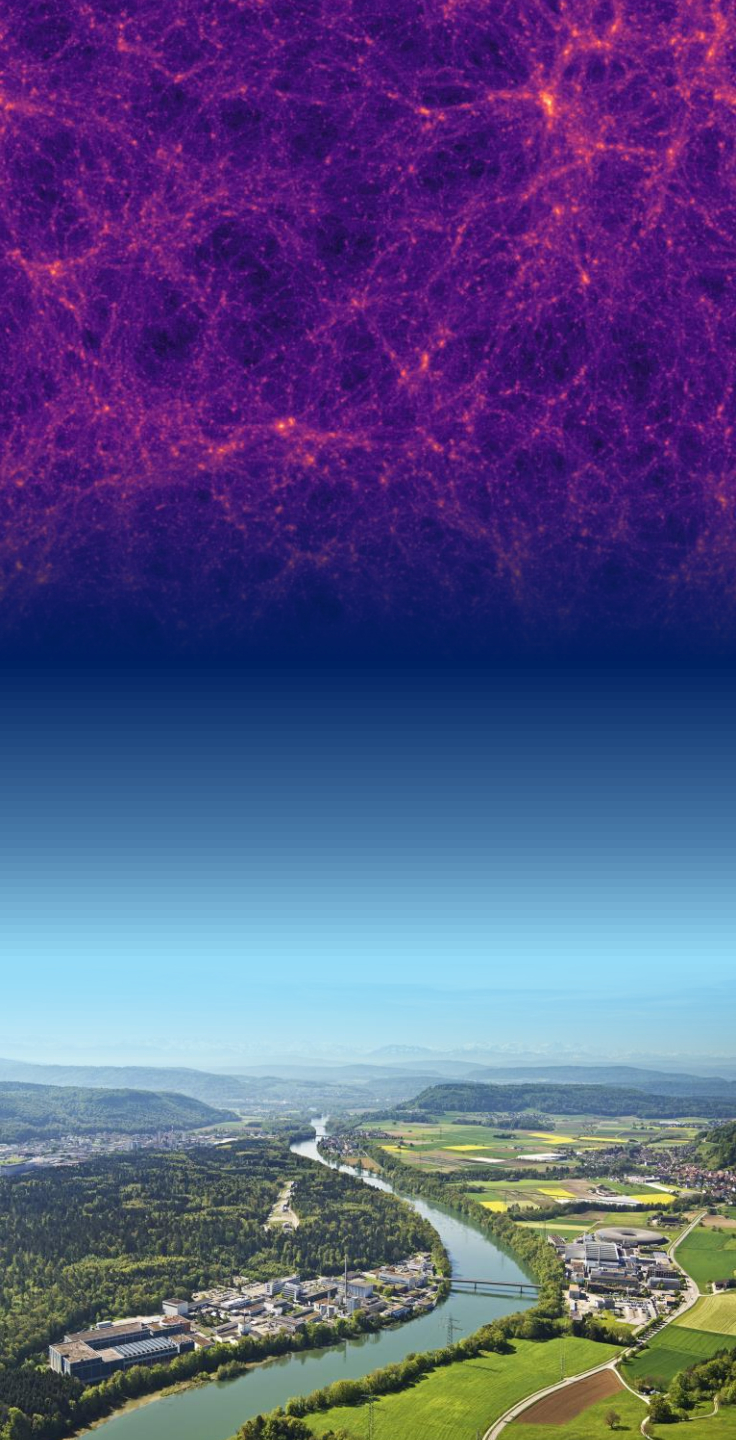


ETH zürich

PSI Center for Scientific Computing, Theory and Data

Final Presentation – Semester Thesis

Cosmological Structure Formation with IPPL

Blanca Crazzolara

18th July 2024

Outline

- Project Objective
- Theoretical Background
- Numerical Implementation
- Scaling Study Results
- Conclusion and Outlook

Thesis Objective

- Implement a **Mini App for Cosmological simulations** using IPPL
  → addition to Alpine for plasma physics

- Motivation: take advantage of similarities in the nature of the physical equations

- Demonstrate versatility of IPPL

Outline

- Project Objective
- Theoretical Background
- Numerical Implementation
- Scaling Study Results
- Conclusion and Outlook

Expansion of the Universe (ΛCDM)

Physical vs. Comoving Coordinates:

$$\mathbf{r}(t) = a(t)\mathbf{x}(t)$$

Hubble parameter (expansion rate):

$$H(t) = \frac{1}{a(t)}\frac{\mathrm{d}a(t)}{\mathrm{d}t}$$

First Friedmann equation:

$$H^2(t) = H_0^2\left(\Omega_{\mathrm{m},0}\frac{1}{a^3} + \Omega_{\Lambda,0}\right)$$

$$\Omega_{\Lambda,0} = \frac{\rho_{\Lambda,0}}{\rho_0} \approx 0.7 \quad \Omega_{m,0} = \frac{\rho_{m,0}}{\rho_0} \approx 0.3$$

Physical Equations for ΛCDM

Electrostatic

$$\Delta\Phi(\mathbf{r}) = -\nabla \cdot \mathbf{E}(\mathbf{r}) = -\frac{\rho(\mathbf{r})}{\epsilon_0}$$

$$\frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t} = \mathbf{v}_i$$

$$\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\nabla\Phi(\mathbf{r}_i)$$

→

Gravity

$$\Delta\Phi(\mathbf{x}) = 4\pi G a^2 \rho(\mathbf{x})$$

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} = \frac{1}{a^2}\mathbf{u}_i$$

$$\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} = -\nabla\Phi(\mathbf{x}_i)$$

Structure Formation

- Small perturbations in initially homogeneous density field grow over time. The overdensity is modelled as **Gaussian random field**.

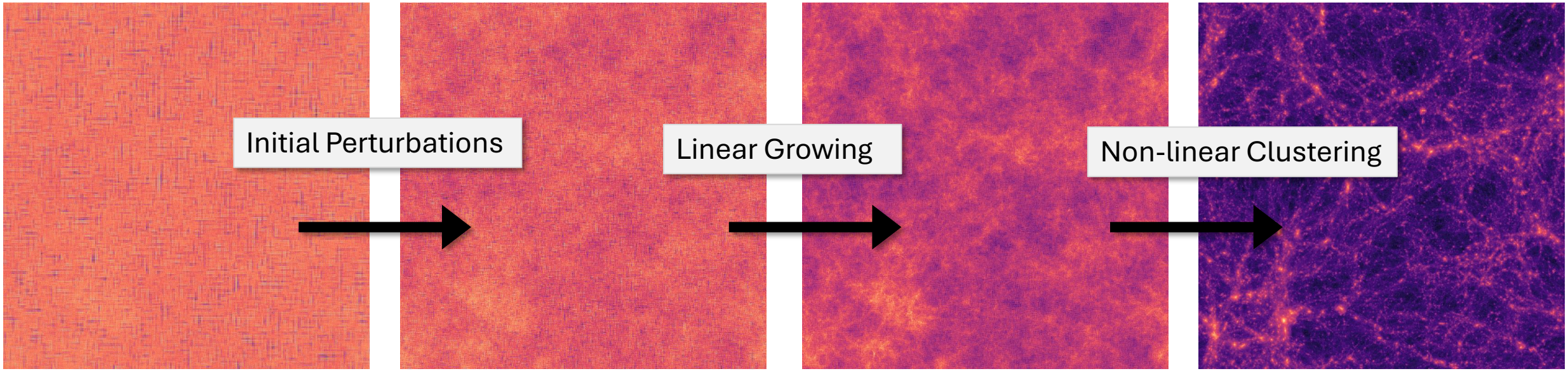
- Linear growing at early times

- Later non-linear clustering due to **gravitational collapse**

  → Formation of Dark Matter haloes and Galaxies

Background

Structure Formation



Scale Factor:	$a = 0.019$	$a = 0.036$	$a = 0.067$	$a = 0.53$
Redshift:	$z = 50.9$	$z = 26.86$	$z = 13.93$	$z = 1.29$

Outline

- Project Objective
- Theoretical Background
- Numerical Implementation
- Scaling Study Results
- Conclusion and Outlook

Initial Conditions

- Perturbations from a homogeneous density field.

- Generation with **N-GenIC** code (developed by the Max-Planck-Institute)
  → particle positions and velocities

- Saved as one CSV file and read into Mini App. Assignment of particles to corresponding processor domain.

Time Evolution

Particle-Mesh method from IPPL

- No. of grid points = No. of particles ($16^3 - 512^3$)
- Poisson equation solved on the grid (heFFTe) with periodic boundary conditions.
- Leapfrog for position and velocity updates of the particles

Leapfrog

- Timestep

$$\Delta t = \frac{1}{H}\Delta(\log a)$$

with fixed $\Delta(\log a)$ (smaller steps during fast expansion)

- Kick-Drift-Kick Scheme

$$\mathbf{u}_i^{(k+1/2)} = \mathbf{u}_i^{(k)} + \mathbf{a}_i^{(k)} \frac{\Delta t}{2} = \mathbf{u}_i^{(k)} - \nabla\Phi|_{\mathbf{x}_i^{(k)}} \frac{\Delta(\log a)}{2H(t_k)}$$

$$\mathbf{x}_i^{(k+1)} = \mathbf{x}_i^{(k)} + \frac{1}{a^2(t_{k+1/2})}\mathbf{u}_i^{(k+1/2)} \Delta t = \mathbf{x}_i^{(k)} + \frac{1}{a^2(t_{k+1/2})}\mathbf{u}_i^{(k+1/2)} \frac{\Delta(\log a)}{H(t_{k+1/2})}$$

$$\mathbf{u}_i^{(k+1)} = \mathbf{u}_i^{(k+1/2)} + \mathbf{a}_i^{(k+1)} \frac{\Delta t}{2} = \mathbf{u}_i^{(k+1/2)} - \nabla\Phi|_{\mathbf{x}_i^{(k+1)}} \frac{\Delta(\log a)}{2H(t_{k+1})}$$

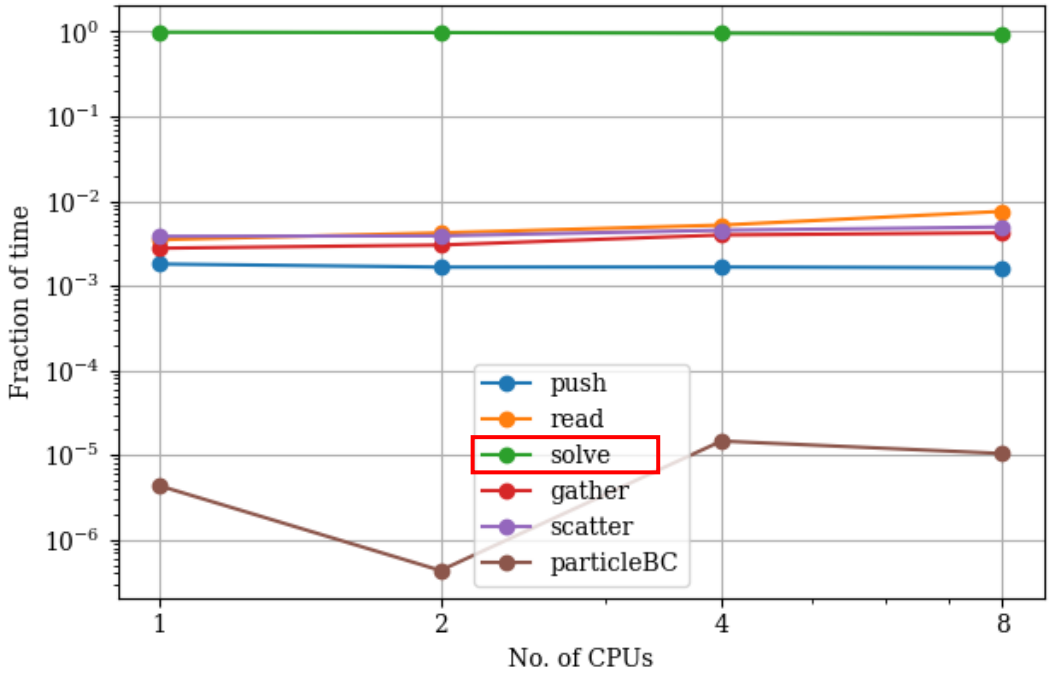
Outline

- Project Objective
- Theoretical Background
- Numerical Implementation
- Scaling Study Results
- Conclusion and Outlook

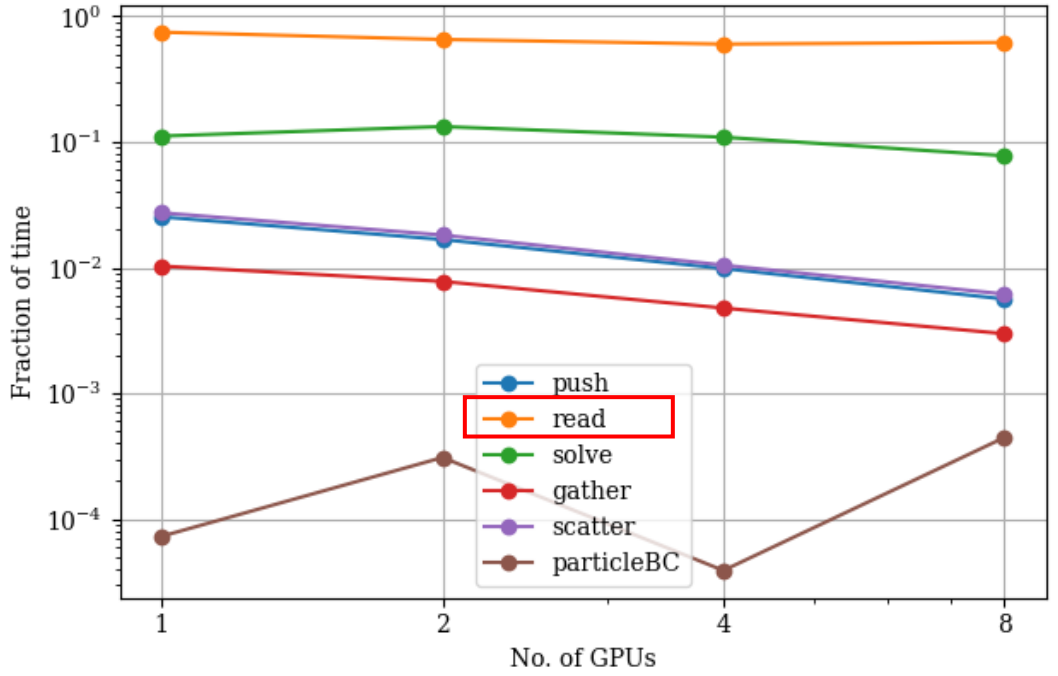
Results

Timing Contributions

CPUs on Merlin



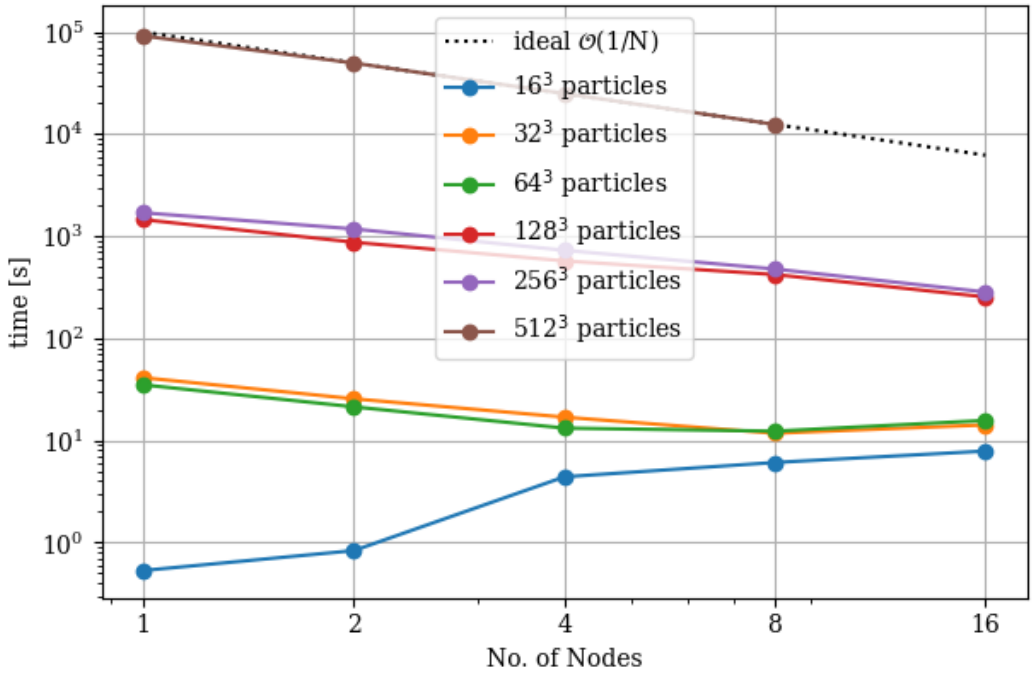
GPUs on Gwendolen

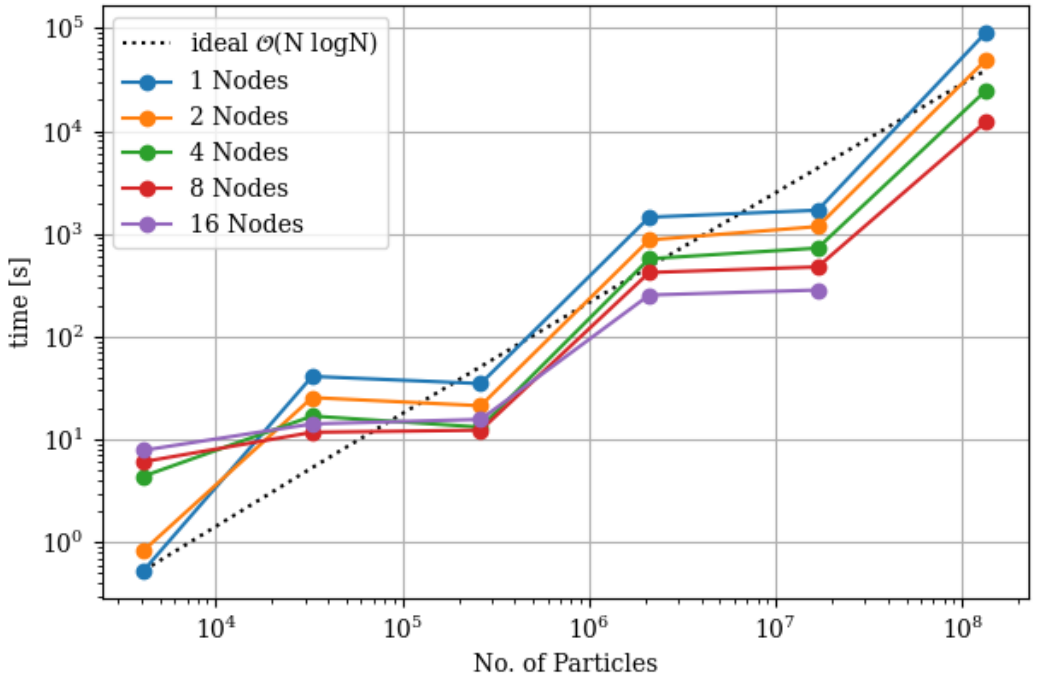


Example Problem size: 512^3 particles, 500 timesteps

Results

CPU Solver Scaling (1 CPU per node)

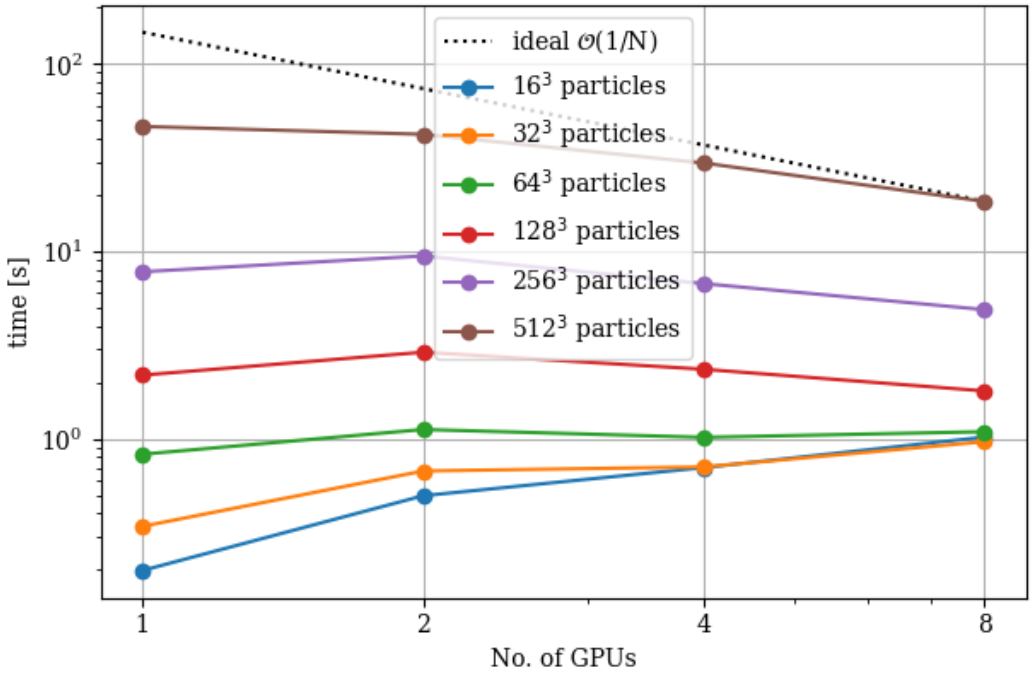


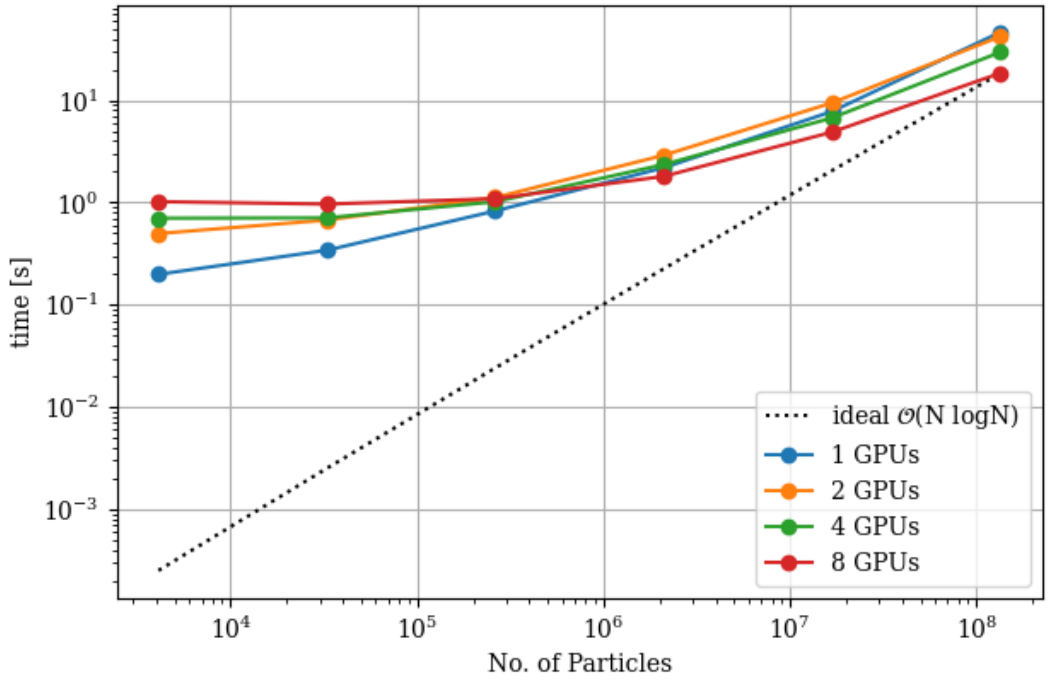


→ Good scaling performance for large problem sizes ($\sim 128^3$)

Results

GPU Solver Scaling (on 1 node)





→ Speed-up only at very large problem sizes ($> 512^3$).
Communication overhead before.

Conclusion

- Code successfully tested on CPUs on Merlin and Euler and GPUs on Gwendolen.
- Solver scales well on CPUs, on GPUs only for very large problem sizes.
- Read-In is extremely inefficient on GPUs. Improvement required.

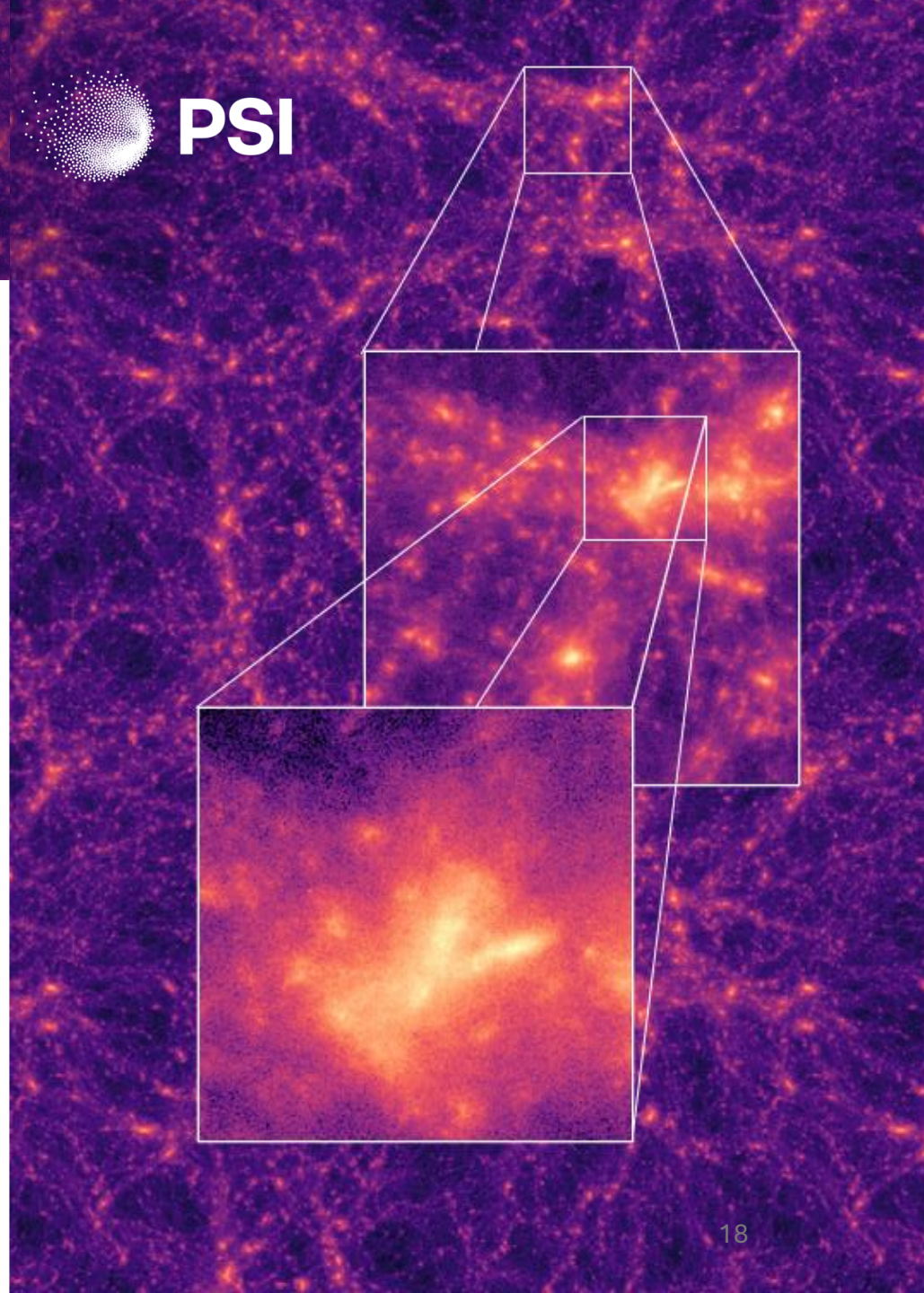
Outlook

- Adaptive Mesh or P^3M method.
  Cosmological simulations have extremely large density differences.

- Include ICs generation inside Mini App
  Improve efficiency, especially for GPUs.

- Adding Baryonic Matter
  Simulation of Galaxies.

- Usage for Cosmology Research
  Dark matter halo function
  Galaxy formation



Thank you for listening.

Questions?