Numerical Galaxy Formation and Cosmology

Lecture 2: Following gravity (on parallel computers)

Ewald Puchwein & Benjamin Moster

Cosmological simulations - recap

- slides of last lecture on lecture website:

  http://www.ast.cam.ac.uk/~puchwein/NumCosmo_lect_2016/

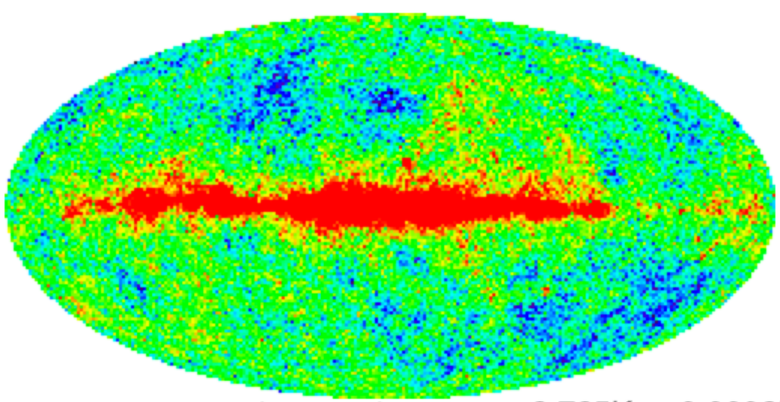
- power spectrum at z~1100 can be constrained from the CMB
- at high redshift perturbations grow in the linear regime (~D(a))

  ➡ power spectrum at some early time (in the linear regime) can be computed

  ➡ from this the density and velocity perturbations at this time can be obtained (at e.g. z~100 with the Zel'dovich approximation)

  ➡ use them as initial conditions for a cosmological simulation

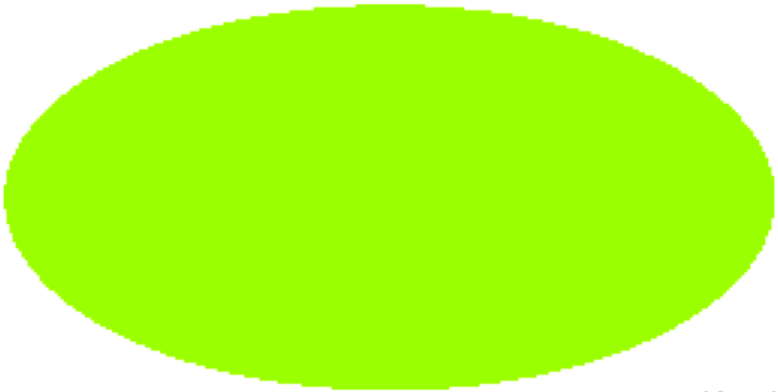
CMB maps as usually shown (z=1100)



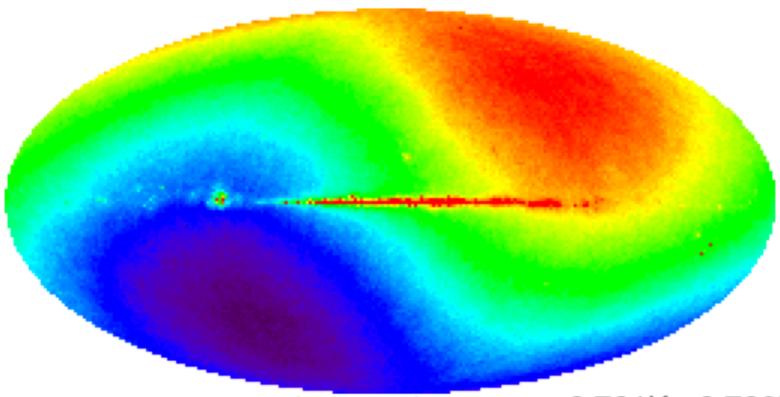
temperature range 2.725K +- 0.0002K

image credit: NASA

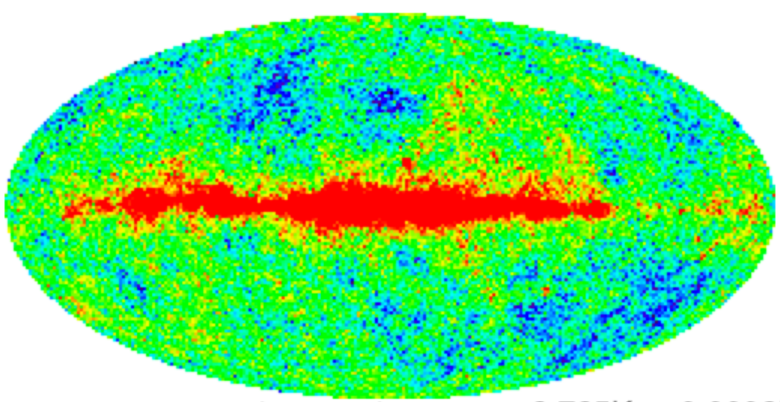
Fluctuations seen by WMAP (Simulated)



temperature range 0K - 4K



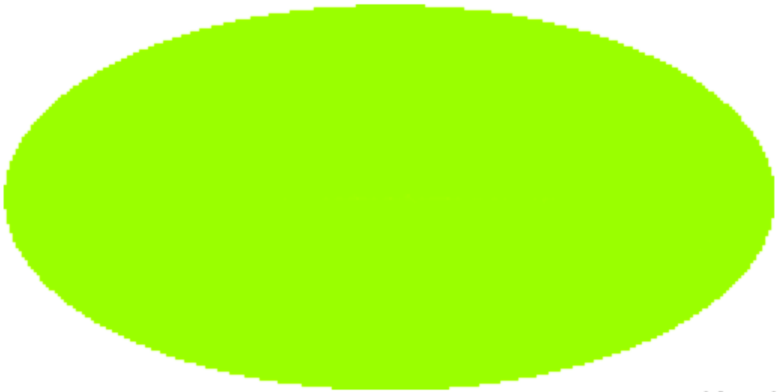
temperature range 2.721K - 2.729K



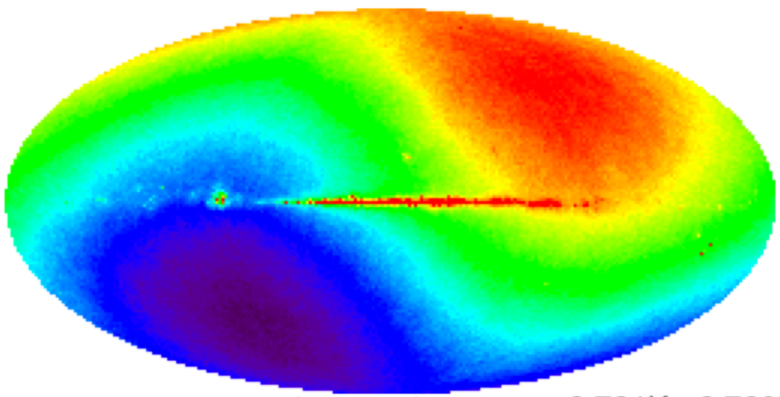
temperature range 2.725K +- 0.0002K

image credit: NASA

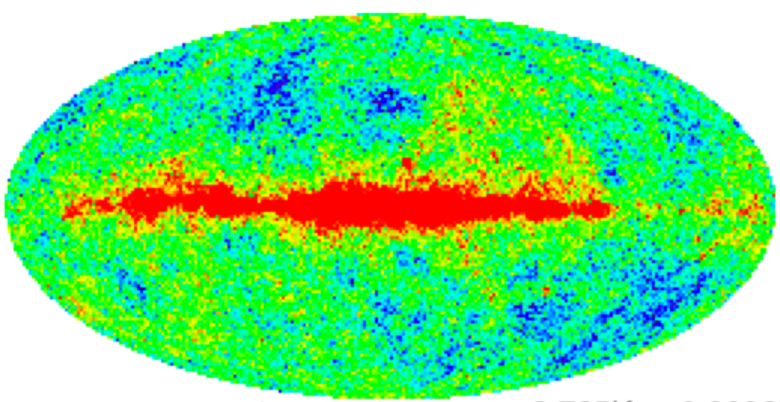
Fluctuations seen by WMAP (Simulated)



temperature range 0K - 4K



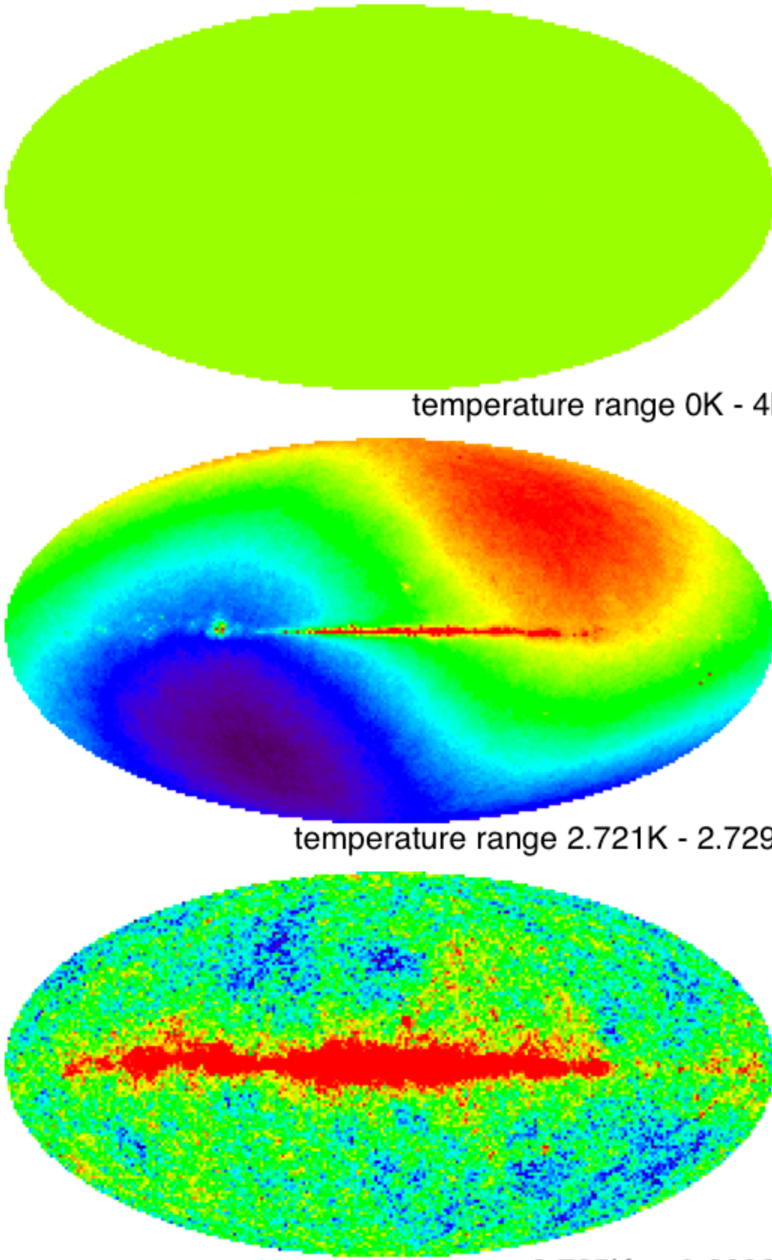
temperature range 2.721K - 2.729K



temperature range 2.725K +- 0.0002K

image credit: NASA

Fluctuations seen by WMAP (Simulated)

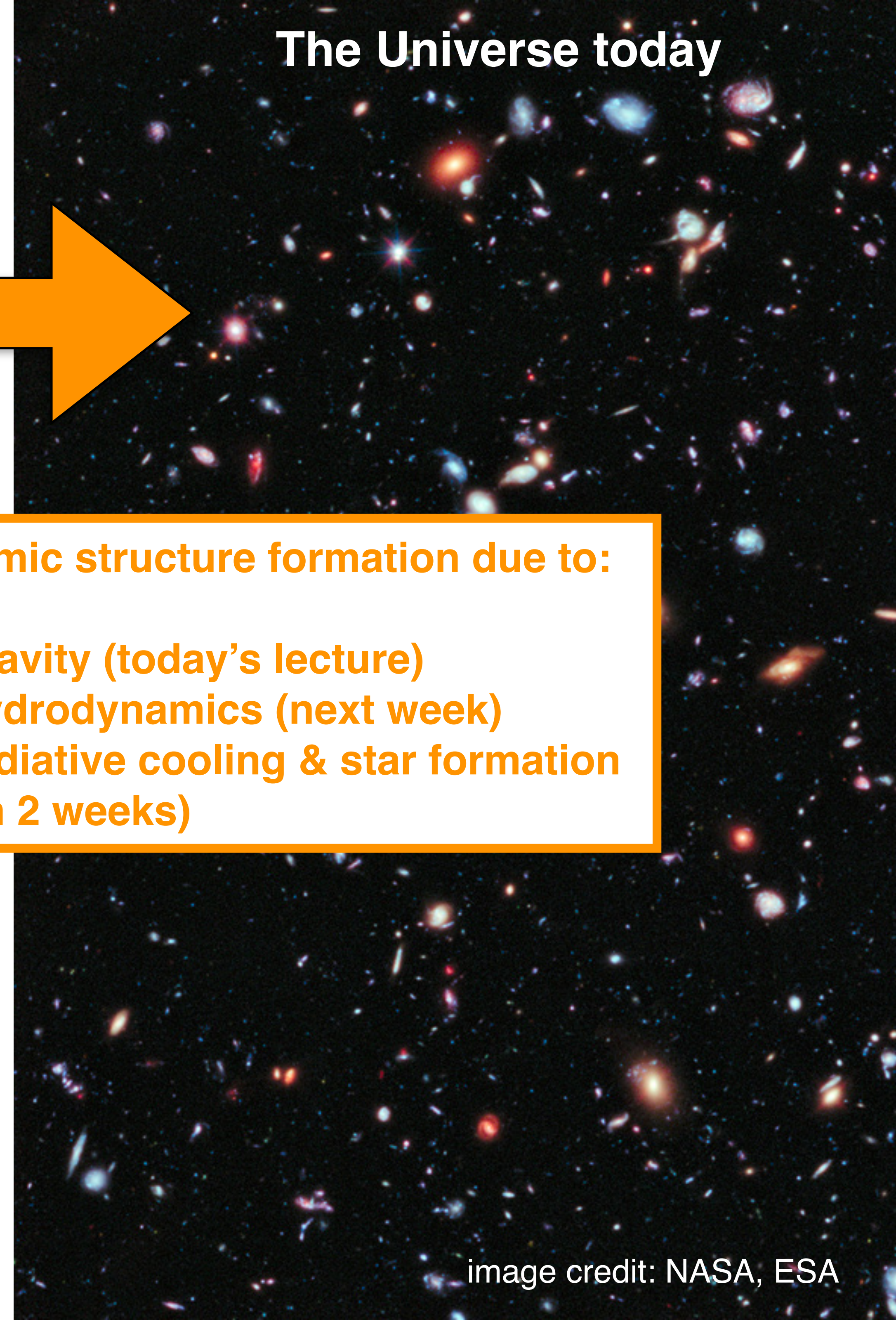


temperature range 0K - 4K

temperature range 2.721K - 2.729K

temperature range 2.725K +- 0.0002K

image credit: NASA



cosmic structure formation due to:

- **gravity (today's lecture)**
- **hydrodynamics (next week)**
- **radiative cooling & star formation (in 2 weeks)**

image credit: NASA, ESA

Cosmological simulations - recap

- (ignoring baryonic physics) the dynamics is given by Newtonian gravity on an expanding background

$$\frac{d\vec{v}}{dt} = -H\vec{v} - \frac{\vec{\nabla}\delta\phi}{a}$$

- the main computational task is calculating the gravitational forces

$$-\frac{\vec{\nabla}\delta\phi}{a}|_{\vec{x}=\vec{x}_j} = \vec{F}_j = \frac{Gm_j}{a^2}\sum_{i \neq j}\frac{m_i(\vec{x}_i - \vec{x}_j)}{|\vec{x}_i - \vec{x}_j|^3}$$

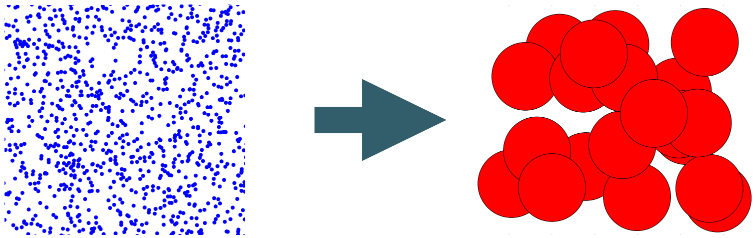
- for N particles, need to calculate N(N-1) ~ N^2 forces

➡ for large N computationally very expensive (e.g. 300 billion particles, highly efficient code ~20 flop/interaction, worlds fastest supercomputer -> 2 years for single computation of forces for particles, need 1000s of timesteps)

The N-body approach

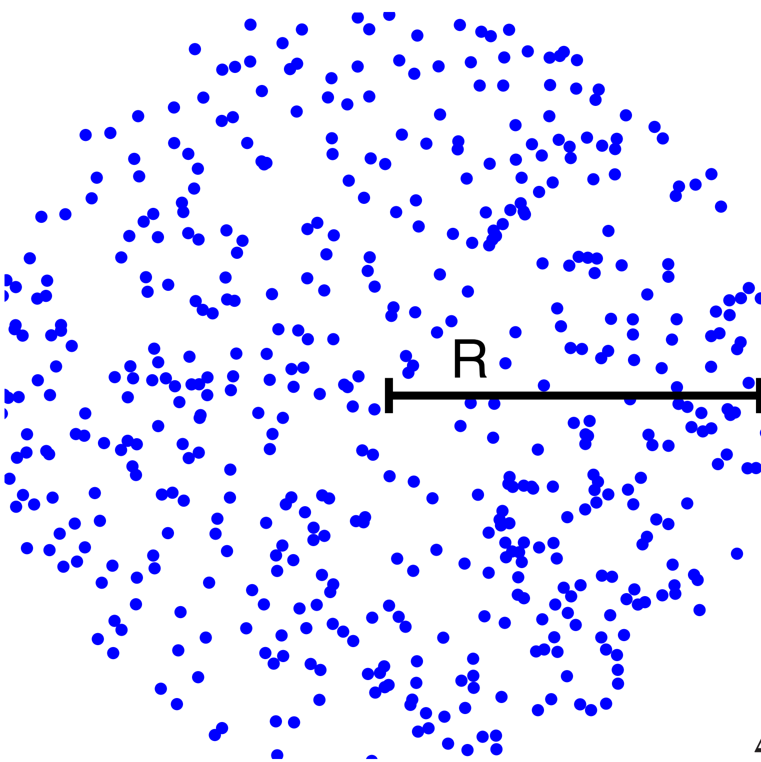
- What do we mean by simulation particles?
- Most of the mass in the Universe is in the form of dark matter
  - e.g. weakly interacting massive particles (WIMPs) may have a mass of ~100 GeV/c^2

    ➡ 10^{12} solar mass galaxy halo consists of 10^{67} dark matter particles
- can only afford to represent if by ~10^2 - 10^9 particles
- Does representing ~10^{60} dark matter particles by 1 simulation particle have unwanted side effects?

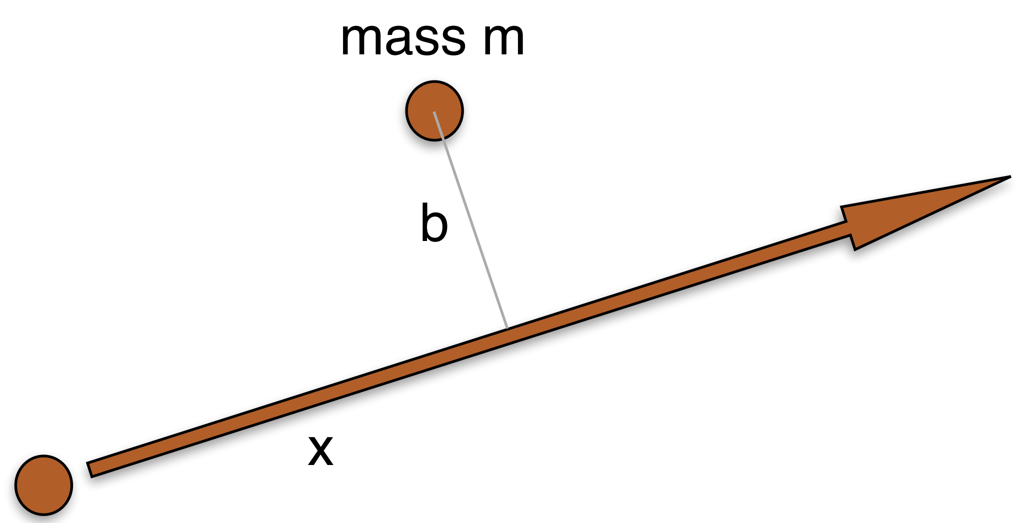


Relaxation time of an N-body system

- time scale on which two body processes play a role in a N-body system



self-gravitating N-body system



$$\Delta v_\perp = \frac{1}{m}\int F_\perp dt = \int \frac{Gm}{x^2+b^2}\frac{b}{\sqrt{x^2+b^2}}\frac{dx}{v} = \frac{2Gm}{bv}$$

Relaxation time of an N-body system

- particles encountered with impact parameter between b and b+db during one crossing

$$dn \approx \frac{2\pi b\,\mathrm{d}b}{\pi R^2} N$$

- individual encounters add incoherently

$$(\Delta v_\perp)^2 = \int \left(\frac{2Gm}{bv}\right)^2 dn = \frac{8G^2m^2}{R^2v^2} N \int \frac{\mathrm{d}b}{b} = \frac{8G^2m^2}{R^2v^2} N \ln\left(\frac{b_{\max}}{b_{\min}}\right)$$

(per crossing time)

$b_{\max} \approx R$ given by system size

$b_{\min}$ controls maximum deflection which should be ~ v $\quad \frac{2Gm}{b_{\min}v} \approx v \quad \rightarrow \quad b_{\min} \approx \frac{2Gm}{v^2}$

Relaxation time of an N-body system

- typical velocity

$$v^2 \approx \frac{GNm}{R}$$

- using this we get

$$b_{\min} \approx \frac{2Gm}{v^2} \approx \frac{2R}{N}$$

and

$$(\Delta v_\perp)^2 = \frac{8G^2m^2}{R^2v^2} N \ln\left(\frac{b_{\max}}{b_{\min}}\right) = \frac{8v^2}{N} \ln\left(\frac{N}{2}\right)$$ (per crossing time)

- the two-body relaxation time is then

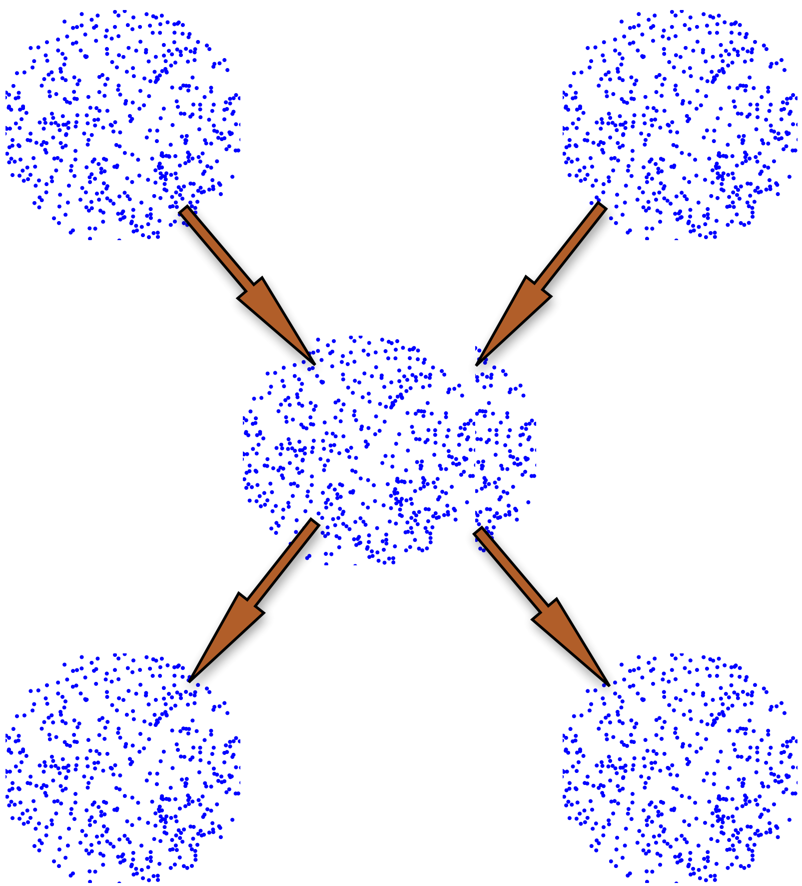
$$t_{\mathrm{relax}} \approx t_{\mathrm{cross}} \frac{v^2}{(\Delta v_\perp)^2} \approx t_{\mathrm{cross}} \frac{N}{8\ln\left(\frac{N}{2}\right)}$$

Relaxation time of an N-body system (examples)

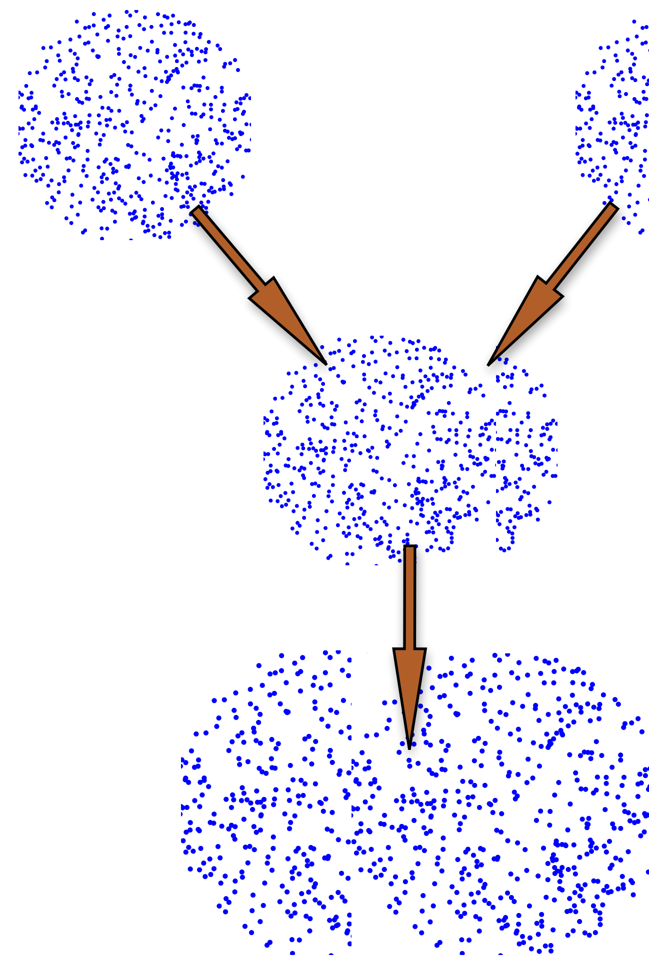
	N	tcross	trelax
star cluster	10^5	~1/2 Myr	~1/2 Gyr collisional
stars in galaxy	10^{11}	~0.01 / H_0	~5x10^6 / H_0 collisionless
dark matter in galaxy	10^{67}	~0.1 / H_0	~10^{63} / H_0 collisionless
galaxy in low-res simulation (without softening)	1000	~0.1 / H_0	~2 / H_0 somewhat collisional (but should be collisionless)

Gravitational softening

dark matter in galaxies and galaxy clusters should be collisionless

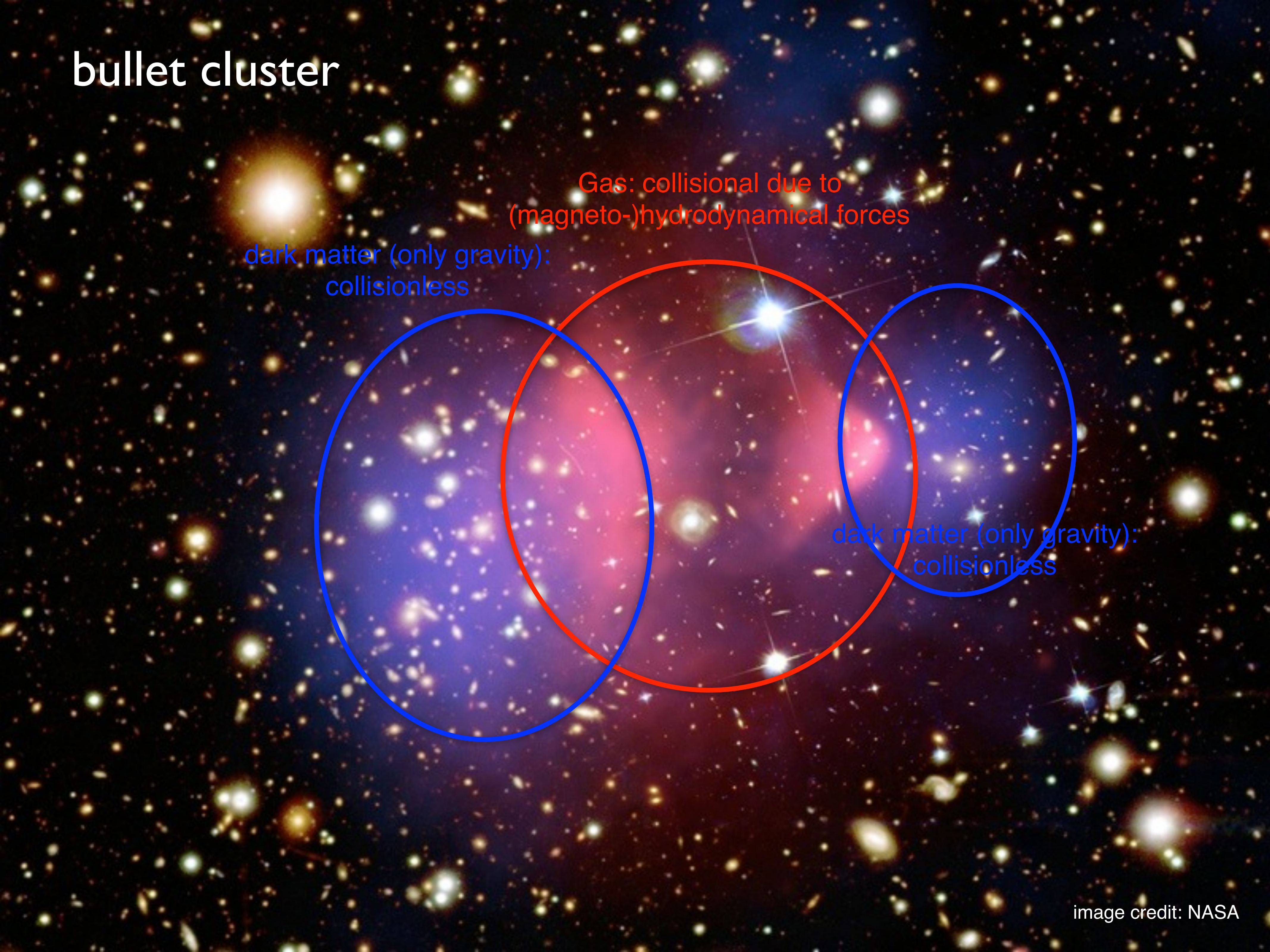


but may become collisional in simulations with a Newtonian force law



bullet cluster

image credit: NASA


bullet cluster
Gas: collisional due to
(magneto-)hydrodynamical forces
dark matter (only gravity):
collisionless
dark matter (only gravity):
collisionless
image credit: NASA

Gravitational softening

- for collisionless systems we need to ensure:

  simulated time << relaxation time

  - prevent large angle scattering
  - prevent formation of bound particle pairs

- want to integrate equations of motion with low-order scheme and reasonably large timesteps

➡ need to soften force law on small scales:

$$\vec{F}_j = \frac{G m_j}{a^2} \sum \frac{m_i(\vec{x}_i - \vec{x}_j)}{((\vec{x}_i - \vec{x}_j)^2 + \epsilon^2)^{\frac{3}{2}}}$$

gravitational softening

Choice of Gravitational softening

- too small softening:
    - system may become collisional
    - time integration more expensive
    - artificial heating
- too large softening:
    - loss of spatial resolution in the simulations
- typical value in cosmological simulations:
    - ~2% to 4% of the mean-interparticle distance $(V/N)^{1/3}$

Size of cosmological simulations over time

- computers double speed every 18 months (Moore's law)
- particle number in simulations doubles every 16-17 months
- only possible with algorithms that scale close to ~N (or N log(N))

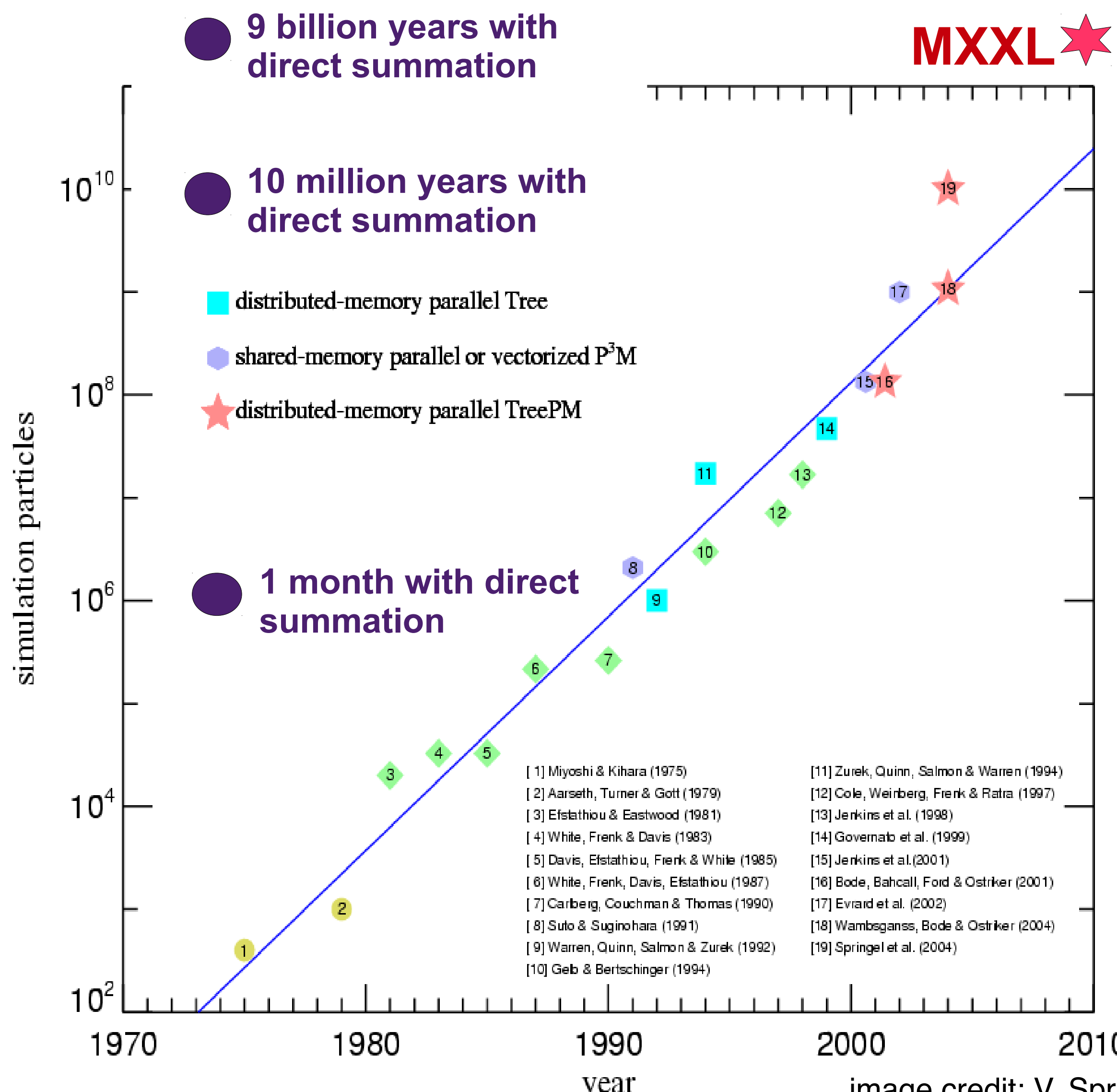


image credit: V. Springel

Requirements for (competitive) N-body simulations

- want large **N** to have high resolution (otherwise small objects are not resolved) and large volume (otherwise no representative volume and no rare objects like galaxy clusters, also the fundamental mode goes non-linear at low-z)
- need efficient self-gravity algorithms with scaling close to ~N (and not N^2)
- need to be able to run it efficiently in parallel on 1000s of CPU cores
- should be memory and communication efficient
- should automatically adapt the size of the timestep to the relevant dynamical time

Overview of self-gravity algorithms

- direct summation ~ N^2 -> not competitive in cosmological runs
- particle-mesh codes
- tree codes

rarely used alone nowadays

- tree particle-mesh codes (e.g. used in the GADGET code)
- multigrid relaxation (e.g. used in the RAMSES code)
- fast multipole codes, …

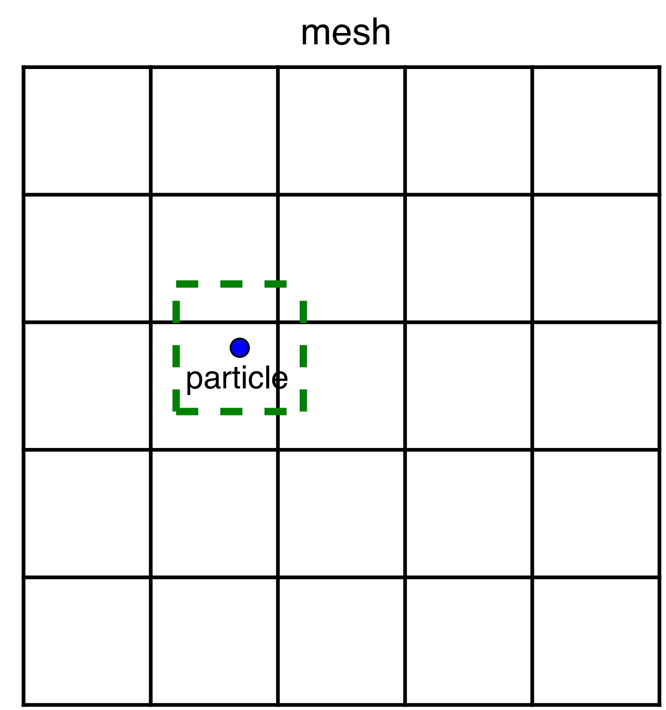
The particle-mesh (PM) method

- particle-mesh method
  - Poisson's equation in real space:
  - Poisson's equation in Fourier space:
  - assign particle mass to grid (e.g. CIC)
  - compute Fourier transform of density contrast (FFT ~ N log N)
  - convert to Fourier transform of potential
  - transform the potential back to real space
  - compute gradient by finite differencing of the potential

average comoving matter density $\bar{\rho}_c$

$$\vec{\nabla}^2 \delta\phi = 4\pi G \bar{\rho}_c \delta a^{-1}$$

$$-k^2 \delta\phi_{\vec{k}} = \frac{4\pi G \bar{\rho}_c \delta_{\vec{k}}}{a}$$



The particle-mesh (PM) method

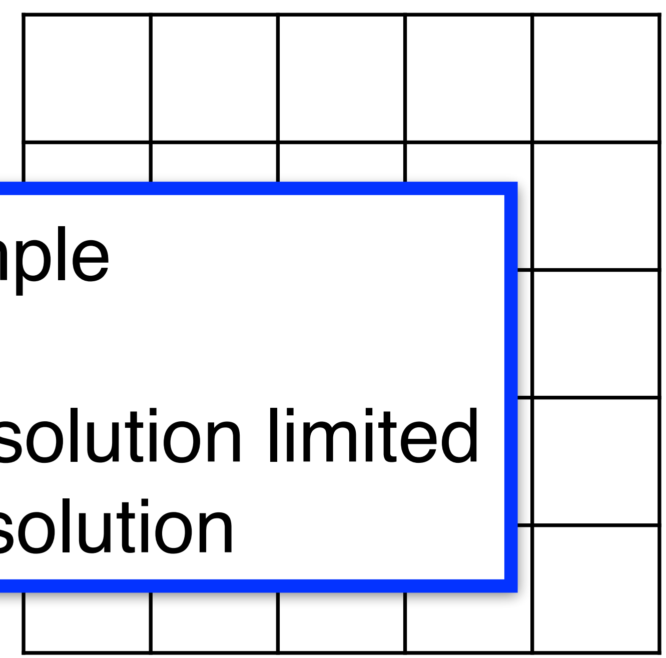
- particle-mesh method
  - Poisson's equation in real space:
  - Poisson's equation in Fourier space:
  - assign particle mass to grid (e.g. CIC)
  - compute Fourier transform of density contrast (FFT ~ N log N)
  - convert to Fourier tra
  - transform the potent space
  - compute gradient by of the potential

average comoving matter density $\bar{\rho}_c$

$$\vec{\nabla}^2 \delta\phi = 4\pi G \bar{\rho}_c \delta a^{-1}$$

$$-k^2 \delta\phi_{\vec{k}} = \frac{4\pi G \bar{\rho}_c \delta_{\vec{k}}}{a}$$

mesh

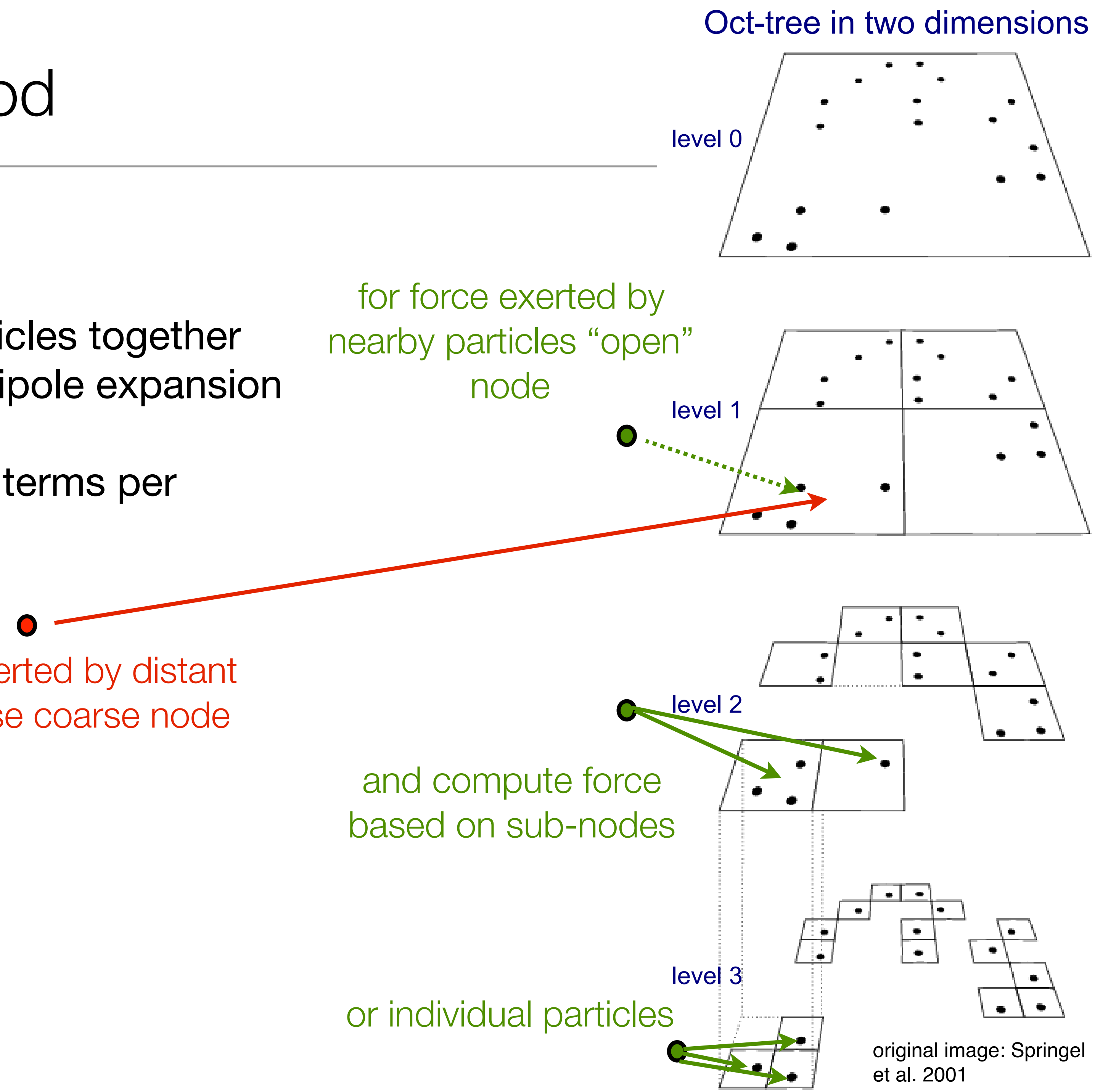


Pros: fast & simple

Cons: spatial resolution limited to grid resolution

The tree method

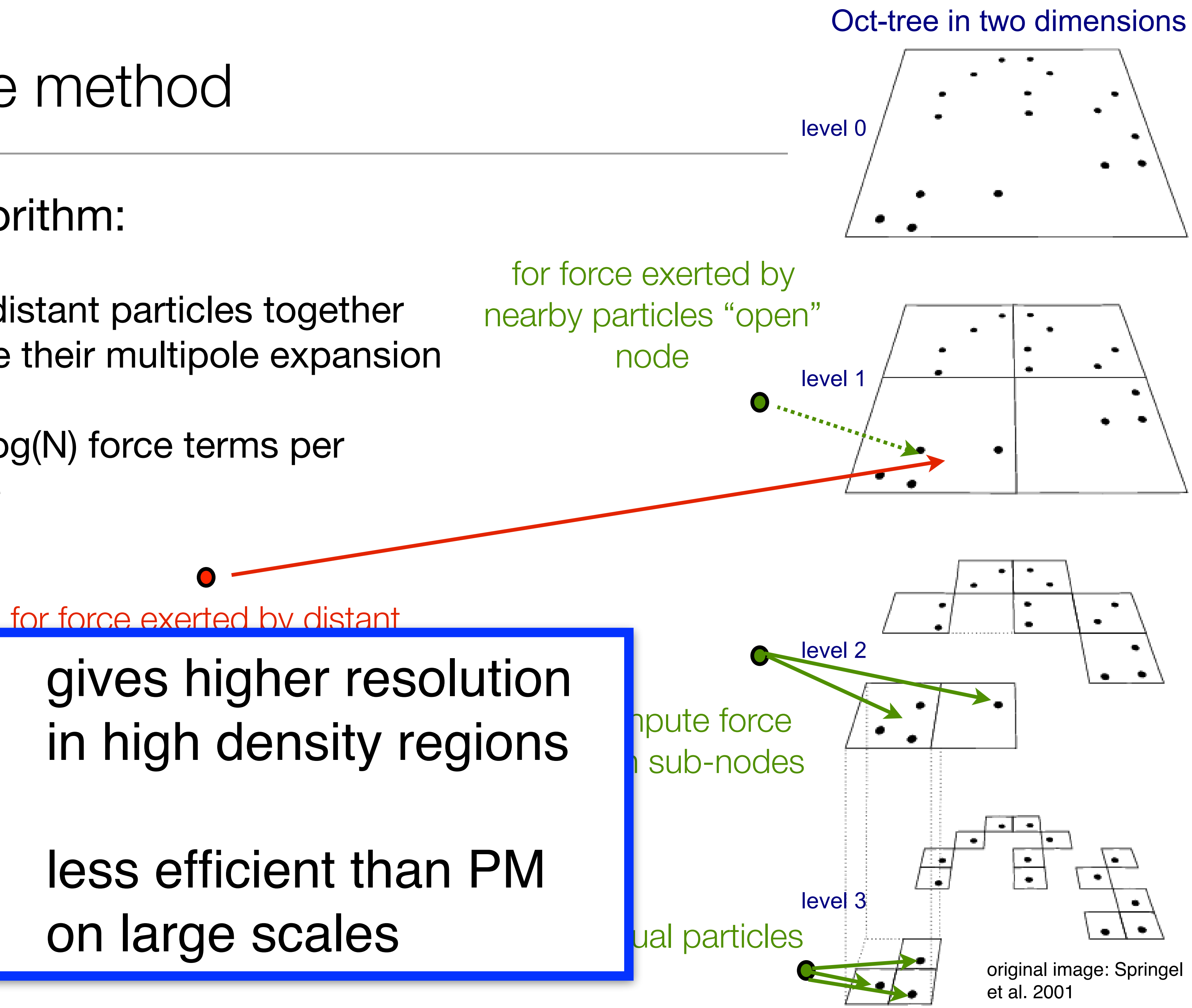
- tree algorithm:
  - group distant particles together and use their multipole expansion
  - only ~log(N) force terms per particle



original image: Springel et al. 2001

The tree method

- tree algorithm:
  - group distant particles together and use their multipole expansion
  - only ~log(N) force terms per particle



Pros: gives higher resolution in high density regions

Cons: less efficient than PM on large scales

The Tree-PM method

- Tree-PM algorithm (used e.g. in Gadget):
  - combine tree and PM methods to get the advantages of both
  - split forces in long range & short range part in Fourier space

long range forces

$$\delta\phi_{\mathbf{k}}^{\mathrm{long}} = \delta\phi_{\mathbf{k}} \exp(-\mathbf{k}^2 r_{\mathrm{s}}^2)$$

solve with particle-mesh method

short range forces

$$\delta\phi_{\mathbf{k}}^{\mathrm{short}} = \delta\phi_{\mathbf{k}} (1 - \exp(-\mathbf{k}^2 r_{\mathrm{s}}^2))$$

in real space (assuming large N_{grid})

$$\phi^{\mathrm{short}} = -G \sum_i \frac{m_i}{r_i} \mathrm{erfc}\left(\frac{r_i}{2 r_{\mathrm{s}}}\right)$$

solve with tree code

The Tree-PM method

- Tree-PM algorithm (used e.g. in Gadget):
  - combine tree and PM methods to get the advantages of both
  - split forces in long range & short range part in Fourier space

long range forces

$$\delta\phi_{\mathbf{k}}^{\mathrm{long}} = \delta\phi_{\mathbf{k}} \exp(-\mathbf{k}^2 r_{\mathrm{s}}^2)$$

solve with particle-

short range forces

$$\delta\phi_{\mathbf{k}}^{\mathrm{short}} = \delta\phi_{\mathbf{k}} (1 - \exp(-\mathbf{k}^2 r_{\mathrm{s}}^2))$$

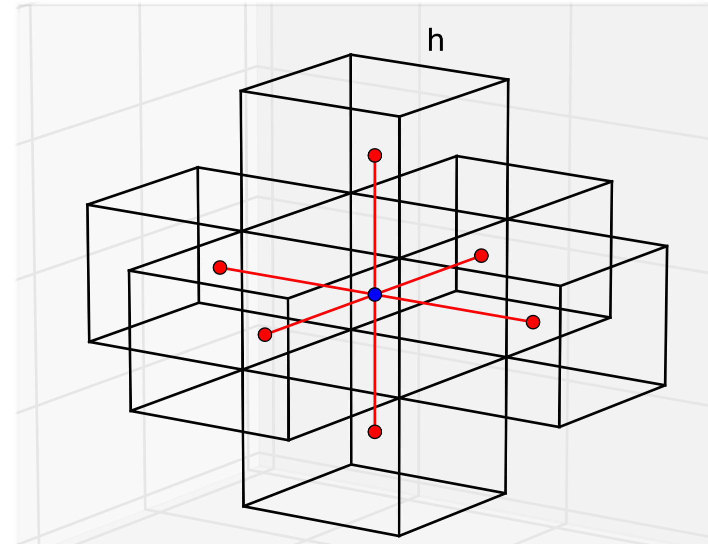
in real space (assuming large N_{grid})

$$\phi^{\mathrm{short}} = -G \sum_i \frac{m_i}{r_i} \mathrm{erfc}\left(\frac{r_i}{2 r_{\mathrm{s}}}\right)$$

olve with tree code

Pros: fast, high resolution, N log(N) scaling

The multigrid relaxation method



seven point stencil for Laplace operator

- multigrid relaxation method (used e.g. in Ramses):
    - discretize Poisson eq. on grid:

$$(\nabla^2 \delta\phi)_{i,j,k} = 4\pi G \delta\rho_{i,j,k}$$

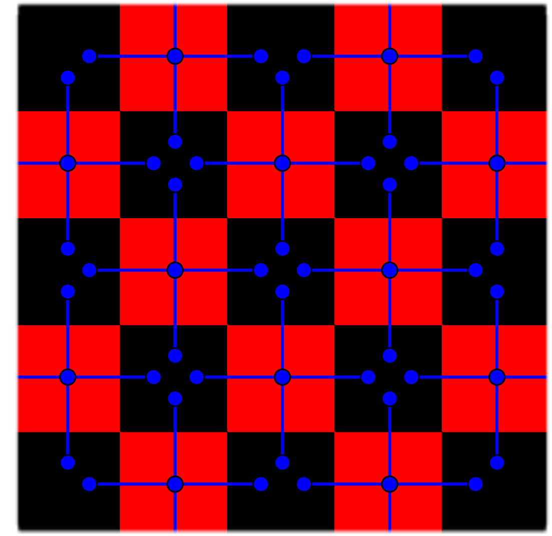
- 7-point stencil for Laplace operator:

$$(\nabla^2 \delta\phi)_{i,j,k} = \frac{\delta\phi_{i+1,j,k} + \delta\phi_{i-1,j,k} + \delta\phi_{i,j+1,k} + \delta\phi_{i,j-1,k} + \delta\phi_{i,j,k+1} + \delta\phi_{i,j,k-1} - 6\delta\phi_{i,j,k}}{h^2}$$

- solve equation iteratively using Newton's method at each grid point

$$f_{i,j,k} \equiv (\nabla^2 \delta\phi)_{i,j,k} - 4\pi G \delta\rho_{i,j,k} \qquad \delta\phi^{(n+1)}_{i,j,k} = \delta\phi^{(n)}_{i,j,k} - \frac{f_{i,j,k}}{\frac{\partial f_{i,j,k}}{\partial(\delta\phi_{i,j,k})}}$$

The multigrid relaxation method



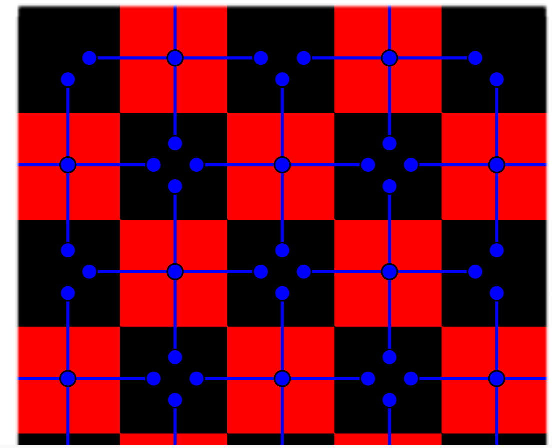
- multigrid relaxation method (continued):
  - perform iterations in red-black sweep:
  - each iteration couples only neighbouring grid points:

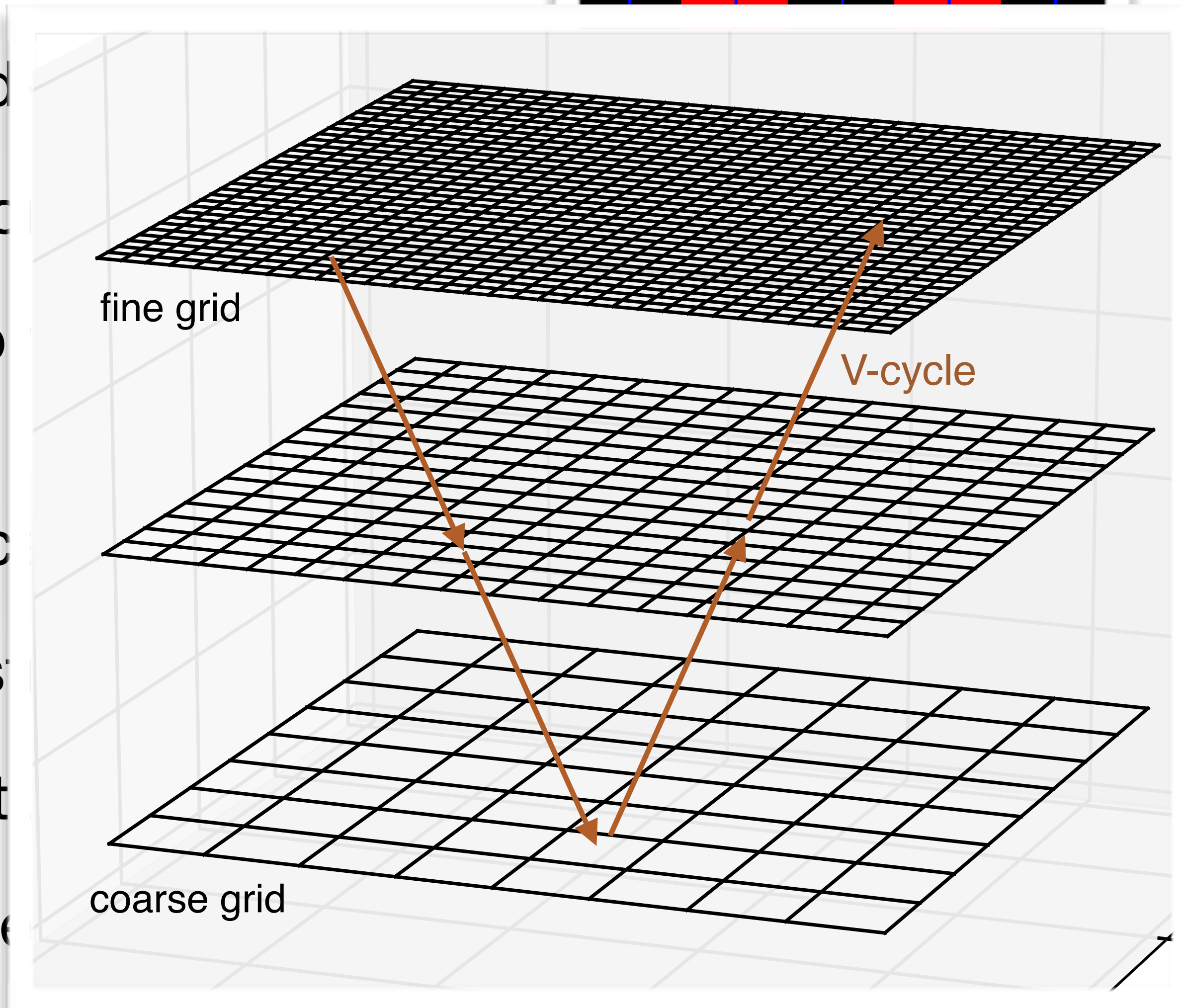
    ➡ it takes many iterations for the solution to “propagate” over a large grid
  - better use multigrid acceleration:
    - recovers large scale structure of solution on coarse grid
    - and small scale structure on fine grid

    ➡ much faster convergence (~N scaling)

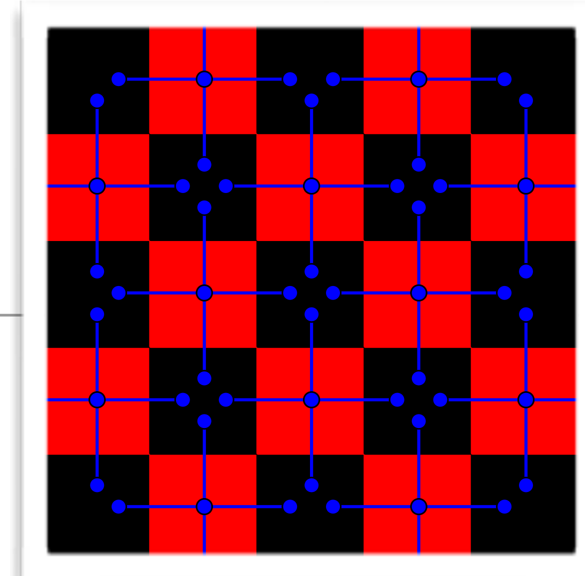
The multigrid relaxation method

- multigrid relaxation method (continued):
  - perform iterations in red
  - each iteration couples c
    - ➡ it takes many iteratio
      a large grid
  - better use multigrid acc
    - recovers large scale s
    - and small scale struct
    - ➡ much faster converg





The multigrid relaxation method



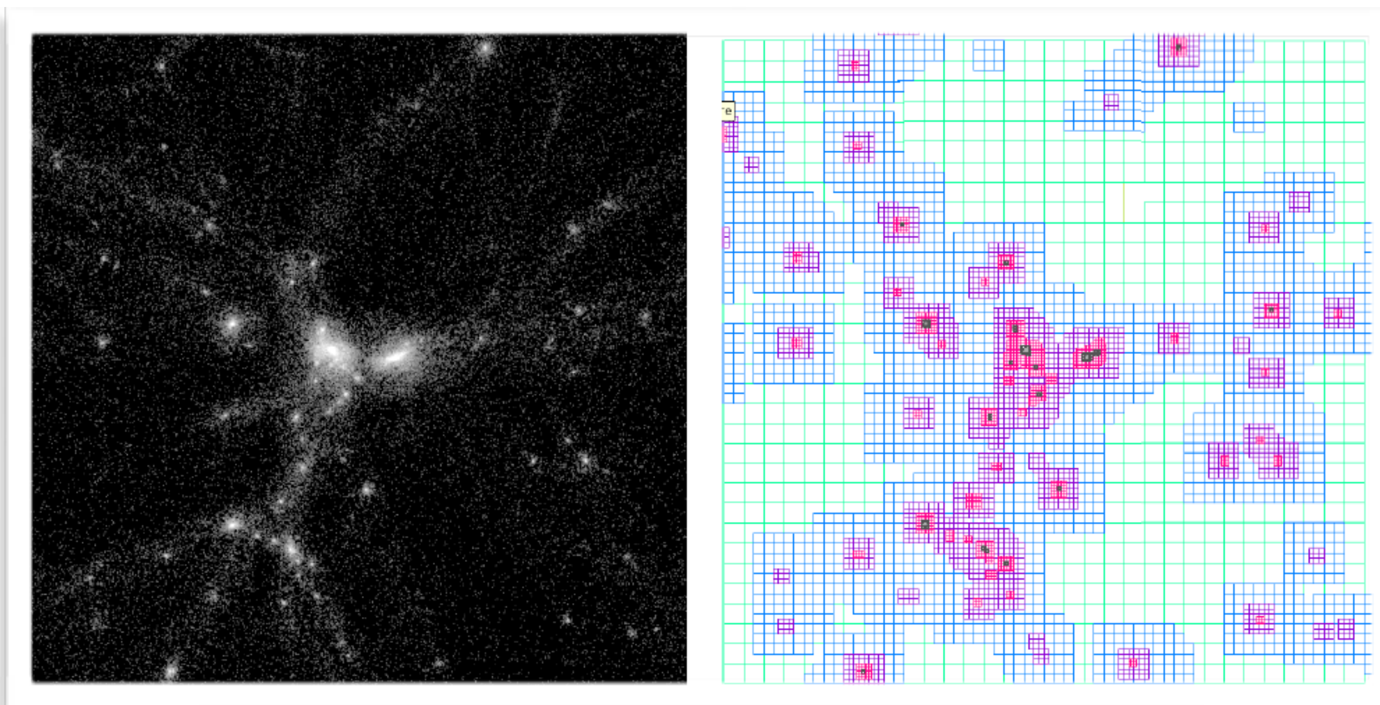
- multigrid relaxation method (continued):
  - perform iterations in red-black sweep:
  - each iteration couples only neighbouring grid points:

    ➡ it takes many iterations for the solution to “propagate” over a large grid
  - better use multigrid acceleration:
    - recovers large scale structure of solution on coarse grid
    - and small scale structure on fine grid

    ➡ much faster convergence (~N scaling)

The multigrid relaxation method

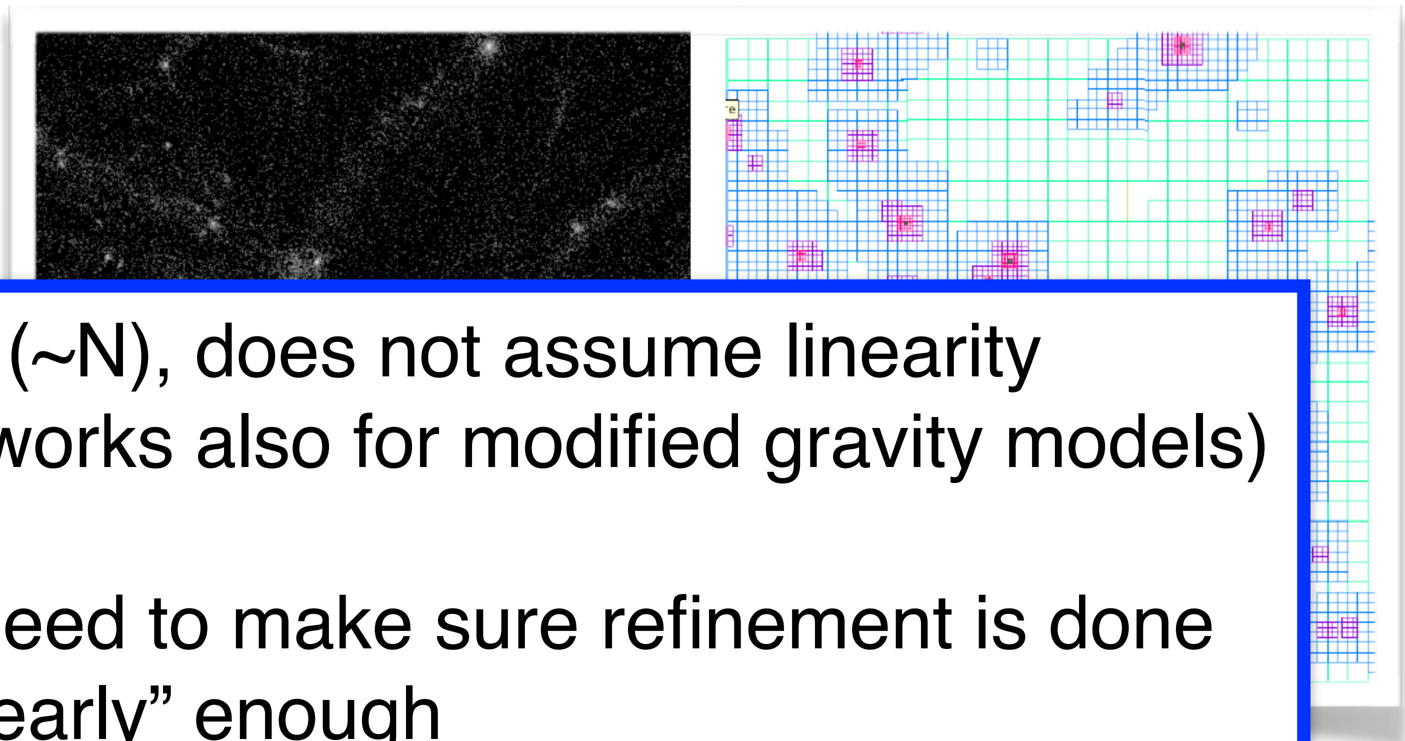
- multigrid relaxation method (continued):
  - so far constant spatial resolution
  - possible to increase resolution using adaptive mesh refinement



source: http://www.deus-consortium.org/a-propos/cosmological-models/run/

The multigrid relaxation method

- multigrid relaxation method (continued):
    - so far constant spatial resolution
    - possible to increase resolution using adaptive mesh refinement



Pros: fast (~N), does not assume linearity
(-> works also for modified gravity models)

Caveats: need to make sure refinement is done "early" enough

Time integration

- How to do the time integration?
  - simplest way - Euler integration:
    $$x_{n+1} = x_n + v_n \Delta t$$
    $$v_{n+1} = v_n + a_n \Delta t$$
    only first order accurate
    -> would need much more time steps for comparable accuracy
  - leap-frog (used in many codes):

$$v_{n+1/2} = v_n + a_n \Delta t/2$$
$$x_{n+1} = x_n + v_{n+1/2} \Delta t$$
$$v_{n+1} = v_{n+1/2} + a_{n+1} \Delta t/2$$

second order accurate

- adaptive timesteps: shortest dynamical timescale changes over time -> code should adapt to it, e.g. $\Delta t \propto 1/\sqrt{a}$
- individual timesteps: dynamical timescale depends on environment (large in low density regions, small in halo centers)

Time integration

- How to do the time integration?
  - simplest way - Euler integration: $x_{n+1} = x_n + v_n \Delta t$
    $v_{n+1} = v_n + a_n \Delta t$

    only first order accurate
    -> would need much more time steps for comparable accuracy
  - leap-frog (used in many codes):

$$v_{n+1/2} = v_n + a_n \Delta t/2$$
$$x_{n+1} = x_n + v_{n+1/2} \Delta t$$
$$v_{n+1} = v_{n+1/2} + a_{n+1} \Delta t/2$$

second order accurate

- adaptive timesteps: shortest dynamical timescale changes over time -> code should adapt to it, e.g. $\Delta t \propto 1/\sqrt{a}$
- individual time

modern codes often use a leapfrog scheme with individual and adaptive time steps

small in halo centers)

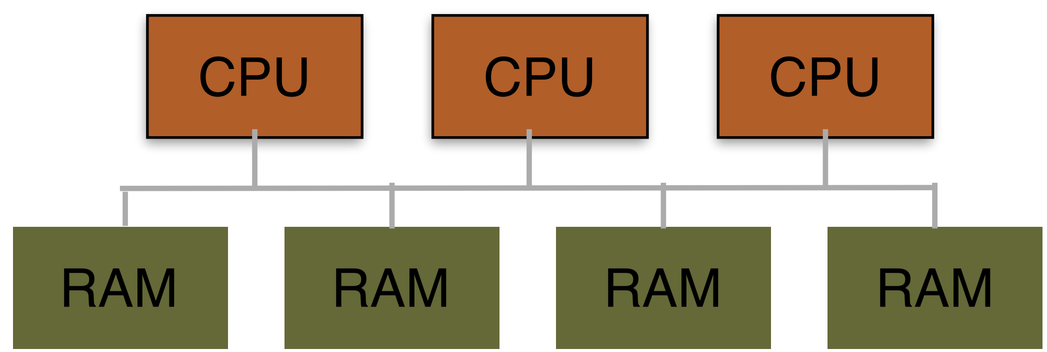
Parallel programming

- shared memory:
  - each CPU can directly access the whole memory
  - communication between tasks via the memory
  - just need to ensure that different tasks do not write to the same memory at the same time
  - e.g. using POSIX threads or OpenMP library

OpenMP examples:

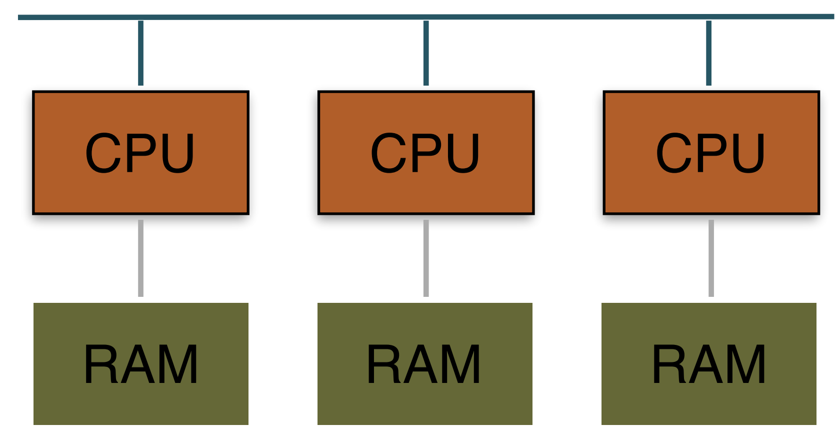
```
#pragma omp parallel for
      for (int i=0; i<100000; i++) a[i] = i*i;

#pragma omp atomic
      count = count+1;
```



Parallel programming

- distributed memory:
  - each CPU can directly access only the local memory (on the same node)
  - communication by explicit commands
  - usually using the MPI library



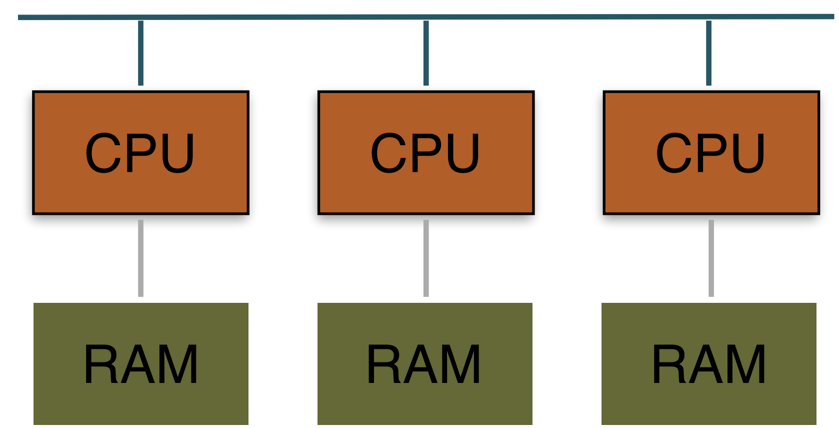
MPI example:

**MPI_Send(void* data, int count, MPI_Datatype datatype, int destination,
    int tag, MPI_Comm communicator)**

**MPI_Recv(void* data, int count, MPI_Datatype datatype, int source,
    int tag, MPI_Comm communicator, MPI_Status* status)**

Parallel programming

- distributed memory:
  - each CPU can directly access only the local memory (on the same node)
  - communication by explicit commands
  - usually using the MPI library



MPI example:

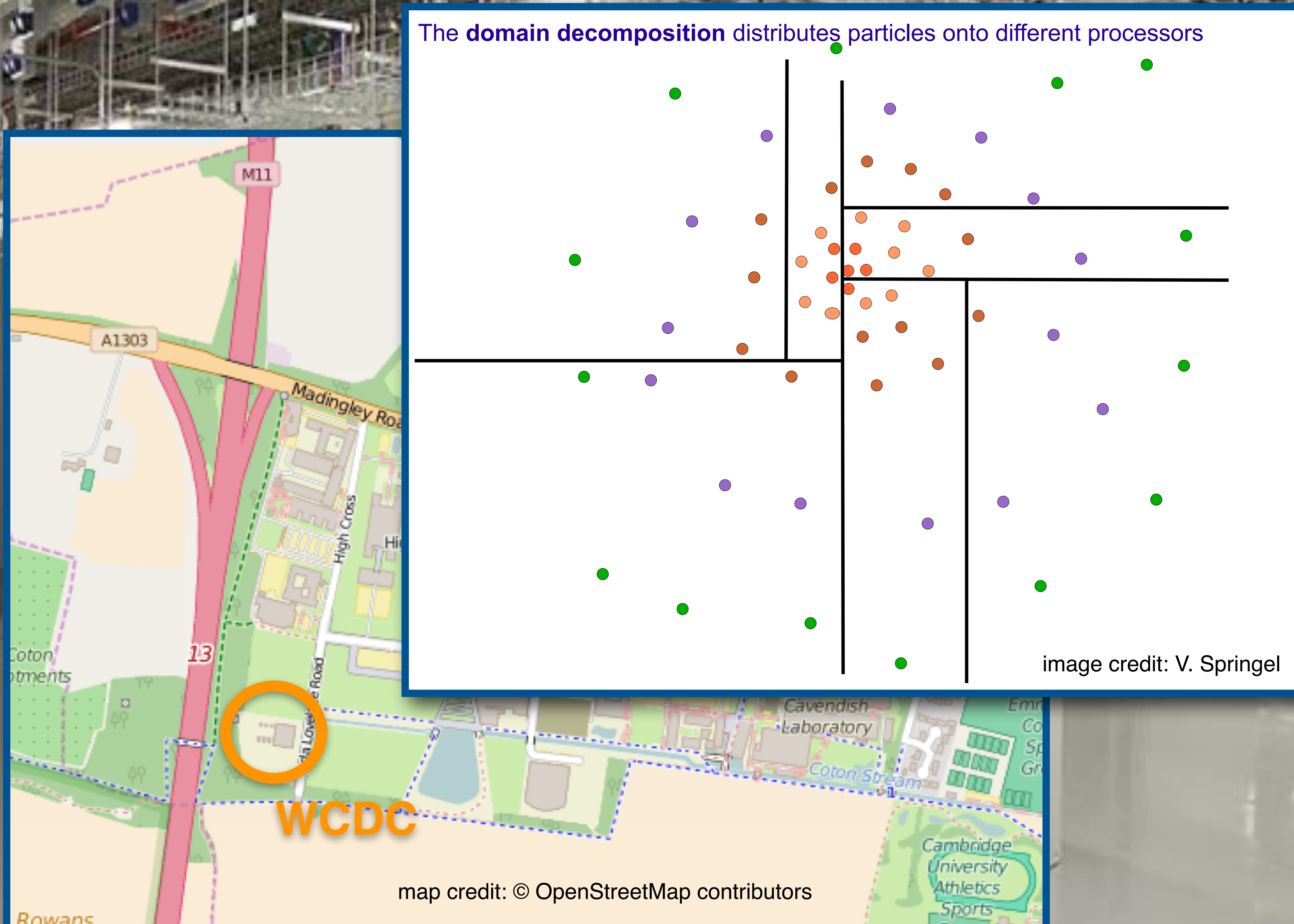
MPI_Send(void* data, int count, MPI_Datatype datatype, int destination,
int tag, MPI_Comm communicator)

MPI_Recv(
int ta

most cosmological simulation codes are MPI parallel or hybrid MPI/OpenMP

image credit: HPCS, University of Cambridge

image credit: HPCS, University of Cambridge



map credit: © OpenStreetMap contributors

The **domain decomposition** distributes particles onto different processors

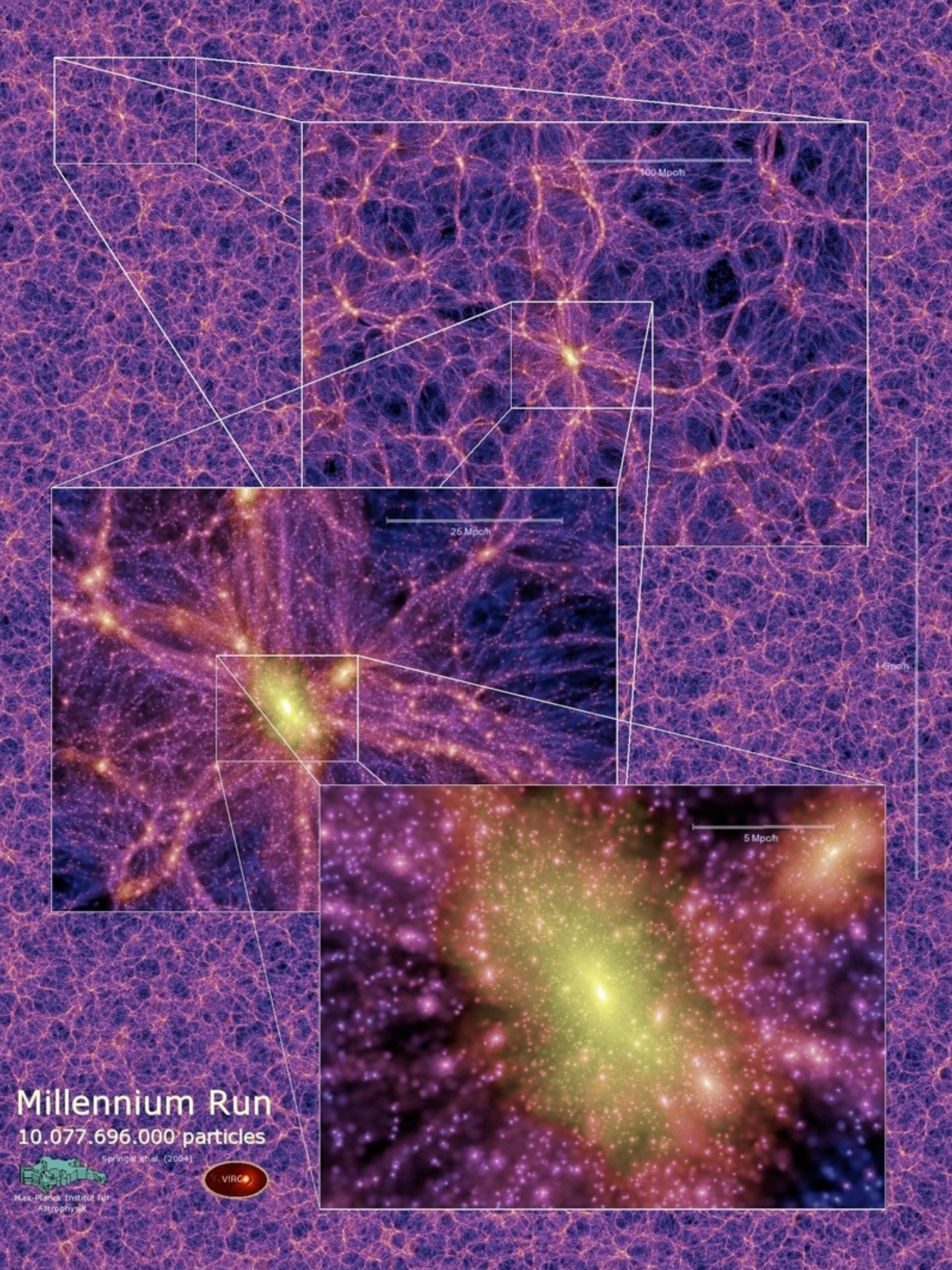
image credit: V. Springel

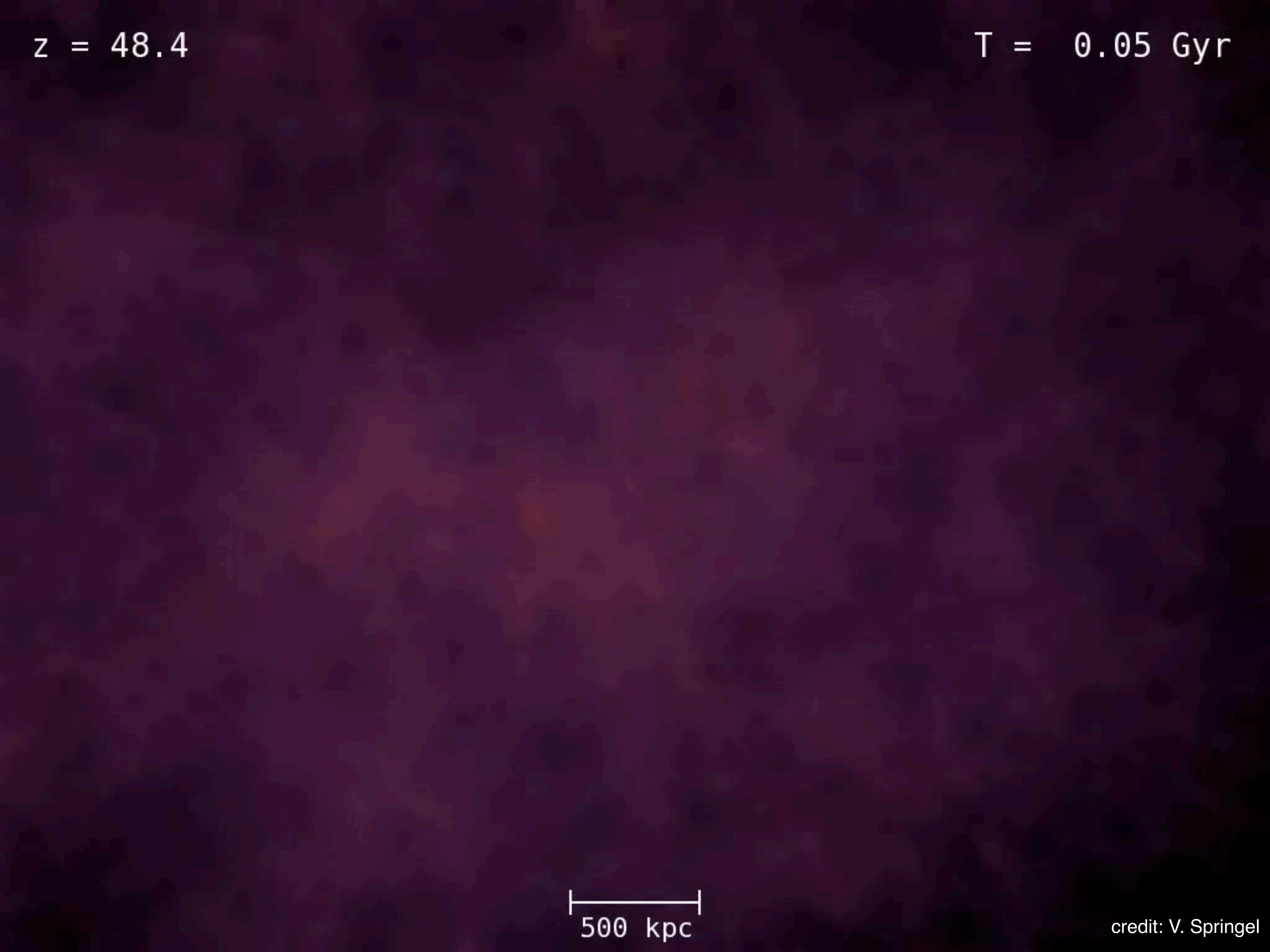
image credit: HPCS, University of Cambridge

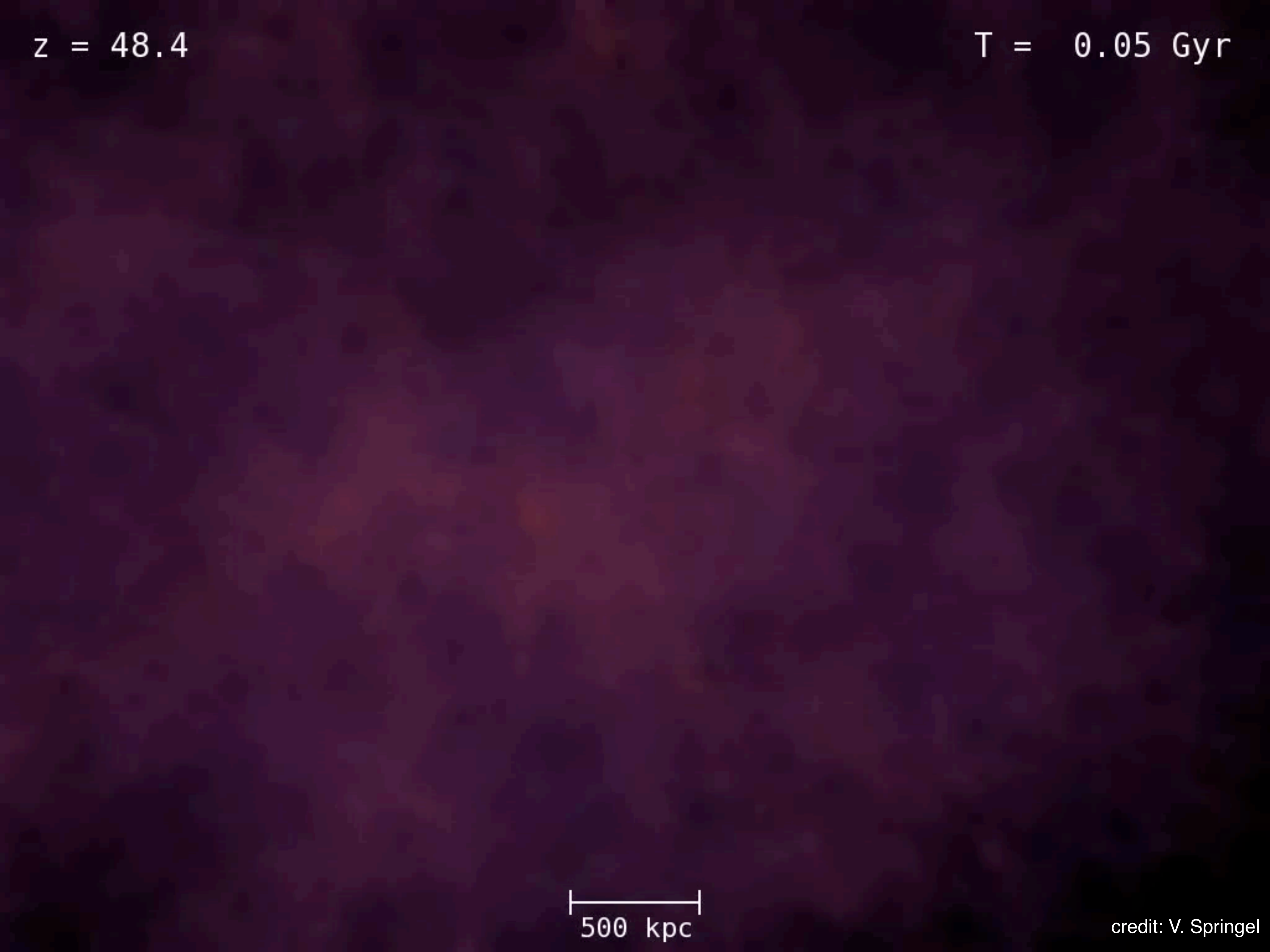
Domain Decomposition

- need to distribute the work to the tasks (each running on one CPU core)
- each task should get the same amount of work for a timestep so that they all finish at about the same time
  - done based on the computational costs of previous timesteps
- memory usage of all tasks should be similar

➡ domain decomposition algorithm tries to balance: gravity work, memory (similar particle number), hydro work


100 Mpc/h
25 Mpc/h
5 Mpc/h
Millennium Run
10.077.696.000 particles
Springel et al. (2004)
Max-Planck Institut für
Astrophysik
VIRGO


z = 48.4
T = 0.05 Gyr
500 kpc
credit: V. Springel

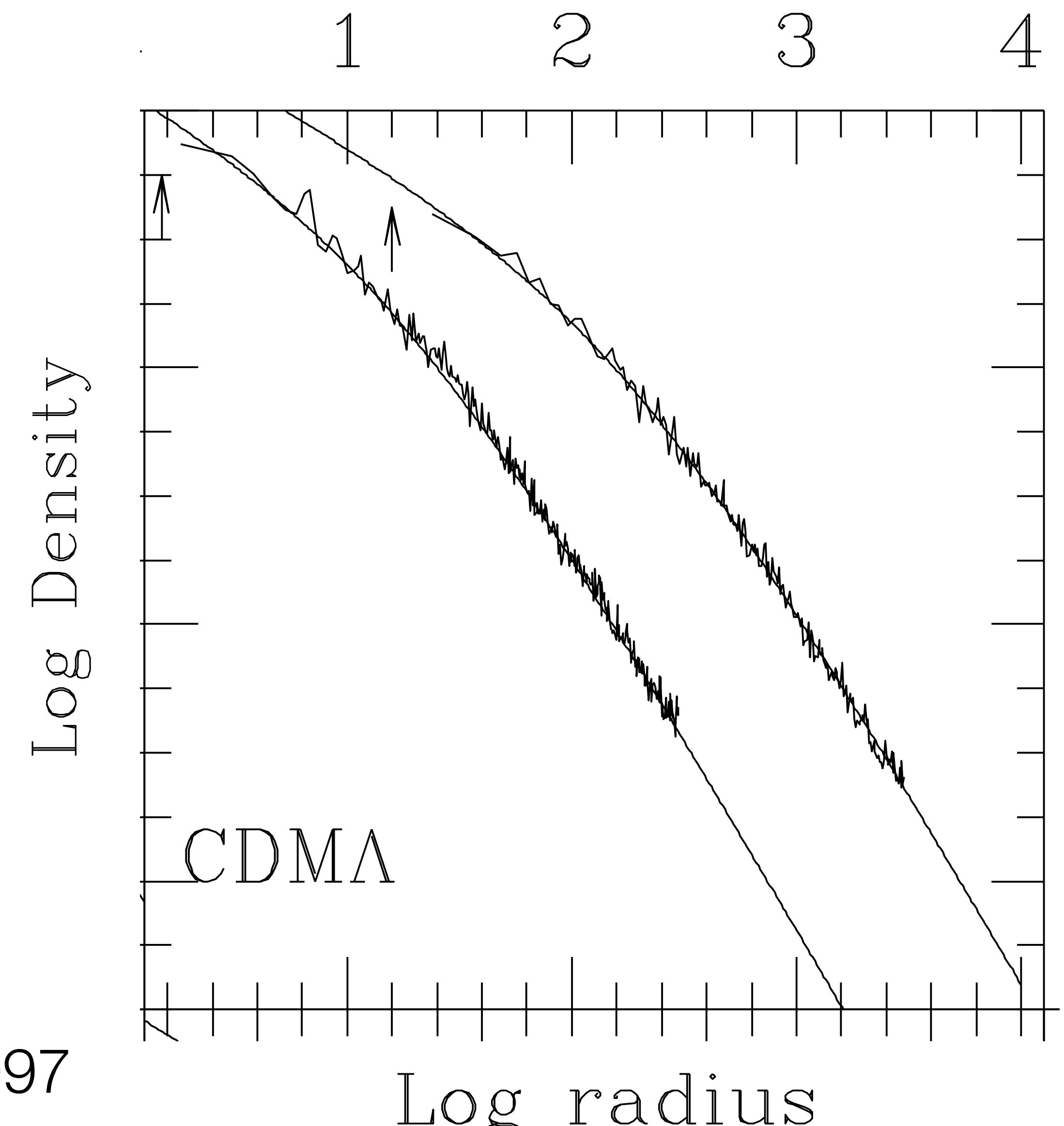

z = 48.4
T = 0.05 Gyr
500 kpc
credit: V. Springel

Some N-body simulation results

dark matter halo density profiles

the NFW profile:

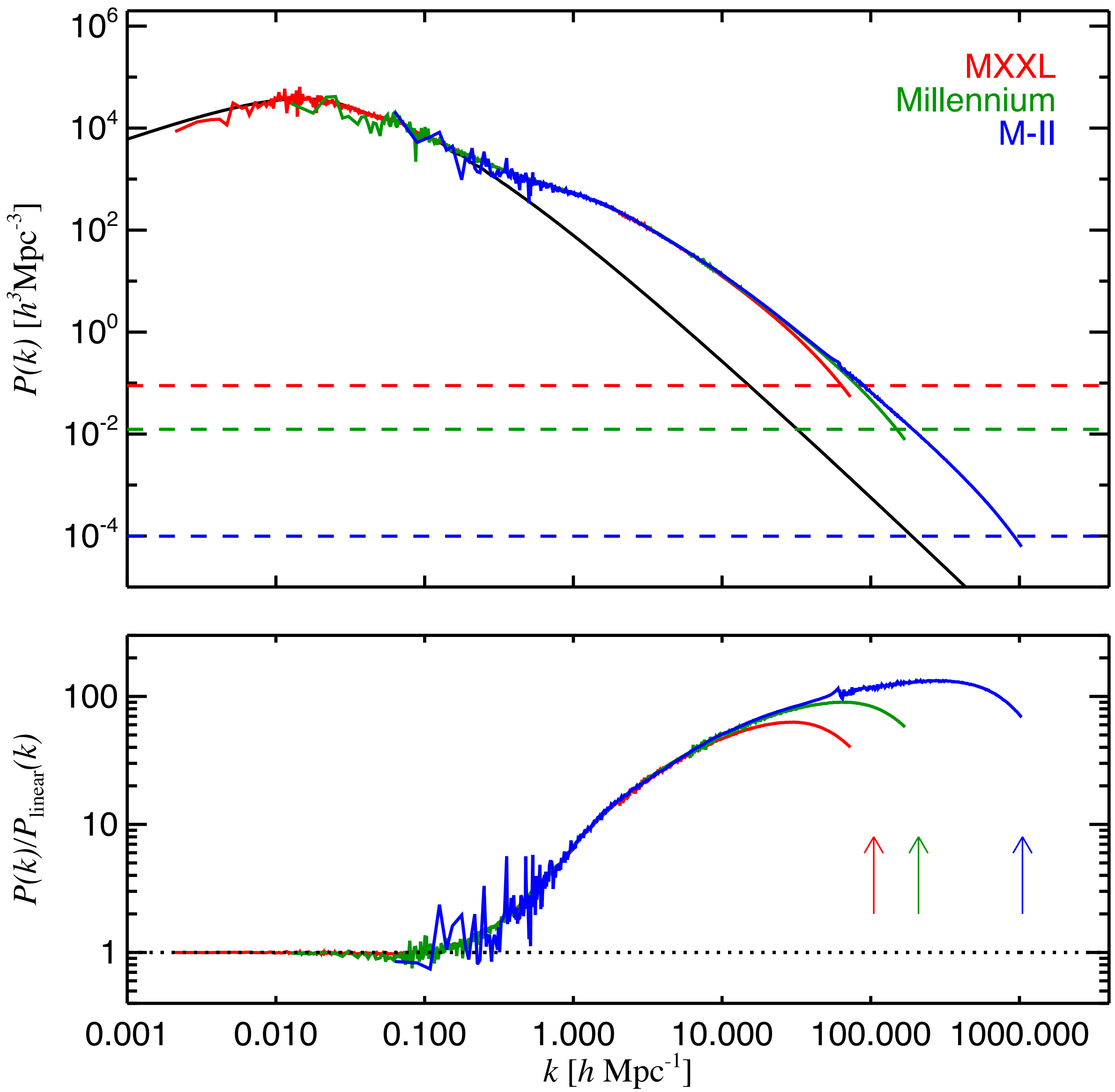
$$\frac{\rho(r)}{\rho_{crit}} = \frac{\delta_c}{(r/r_s)(1+r/r_s)^2}$$



Navarro et al. 1997

Some N-body simulation results

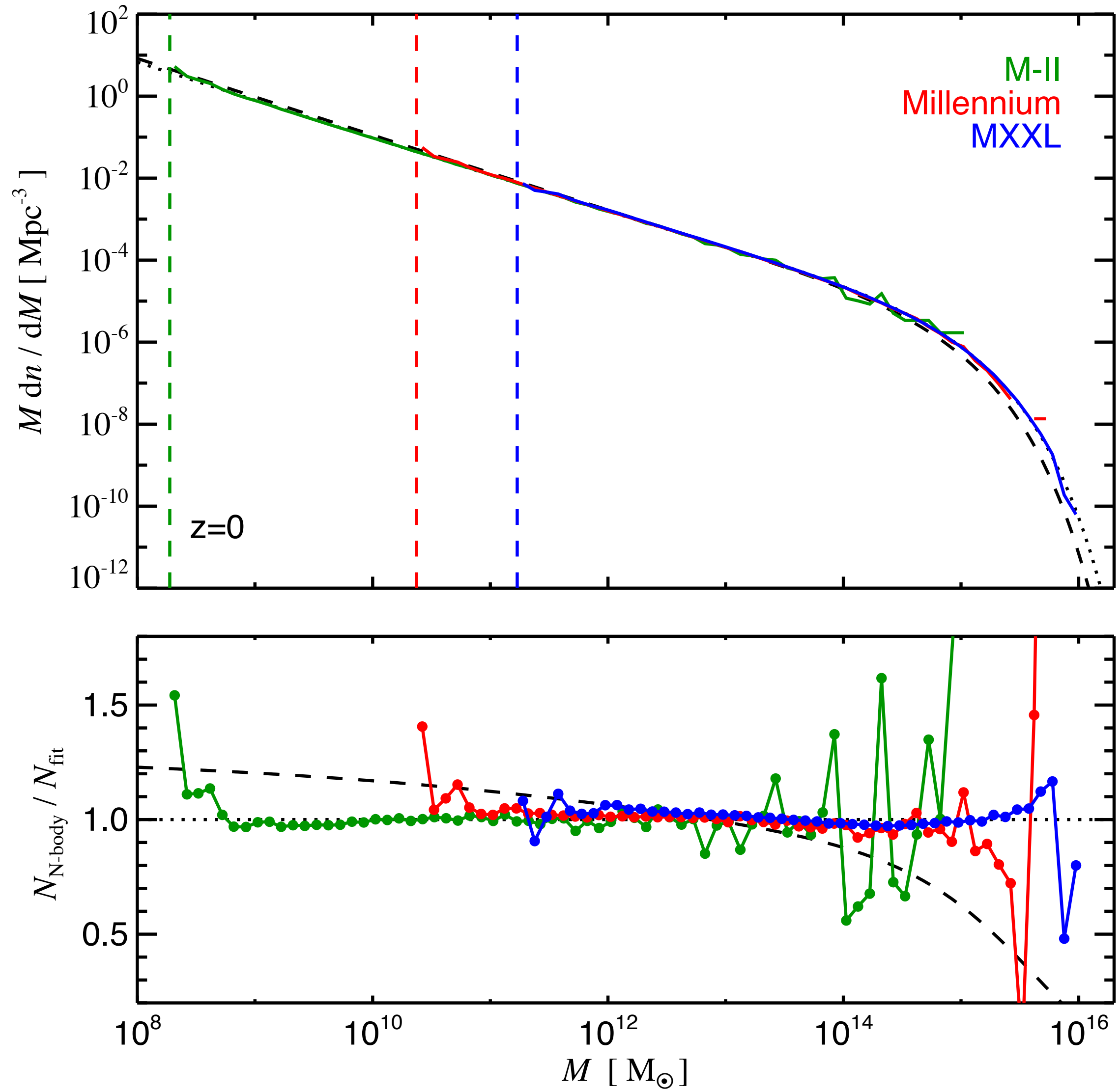
non-linear matter power spectrum



Angulo et al. 2012

Some N-body simulation results

halo mass function



Angulo et al. 2012

Literature & Outlook

- “The cosmological simulation code GADGET-2”, V. Springel, 2005, MNRAS, 364, 1105, arXiv:astro-ph/0505010

- “Simulation techniques for cosmological simulations”, K. Dolag et al. 2008, arXiv:0801.1023

- MPI tutorial: www.zib.de/zibdoc/mpikurs/mpi-course.pdf

- Next lecture:
  - ***Hydrodynamic simulations (on a grid)***