SWIFT

Scaling Cosmological Simulations

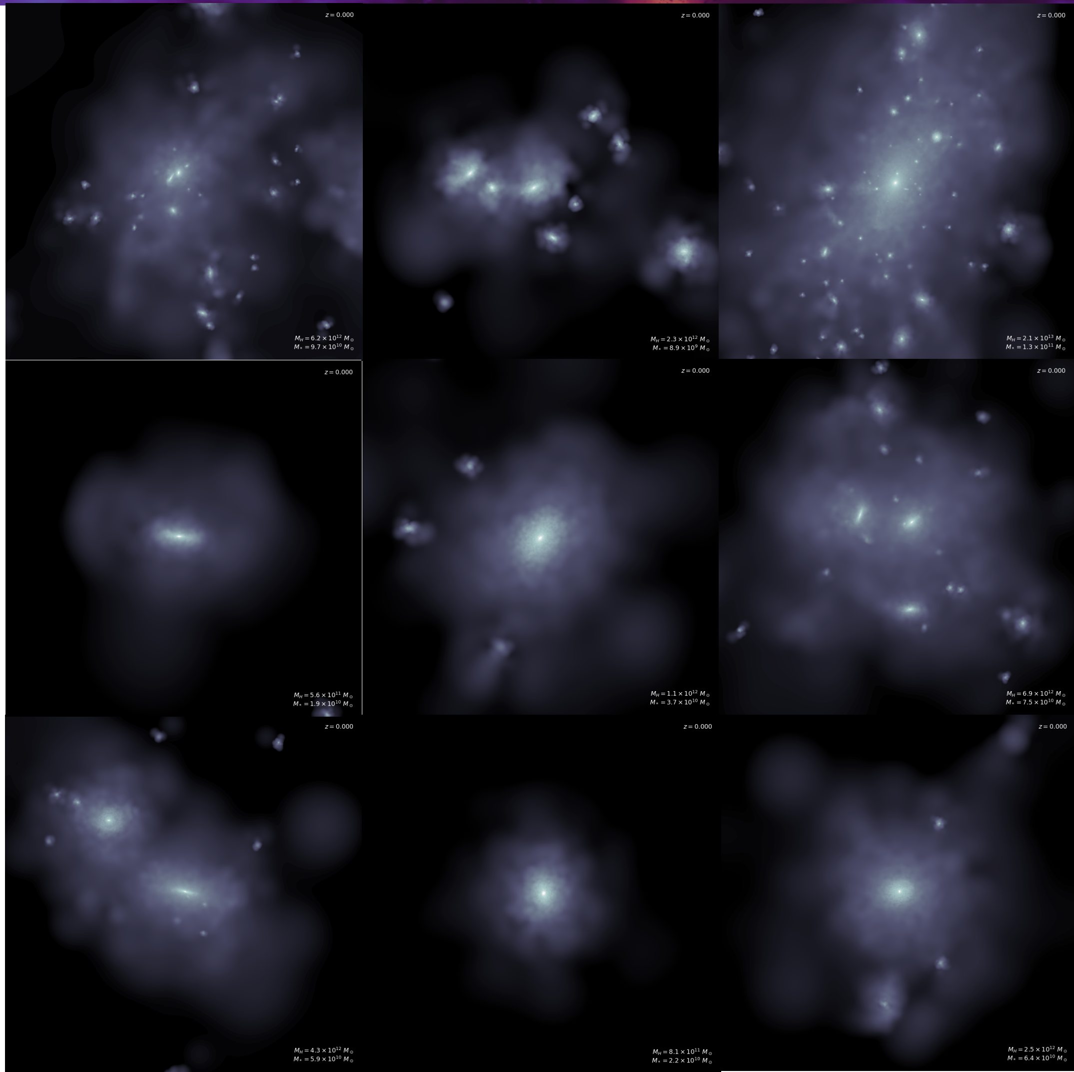
Josh Borrow *Institute for Computational Cosmology, Durham University*

Matthieu Schaller, Pedro Gonnet, Richard Bower, Alexei Borrisov, … (etc. etc. etc.).

What is SWIFT?

- SWIFT is a new cosmological simulation code that will be used to run the next generation of EAGLE, GEAR, and (maybe?) SIMBA simulations.

- It's a replacement for GADGET/GIZMO in many ways and aims to be much easier to work with.

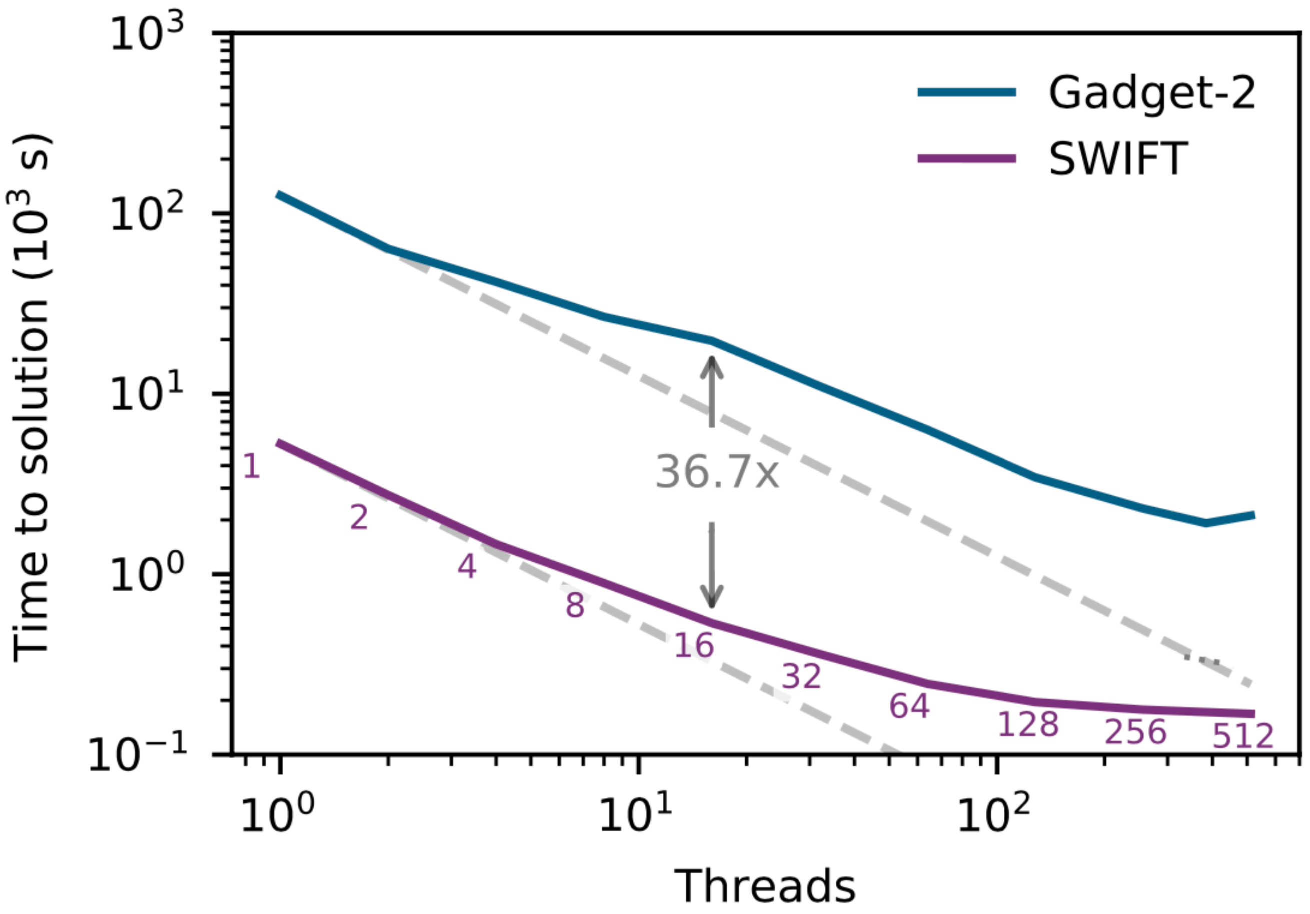


Challenges for Cosmological Simulations

- Scalability
  - GADGET-based codes struggle to scale further than about 4000 cores.
- Data size
  - EAGLE-XL, our next project, will take up around 1 Pb of disk space.
- Code complexity and quality
  - EAGLE (and SIMBA) codes have exploded in complexity without much maintenance time - we have *lots* of technical debt as a community.

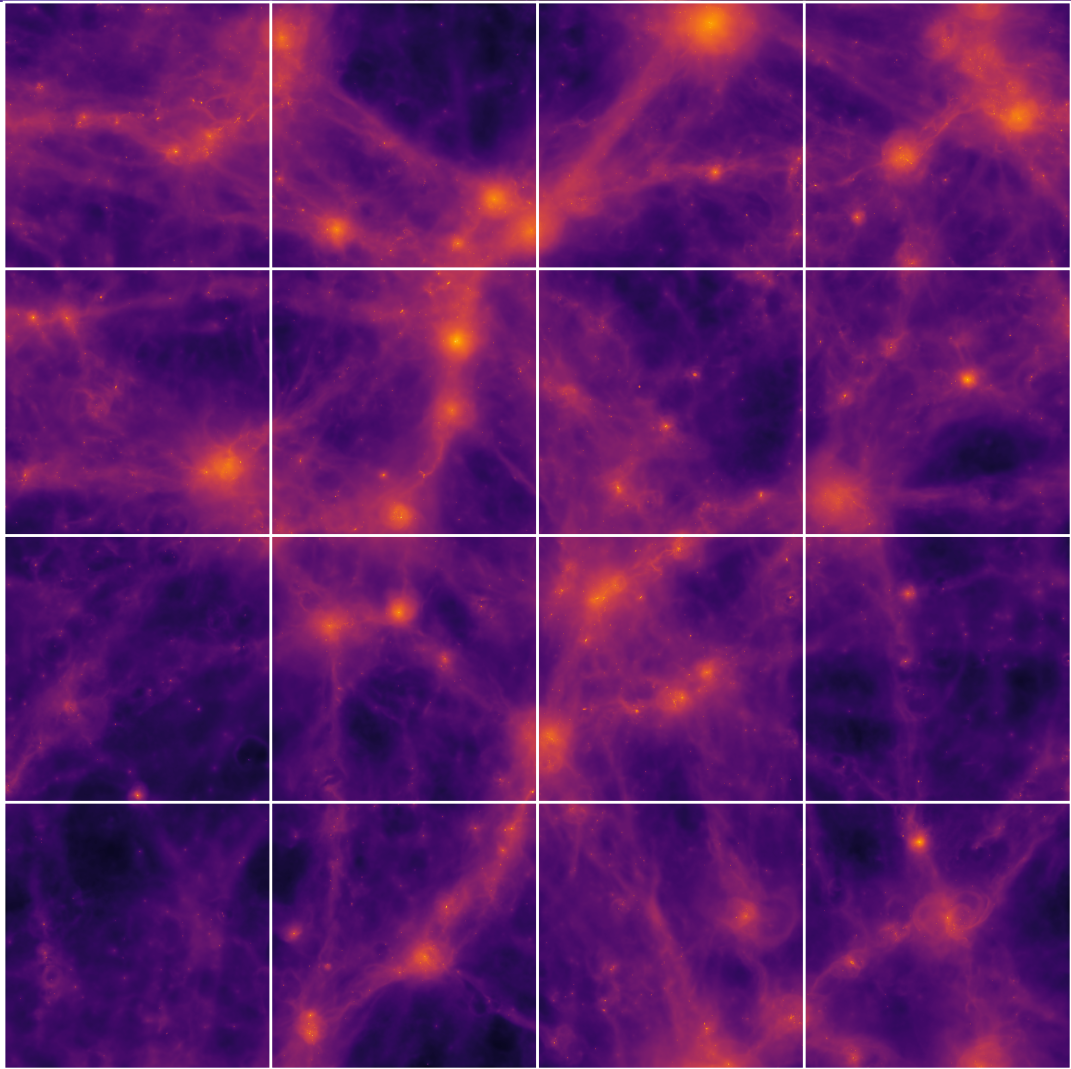
Code Speed

- SWIFT uses improved algorithms to gain an up-to 37x speed-up against Gadget-2
- See Borrow+2018 for more information.



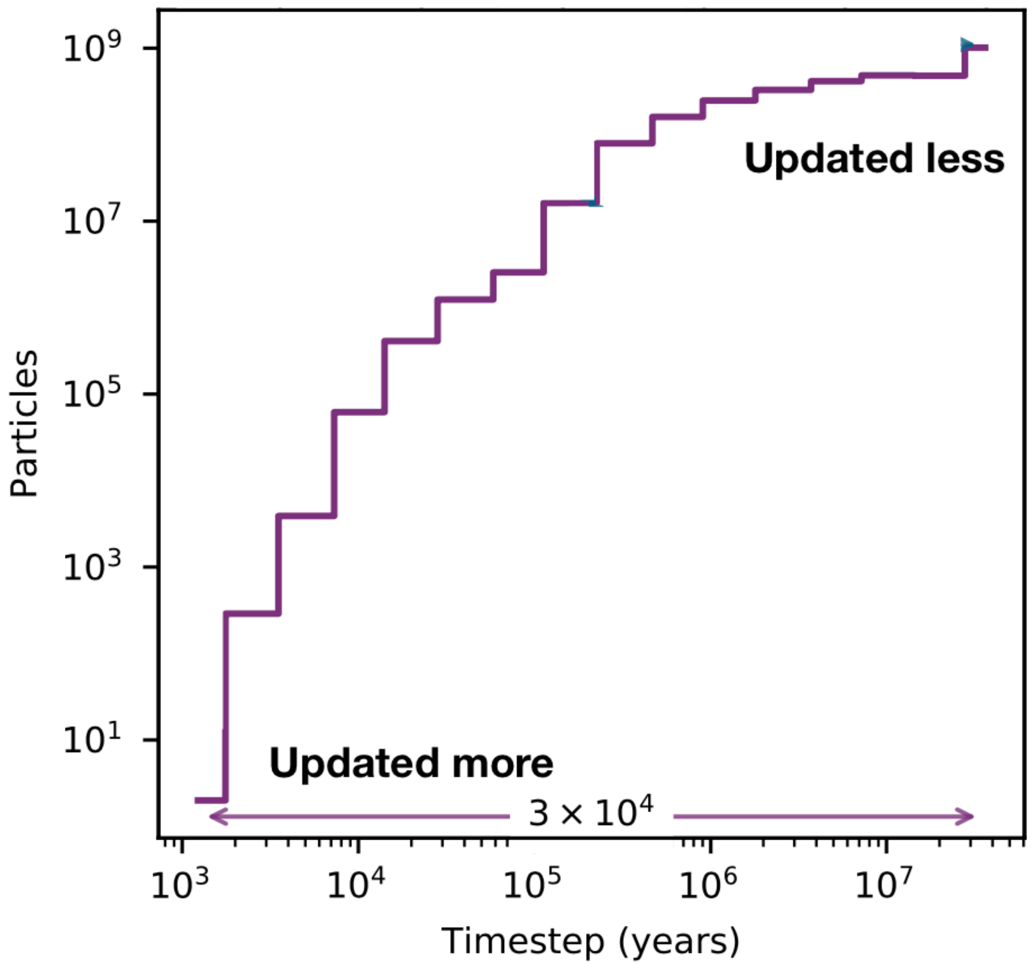
Scalability

- Gadget-based codes use only "Data Parallelism"
- In the age of high core-count CPUs this is no longer a viable model.
- SWIFT uses data parallelism on a node-by-node basis, but task-based parallelism within nodes.



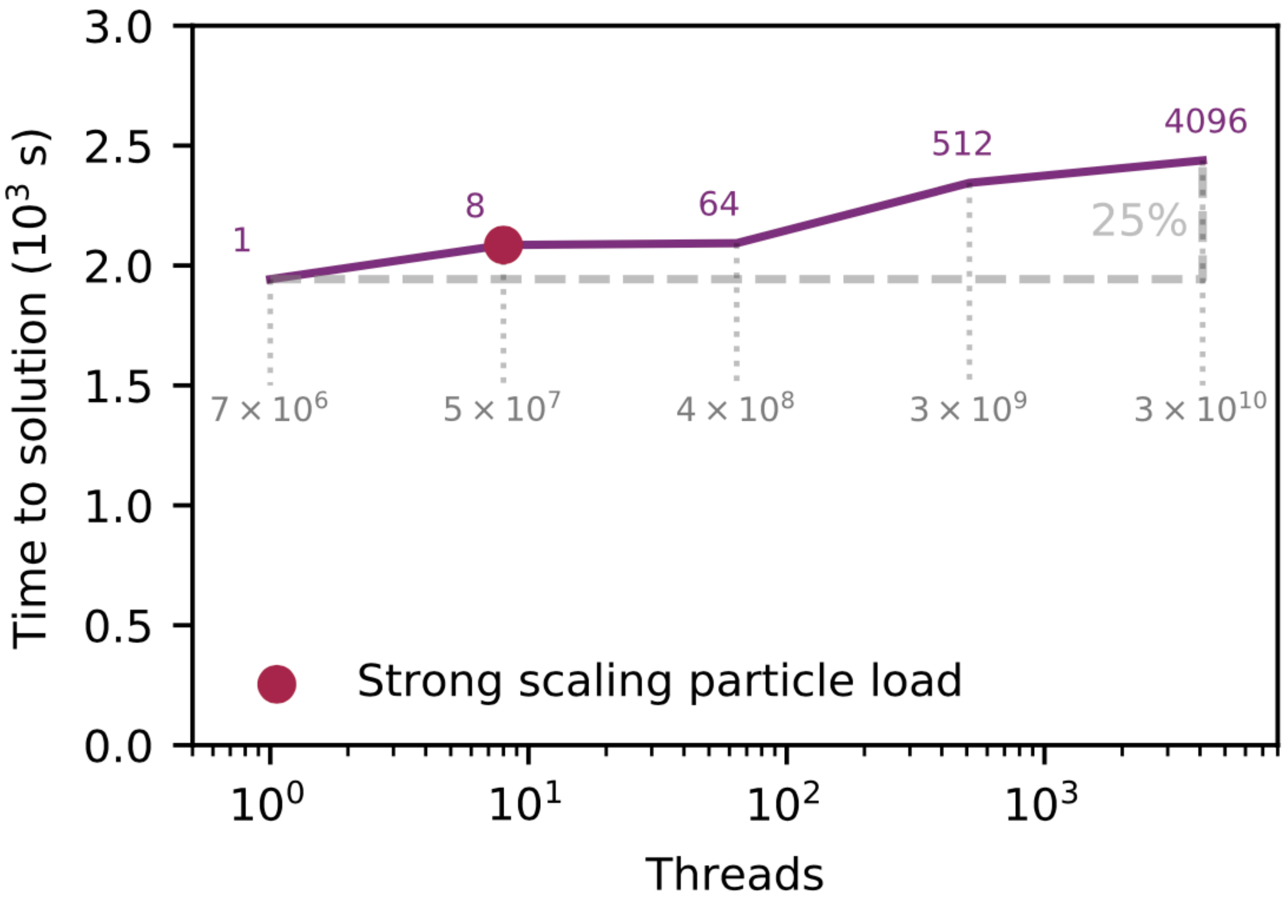
Scalability

- Why does data parallelism fail?
- Cosmological simulations are a parallelization nightmare!
- In the vast majority of steps there is almost nothing to do - need the largest domains possible to reduce communication.



Scalability

- Weak-scaling cosmological simulations is challenging because of the multiple time-stepping.
- Thanks to the larger domains and lack of unnecessary drifts, SWIFT can weak-scale almost perfectly.

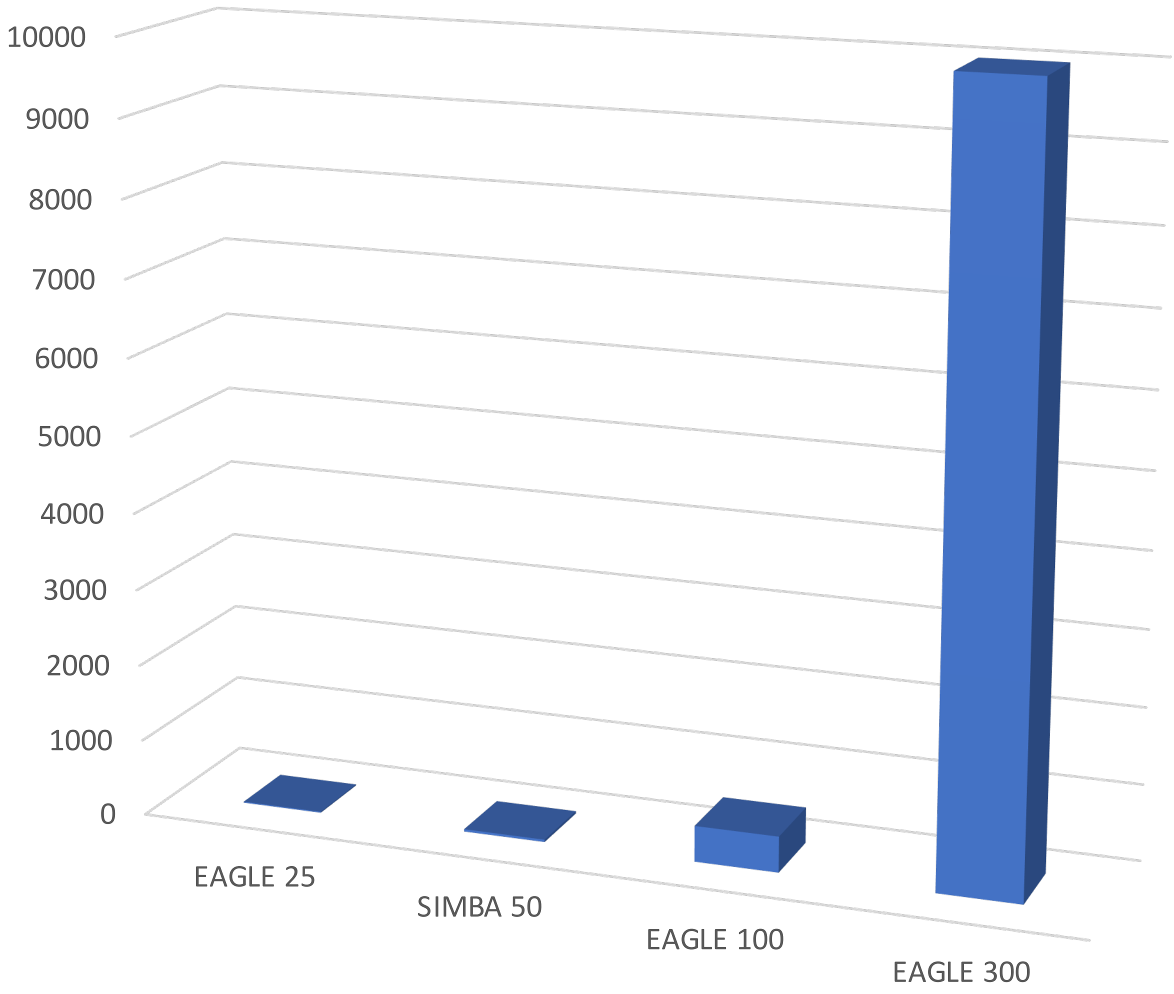


Data Size

- We are now producing ‘big data’ all by ourselves; EAGLE-XL’s production run has snapshots that are around 10 Tb each.

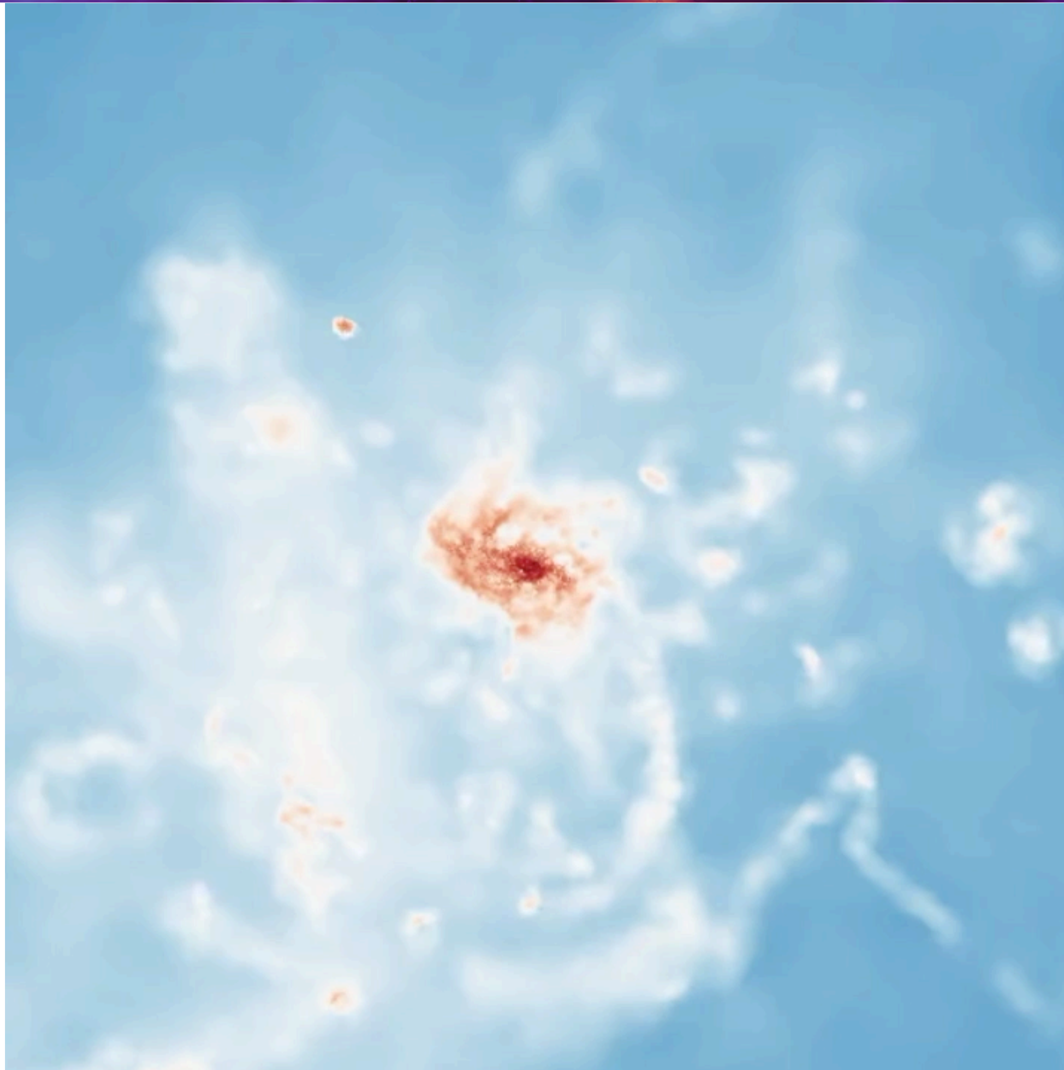
- This makes on-the-fly (post?)-processing essential!

- SWIFT runs VELOCIraptor on the fly to calculate galaxy properties (and profiles!).



Postprocessing Tools

- We have python libraries for both SWIFT snapshots and VELOCIraptor catalogues that are available on GitHub!
- Cool stuff like:
  - Automatic use of symbolic units
  - Ability to generate all the halo catalogue plots you could ever want from one line
  - Visualisation (see SIGAME!)



Postprocessing Tools

- We have python libraries for both SWIFT snapshots and VELOCIraptor catalogues that are available on GitHub!

- Cool stuff like:

  - Ability to analyse any size dataset on a laptop

  - Database integration for halo catalogues

Code Quality

- EAGLE source code:
- SIMBA is similar in many ways...
- Code quality is as important as mathematical rigor - they are the same thing!

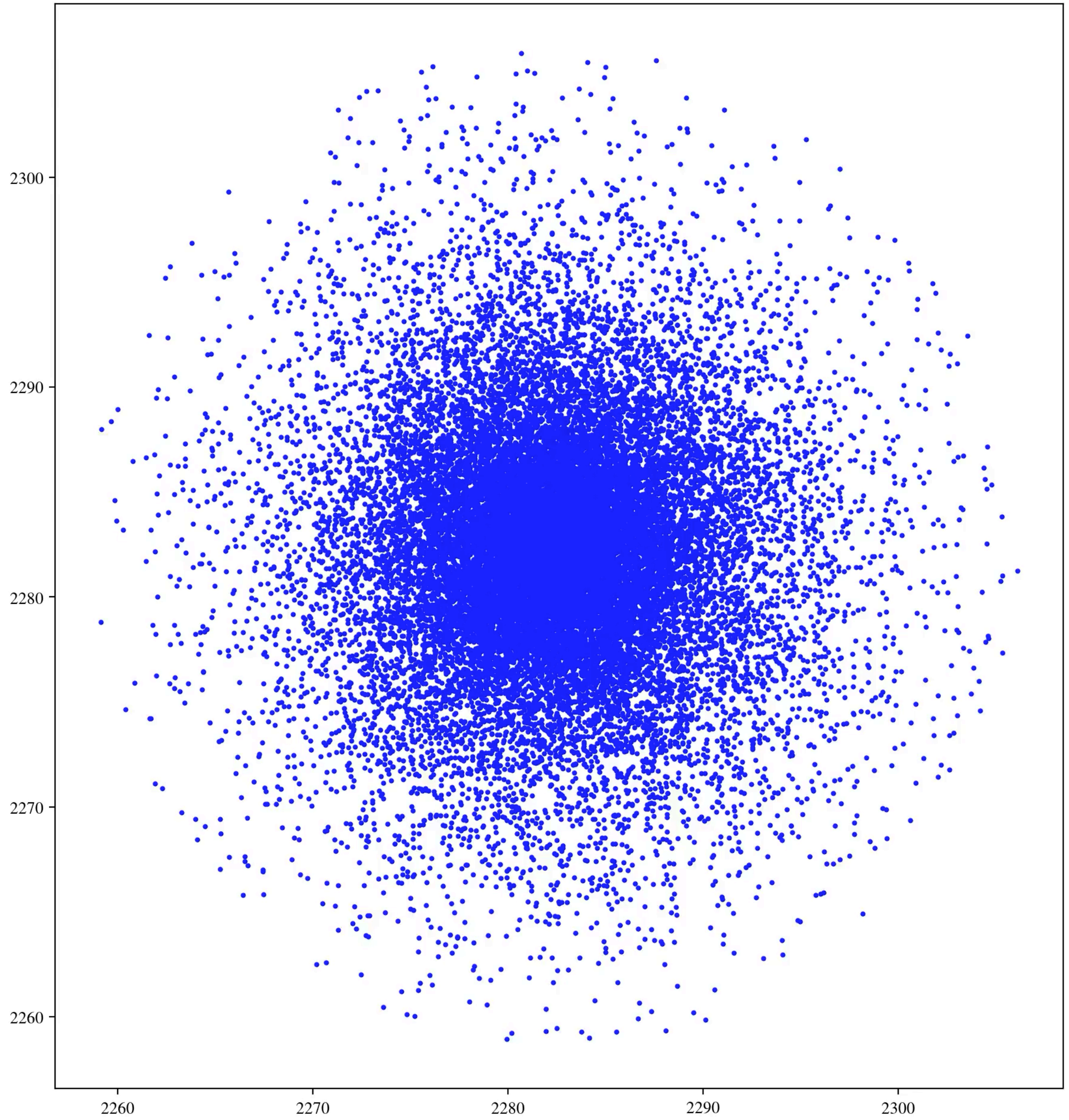
```
#ifndef NOVISCOSITYLIMITER
              double dt = 2 * IMAX(local.Timestep,
                                   (P[j].TimeBin ? (1 << P[j].TimeBin) : 0)) *
                          All.Timebase_interval;
              if (dt > 0 && kernel.dwk_ij < 0) {
#ifdef BLACK_HOLES
                if ((local.Mass + P[j].Mass) > 0)
#endif
                  visc = DMIN(visc, 0.5 * fac_vsic_fix * kernel.vdotr2 /
                                        ((local.Mass + P[j].Mass) *
                                         kernel.dwk_ij * kernel.r * dt));
#ifdef COLIBRE_DIFFUSION
                  visc_thermal = DMIN(visc_thermal, 0.5 * fac_vsic_fix * kernel.vdotr2 /
                                        ((local.Mass + P[j].Mass) *
                                         kernel.dwk_ij * kernel.r * dt));
#endif
              }
#endif
            } else {
              visc = 0;
            }

#ifdef SPHS
            /* Pressure limiter */
            double pressurelimiter = fabs(local.Pressure - SphP[j].Pressure) /
                                     (local.Pressure + SphP[j].Pressure);
```

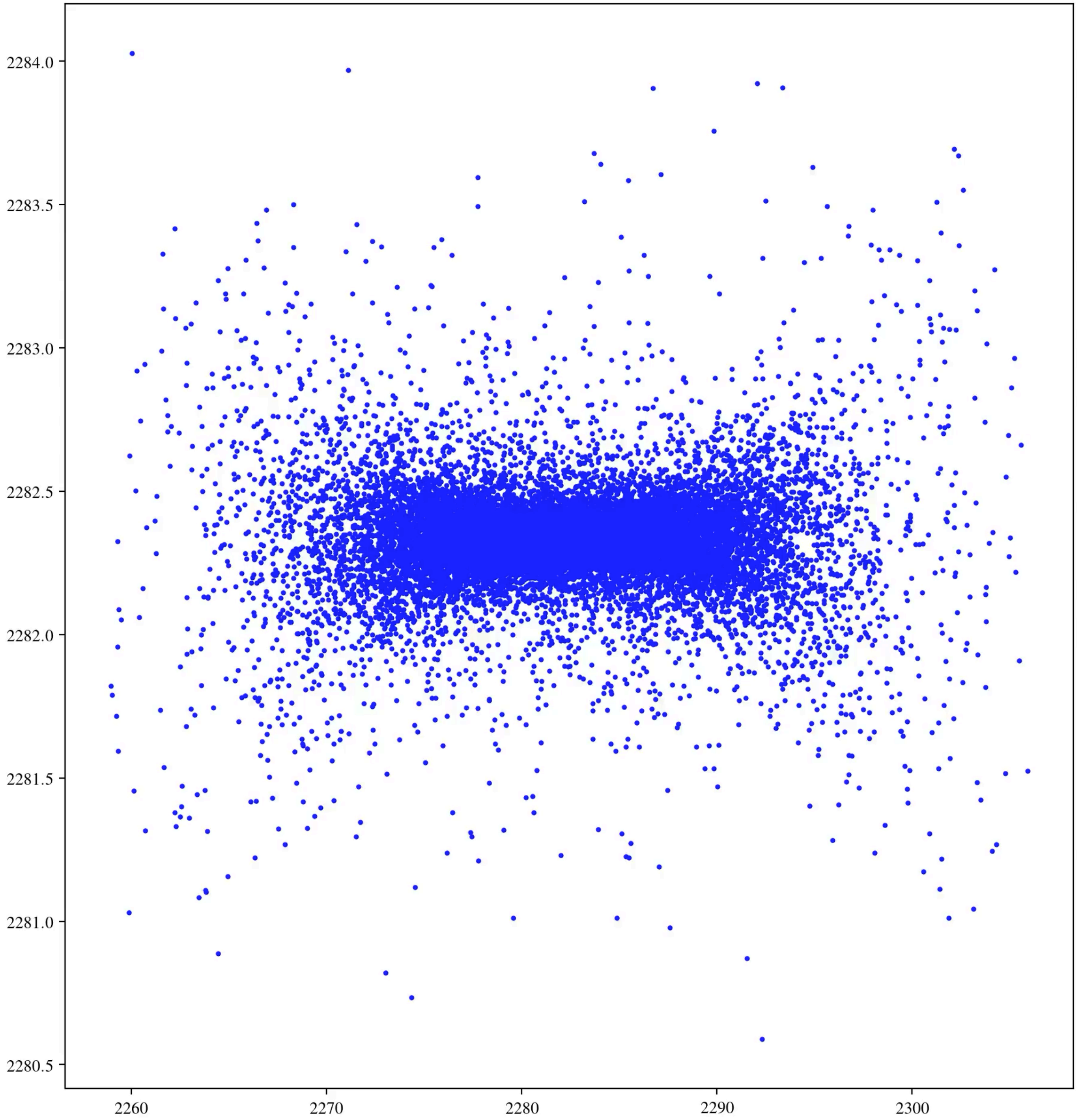
SWIFT

- SWIFT uses unit tests to ensure correctness and is much easier to work with.
- #ifdef is pretty much banned, replaced by real modularity.

```c
/* Balsara term */
const float balsara_i = pi->force.balsara;
const float balsara_j = pj->force.balsara;

/* Construct the full viscosity term */
const float rho_ij = rhoi + rhoj;
const float alpha = pi->viscosity.alpha + pj->viscosity.alpha;
const float visc =
    -0.25f * alpha * v_sig * mu_ij * (balsara_i + balsara_j) / rho_ij;

/* Convolve with the kernel */
const float visc_acc_term = 0.5f * visc * (wi_dr + wj_dr) * r_inv;

/* Compute gradient terms */
const float P_over_rho2_i = pressurei / (rhoi * rhoi) * pi->force.f;
const float P_over_rho2_j = pressurej / (rhoj * rhoj) * pj->force.f;

/* SPH acceleration term */
const float sph_acc_term =
    (P_over_rho2_i * wi_dr + P_over_rho2_j * wj_dr) * r_inv;

/* Assemble the acceleration */
const float acc = sph_acc_term + visc_acc_term;

/* Use the force Luke ! */
pi->a_hydro[0] -= mj * acc * dx[0];
pi->a_hydro[1] -= mj * acc * dx[1];
pi->a_hydro[2] -= mj * acc * dx[2];
```

Current status of SIMBA in SWIFT





Conclusions

- SWIFT is an open-source, modern replacement to Gadget-like codes that can run petascale simulations
- EAGLE and GEAR sub-grid models already in SWIFT
- SIMBA is currently being ported
- We have a huge suite of analysis tools all ready to go!
- Any questions please just ask!