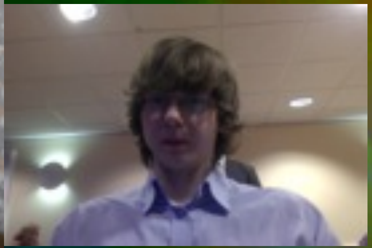
Swift: task-based hydrodynamics at Durham's IPCC

For the cosmological simulations of the formation of galaxies






movie: Richard Bower (Durham)

Tom Theuns
Institute for Computational Cosmology
Durham

DiRAC VIRGO belspo ICC Institute for Computational Cosmology SEVENTH FRAMEWORK PROGRAMME PRACE COSMOCOMP intel

Contents:

- Introduction: cosmological simulations: aims and methods
- computational challenges
- load - (im) balance
- Swift: task-based hydrodynamics and gravity

related talks/posters:

keynote: Simon Portegies-Zwart:

Massively-parallel GPU-accelerated galaxy simulation

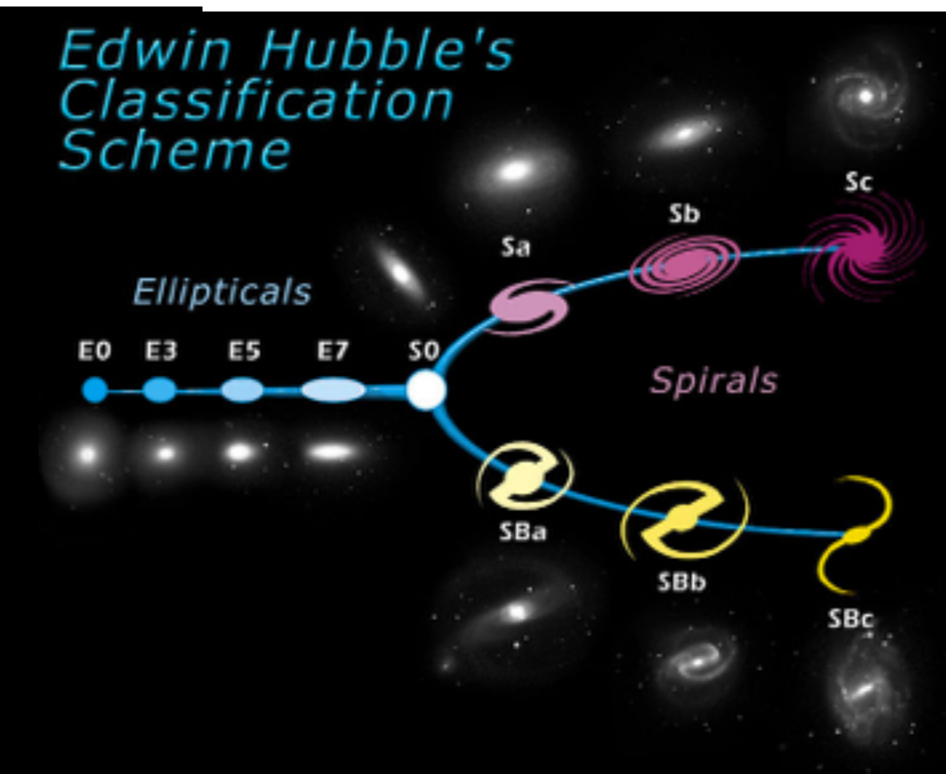
Poster: Karakasis et al

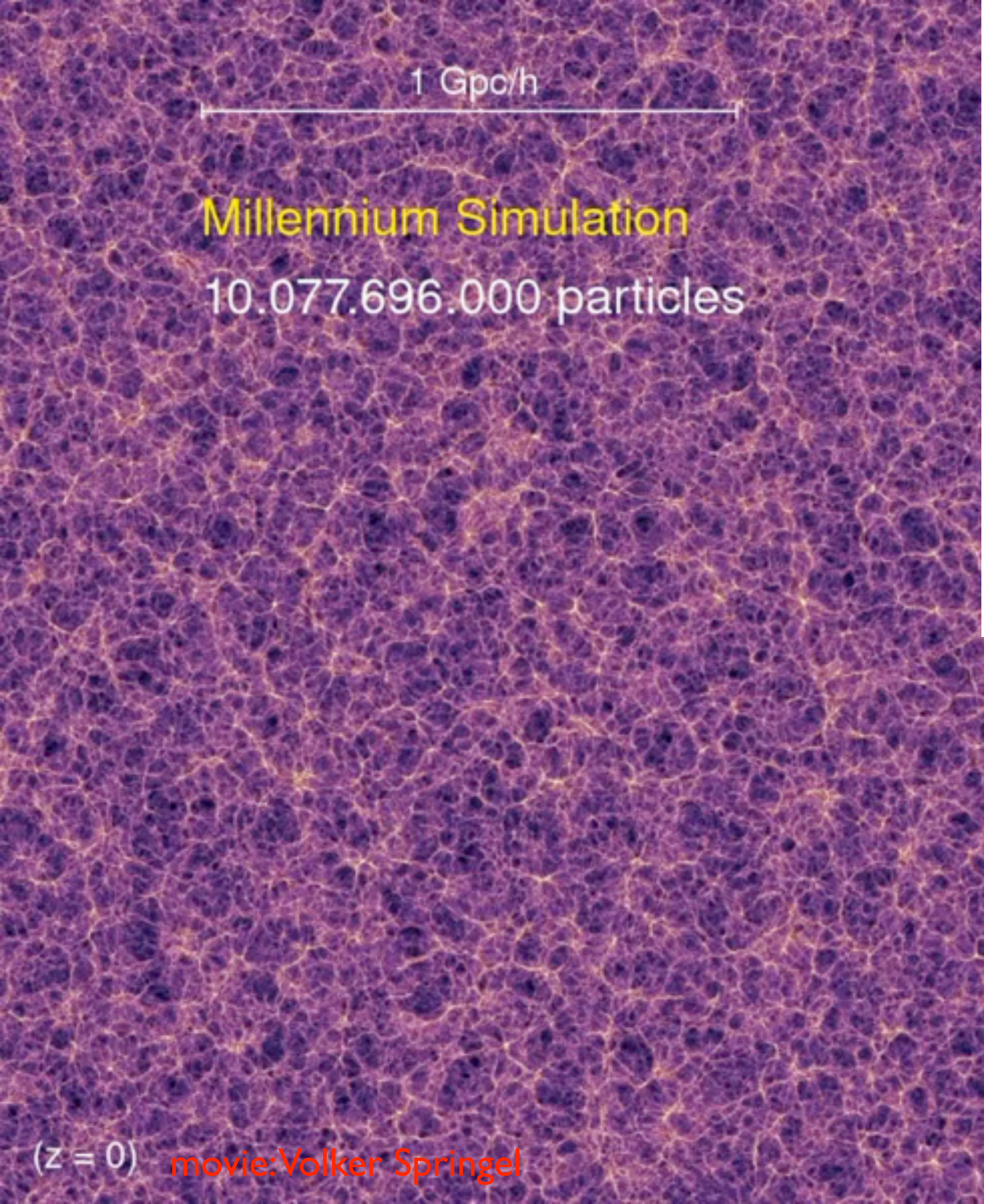
Gadget on the Mike

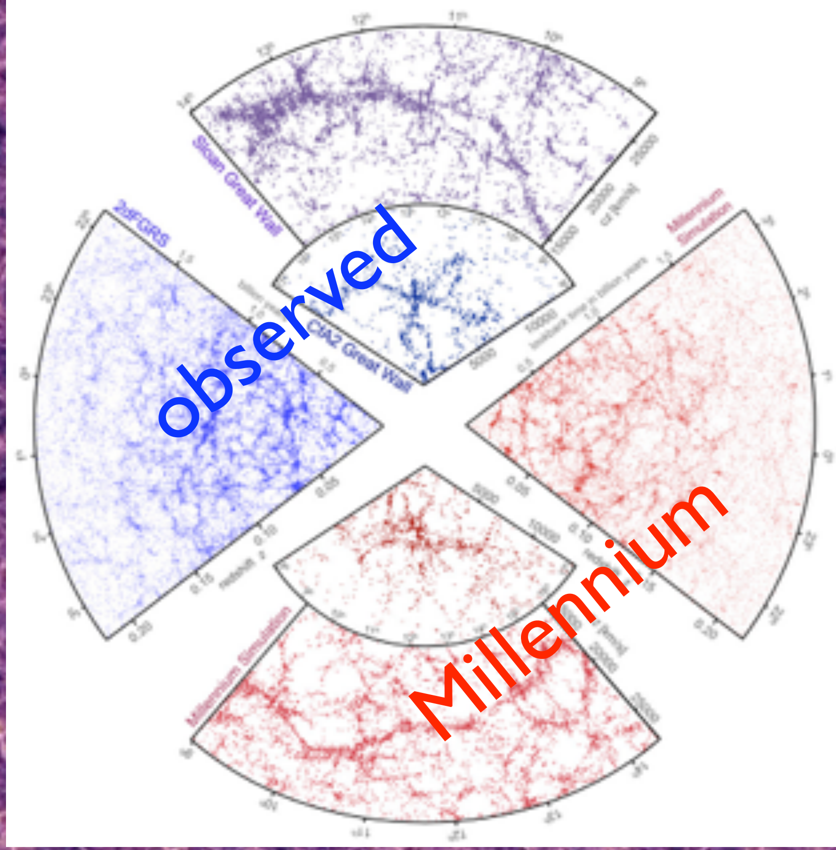
Several talks/posters on SPH (hydrodynamics scheme)

.e.g Guo: Exploring the Memory-Efficient Implementation Model for Incompressible Smoothed Particle Hydrodynamics(ISPH)

Introduction



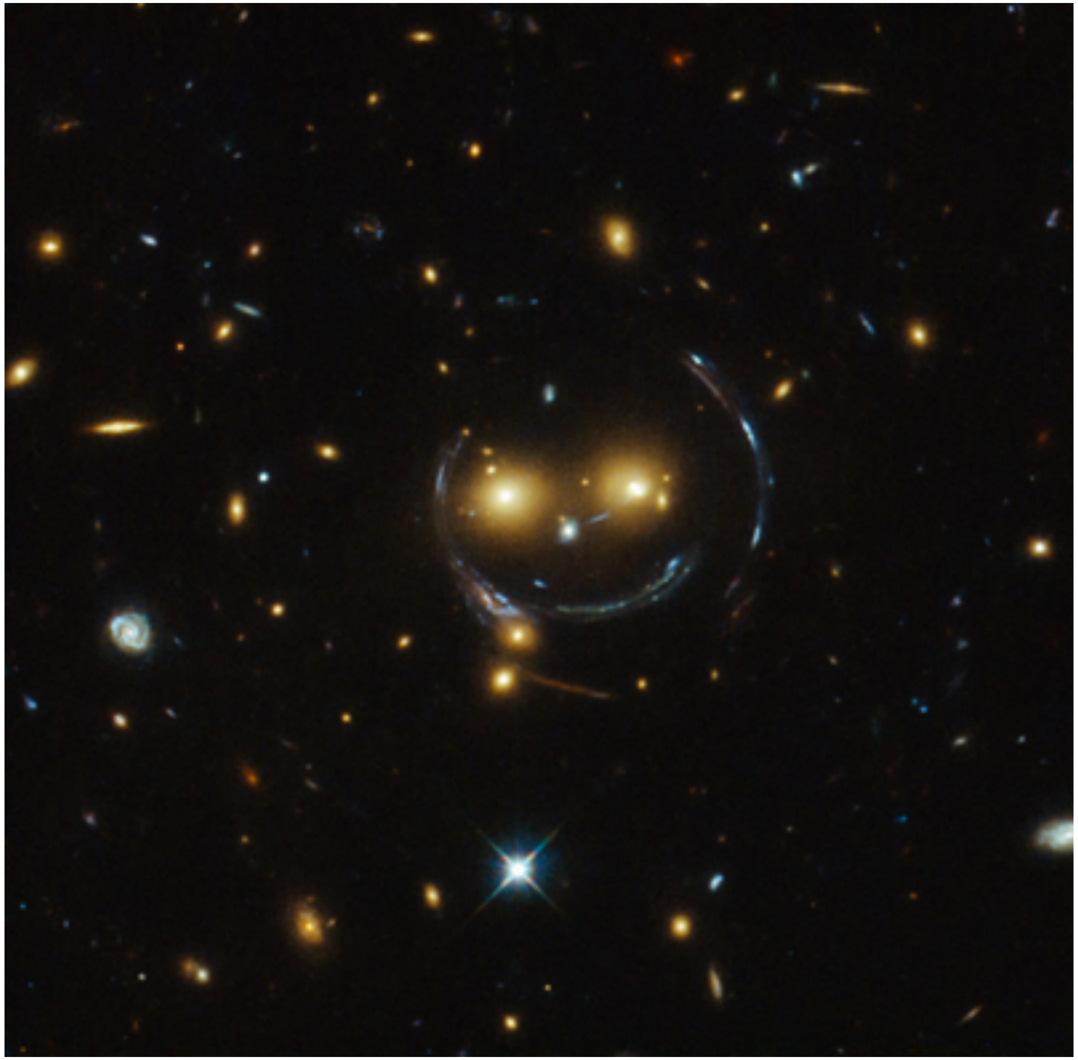




movie: Volker Springel

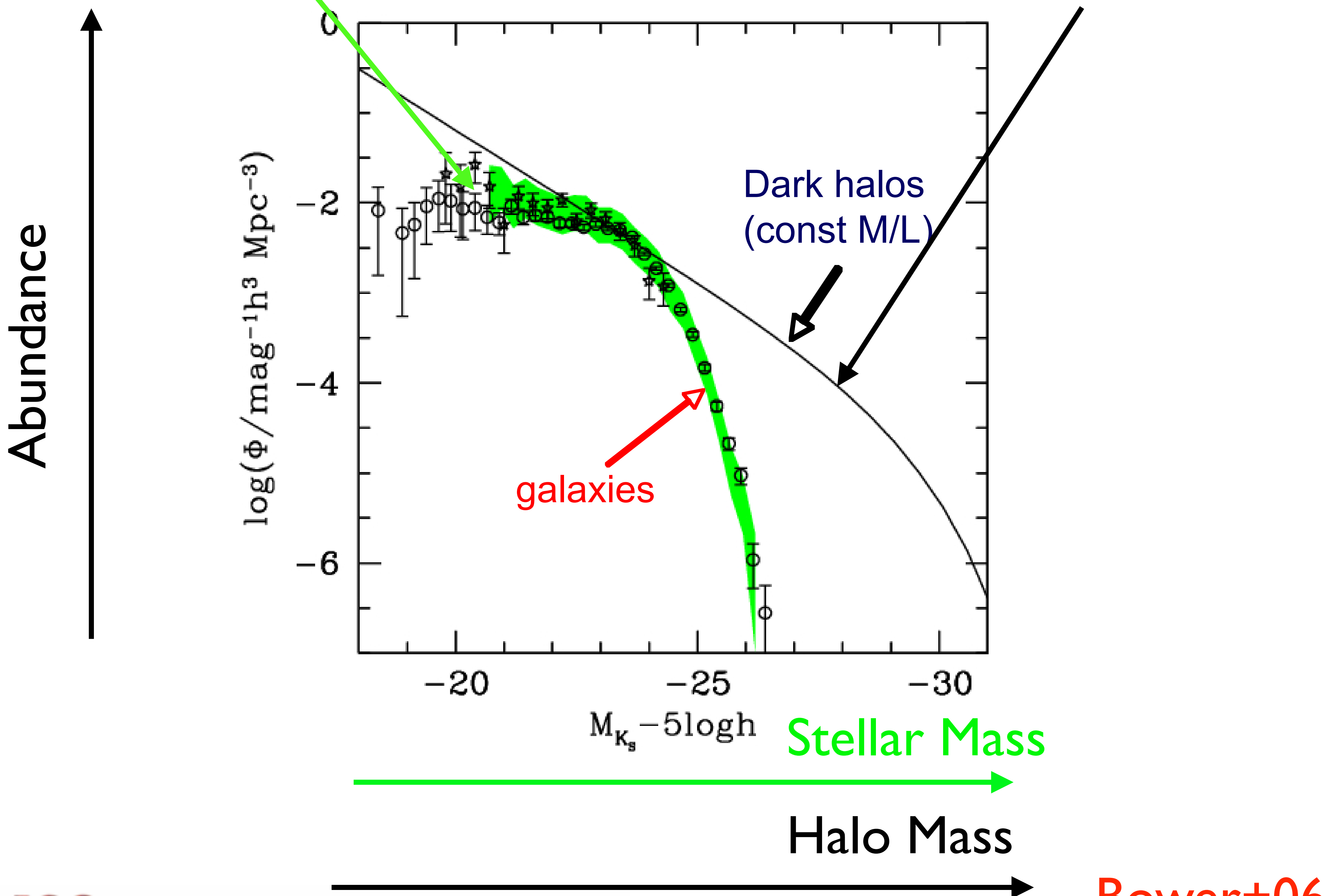
Springel +05
Virgo project

Smiling galaxies (because they contain dark matter)



SDSS J1038+4849

Galaxy stellar mass function versus dark matter halo mass function



Physics of galaxy formation

Aims:

- How do galaxies form?
- How do they evolve?
- Which physical processes operate?

Basic paradigm

(White & Rees, White & Frenk)

- Dark haloes form
- Cool(ed) gas forms discs
- Discs fragment to form stars

Multi-scale/complex/rich problem

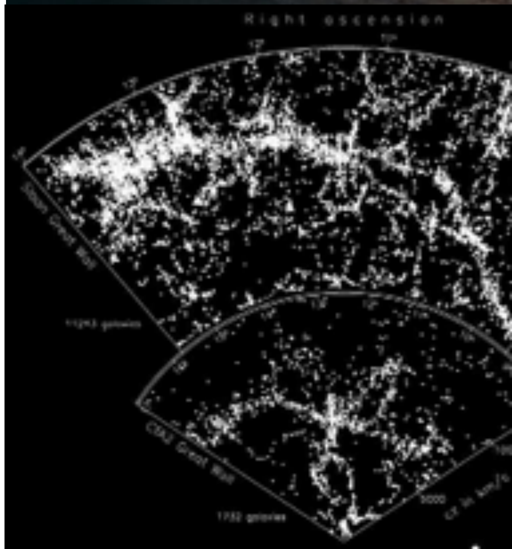
2 pc

x 10000

20 kpc

x 10000

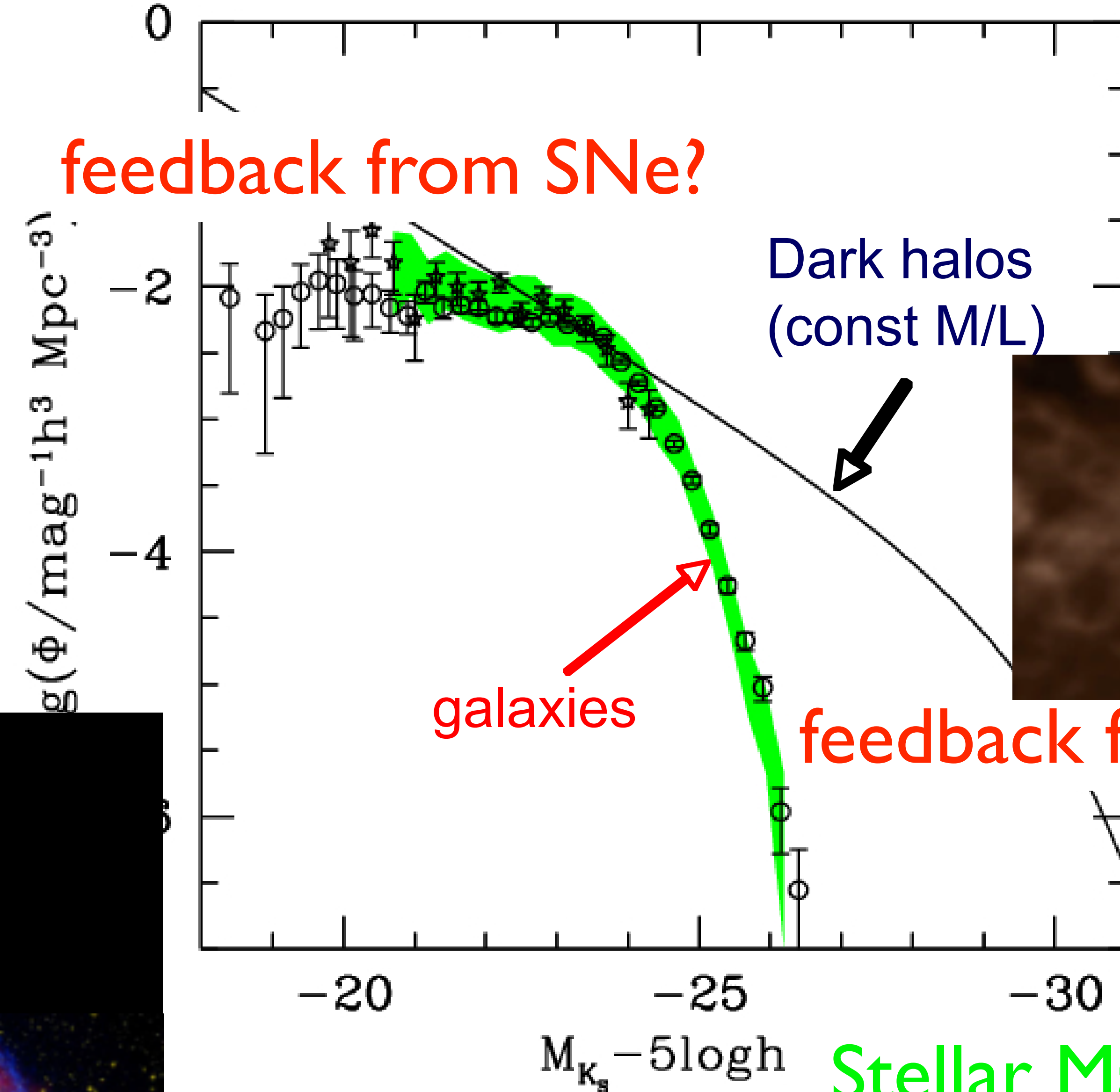
200 Mpc



Galaxy stellar mass function versus dark matter halo mass function

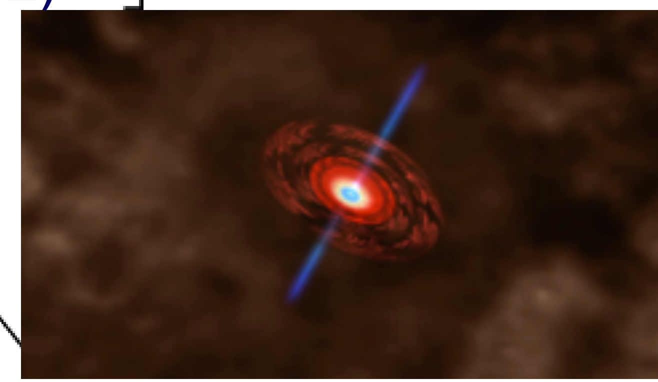
Abundance

feedback from SNe?



Dark halos
(const M/L)

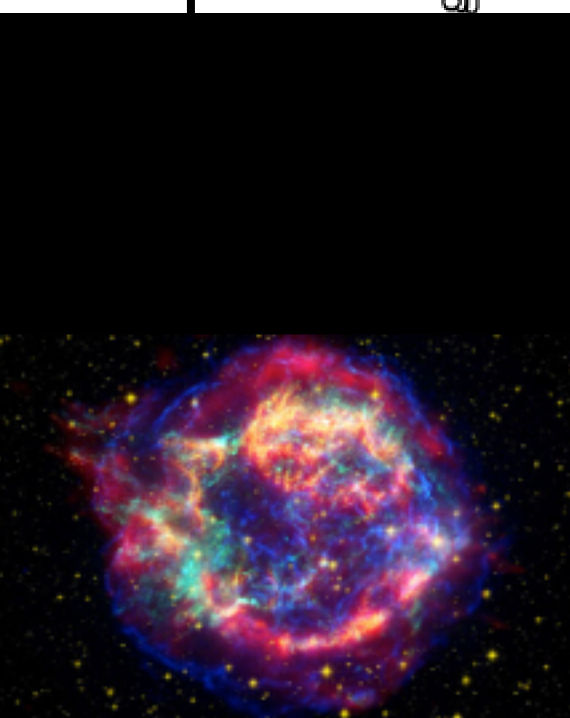
galaxies



feedback from AGN?

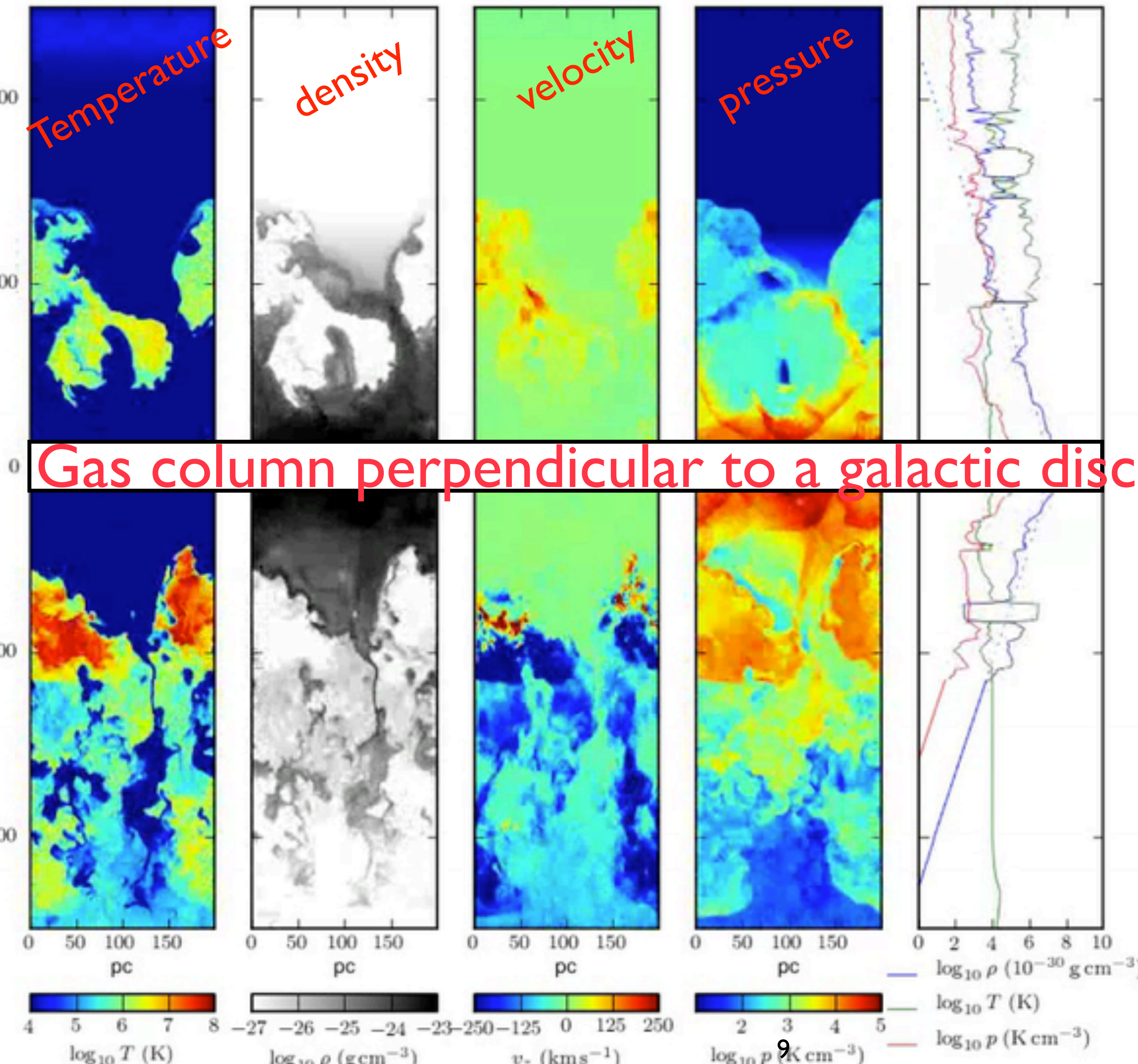
Stellar Mass

Halo Mass



Bower+06

Physics of supernova feedback



Gas column perpendicular to a galactic disc

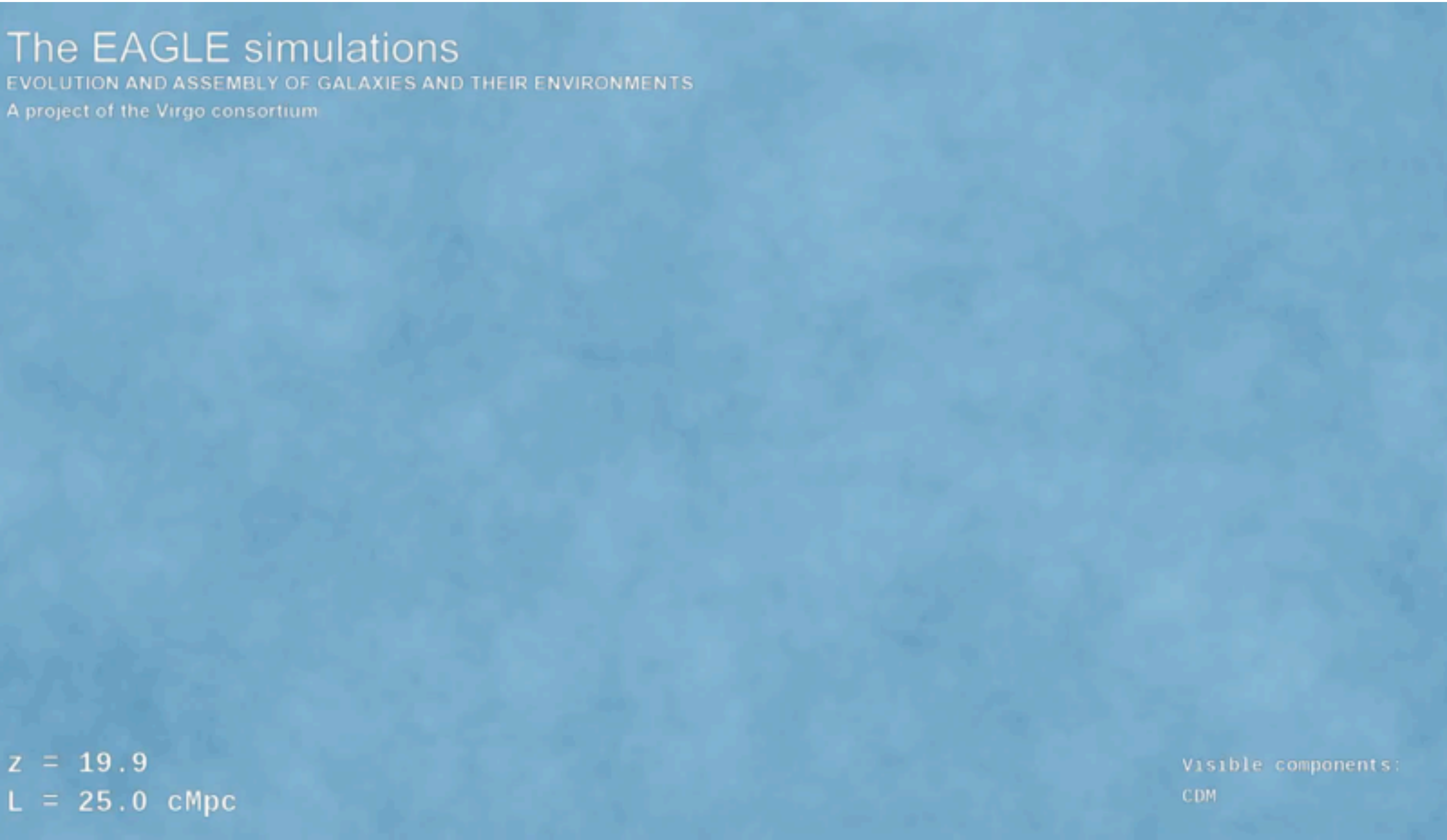
Creasey+13, 15

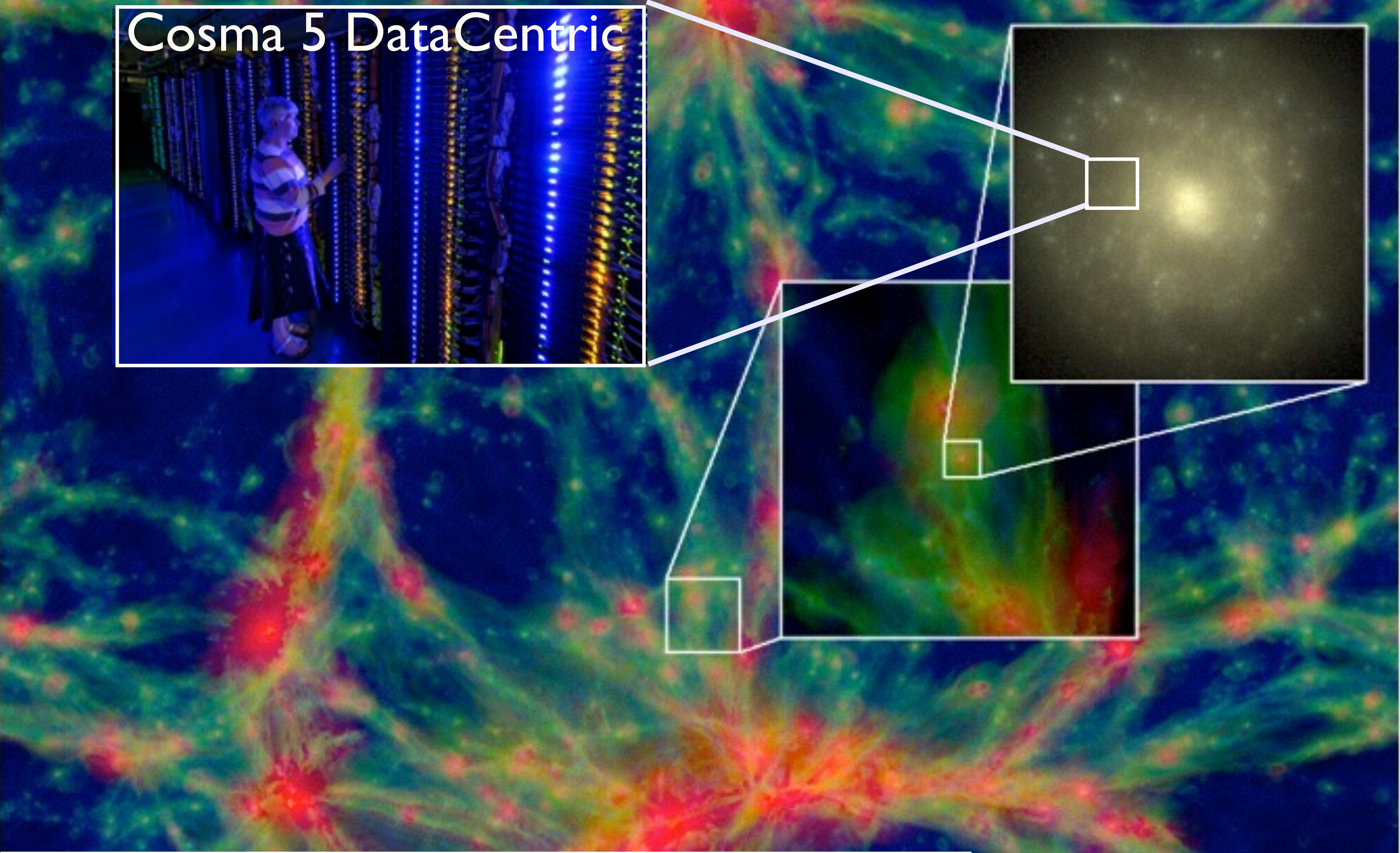
FLASH

30 Jupiter mass resolution

Current state of the art: Eagle simulations (Schaye +15)

http://icc.dur.ac.uk/Eagle/



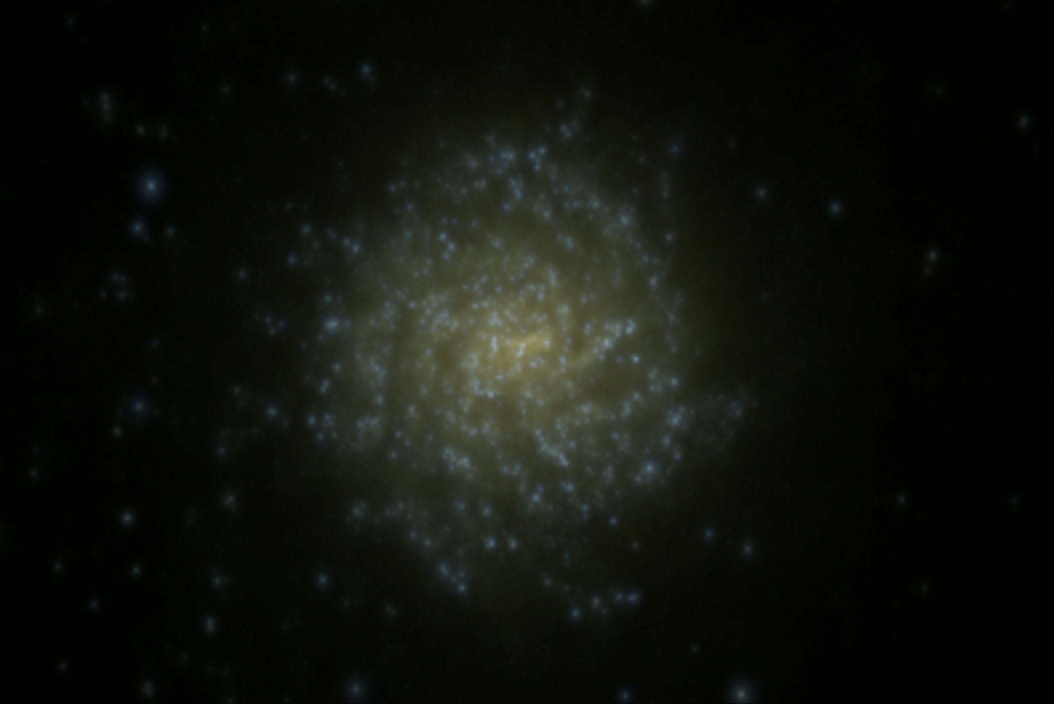


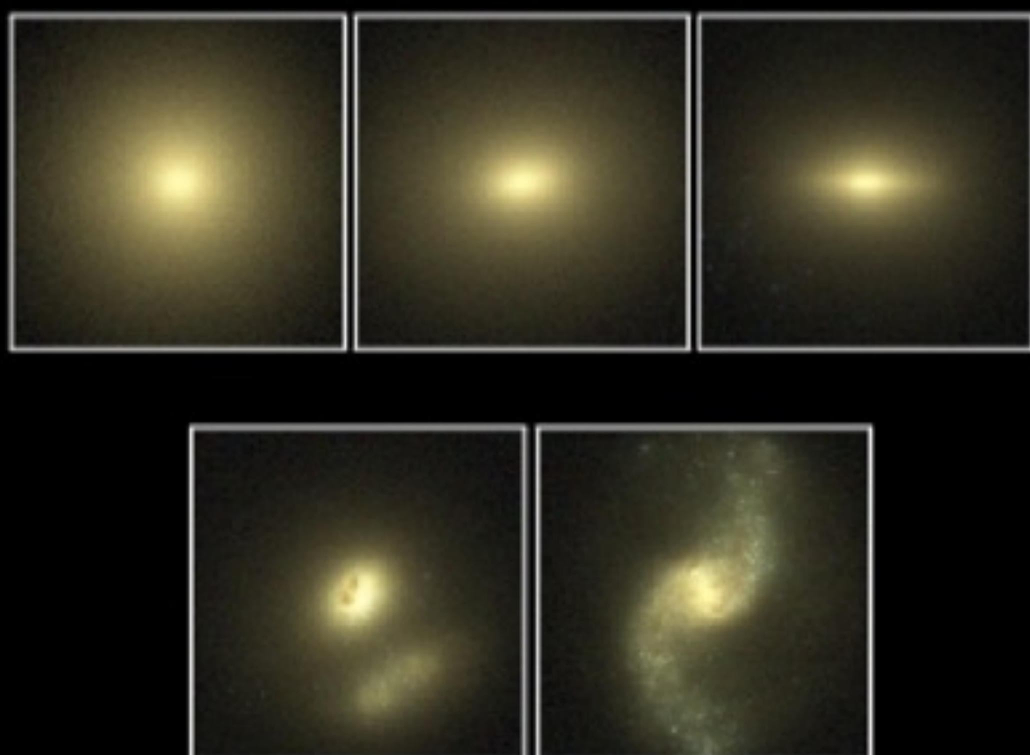
Name	L (comoving Mpc)	N	$m_{\rm g}$ $(\rm M_\odot)$	$m_{\rm dm}$ $(\rm M_\odot)$	$\epsilon_{\rm com}$ (comoving kpc)	$\epsilon_{\rm prop}$ (proper kpc)
L025N0376	25	376^3	1.81×10^6	9.70×10^6	2.66	0.70
L025N0752	25	752^3	2.26×10^5	1.21×10^6	1.33	0.35
L050N0752	50	752^3	1.81×10^6	9.70×10^6	2.66	0.70
L100N1504	100	1504^3	1.81×10^6	9.70×10^6	2.66	0.70

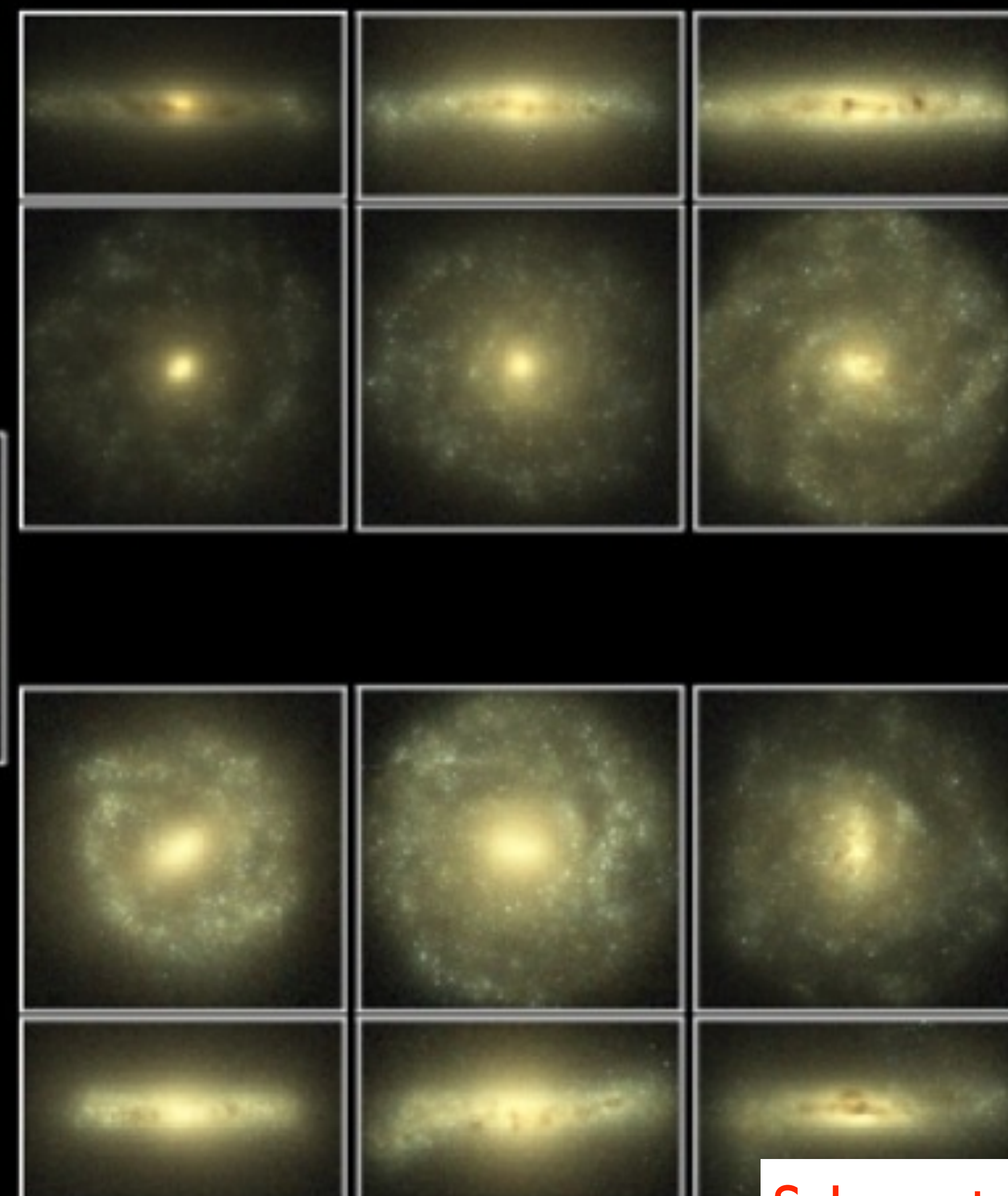
7 M CPU hours

Schaye +15

The Hubble Sequence







Schaye +15

Contents:

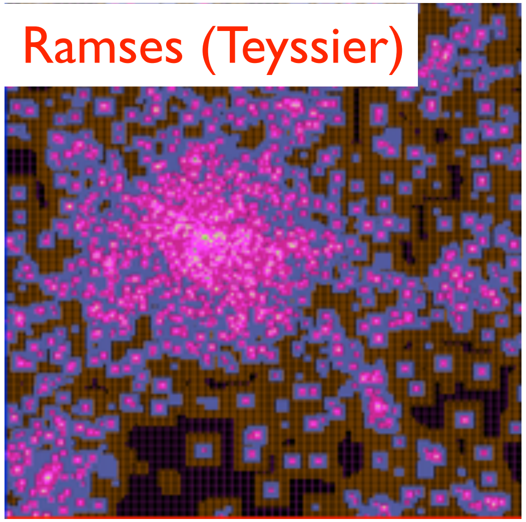
- Introduction: cosmological simulations: aims and methods
- computational challenges
- load - (im) balance
- Swift: task based hydrodynamics and gravity

Methods: algorithms and implementations

- gravity (from gas, stars and dark matter)
- hydrodynamics (gas accretion, gas cooling)

$$\frac{d\rho}{dt} \equiv \frac{\partial \rho}{\partial t} - \mathbf{v} \cdot \nabla \rho = -\rho \nabla \mathbf{v}$$

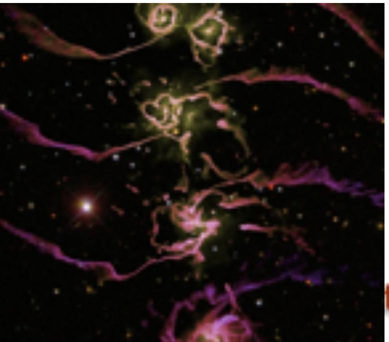
Ramses (Teyssier)



Eulerian hydrodynamics (AMR codes)

Highly optimised algorithms - minimise calculations!

Gadget (Springel)



Lagrangian hydrodynamics (SPH codes)

Lagrangian hydrodynamics

Mon. Not. R. Astron. Soc. **364,** 1105–1134 (2005)

The cosmological simulation code GADGET-2

Volker Springel★
Max-Planck-Institut für Astrophysik, Karl-Schwarzschild-Straße 1, 85740 Garching bei München, Germany

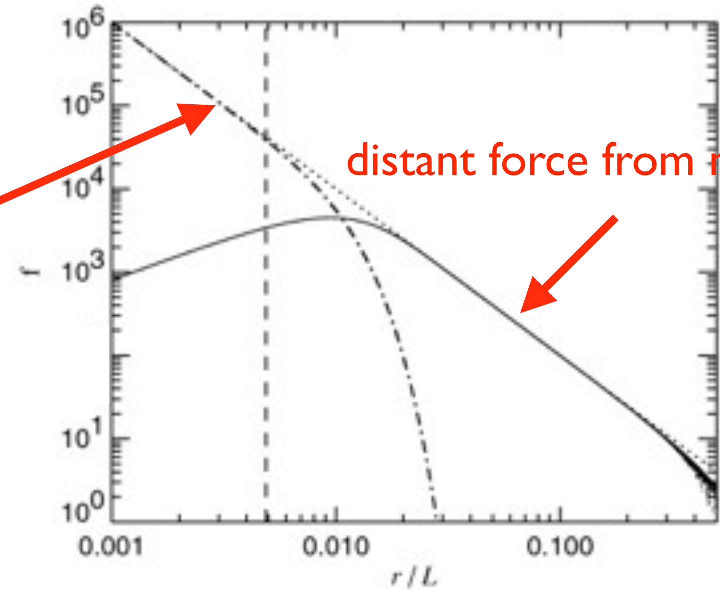
$$\rho(\mathbf{r}_i) = \sum_j m_j W(|\mathbf{r}_i - \mathbf{r}_j|, h_i)$$

density is found by summing weighted mass over neighbours

$$W(r,h) = \frac{8}{\pi h^3}\begin{cases} 1-6\left(\frac{r}{h}\right)^2+6\left(\frac{r}{h}\right)^3, & 0 \leqslant \frac{r}{h} \leqslant \frac{1}{2}, \\ 2\left(1-\frac{r}{h}\right)^3, & \frac{1}{2} < \frac{r}{h} \leqslant 1, \\ 0, & \frac{r}{h} > 1. \end{cases}$$

Gravity:

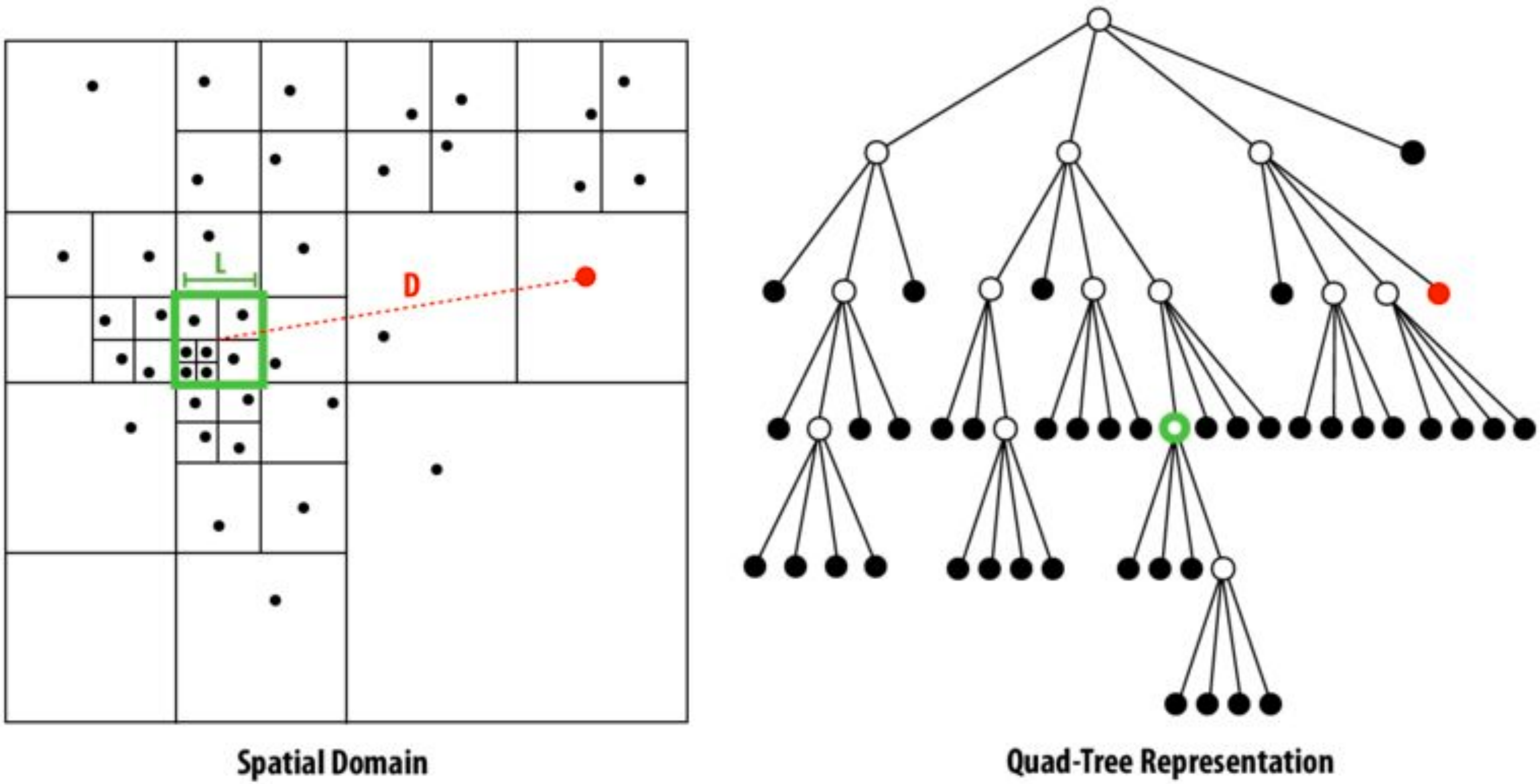
$$\mathbf{F}_{\mathrm{grav},i} = \sum_j \frac{m_j}{r_{ij}^3}\,\mathbf{r_{ij}}$$



distant force from mesh (FFTs)

nearby force calculated from tree

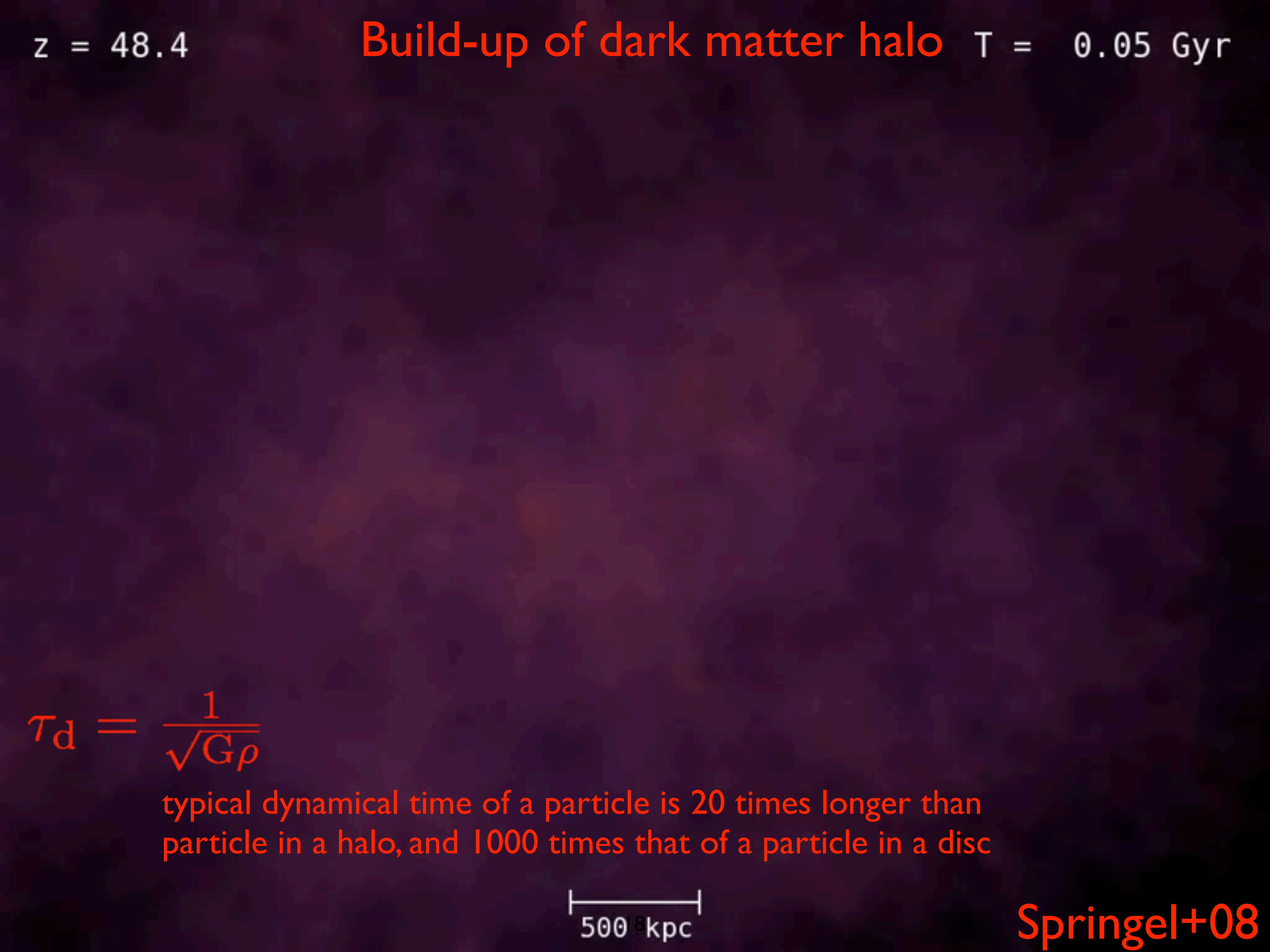
Tree calculation (in 2 dimensions)



Contents:

- Introduction: cosmological simulations: aims and methods
- computational challenges
- load - (im) balance
- Swift: task based hydrodynamics and gravity

Build-up of dark matter halo



$$\tau_{\mathrm{d}} = \frac{1}{\sqrt{G\rho}}$$

typical dynamical time of a particle is 20 times longer than particle in a halo, and 1000 times that of a particle in a disc

Springel+08

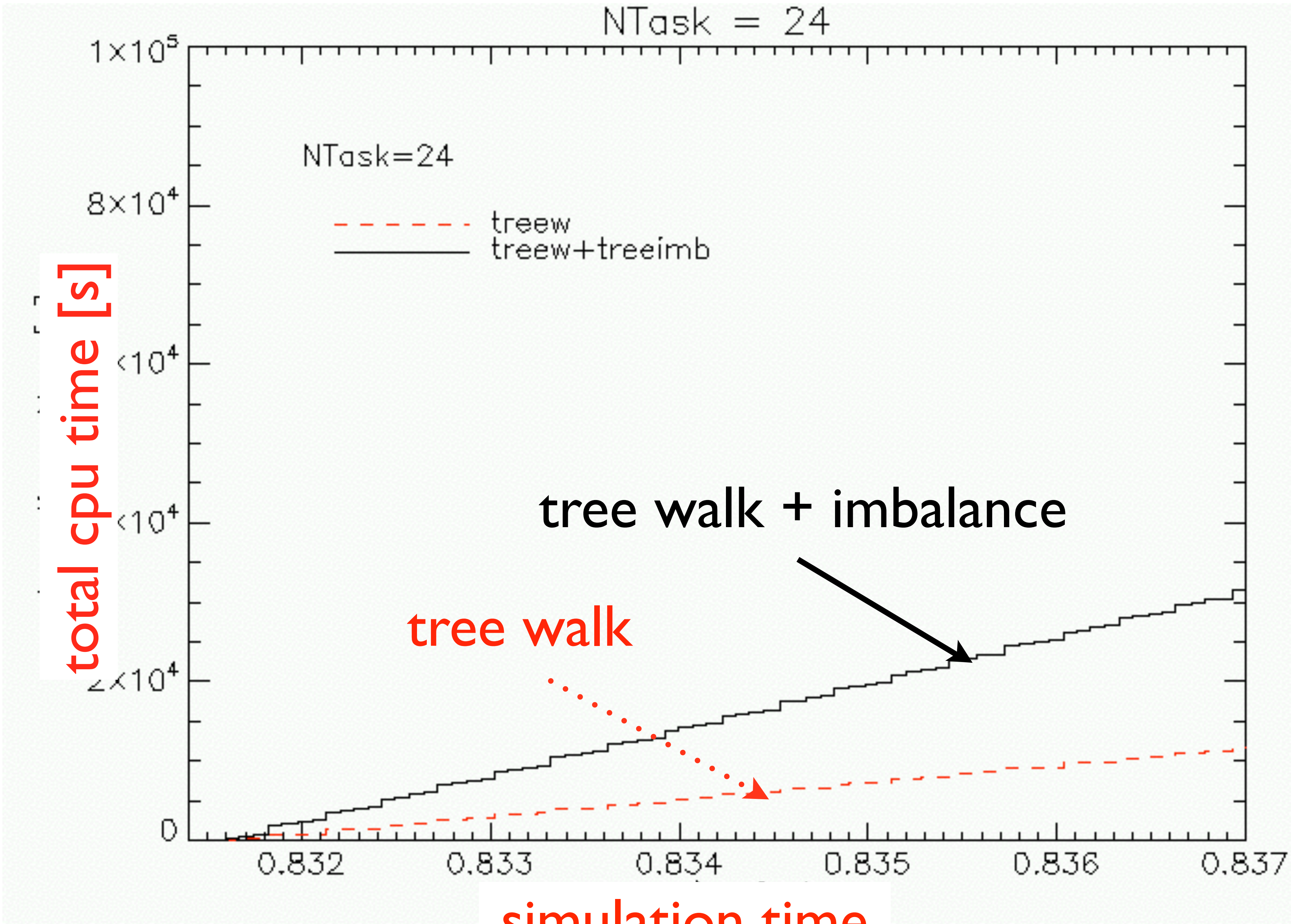
$$\rho(\mathbf{r}_i) = \sum_j m_j W(|\mathbf{r}_i - \mathbf{r}_j|, h_i)$$

- Amount of memory per particle is large (many diagnostic properties stored, such as metallicity, star formation rate, etc), little computational time spent on interaction between two particles
- Runs require a lot of memory (and hence many cores/ nodes): currently 10 billion particle runs
- Lots of time spent in **finding** neighbours: leads to load-imbalance

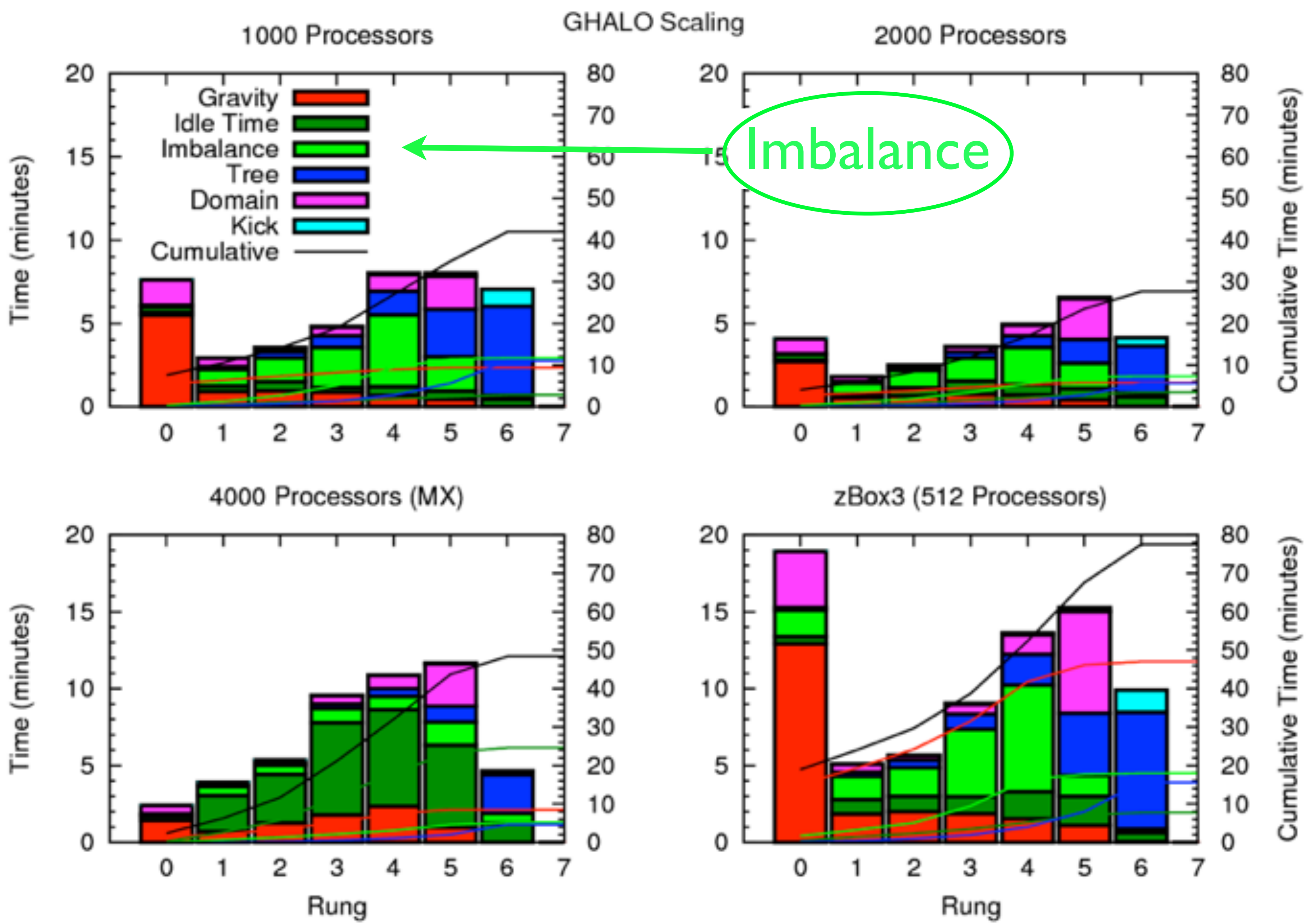
Contents:

- Introduction: cosmological simulations: aims and methods
- computational challenges
- load - (im) balance
- Swift: task based hydrodynamics and gravity

Load (im) balance in gravity calculation (Gadget-2)

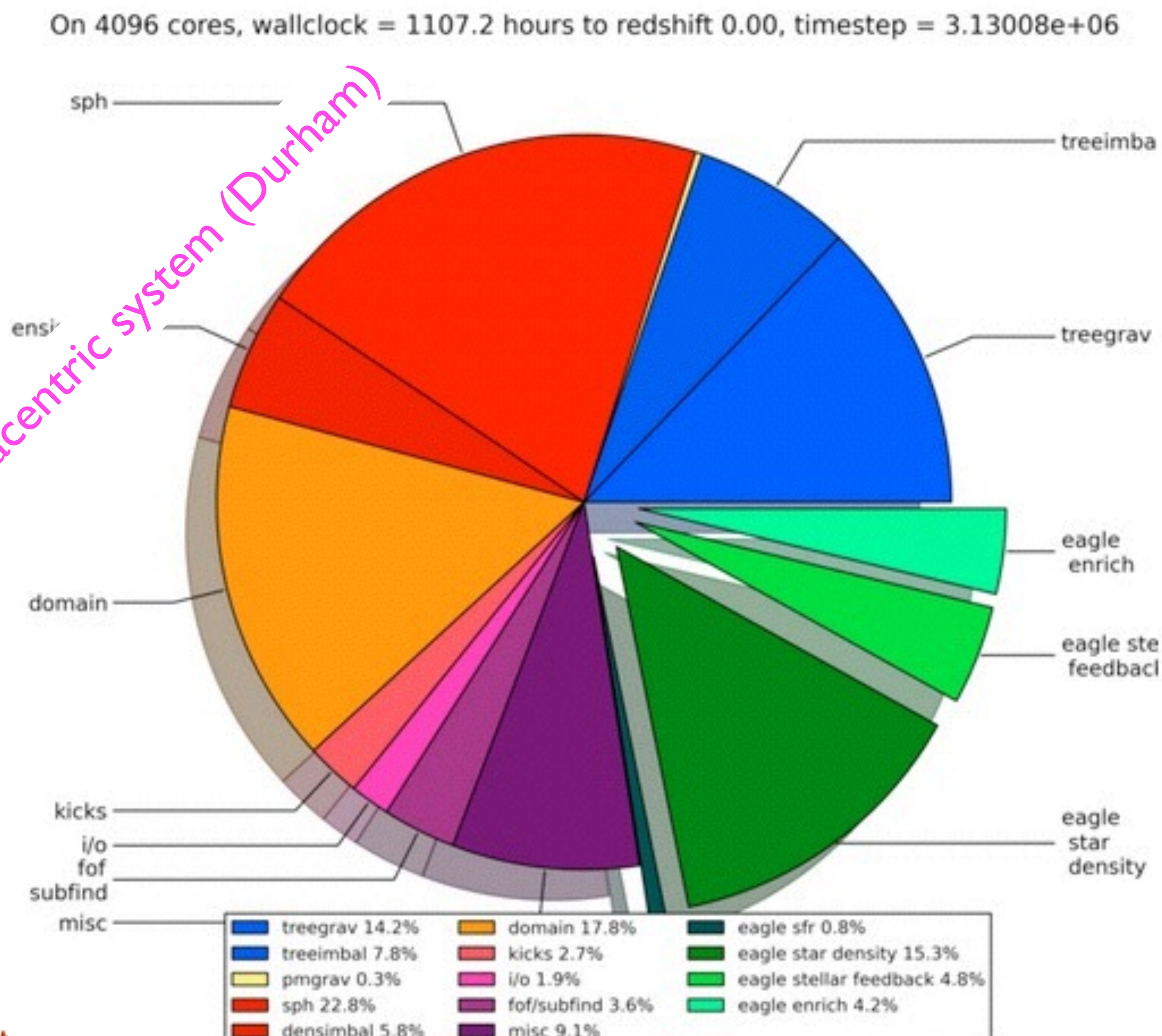


Timings of pkdgrav on G-halo from Stadel



1500^3 Eagle reference run

4096 cores



Cosma 5, Datacentric system (Durham)

Contents:

- Introduction: cosmological simulations: aims and methods
- computational challenges
- load - (im) balance
- Swift: task based hydrodynamics and gravity

Task based parallelism for SPH/Gravity

SWIFT: Fast algorithms for multi-resolution SPH on multi-core architectures

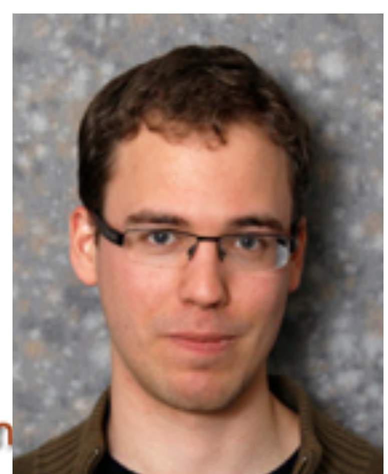
Pedro Gonnet[*], Matthieu Schaller[†], Tom Theuns[†‡], Aidan B. G. Chalk[*]

EFFICIENT AND SCALABLE ALGORITHMS FOR SMOOTHED PARTICLE HYDRODYNAMICS ON HYBRID SHARED/DISTRIBUTED-MEMORY ARCHITECTURES

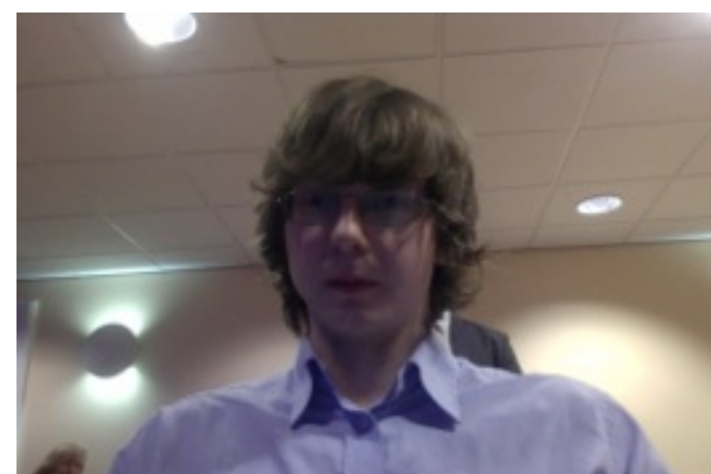
PEDRO GONNET[*]

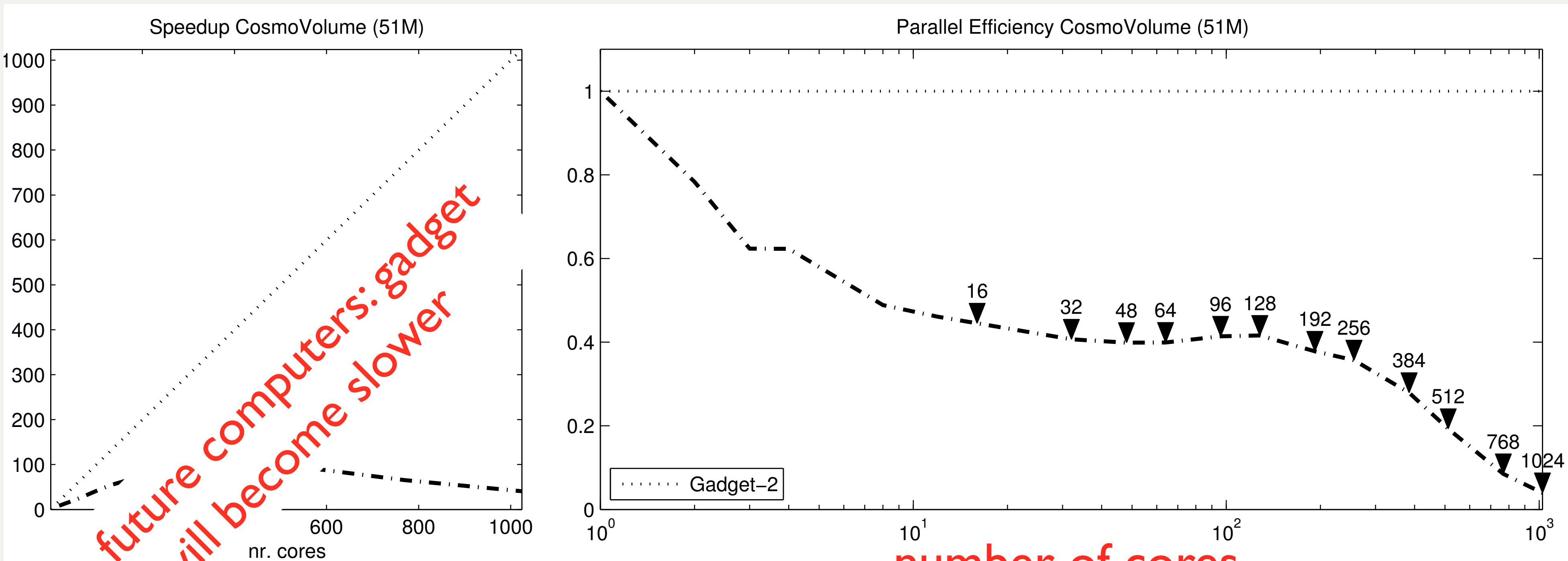
Gonnet

Schaller



Chalk



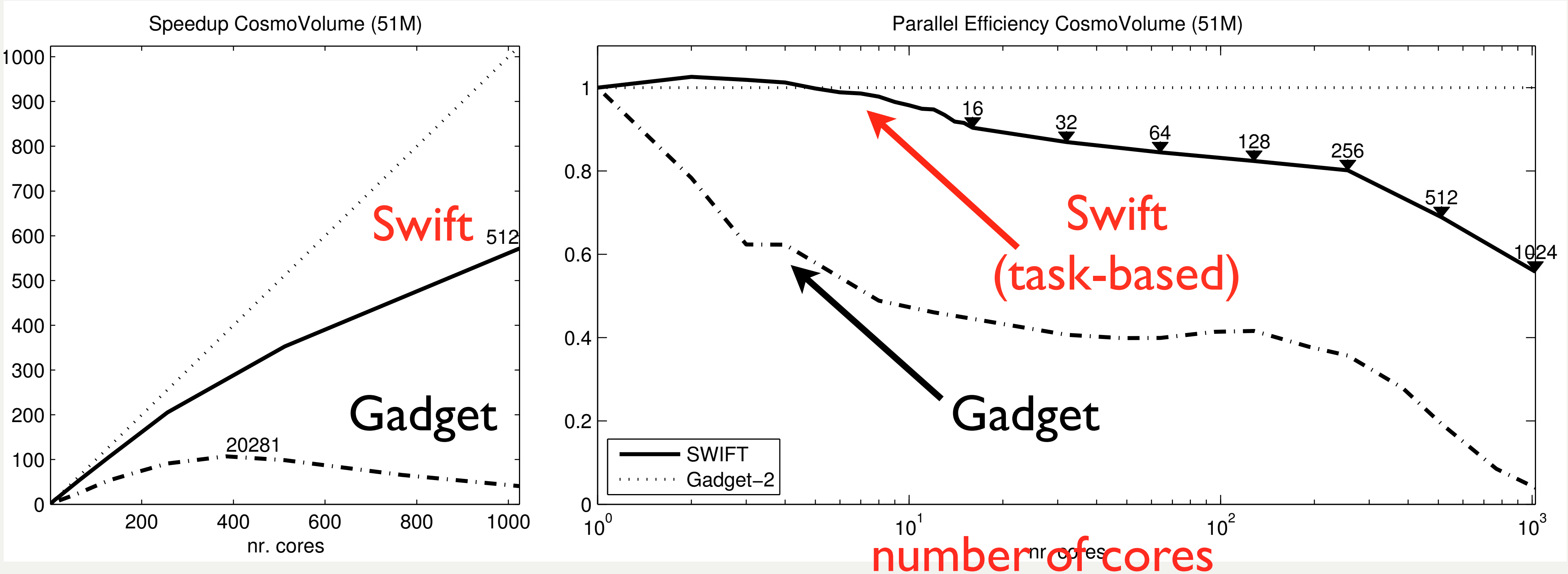


on future computers: gadget will become slower

number of cores

...particle EAGLE box ($z = 0.5$) SPH-only simulation on the COSMA5 cluster.

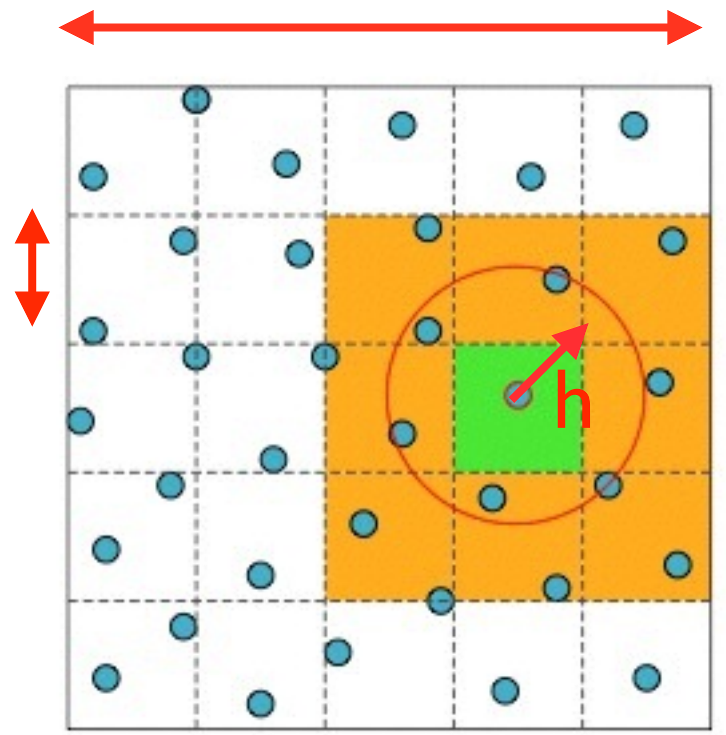
(Pete Beckman this morning)



- 51M particle EAGLE box ($z = 0.5$) SPH-only simulation on the COSMA5 cluster.
- SWIFT: Strong scaling up to 1024 cores with 60% parallel efficiency.

computational domain

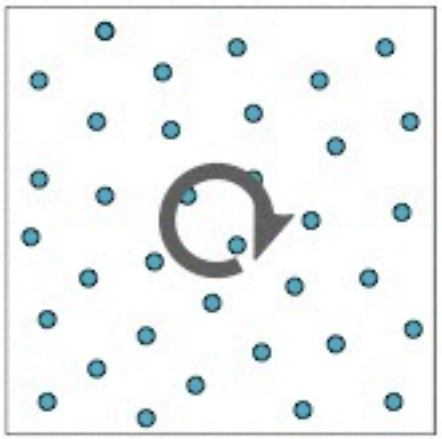
choose cell to be larger than distance to neighbours



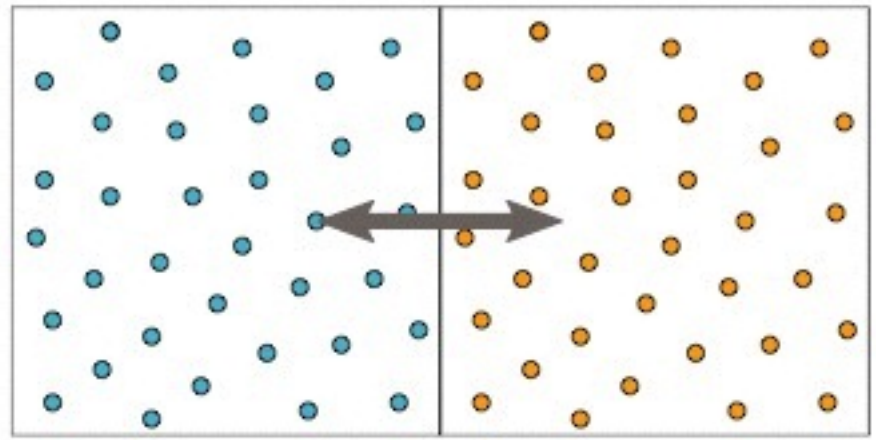
$$\rho(\mathbf{r}_i) = \sum_j m_j W(|\mathbf{r}_i - \mathbf{r}_j|, h_i)$$

Task type I

Task type II



i



i

$$\rho(\mathbf{r}_i) = \sum_j m_j W(|\mathbf{r}_i - \mathbf{r}_j|, h_i)$$

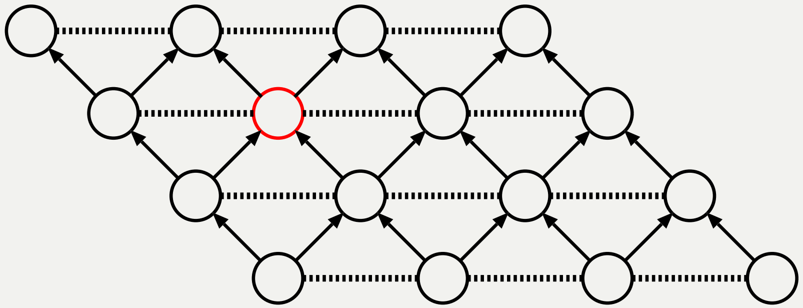
j

re-use particle data whenever possible in next task

Task-based parallelism

Main concepts

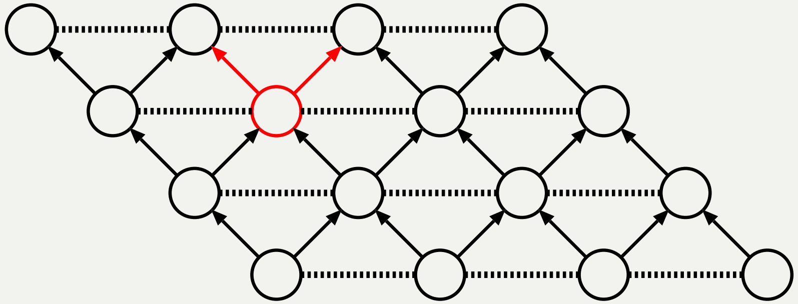
- Shared-memory parallel programming paradigm in which the computation is formulated in an implicitly parallelizable way that automatically avoids most of the problems associated with concurrency and load-balancing.
- We first reduce the problem to a set of inter-dependent tasks.
- For each task, we need to know:
  - Which tasks it depends on,
  - Which tasks it conflicts with.
- Each thread then picks up a task which has no unresolved dependencies or conflicts and computes it.



Task-based parallelism

Main concepts

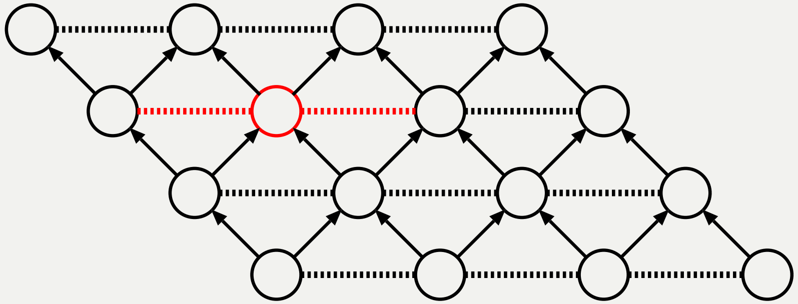
- Shared-memory parallel programming paradigm in which the computation is formulated in an implicitly parallelizable way that automatically avoids most of the problems associated with concurrency and load-balancing.
- We first reduce the problem to a set of inter-dependent tasks.
- For each task, we need to know:
  - Which tasks it depends on,
  - Which tasks it conflicts with.
- Each thread then picks up a task which has no unresolved dependencies or conflicts and computes it.



Task-based parallelism

Main concepts

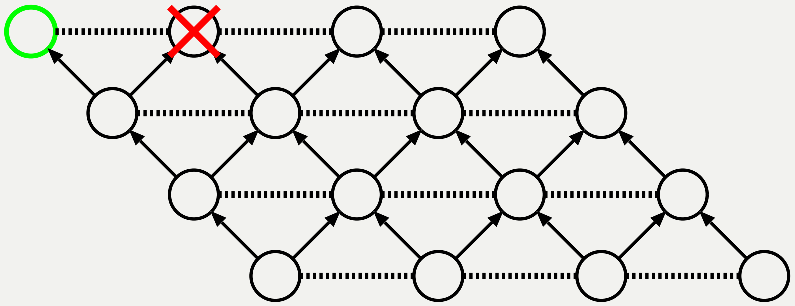
- Shared-memory parallel programming paradigm in which the computation is formulated in an implicitly parallelizable way that automatically avoids most of the problems associated with concurrency and load-balancing.
- We first reduce the problem to a set of inter-dependent tasks.
- For each task, we need to know:
  - Which tasks it depends on,
  - Which tasks it conflicts with.
- Each thread then picks up a task which has no unresolved dependencies or conflicts and computes it.



Task-based parallelism

Main concepts

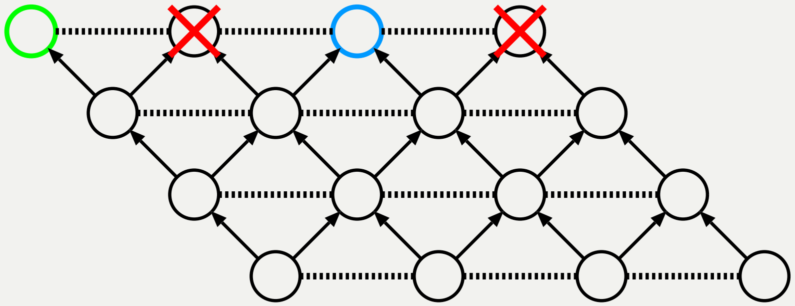
- Shared-memory parallel programming paradigm in which the computation is formulated in an implicitly parallelizable way that automatically avoids most of the problems associated with concurrency and load-balancing.
- We first reduce the problem to a set of inter-dependent tasks.
- For each task, we need to know:
  - Which tasks it depends on,
  - Which tasks it conflicts with.
- Each thread then picks up a task which has no unresolved dependencies or conflicts and computes it.



Task-based parallelism

Main concepts

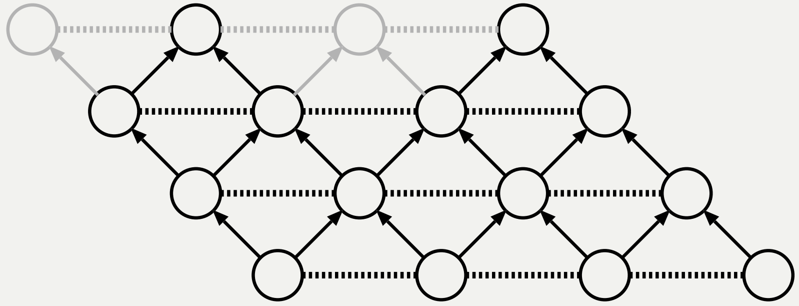
- Shared-memory parallel programming paradigm in which the computation is formulated in an implicitly parallelizable way that automatically avoids most of the problems associated with concurrency and load-balancing.
- We first reduce the problem to a set of inter-dependent tasks.
- For each task, we need to know:
  - Which tasks it depends on,
  - Which tasks it conflicts with.
- Each thread then picks up a task which has no unresolved dependencies or conflicts and computes it.



Task-based parallelism

Main concepts

- Shared-memory parallel programming paradigm in which the computation is formulated in an implicitly parallelizable way that automatically avoids most of the problems associated with concurrency and load-balancing.
- We first reduce the problem to a set of inter-dependent tasks.
- For each task, we need to know:
  - Which tasks it depends on,
  - Which tasks it conflicts with.
- Each thread then picks up a task which has no unresolved dependencies or conflicts and computes it.

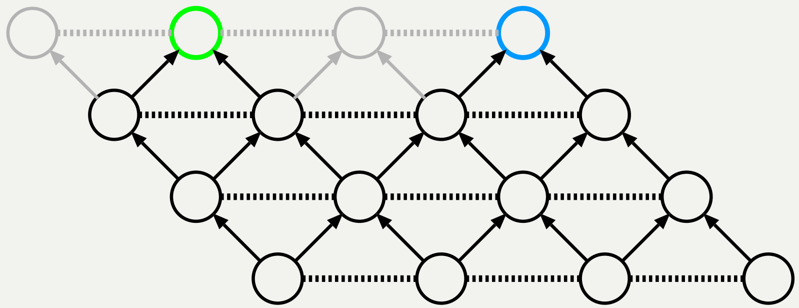


QuickSched library

Task-based parallelism

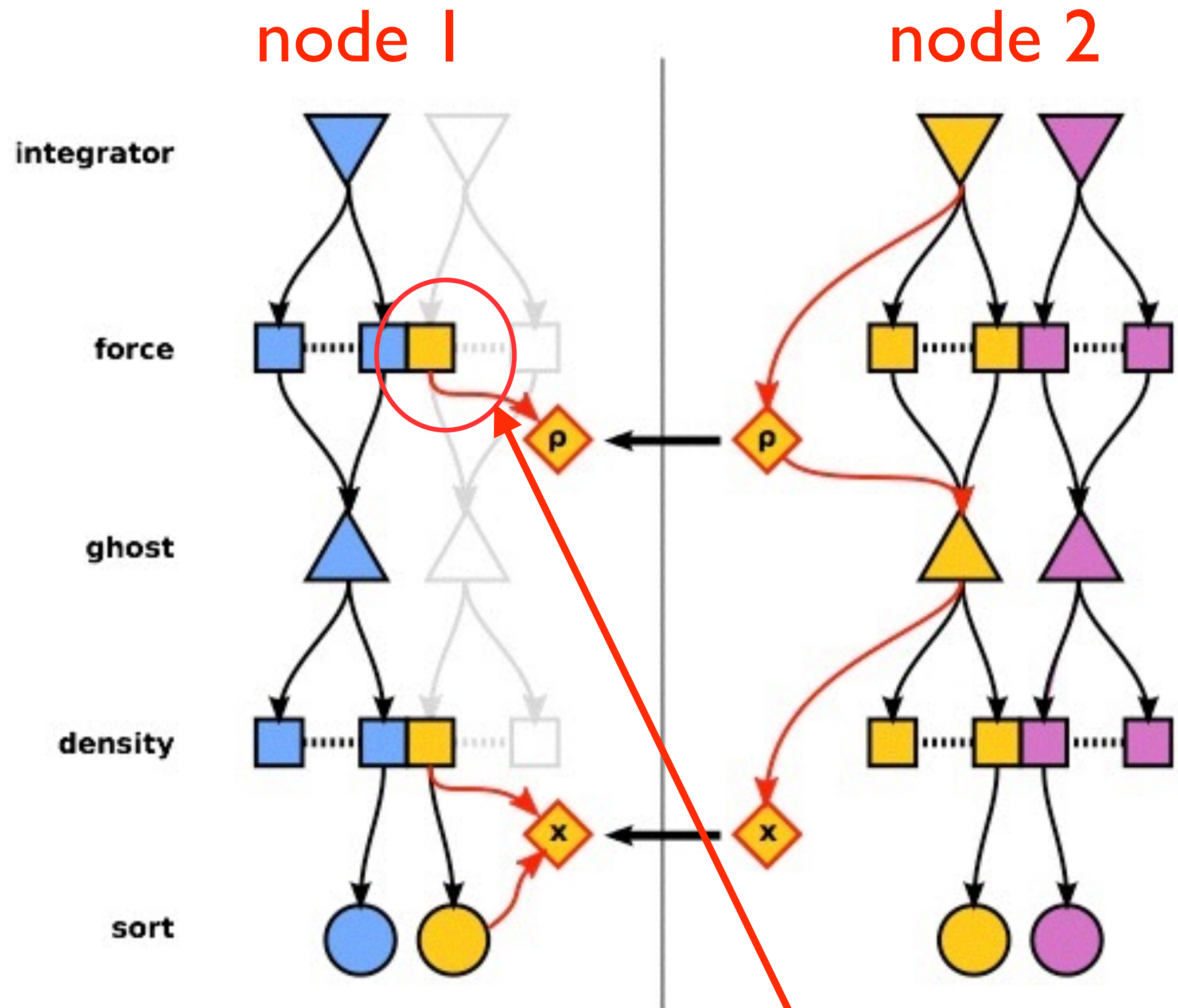
Main concepts

- Shared-memory parallel programming paradigm in which the computation is formulated in an implicitly parallelizable way that automatically avoids most of the problems associated with concurrency and load-balancing.
- We first reduce the problem to a set of inter-dependent tasks.
- For each task, we need to know:
  - Which tasks it depends on,
  - Which tasks it conflicts with.
- Each thread then picks up a task which has no unresolved dependencies or conflicts and computes it.



ParMETIS library distributes tasks

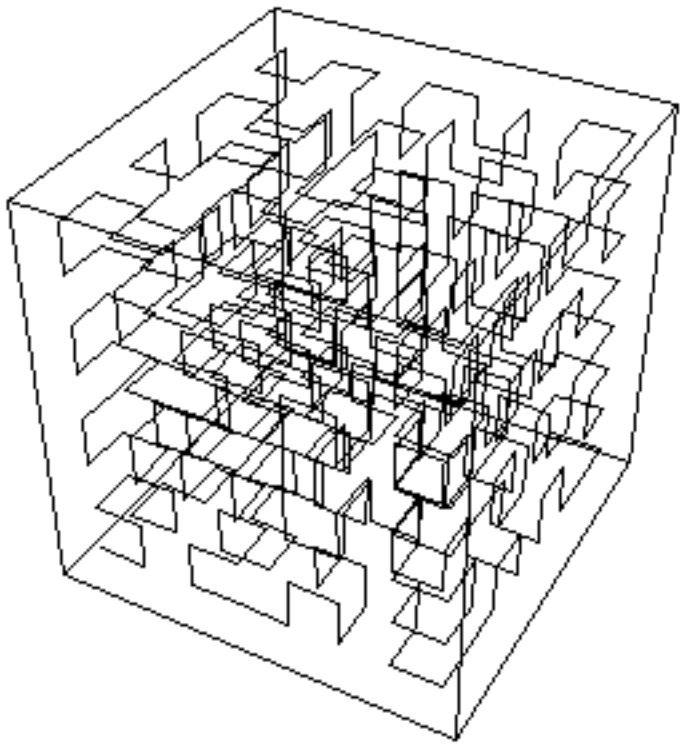
MPI implementation uses asynchronous comms



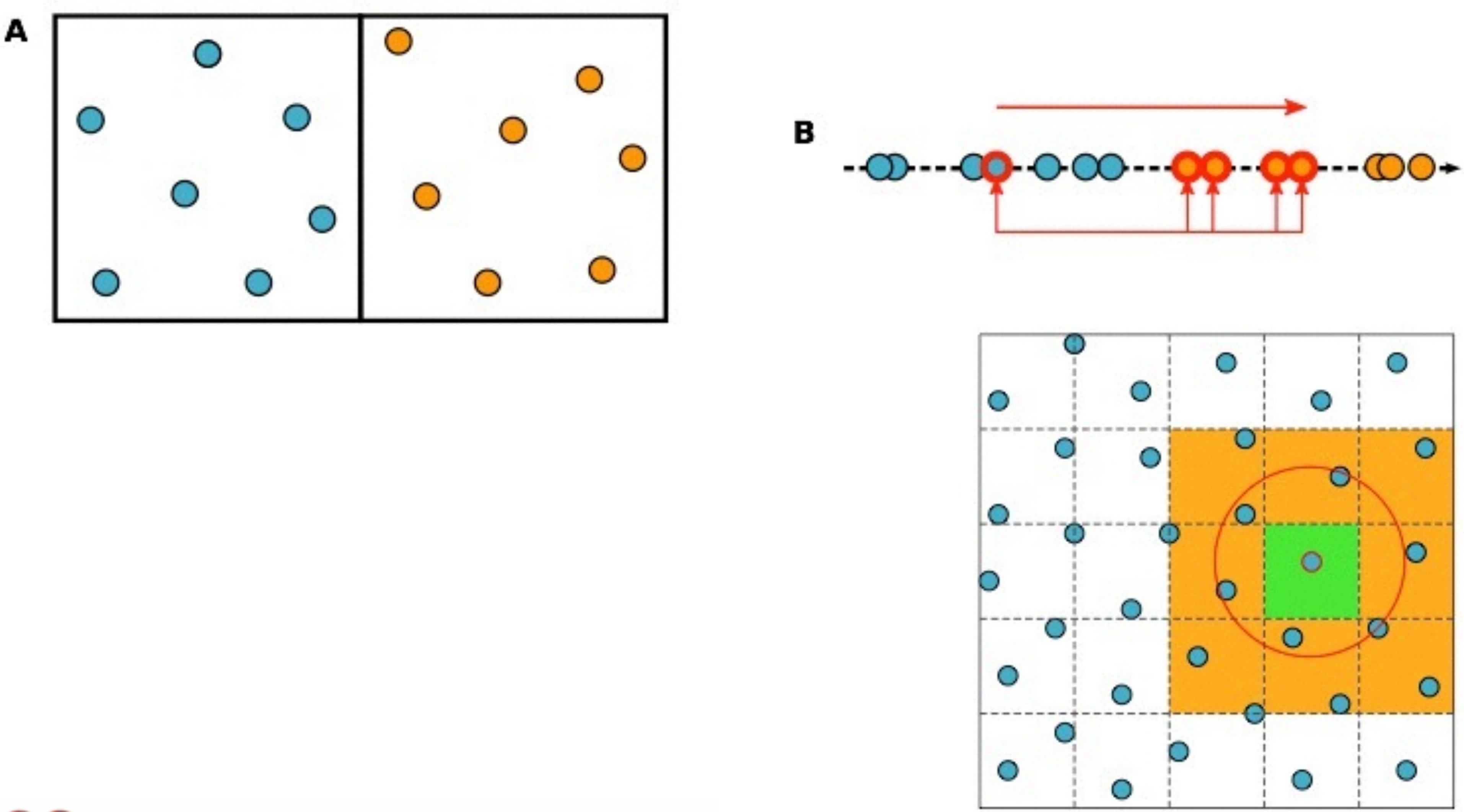
particles from neighbour cell on other node imported by comm task

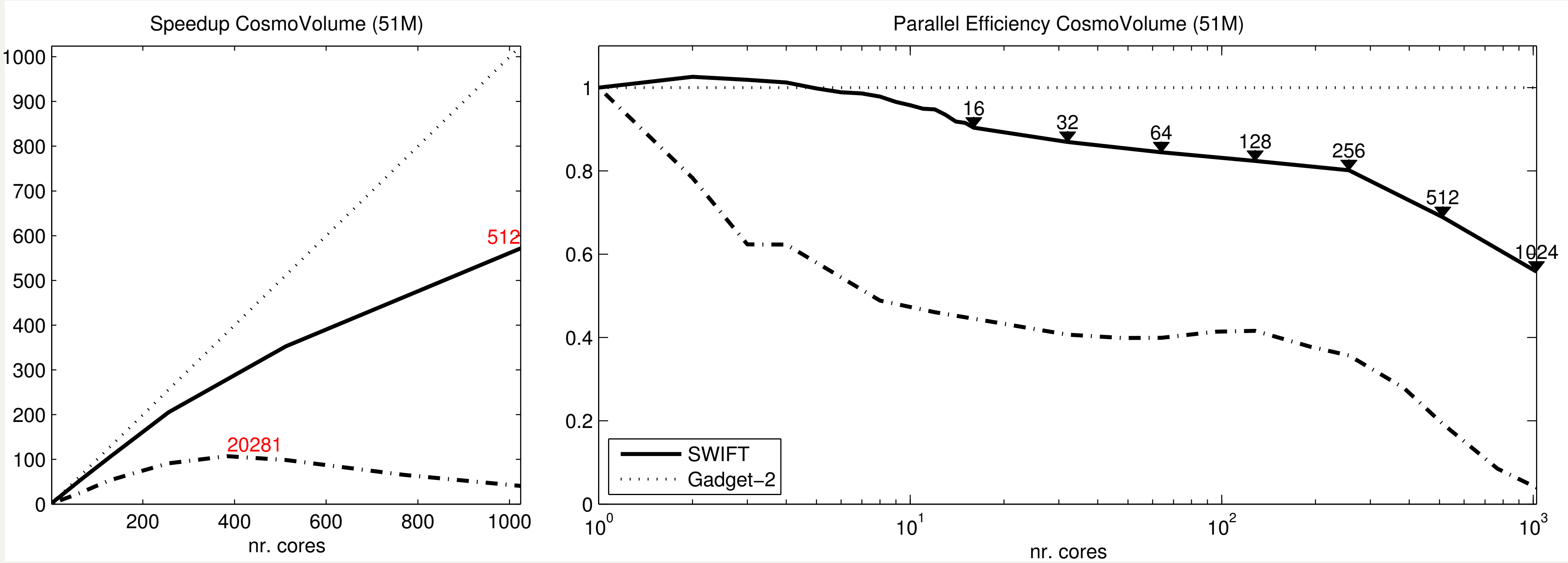
QuickSched library

tasks are distributed over resources to maximise throughput (and not spatially)



Sorting particles in cells cuts down on unnecessary neighbour testing

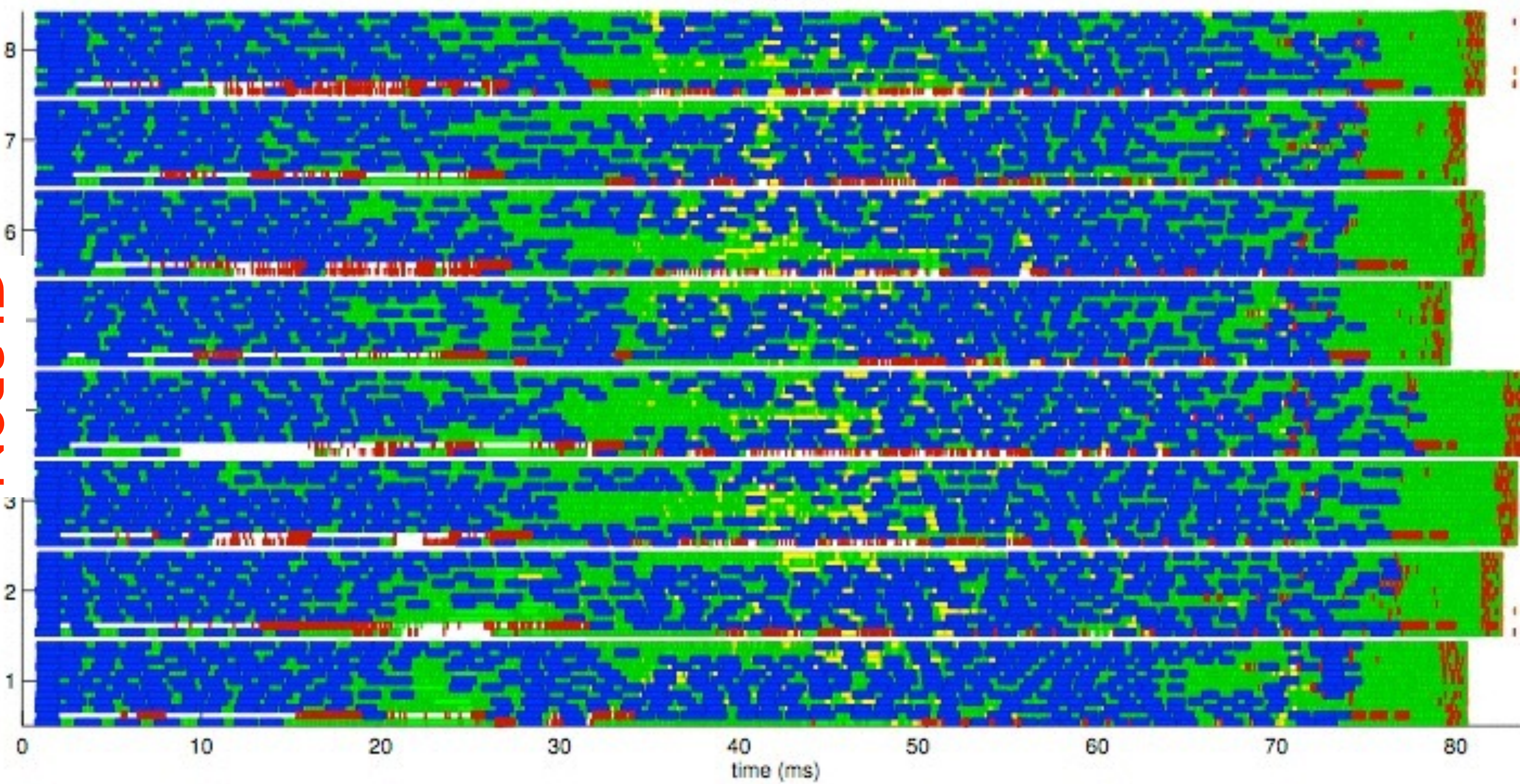




- 51M particle EAGLE box ($z = 0.5$) SPH-only simulation on the COSMA5 cluster.
- SWIFT: Strong scaling up to 1024 cores with 60% parallel efficiency.
- $\sim 40\times$ faster than GADGET.

Swift tasks

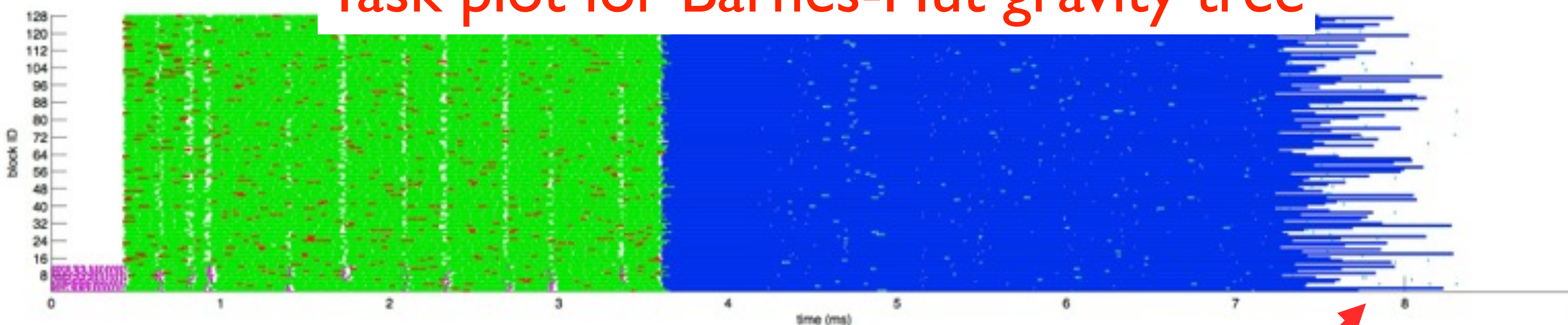
Node ID



wall clock time (ms)

Single GPU results for Barnes-Hut

Task plot for Barnes-Hut gravity tree



block ID

time (ms)

purple=data movement

red, green, blue=different tasks

big tasks to be moved

Timings of gravity part (Barnes-Hut) of code

Simulation type	1M parts	3M parts	10M parts
1 CPU with Quicksched	15.9s	50.5s	174.5s
16 CPUs with Quicksched	1.217s	3.489s	12.0s
GTX690 GPU	0.239s	0.677s	2.636 s
GTX690 GPU Single precision	0.116s	0.344s	1.414s
Tesla K40c GPU	0.099s	0.271s	2.025s
Gadget-2, 16 CPUs	2.25s	6.59s	47.91s
Bonsai-2 GTX690	0.069s	0.228s	Error.

- QuickSched implementation uses quadrupoles, Gadget uses monopoles
- Bonsai 2:

Contents:

- Introduction: cosmological simulations: aims and methods
- computational challenges
- load - (im) balance
- Swift: task based hydrodynamics and gravity

Future:

- vectorisation of interaction kernels
- optimizing cache performance
- optimizing MPI performance
- self-tuning strategy to exploit hardware specifics

Swift: task-based hydrodynamics at Durham's IPCC

For the cosmological simulations of the formation of galaxies

Gonnet Schaller Chalk Bower

movie: Richard Bower (Durham)

Tom Theuns
Institute for Computational Cosmology
Durham