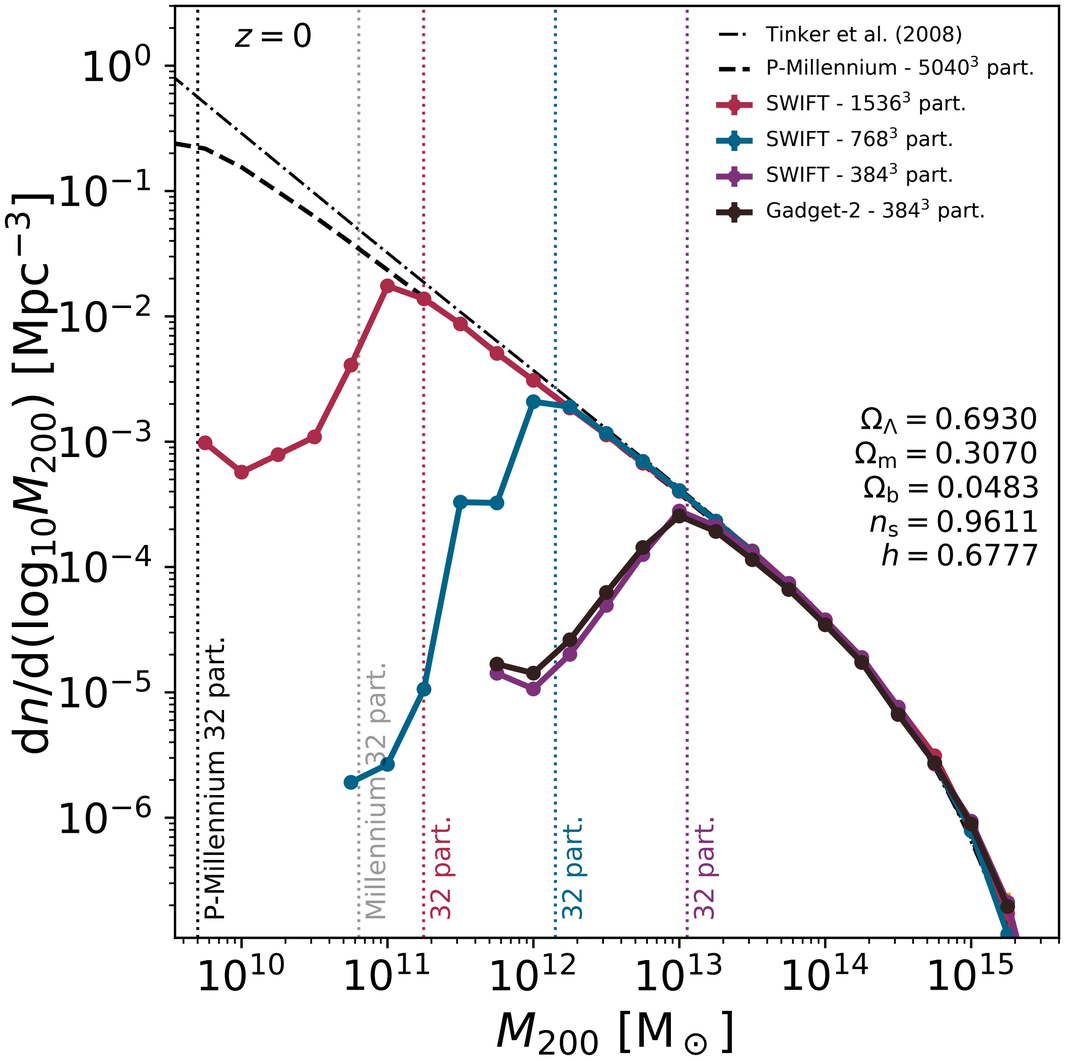
Dark Matter Only Runs

- Re-run the P-Millennium simulations (DMO)
- Running ~10x faster than with GADGET-2
- We can match the HMF exactly
- This includes running halo-finders on the fly (VELOCIraptor, 6D FoF)



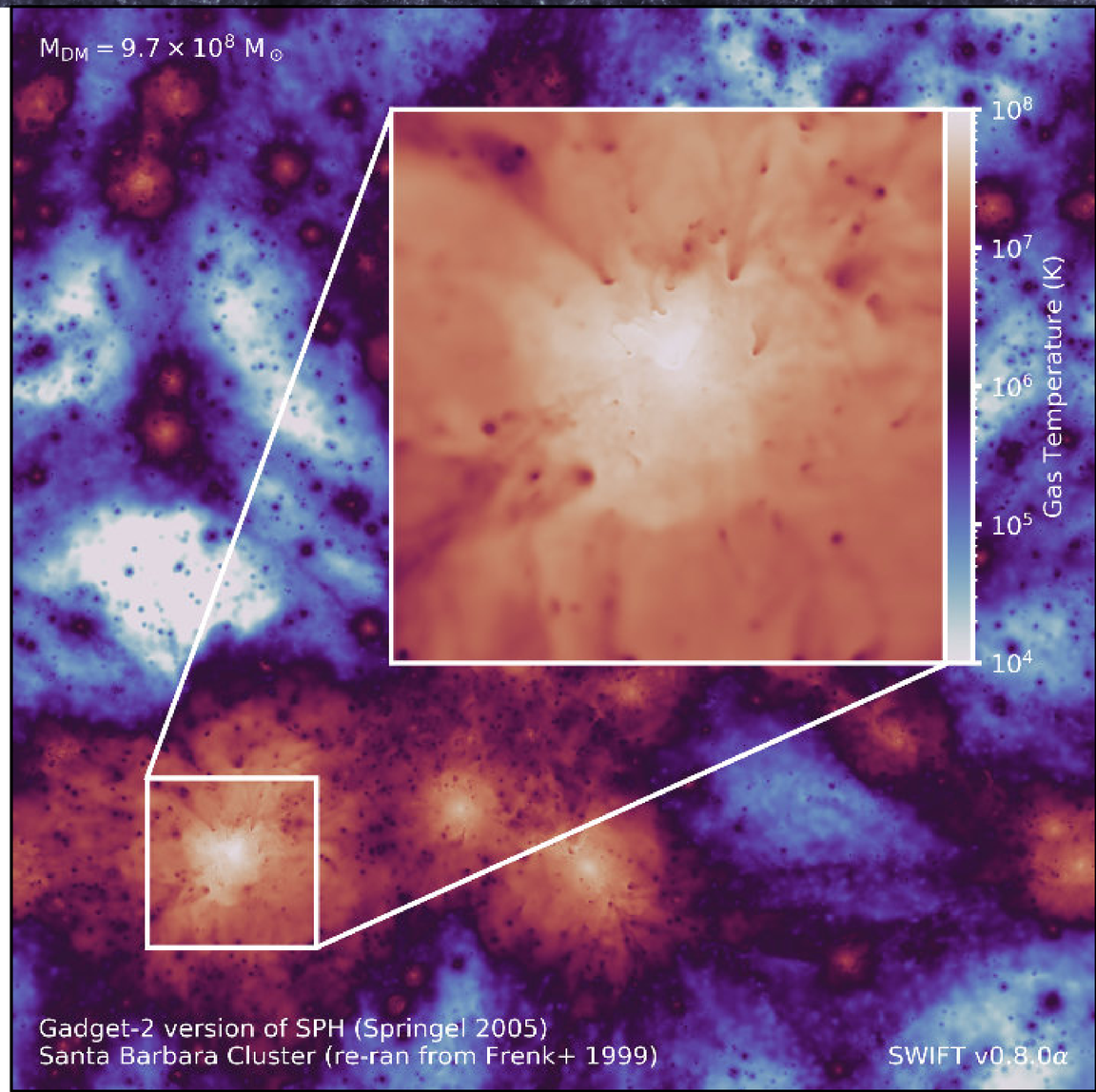
Dark Matter Only Runs

- Re-run the P-Millennium (DMO) simulations
- Running ~10x faster than with GADGET-2
- We can match the HMF exactly
- This includes running halo-finders on the fly (VELOCIraptor, 6D FoF)



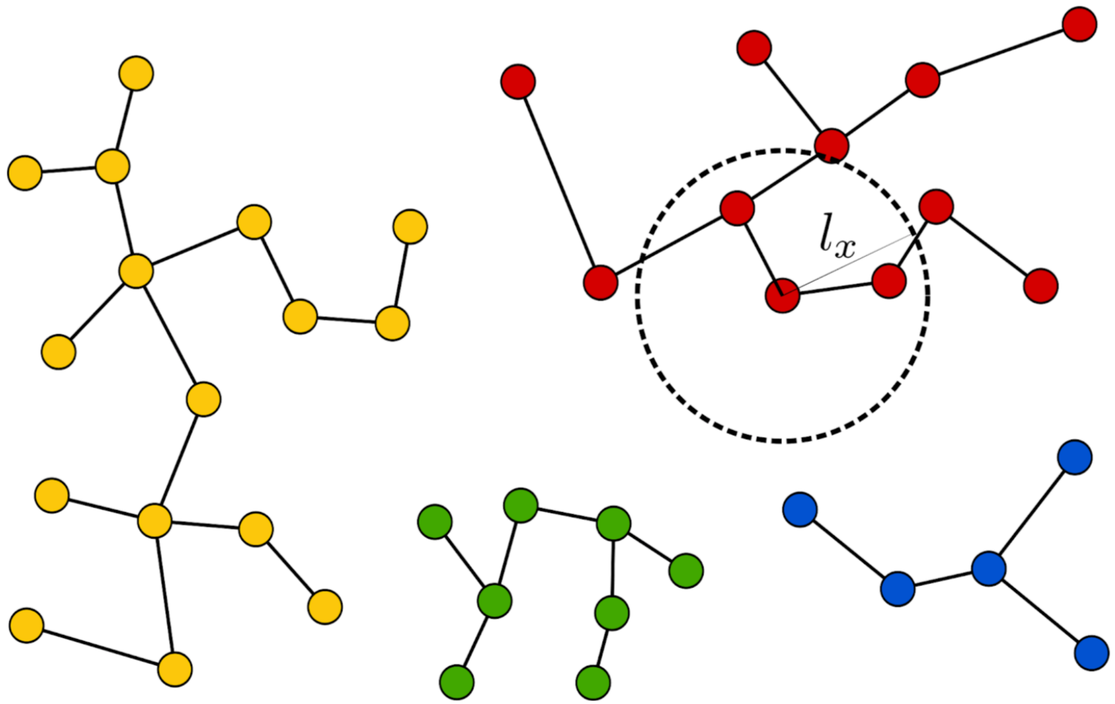
Santa-Barbara Cluster

- Everyone’s favourite cluster
- Now simulated with SWIFT!
- A great test of cosmology, hydrodynamics, and gravity all working together.



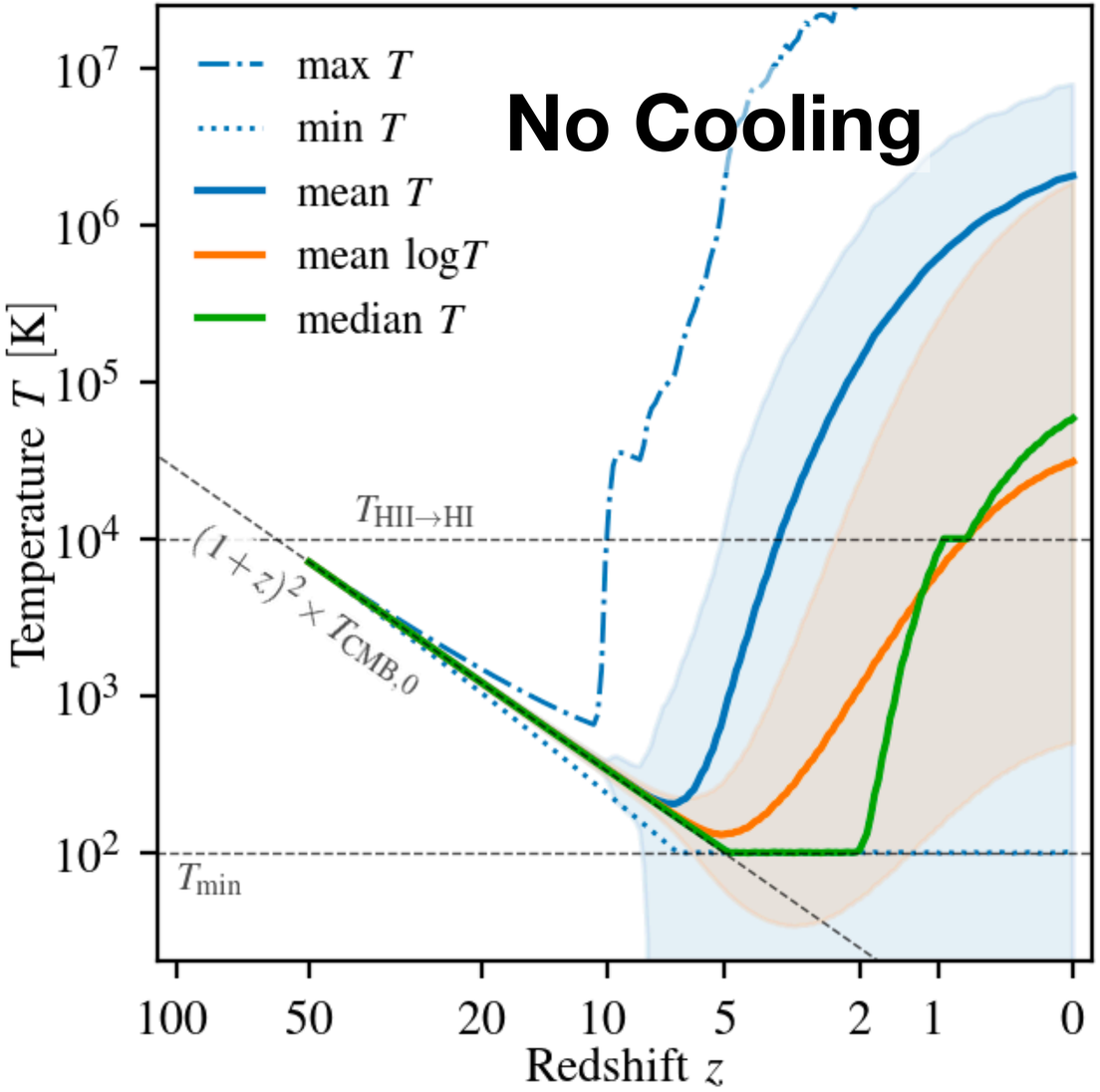
On-The-Fly FoF

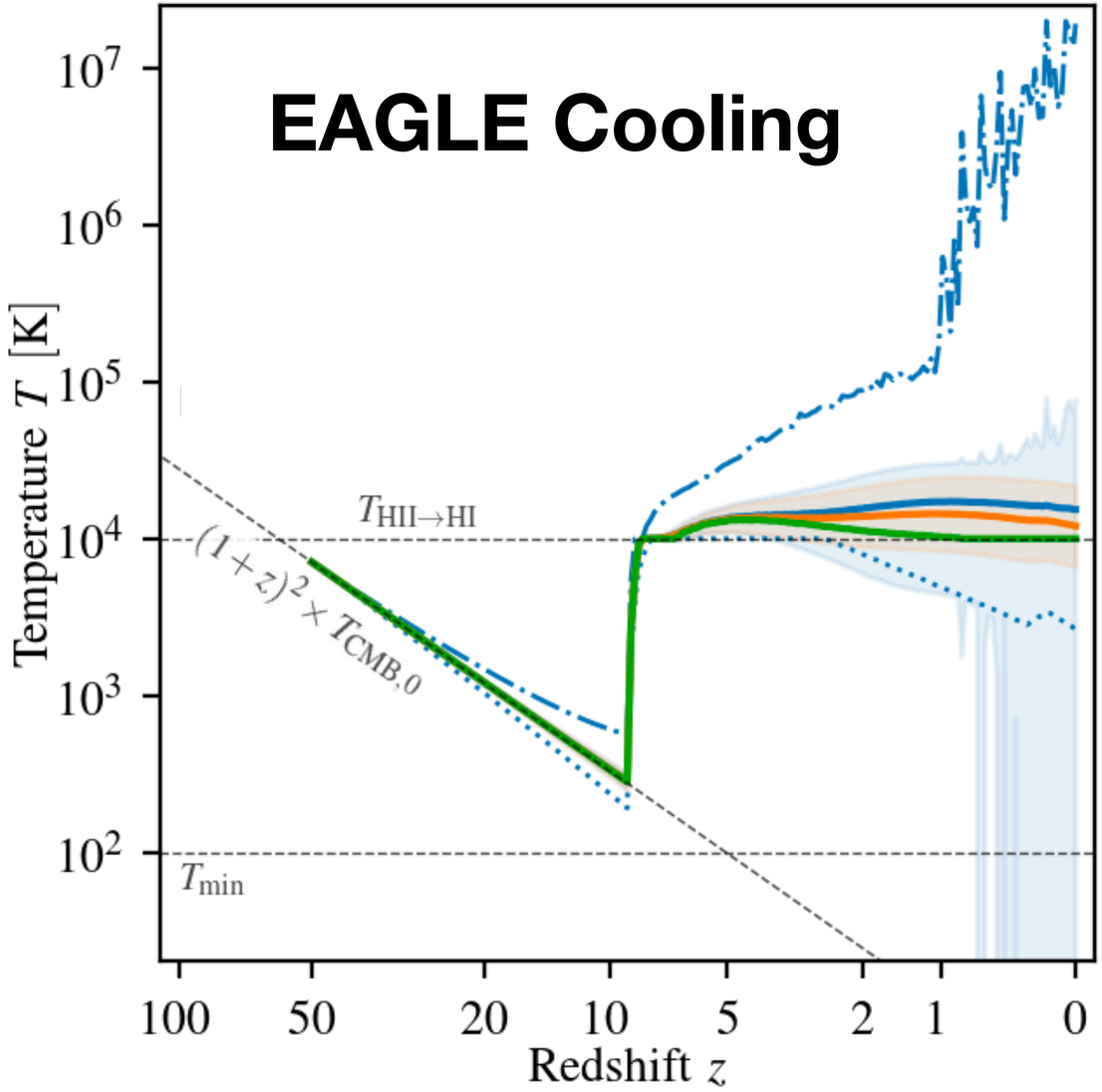
- Need a faster, simpler FoF than VELOCIraptor to run on-the-fly
- Used for seeding black holes
- FoF in SWIFT takes the same time on 56 cores as the GADGET/EAGLE one took on 2048 cores.



Friends-of-Friends algorithm visualised
Willis+, 2019

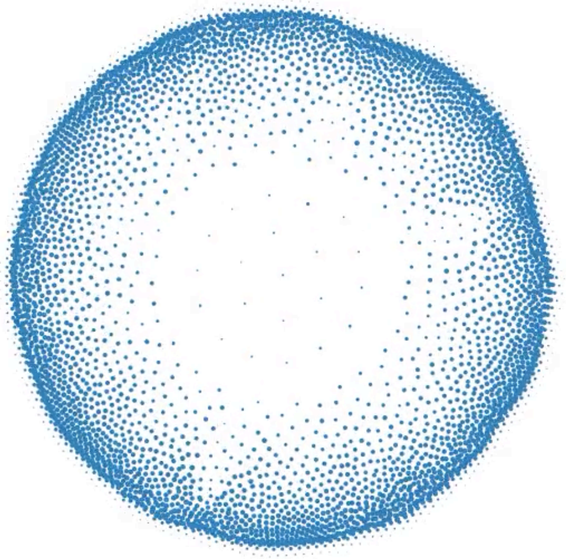
EAGLE Cooling





ANARCHY++

- Modified version of the original ANARCHY SPH scheme implemented
- Uses Pressure-Energy instead of Pressure-Entropy (makes feedback implementation much easier)
- Includes the Cullen & Denhen (2010) AV switch
- Also includes a small amount of energy diffusion



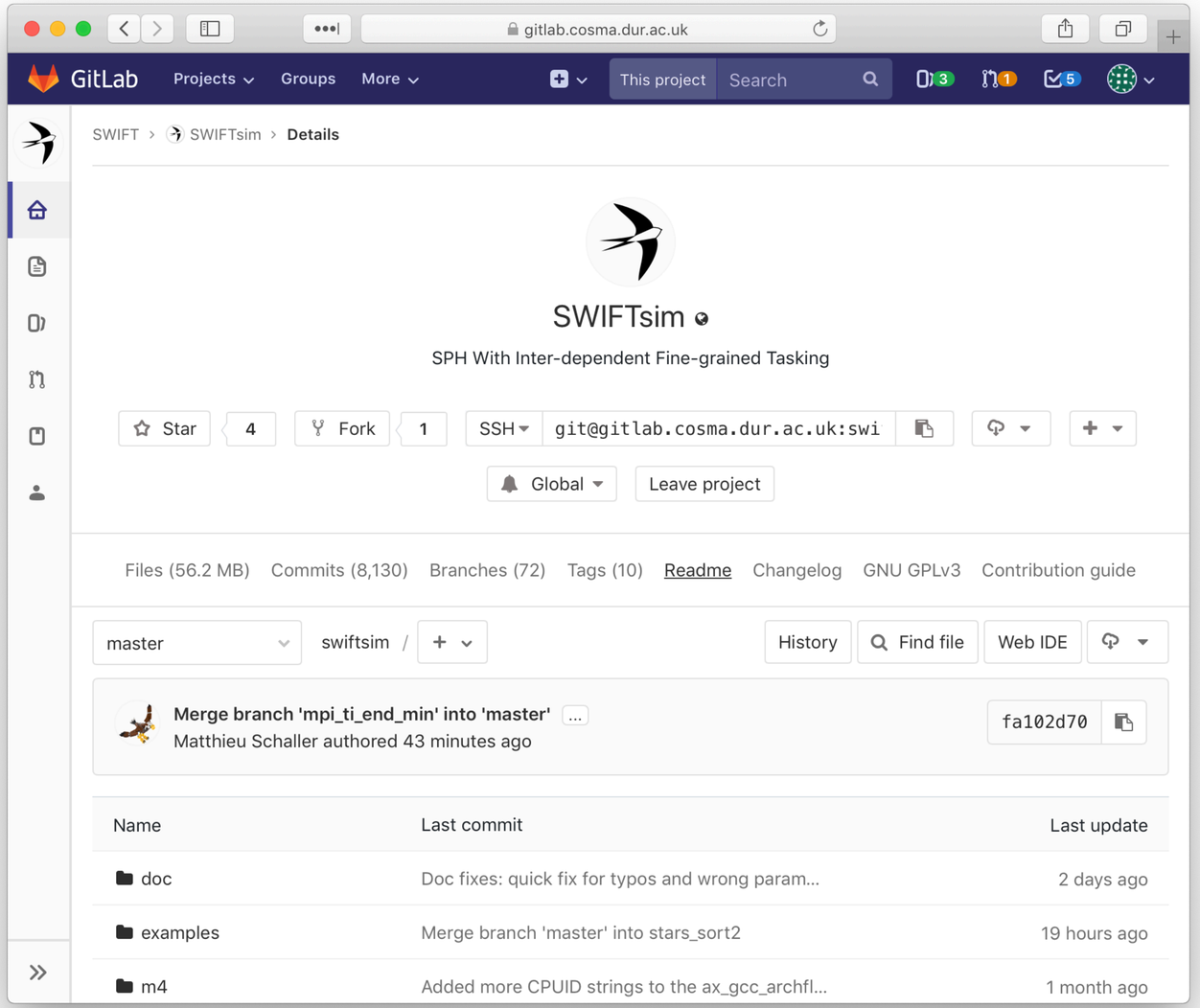
Sedov-Taylor blastwave in SWIFT

What do we need for EAGLE-XL?

Physics	Theory	Foundational Work	Complete?
Hydrodynamics (ANARCHY)	✅	✅	80%
Gravity (FMM)	✅	✅	✅
Cosmology	✅	✅	90%
Cooling	✅	✅	✅
Stars	✅	✅	25%
Feedback	✅	20%	
Black Holes	✅	20%	
AGN	✅		

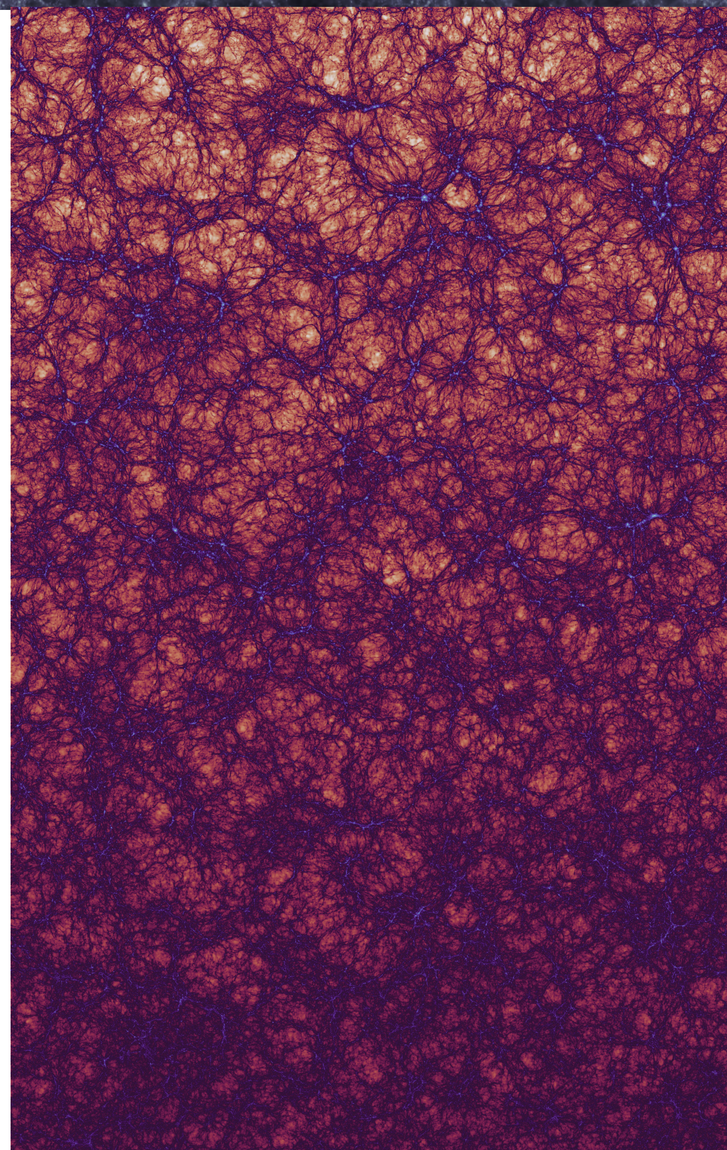
Did we mention?

- SWIFT is fully open source and in *open development*
- You can see all the issues. All the commits. Everything.
- swiftsim.com



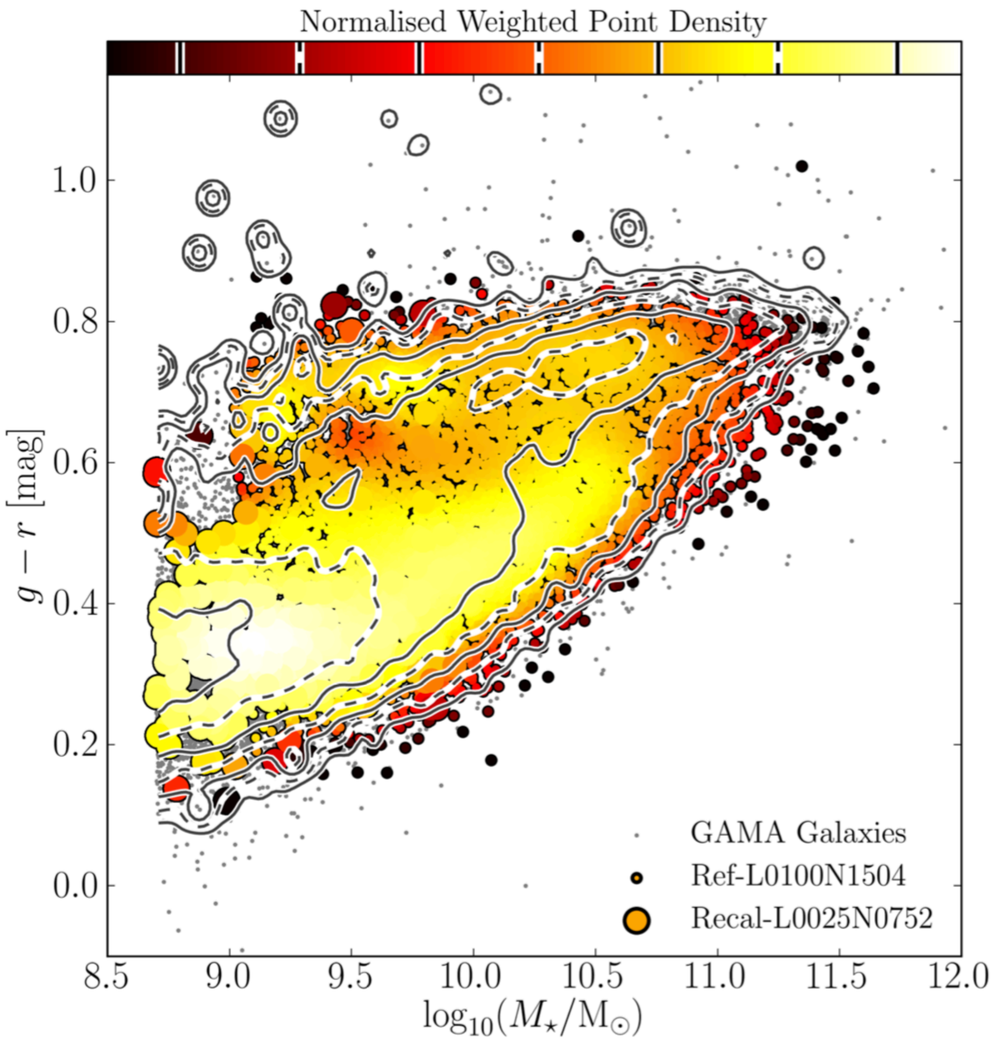
Conclusions

- EAGLE-XL is only possible with SWIFT
- EAGLE-XL will open up a new era of statistics in computational cosmology
- SWIFT will allow us to explore other projects that were previously intractable
- Plans to implement other sub-grid models in SWIFT to compare directly; EAGLE cooling with SIMBA black holes?



Parameter Tuning

- Common criticism: “don’t you just tune the parameters to get what you want”?
- It’s not quite as simple as that
- Some data is “training” (GSMF at z=0.1), some “validation” (GSMF at higher z), and some “results” (galaxy colours).



Mass-colour plane, Trayford+ 2015