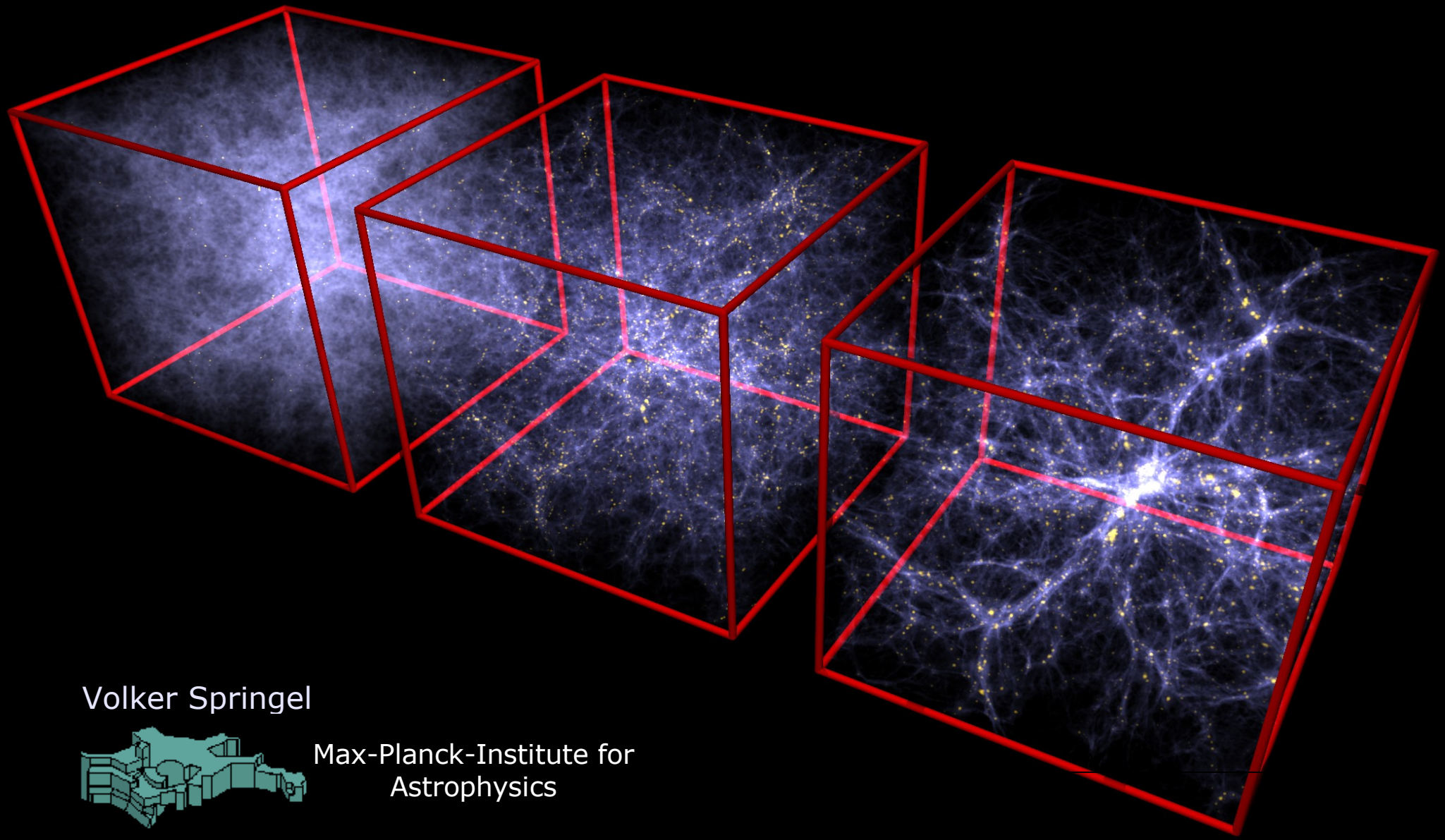
Summer school on cosmological numerical simulations
3rd week – MONDAY

Helmholtz School of Astrophysics
Potsdam, July/August 2006



Volker Springel

Max-Planck-Institute for
Astrophysics

Summer school on cosmological numerical simulations

Tentative plan for lectures of the third week

Volker Springel

Monday ▶ **The GADGET code: Usage, capabilities & limitations, algorithmic aspects**

Tuesday ▶ **Smoothed particle hydrodynamics as a tool to model baryonic physics**

Wednesday ▶ **Time integration methods, semi-analytic simulations**

Thursday ▶ **Galaxy merger simulations, construction of isolated galaxies, physics in mergers**

Friday ▶ **Rounding and debugging issues, version control systems, visualization of particle data**

The GADGET code: Usage, capabilities & limitations, algorithmic aspects

MONDAY-Lecture of 3rd week

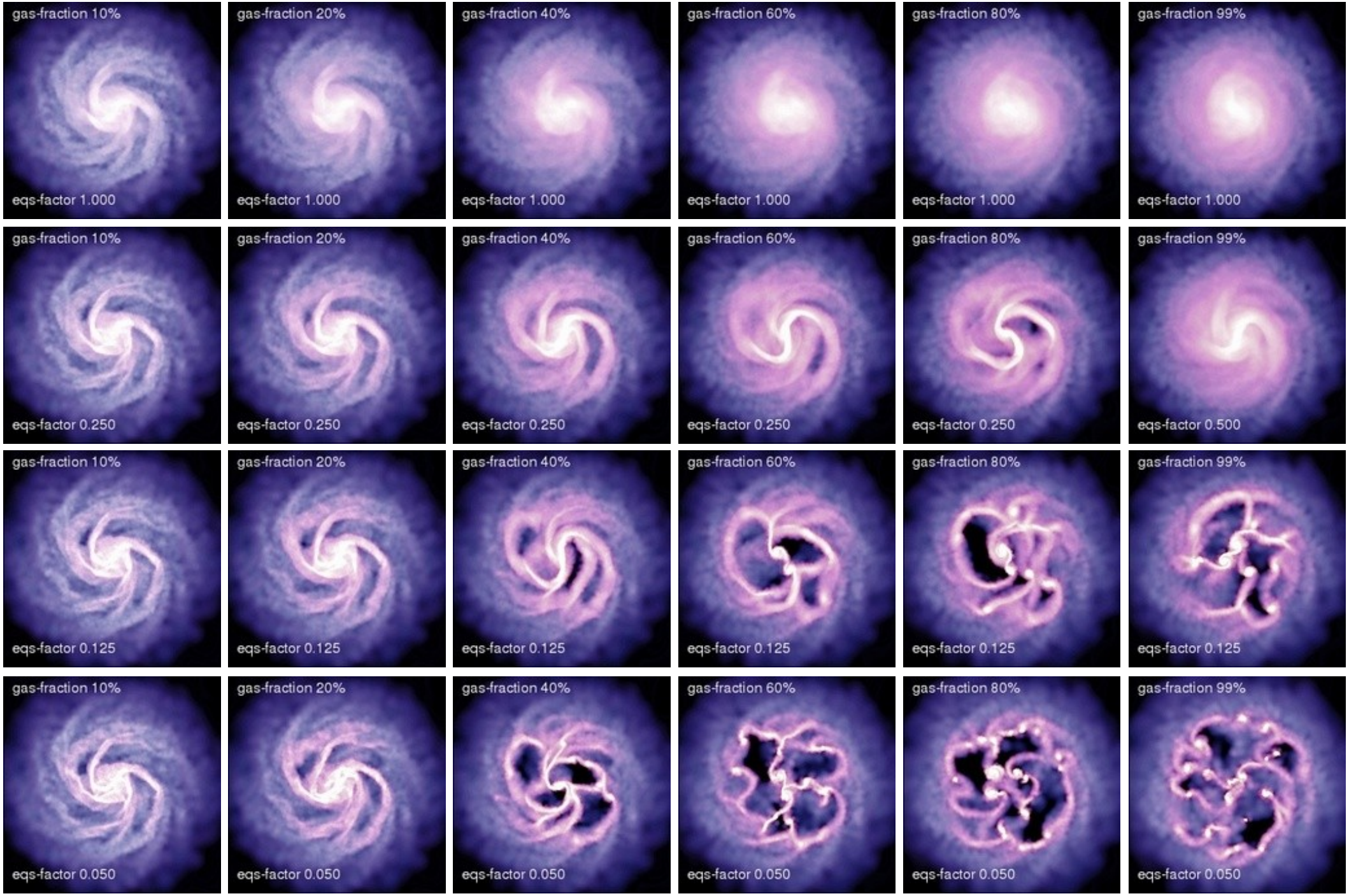
Volker Springel

- Introduction to types of simulations possible with GADGET
- History and features of GADGET- II
- Gravitational force calculation algorithms
- Practical hints for the usage of GADGET- II
- Parallelization algorithms and limits for the scalability

Examples for GADGET simulations

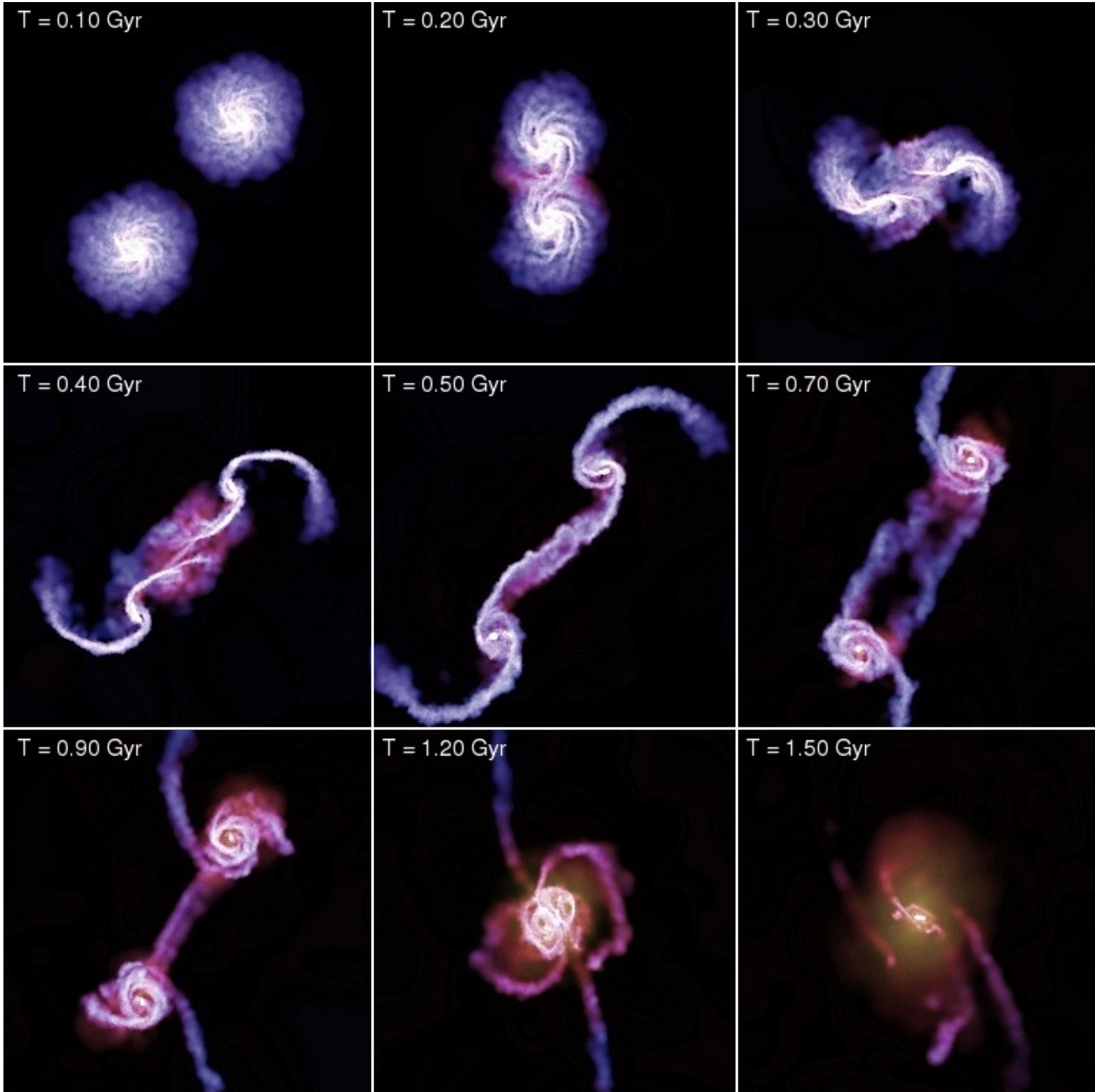
The multiphase-model allows stable disk galaxies even for very high gas surface densities

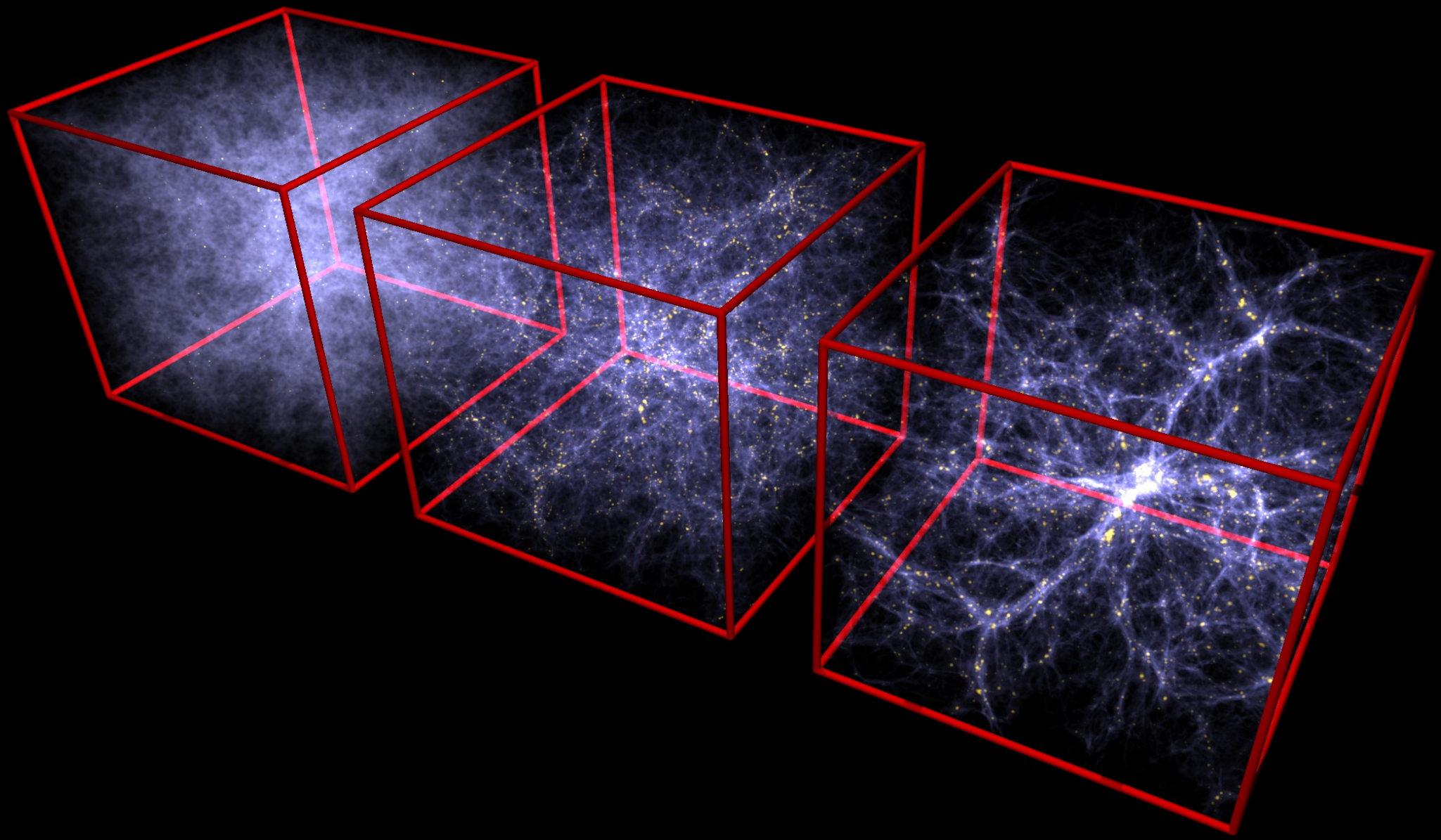
STABILITY OF DISKS AS A FUNCTION OF GAS FRACTION AND EQUATION OF STATE



In major-mergers between two disk galaxies, tidal torques extract angular momentum from cold gas, providing fuel for nuclear starbursts

TIME EVOLUTION OF A PROGRADE MAJOR MERGER



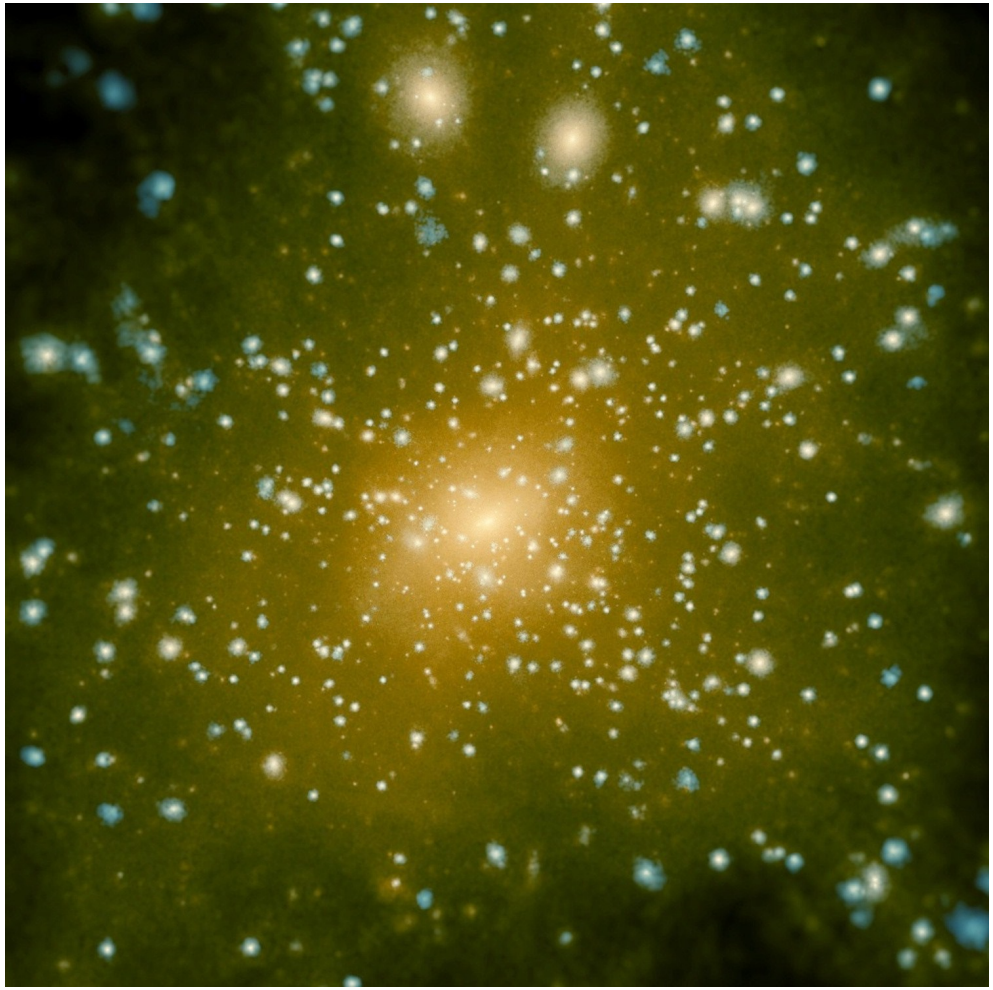


Simulating very high-mass resolution down to low redshift requires a multi-resolution technique

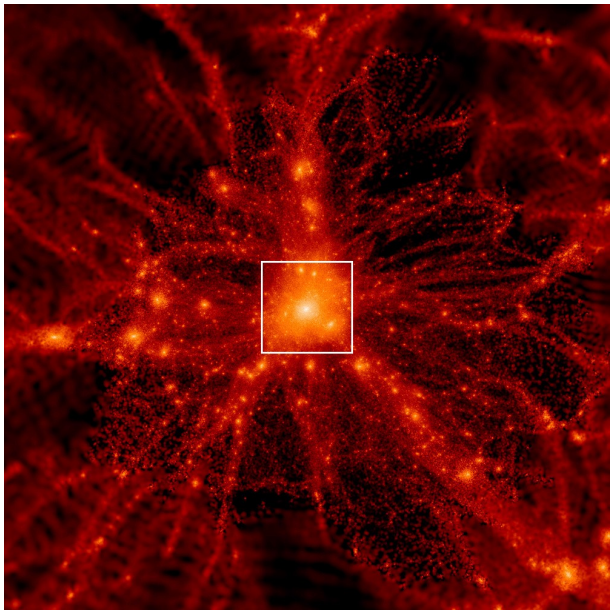
ZOOMING IN ON HALOS OF INTEREST - RESIMULATION TECHNIQUE

Rerun with locally higher resolution:
achieve ~ 10^2-10^3 increase in mass resolution

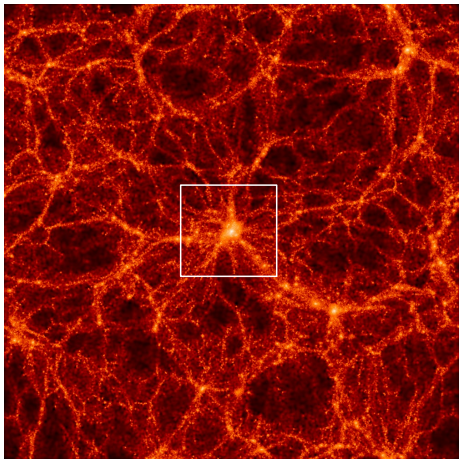
Internal structure of indivudal objects can be studied with very high resolution



Resimulation with spatially varying resolution



Parent simulation:
large-scale structure



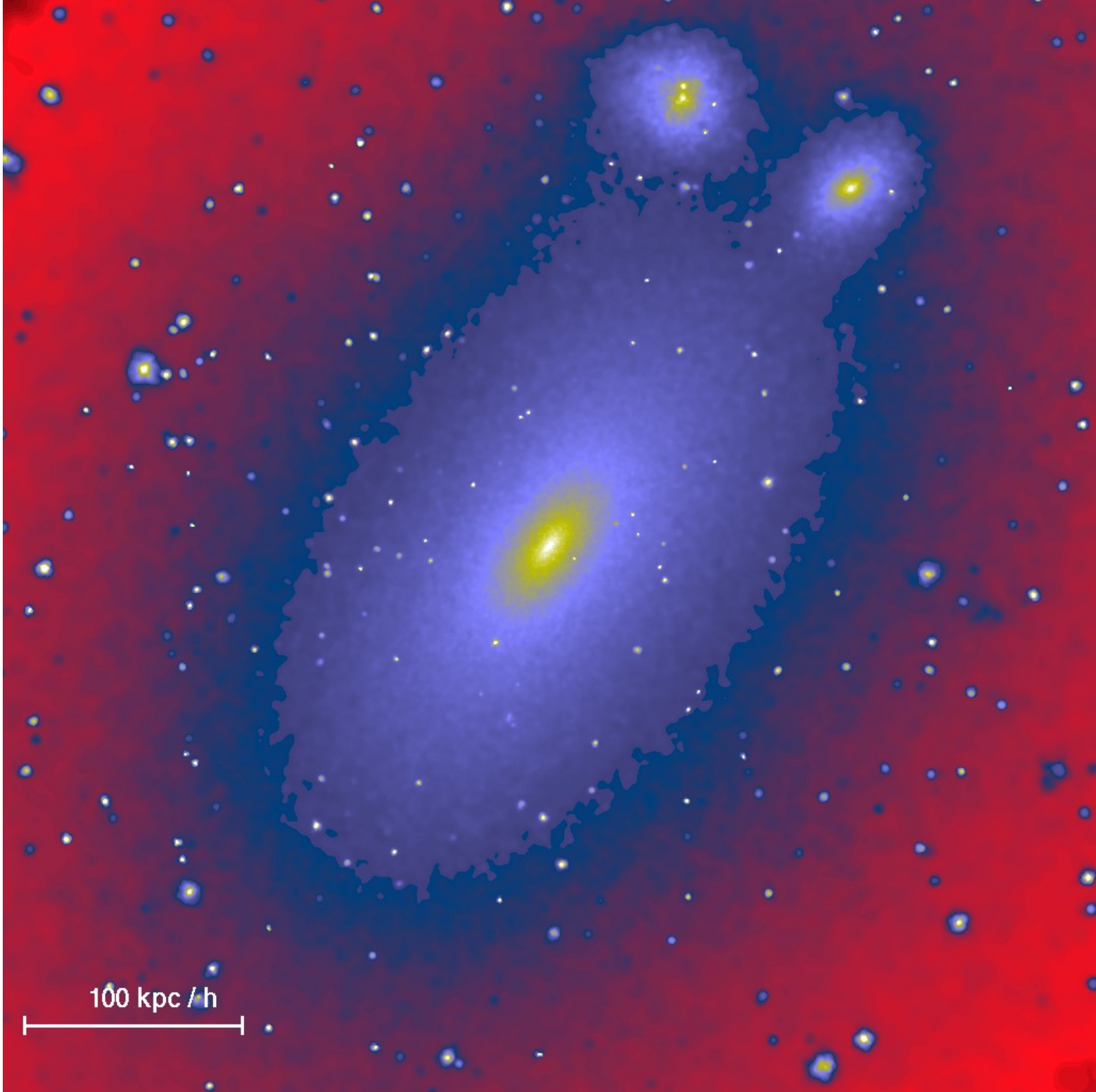
Springel et al. (2001)

Lagrangian simulations allow very high spatial dynamic range in 3D

ZOOM INTO A CLUSTER

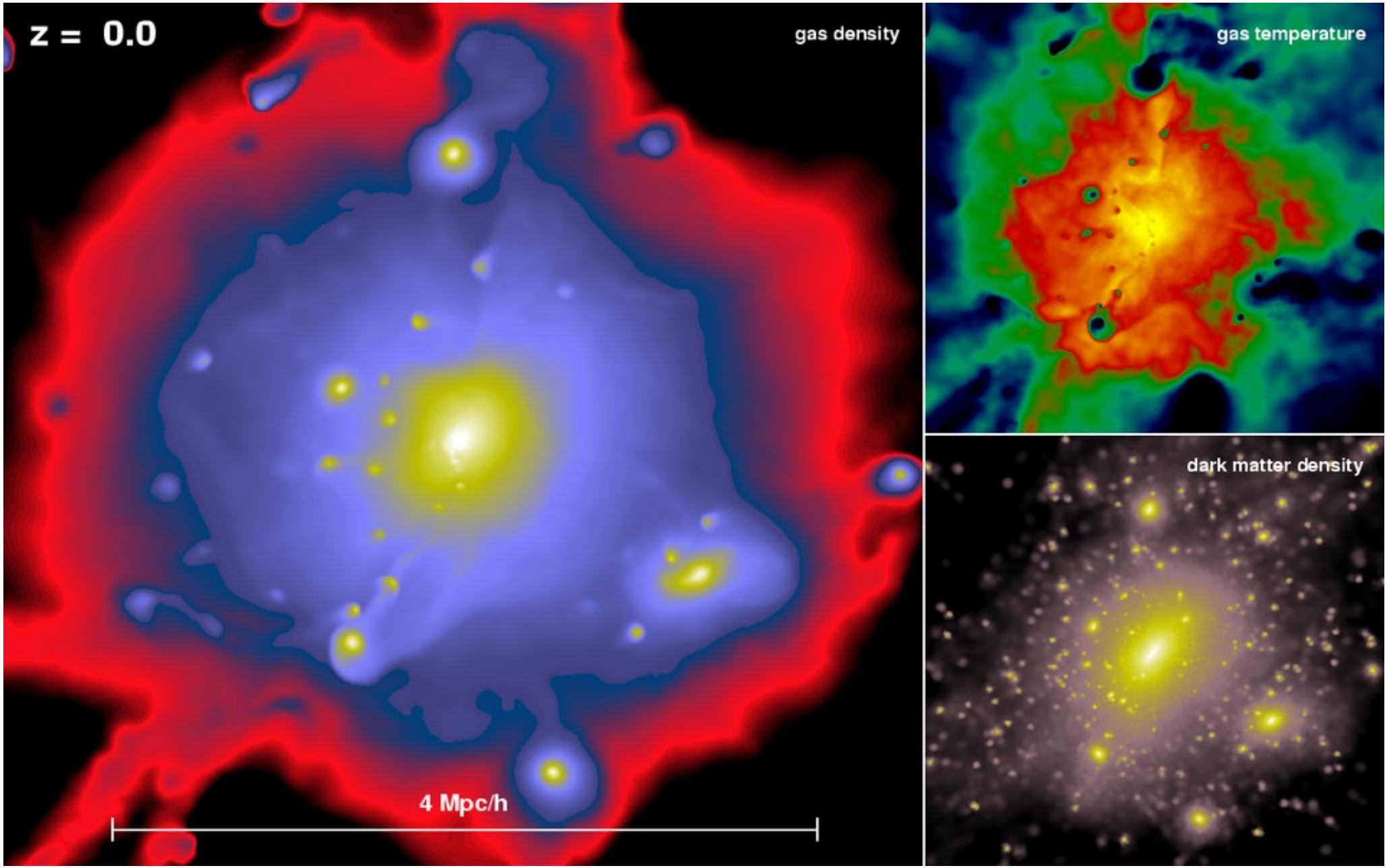
~ 20 million particles within virial radius of cluster
~10^5 spatial dynamic range

Springel, White, Kauffmann, Tormen (2000)



Adiabatic gasdynamics can be readily incorporated in zoom simulations

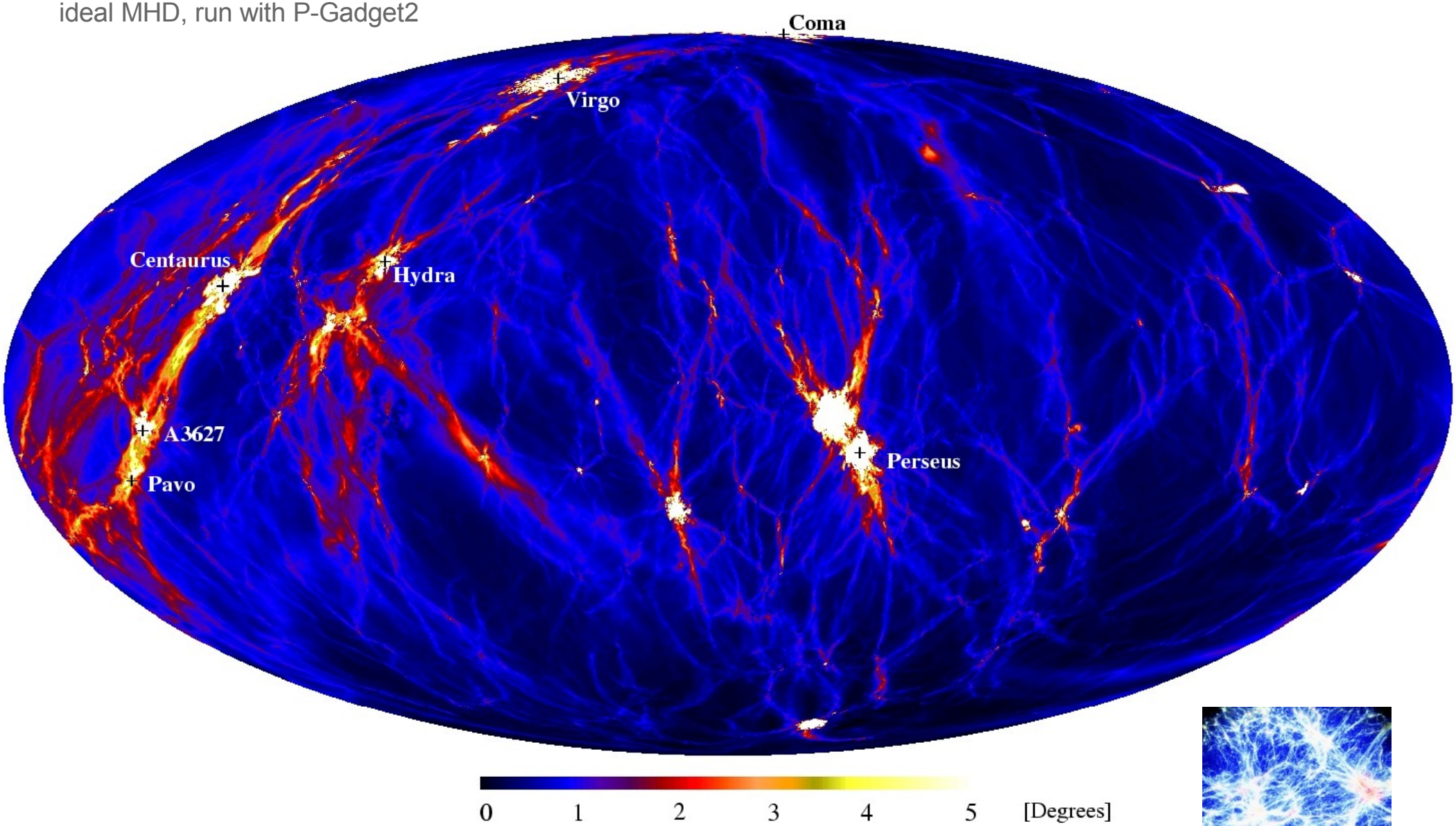
A SIMULATED CLUSTER WITH GAS

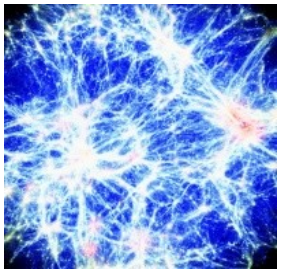


Weak magnetic fields are ubiquitious in the universe

DEFLECTION MAP OF UHECR IN THE LOCAL UNIVERSE

Contrained Simulation of the Local Universe
ideal MHD, run with P-Gadget2



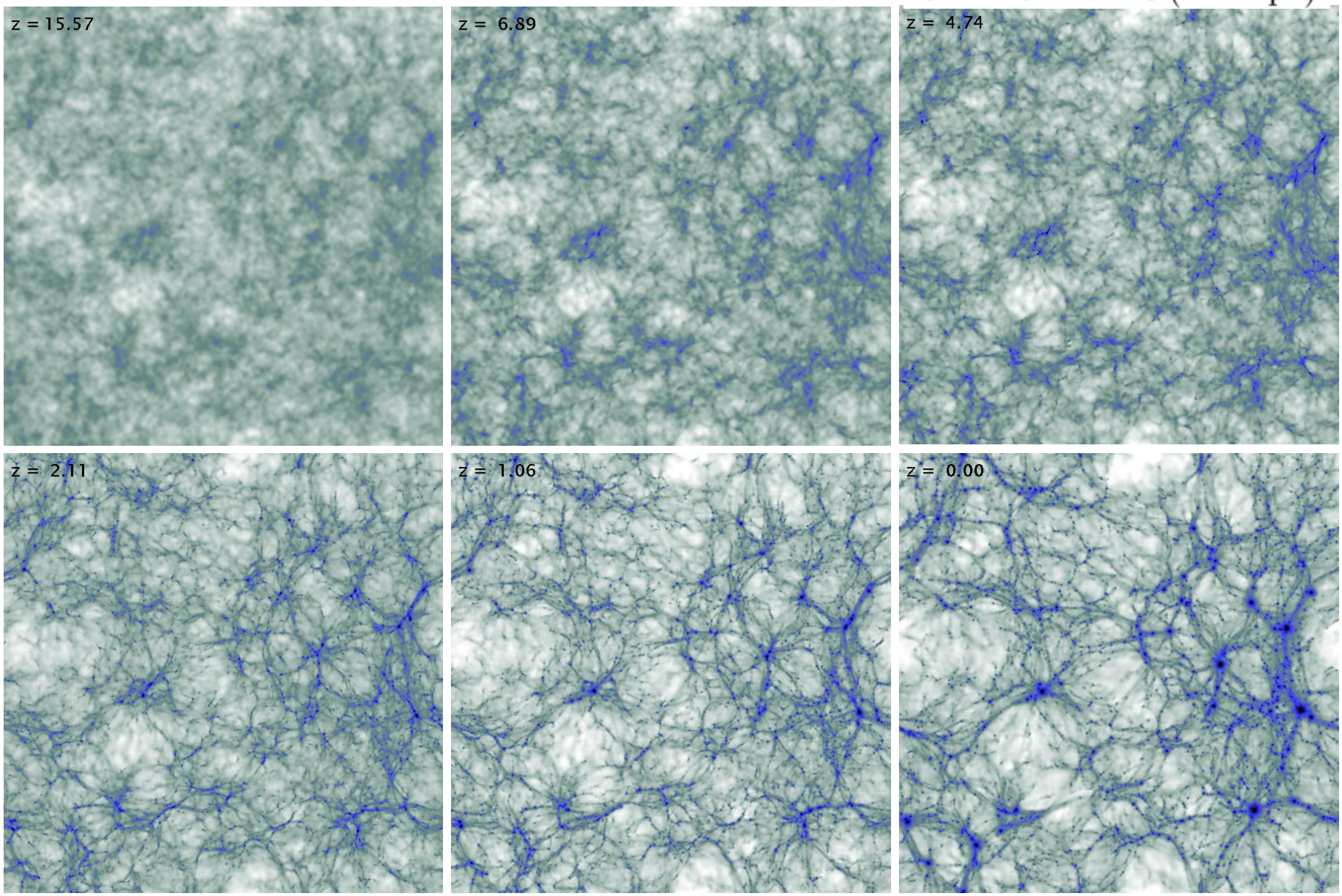


Dolag, Grasso, Springel & Tkachev (2003):

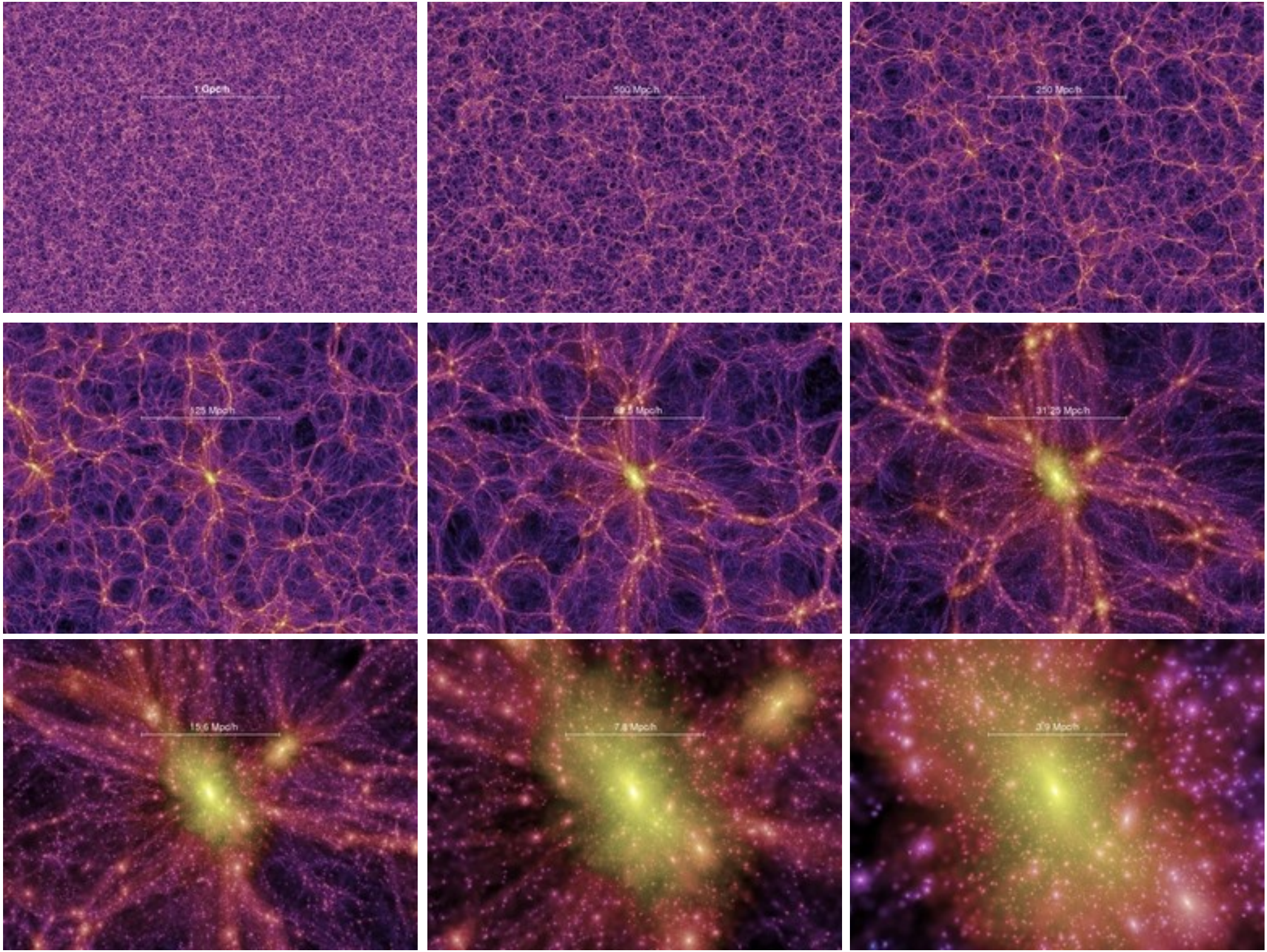
Simulations on scales of order 100 Mpc are the workhorses of studies of large-scale structure formation

EVOLUTION OF STRUCTURE IN THE GAS DISTRIBUTION

ΛCDM, $N = 2 \times 224^3$
$134 \times 134 \times 22.3\,(h^{-1}\mathrm{Mpc})^3$

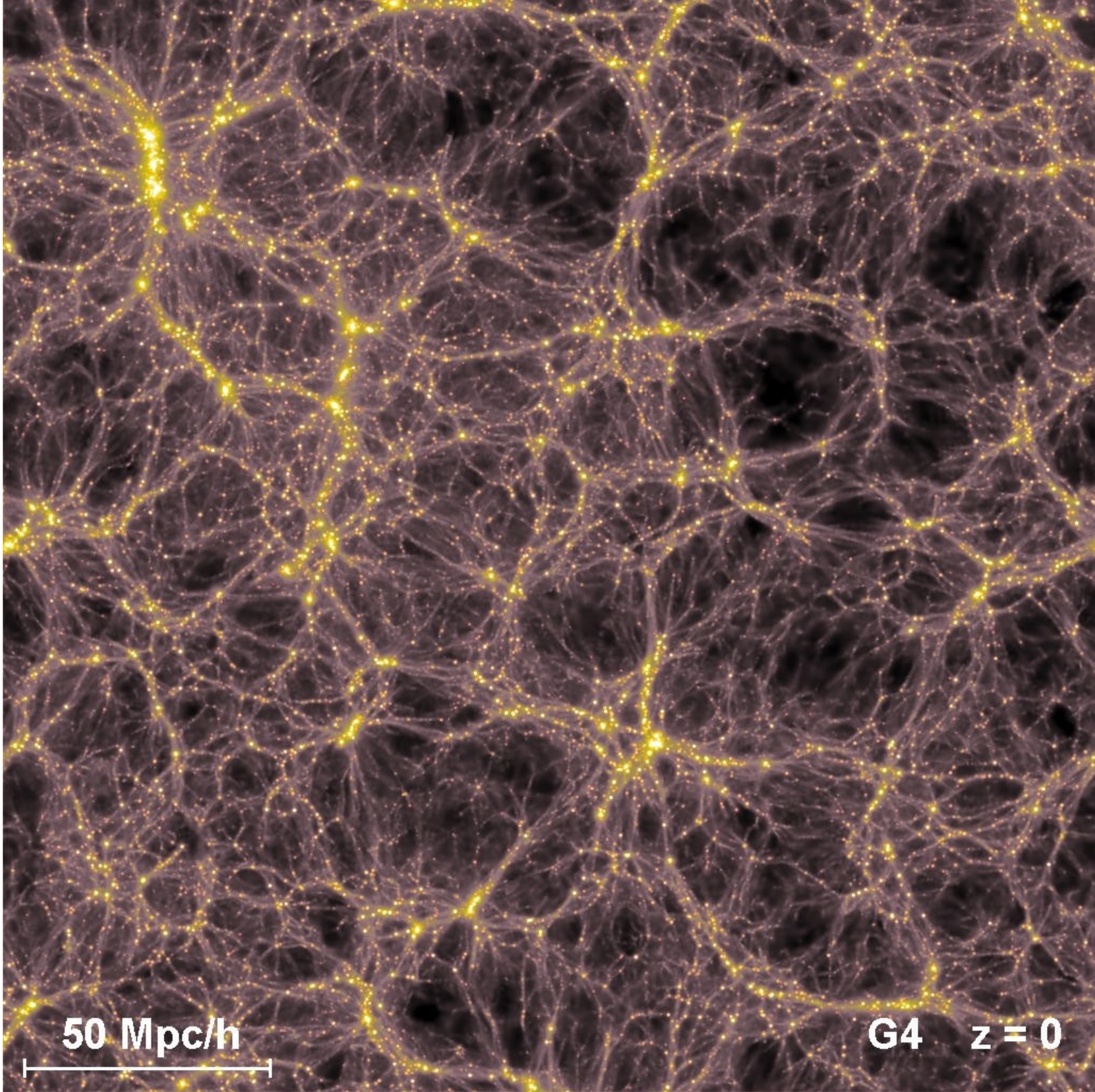


Springel, Hernquist & White (2000)


1 Gpc/h
500 Mpc/h
250 Mpc/h
125 Mpc/h
62.5 Mpc/h
31.25 Mpc/h
15.6 Mpc/h
7.8 Mpc/h
3.9 Mpc/h

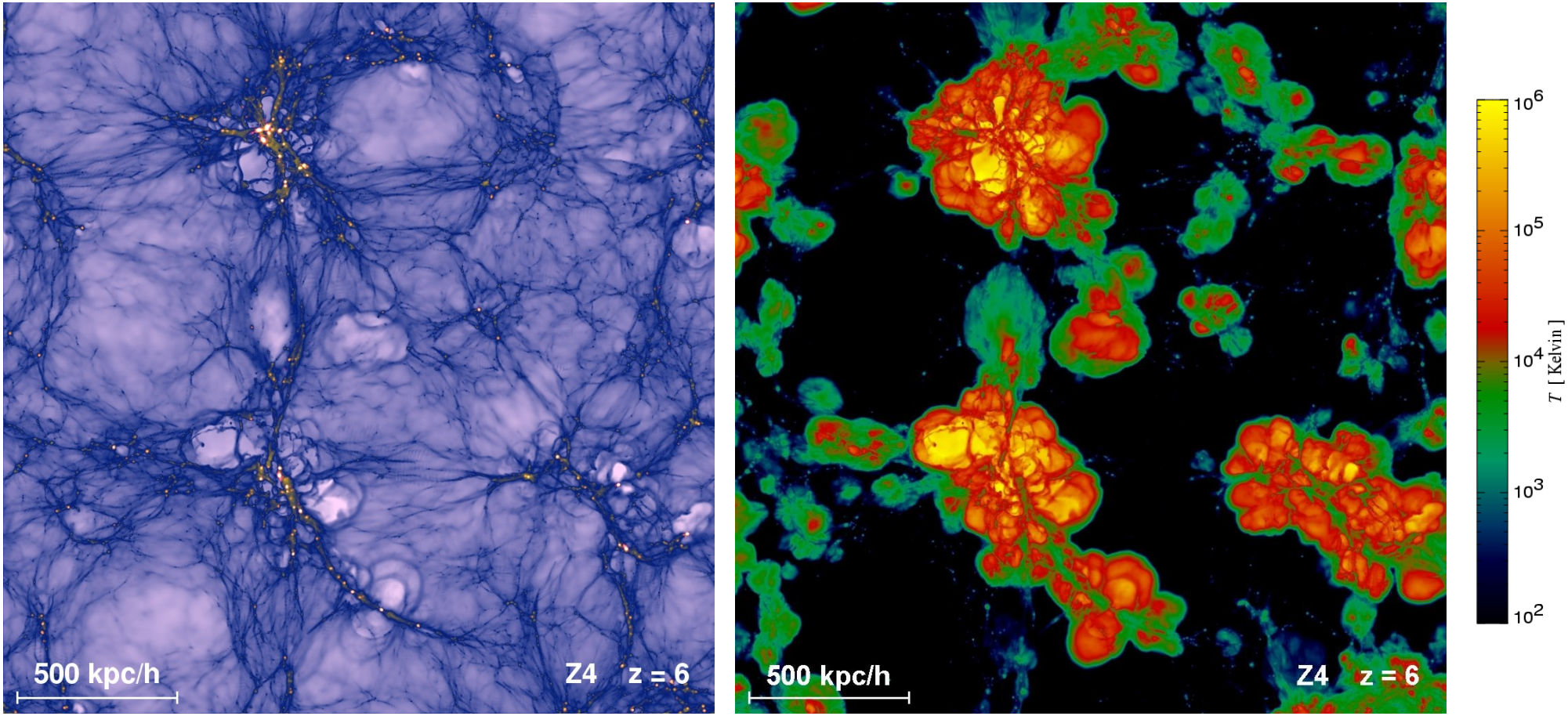
Cosmological hydrodynamical simulations can directly follow galaxy formation

BARYONIC DENSITY IN SIMULATIONS WITH RADIATIVE COOLING, STAR FORMATION AND FEEDBACK



Galactic winds reduce the star formation efficiency of low-mass galaxies

HOT BUBBLES IN THE IGM AROUND SMALL PRIMEVAL GALAXIES



Springel & Hernquist (2003)

Features and history of GADGET

GADGET is a versatile TreeSPH N-body code for cosmological applications

PRINCIPLE CHARACTERISTICS OF GADGET

- Gravity solver based on a TREE or TreePM algorithm
- Hydrodynamics is followed by means of SPH
- Timesteps can be individual and adaptive
- Code is parallelized with MPI for distributed memory architectures
- Code is written in C and is highly portable
- A basic version of the code is publicly available

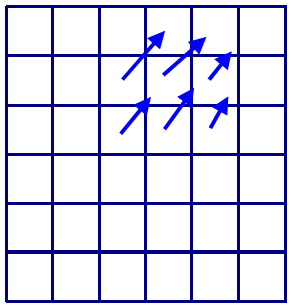
What is smoothed particle hydrodynamics?

DIFFERENT METHODS TO DISCRETIZE A FLUID

Eulerian

discretize space

representation on a mesh
(volume elements)



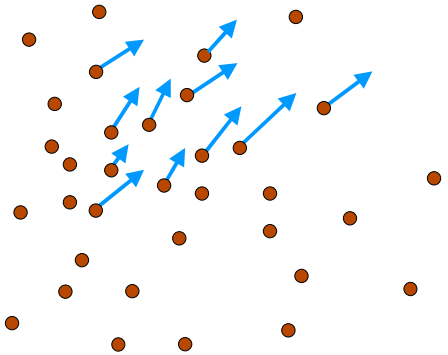
principle advantage:

high accuracy (shock capturing), low numerical viscosity

Lagrangian

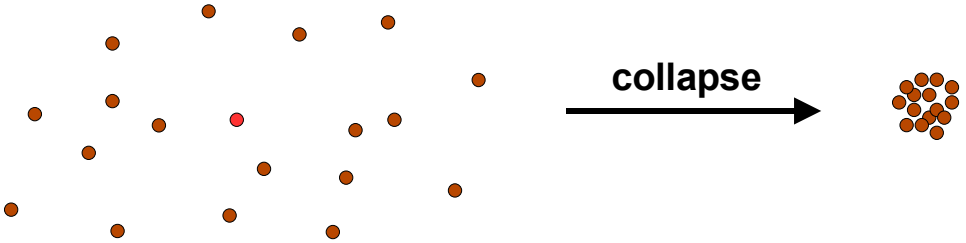
discretize mass

representation by fluid elements
(particles)



principle advantage:

resolutions adjusts automatically to the flow



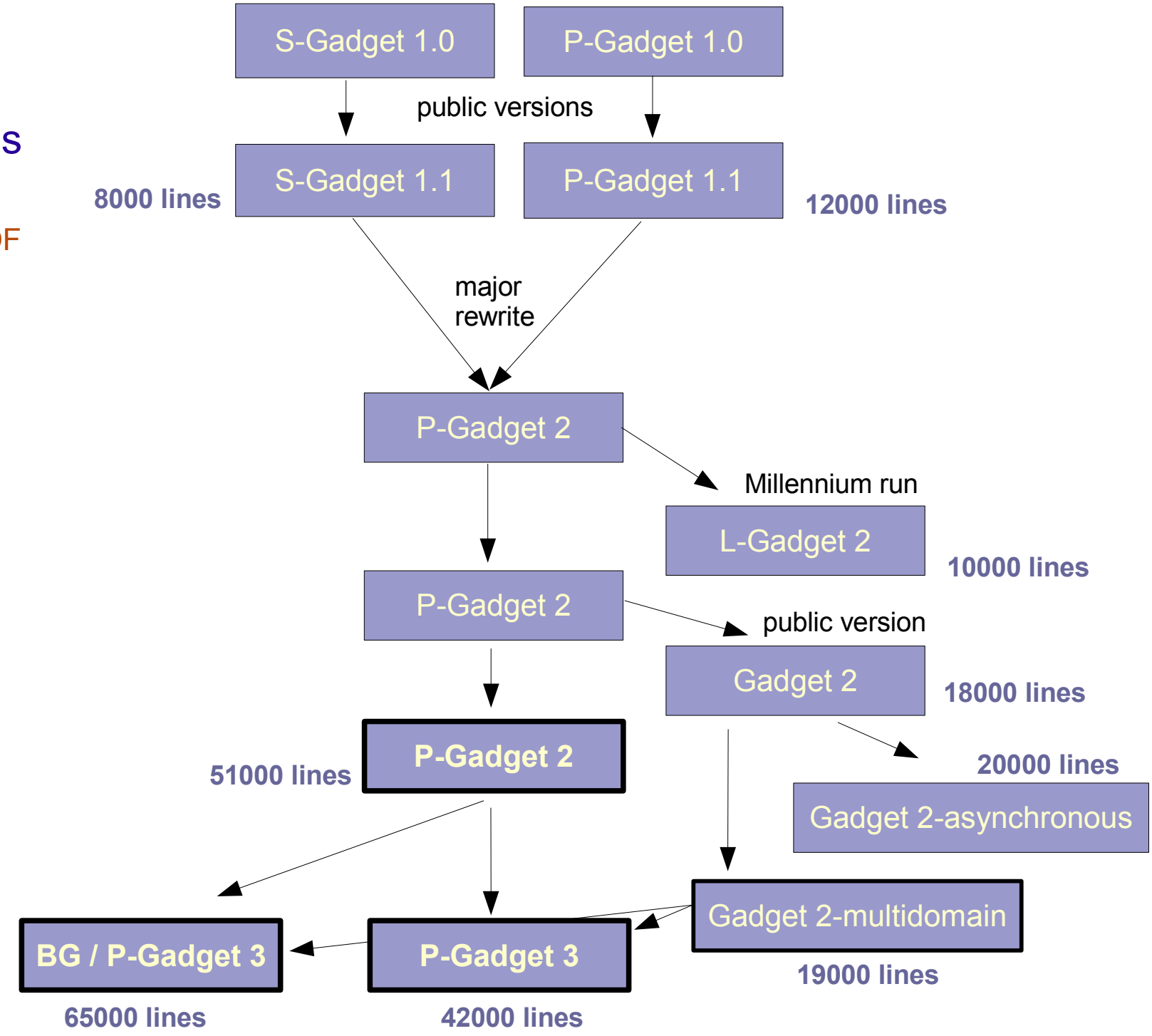
GADGET has evolved over the years and is in a process of continuous change

MAJOR EVENTS IN GADGET'S DEVELOPMENT

- 1997 first version, serial GRAPE code
- 1998 first tree code, first parallel MPI code, serial SPH
- 1999 large production runs on 512 processors Cray T3E, SPH in parallel code
- 2000 Gadget-1.0 released to the public
- 2001 Gadget-1.1 released
- 2002 complete rewrite, first versions of Gadget-2
- 2003 TreePM finalized, code for Millennium Run written
- 2004 lots of physics added to code
- 2005 public version Gadget2 released
- **2006 lots of physics added to code, Gadget3?**

There is now a collection of different versions of GADGET

THE FAMILY TREE OF GADGET



We recently developed a largely new cosmological code: GADGET-II

NEW FEATURES OF GADGET-II

- New symplectic integration method
- Higher speed of the tree algorithm
- Less memory consumption for tree and particle storage (~100% saving)
- **Code may be run optionally as a TreePM hybrid code** ← **Key feature for Millenium Run**
- SPH neighbour search faster
- Conservative SPH formulation
- Fully consistent dynamic tree updates
- Additional types of runs possible (e.g. 2D, hydrodynamics-only, long periodic boxes)
- Efficient and clean formation of star particles
- More physics
- More output options, including HDF5
- Still fully standard C & standard MPI. The FFTW and GSL libraries are needed.
- Reduced communication overhead, better scalability, arbitrary number of cpus
- Built in parallel group finder

The new code is quite a bit better than the old version...

Physics in GADGET-II for simulations of galaxy formation

- Radiative cooling, UV background (homogeneous)
- Subresolution multiphase model for the ISM: Star formation and feedback
- Phenomenological model for galactic winds
- Detailed chemical enrichment
- Thermal conduction
- Magneto-hydrodynamics
- Non-thermal relativistic component (cosmic rays)
- Growth of supermassive black holes and AGN feedback
- Bubble heating and feedback by AGN
- Shock detection
- Physical viscosity via Navier-Stokes equation

Hopefully additional in the future...

Why is GADGET written in C ?

SOME PROS AND CONS FOR DIFFERENT LANGUAGES

C

- highly portable (ANSI standardized)
- free, efficient, and reliable compilers on all platforms
- compact syntax, free format source code
- powerful pointer arithmetic
- dynamic memory allocation
- powerful low-level bit operations
- call by reference or call by value
- easy to learn
- direct access to all UNIX functionality
- subset of C++
- used a lot outside of physics (good for a career outside research if needed)

- no run-time library for automatic I/O error checking
- no array bound checks
- easy to write code that will seg-fault
- language is not really designed for numerical work

Fortran

- easy to learn
- run-time library helps in tracking down I/O errors
- language is efficient for numerical work, Fortran90 matrix arithmetic is convenient and fast
- large body of legacy code available in physics
- limited feature set and possibility to do array bound checks help to avoid simple coding mistakes
- allows implicit types and quick & dirty coding

- compilers are often proprietary and buggy (small user base)
- easy to write code that is not well portable
- wordy syntax, anachronistic restrictions on source code format
- static memory model (Fortran90 allows also dynamic allocation)
- dying language, not used a lot any more outside of physics
- error-prone treatment of global variables (common blocks)
- cumbersome interfaces to UNIX-libraries

C++

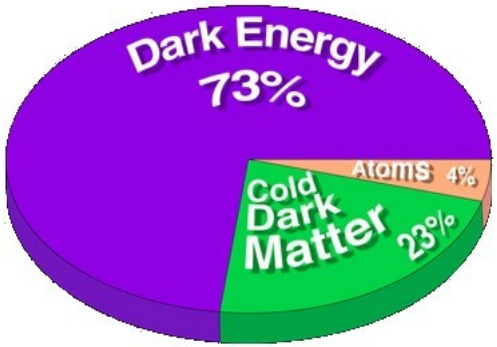
- strong typing enforced
- object oriented, includes features such as operator overloading, inheritence, virtual functions and classes, etc.
- allows code encapsulation, supports writing reusable, modular and extensible code
- powerful standard template library

- difficult to learn (numerous abstract concepts and subtle syntax)
- often lower performance in numerical applications than C or Fortran

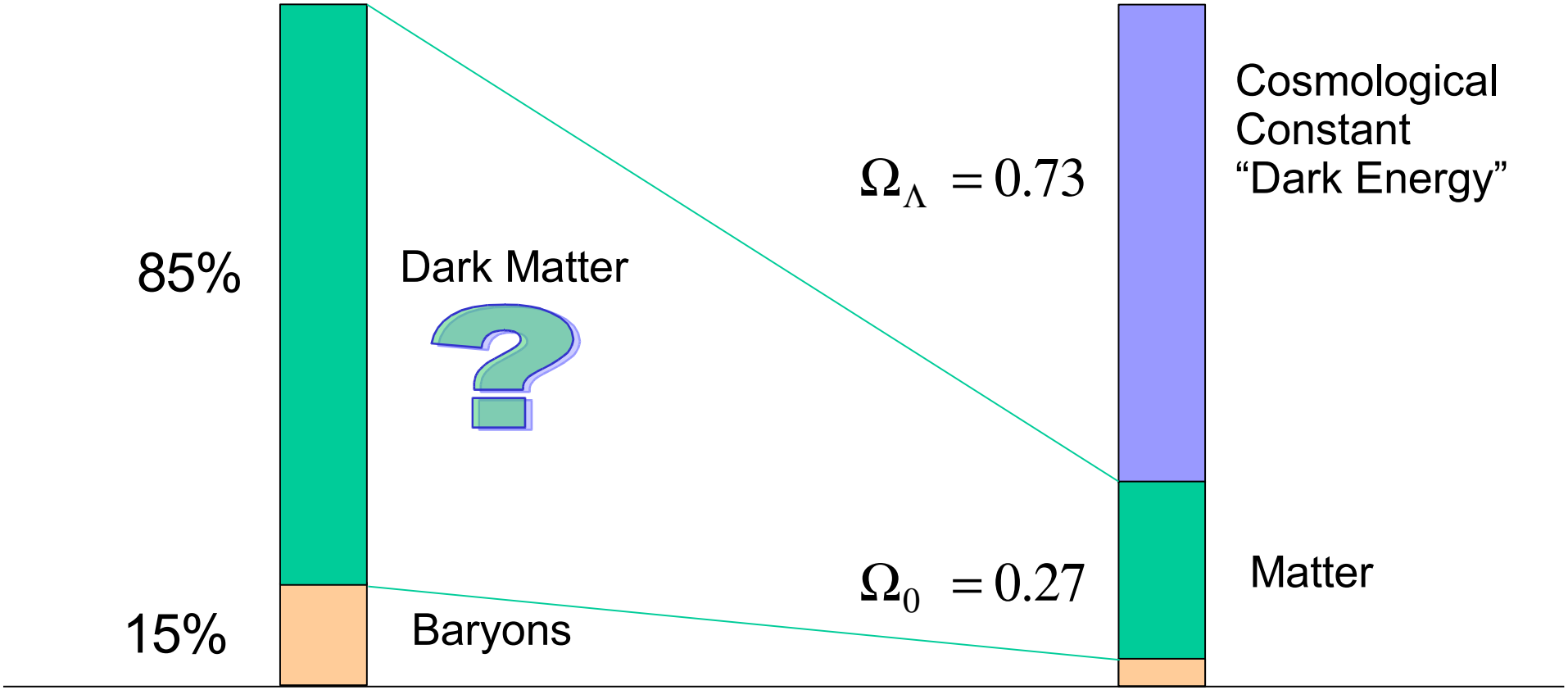
What do we simulate?

Simulations need to account for the full cosmic matter-energy content

MATERIAL IN THE UNIVERSE



$\Omega_{tot} = 1$
flat space-time



We assume that the only appreciable interaction of dark matter particles is **gravity**

COLLISIONLESS DYNAMICS

Because there are **so many** dark matter particles, it's best to describe the system in terms of the **single particle distribution function**

$$f = f(\mathbf{x}, \mathbf{v}, t)$$

There are so many dark matter particles that they do not scatter locally on each other, they just respond to their collective gravitational field

Collisionless Boltzmann equation

Poisson-Vlasov System

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial \mathbf{x}} \cdot \mathbf{v} + \frac{\partial f}{\partial \mathbf{v}} \cdot \left(-\frac{\partial \Phi}{\partial \mathbf{x}} \right) = 0$$

$$\nabla^2 \Phi(\mathbf{x}, t) = 4\pi G \int f(\mathbf{x}, \mathbf{v}, t)\, \mathrm{d}\mathbf{v}$$

Phase-space is conserved along each characteristic (i.e. particle orbit).

The number of stars in galaxies is so large that the two-body relexation time by far exceeds the Hubble time. Stars in galaxies are therefore also described by the above system.

This system of partial differential equations is very difficult (impossible) to solve directly in non-trivial cases.

The N-body method uses a finite set of particles to sample the underlying distribution function

"MONTE-CARLO" APPROACH TO COLLISIONLESS DYNAMICS

We discretize in terms of N particles, which approximately move along characteristics of the underlying system.

$$\ddot{\mathbf{x}}_i = -\nabla_i \Phi(\mathbf{x}_i)$$

$$\Phi(\mathbf{x}) = -G \sum_{j=1}^{N} \frac{m_j}{[(\mathbf{x} - \mathbf{x}_j)^2 + \epsilon^2]}$$

The need for **gravitational softening**:

- Prevent large-angle particle scatterings and the formation of bound particle pairs.
- Ensure that the two-body relexation time is sufficiently large.
- Allows the system to be integrated with low-order intergations schemes.

} Needed for faithful collisionless behaviour

The dynamics of structure formation is driven by gravity

Gravity
general relativity, but Newtonian approximation in expanding space usually sufficient

Hydrodynamics
shock waves
radiation processes
star formation
supernovae,
black holes, etc...

dark matter is collisionless

Monte-Carlo integration as an **N-body system**

3N **coupled**, non-linear differential equations of second order

Problems:

- N is very large
- All equations are coupled to each other

Two conflicting requirements complicate the study of **hierarchical** structure formation

DYNAMIC RANGE PROBLEM FACED BY COSMOLOGICAL SIMULATIONS

Want **small particle mass** to resolve internal structure of halos

Want **large volume** to obtain respresentative sample of universe

need large N

where N is the particle number

Problems due to a small box size:

- Fundamental mode goes non-linear soon after the first halos form. ⇨ Simulation cannot be meaningfully continued beyond this point.
- No rare objects (the first halo, **rich** galaxy clusters, etc.)

Problems due to a large particle mass:

- Physics cannot be resolved.
- Small galaxies are missed.

At any given time, halos exist on a large range of mass-scales !

Several questions come up when we try to use the N-body approach for cosmological simulations

- How do we compute the gravitational forces efficiently and accurately?
- How do we integrate the orbital equations in time?
- How do we generate appropriate initial conditions?

$$\ddot{\mathbf{x}}_i = -\nabla_i \, \Phi(\mathbf{x}_i)$$

$$\Phi(\mathbf{x}) = -G \sum_{j=1}^{N} \frac{m_j}{[(\mathbf{x} - \mathbf{x}_j)^2 + \epsilon^2]}$$

Note: The naïve computation of the forces is an N^2 - task.

The particle mesh (PM) force calculation

The particle-mesh method

Poisson's equation can be solved in real-space by a convolution of the density field with a Green's function.

$$\Phi(\mathbf{x}) = \int g(\mathbf{x} - \mathbf{x}')\, \rho(\mathbf{x})\, \mathrm{d}\mathbf{x}'$$

Example for vacuum boundaries:

$$\Phi(\mathbf{x}) = -G \int \frac{\rho(\mathbf{x})}{|\mathbf{x} - \mathbf{x}'|}\, \mathrm{d}\mathbf{x}' \qquad g(\mathbf{x}) = -\frac{G}{|\mathbf{x}|}$$

In Fourier-space, the convolution becomes a simple multiplication!

$$\hat{\Phi}(\mathbf{k}) = \hat{g}(\mathbf{k}) \cdot \hat{\rho}(\mathbf{k})$$

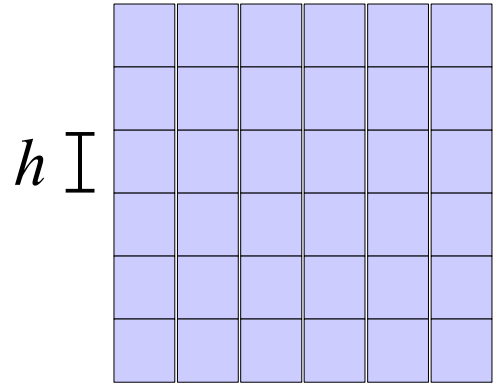
→ **Solve the potential in these steps:**

(1) FFT forward of the density field
(2) Multiplication with the Green's function
(3) FFT backwards to obtain potential

The four steps of the PM algorithm

(a) Density assignment
(b) Computation of the potential
(c) Determination of the force field
(d) Assignment of forces to particles

Density assignment



$\{\mathbf{x}_m\}$ set of discrete mesh centres

Give particles a “shape” S(x). Then to each mesh cell, we assign the fraction of mass that falls into this cell. The overlap for a cell is given by:

$$W(\mathbf{x_m} - \mathbf{x}_i) = \int_{\mathbf{x_m}-\frac{h}{2}}^{\mathbf{x_m}+\frac{h}{2}} S(\mathbf{x}' - \mathbf{x}_i)\,\mathrm{d}\mathbf{x}' = \int \Pi\left(\frac{\mathbf{x}' - \mathbf{x_m}}{h}\right) S(\mathbf{x}' - \mathbf{x}_i)\,\mathrm{d}\mathbf{x}'$$

The assignment function is hence the convolution:

$$W(\mathbf{x}) = \Pi\left(\frac{\mathbf{x}}{h}\right) \star S(\mathbf{x})$$

where

$$\Pi(x) = \begin{cases} 1 & \text{for } |x| \leq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

The density on the mesh is then a sum over the contributions of each particle as given by the assignment function:

$$\rho(\mathbf{x_m}) = \frac{1}{h^3} \sum_{i=1}^{N} m_i\, W(\mathbf{x}_i - \mathbf{x_m})$$

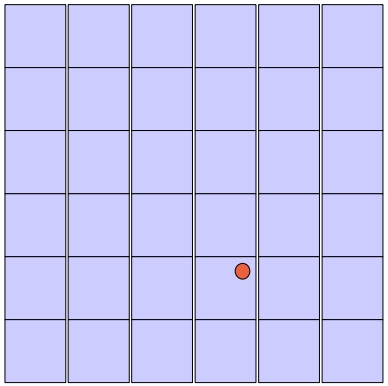
Commenly used particle shape functions and assignment schemes

Name	Shape function S(x)	# of cells involved	Properties of force
NGP Nearest grid point	$\delta(\mathbf{x})$	$1^3 = 1$	piecewise constant in cells
CIC Clouds in cells	$\frac{1}{h^3}\Pi\left(\frac{\mathbf{x}}{h}\right) \star \delta(\mathbf{x})$	$2^3 = 8$	piecewise linear, continuous
TSC Triangular shaped clouds	$\frac{1}{h^3}\Pi\left(\frac{\mathbf{x}}{h}\right) \star \frac{1}{h^3}\Pi\left(\frac{\mathbf{x}}{h}\right)$	$3^3 = 27$	continuous first derivative

Note: For interpolation of the grid to obtain the forces, the same assignment function needs to be used to ensure momentum conservation. (In the CIC case, this is identical to tri-linear interpolation.)

Finite differencing of the potential to get the force field

Approximate the force field $\mathbf{f} = -\nabla\Phi$ with finite differencing



2nd order accurate scheme:

$$f^{(x)}_{i,j,k} = -\frac{\Phi_{i+1,j,k} - \Phi_{i-1,j,k}}{2h}$$

4th order accurate scheme:

$$f^{(x)}_{i,j,k} = -\frac{4}{3}\frac{\Phi_{i+1,j,k} - \Phi_{i-1,j,k}}{2h} + \frac{1}{3}\frac{\Phi_{i+2,j,k} - \Phi_{i-2,j,k}}{4h}$$

Interpolating the mesh-forces to the particle locations

$$F(\mathbf{x}_i) = \sum_{\mathbf{m}} W(\mathbf{x}_i - \mathbf{x}_\mathbf{m})\, f_\mathbf{m}$$

The interpolation kernel needs to be the same one used for mass-assignment to ensure force anti-symmetry.

Advantages and disadvantages of the PM-scheme

Pros: **SPEED** and simplicity

Cons:

- Spatial force resolution limited to mesh size.
- Force errors somewhat anisotropic on the scale of the cell size

→ *serious problem:*

cosmological simulations cluster strongly and have a very large dynamic range

cannot make the PM-mesh fine enough and resolve internal structure of halos as well as large cosmological scales

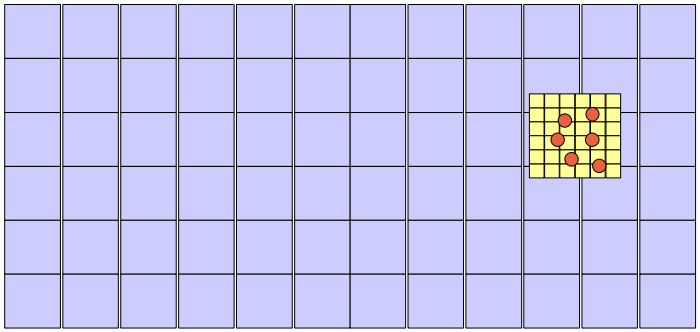
→ we need a method to increase the **dynamic range** available in the force calculation

Particle-Particle PM schemes (P^3M)

Idea: Supplement the PM force with a direct summation short-range force at the scale of the mesh cells. The particles in cells are linked together by a chaining list.

Offers much higher dynamic range, but becomes slow when clustering sets in.

In AP^3M, mesh-refinements are placed on clustered regions



Can avoid clustering slow-down, but has higher complexity and ambiguities in mesh placement

Codes that use AP^3M: **HYDRA** (Couchman)

Iterative Poisson solvers determine the potential directly on a (hierarchical grid)

Idea: Start with a trial potential and then iteratively relax the solution by updating with a finite difference approximation to the Laplacian.

$$\Phi'_{i,j,k} = \frac{1}{6}\left(\Phi_{i+1,j,k} + \Phi_{i-1,j,k} + \Phi_{i,j+1,k} + \Phi_{i,j-1,k} + \Phi_{i,j,k+1} + \Phi_{i,j,k-1} - 4\pi G h^2 \rho_{i,j,k}\right)$$

This updating eliminates errors on the scale of a few grid cells rapidly, but longer-range fluctuations die out much more slowly.

In **multigrid methods**, a hierarchy of meshes is used to speed up convergence, resulting in a fast method that allows for locally varying resolution.

Codes that use a real-space Poisson solver: **ART** (Kravtsov)
MLAPM (Knebe)

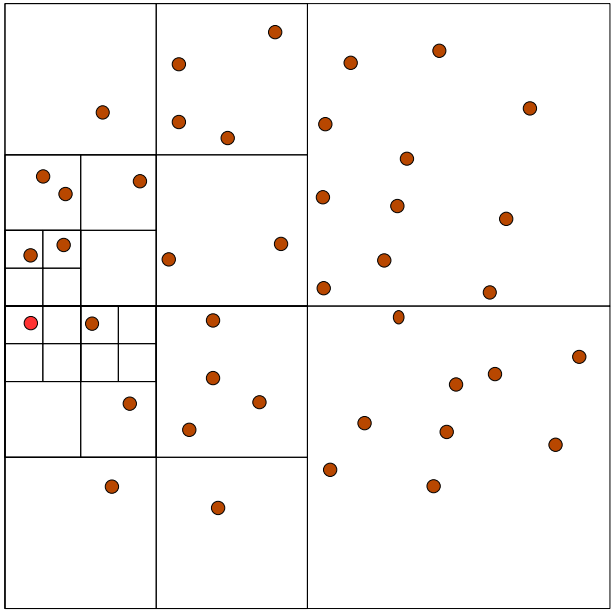
TREE algorithms

Gravity is the driving force for structure formation in the universe

HIERARCHICAL TREE ALGORITHMS

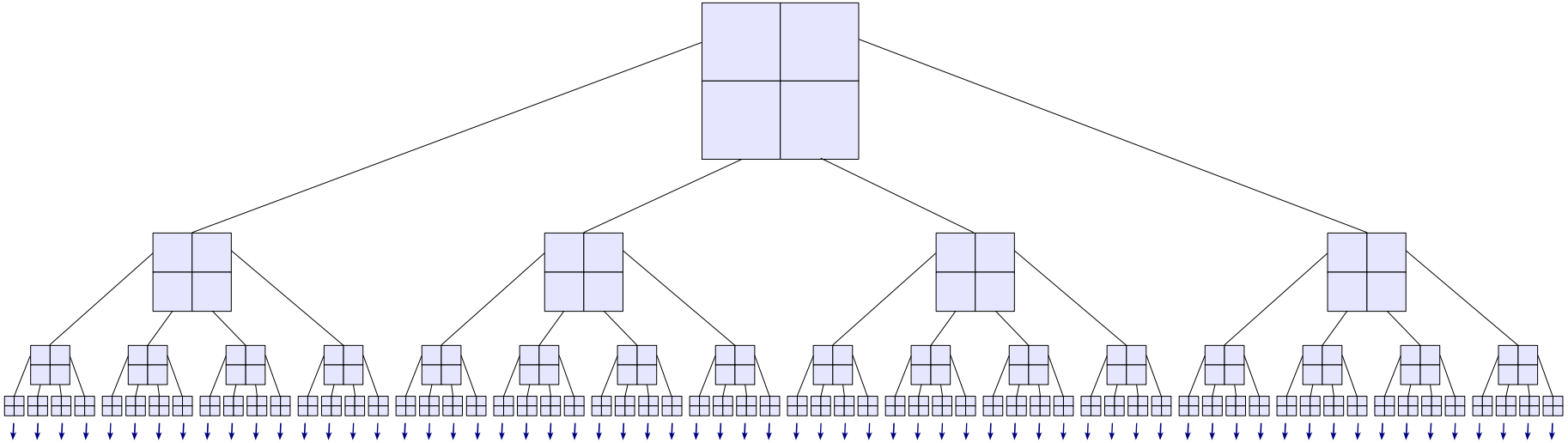
The N^2 - scaling of direct summation puts serious limitations on N...

But we want N ~ 10^6-10^{10} for collisionless dynamics of dark matter !

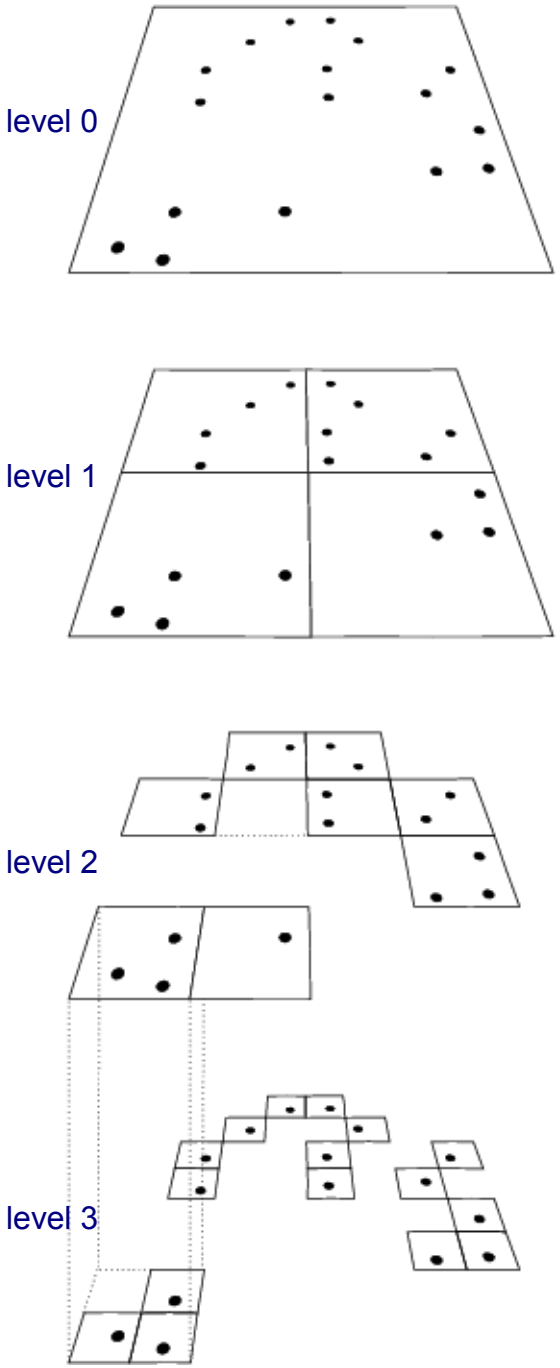


Idea: Group distant particles together, and use their multipole expansion.

→ Only ~ log(N) force terms per particle.

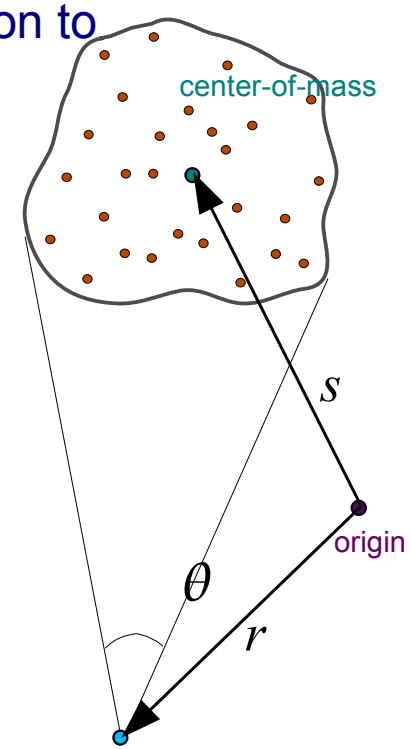


Oct-tree in two dimensions



Tree algorithms

Idea: Use hierarchical multipole expansion to account for distant particle groups



$$\Phi(\mathbf{r}) = -G \sum_i \frac{m_i}{|\mathbf{r} - \mathbf{x}_i|}$$

We expand:

$$\frac{1}{|\mathbf{r} - \mathbf{x}_i|} = \frac{1}{|(\mathbf{r} - \mathbf{s}) - (\mathbf{x}_i - \mathbf{s})|}$$

for $\quad |\mathbf{x}_i - \mathbf{s}| \ll |\mathbf{r} - \mathbf{s}| \qquad \mathbf{y} \equiv \mathbf{r} - \mathbf{s}$

and obtain:

$$\frac{1}{|\mathbf{y} + \mathbf{s} - \mathbf{x}_i|} = \frac{1}{|\mathbf{y}|} - \frac{\mathbf{y} \cdot (\mathbf{s} - \mathbf{x}_i)}{|\mathbf{y}|^3} + \frac{1}{2} \frac{\mathbf{y}^T \left[3(\mathbf{s} - \mathbf{x}_i)(\mathbf{s} - \mathbf{x}_i)^T - \mathbf{I}(\mathbf{s} - \mathbf{x}_i)^2 \right] \mathbf{y}}{|\mathbf{y}|^5} + \ldots$$

the dipole term vanishes when summed over all particles in the group

The multipole moments are computed for each node of the tree

Monpole moment:

$$M = \sum_i m_i$$

Quadrupole tensor:

$$Q_{ij} = \sum_k m_k \left[3(\mathbf{x}_k - \mathbf{s})_i (\mathbf{x}_k - \mathbf{s})_j - \delta_{ij} (\mathbf{x}_k - \mathbf{s})^2 \right]$$

Resulting potential/force approximation:

$$\Phi(\mathbf{r}) = -G \left[\frac{M}{|\mathbf{y}|} + \frac{1}{2} \frac{\mathbf{y}^T \mathbf{Q}\, \mathbf{y}}{|\mathbf{y}|^5} \right]$$

For a single force evaluation, not *N* single-particle forces need to be computed, but **only of order *log(N)* multipoles**, depending on opening angle.

- The tree algorithm has no intrinsic restrictions for its dynamic range
- force accuracy can be conveniently adjusted to desired level
- the speed does depend only very weakly on clustering state
- geometrically flexible, allowing arbitrary geometries

TreePM force calculation algorithm

Particularly at high redshift, it is expensive to obtain accurate forces with the tree-algorithm

THE TREE-PM FORCE SPLIT

Periodic peculiar potential

$$\nabla^2\phi(\mathbf{x}) = 4\pi G[\rho(\mathbf{x}) - \overline{\rho}] = 4\pi G \sum_{\mathbf{n}} \sum_i m_i \left[\tilde{\delta}(\mathbf{x} - \mathbf{x}_i - \mathbf{n}L) - \frac{1}{L^3}\right]$$

Idea: Split the potential (of a single particle) in Fourier space into a long-range and a short-range part, and compute them separately with PM and TREE algorithms, respectively.

Poisson equation in Fourier space:

$$\phi_{\mathbf{k}} = -\frac{4\pi G}{\mathbf{k}^2}\rho_{\mathbf{k}} \qquad (\mathbf{k} \neq 0)$$

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

Solve with PM-method

- CIC mass assignment
- FFT
- multiply with kernel
- FFT backwards
- Compute force with 4-point finite difference operator
- Interpolate forces to particle positions

$$\phi_{\mathbf{k}}^{\mathrm{short}} = \phi_{\mathbf{k}} \left[1 - \exp(-\mathbf{k}^2 r_s^2)\right]$$

FFT to real space

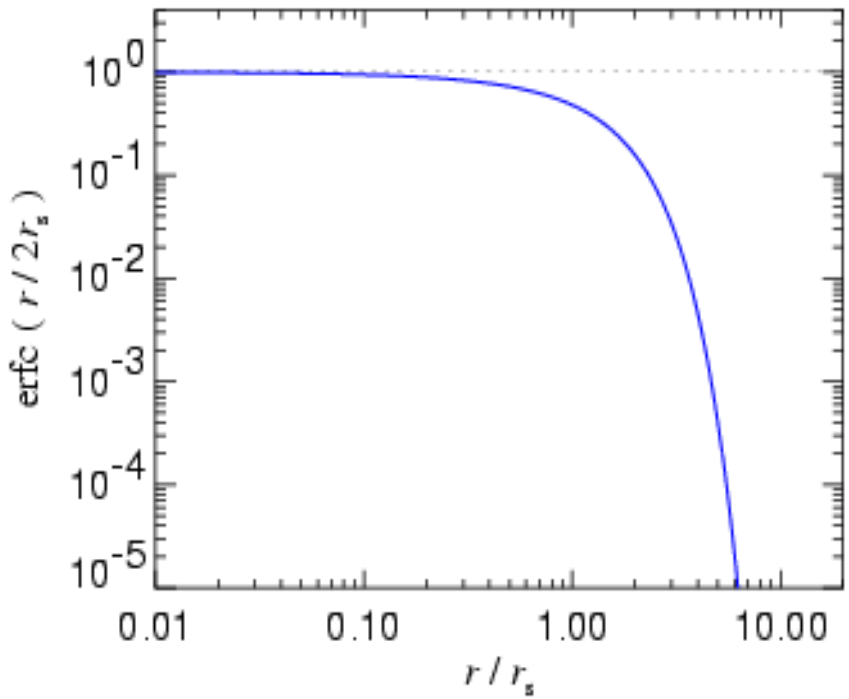
$$\phi(r) = -\frac{Gm}{r}\,\mathrm{erfc}\left(\frac{r}{2r_s}\right)$$

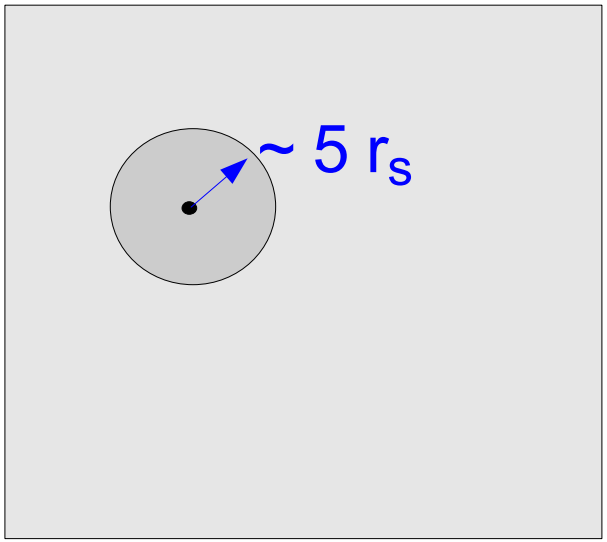
Solve in real space with TREE

In the TreePM algorithm, the tree has to be walked locally only

PERFORMANCE GAIN DUE TO LOCAL TREE WALK

$$\phi(r) = -\frac{Gm}{r}\,\mathrm{erfc}\left(\frac{r}{2r_s}\right)$$





Advantages of TreePM include:

- Accurate and fast long-range force
- No force anisotropy
- Speed is largely insensitive to clustering (as for tree algorithm)
- No Ewald correction necessary for periodic boundary conditions

Using zero-padding and a different Greens-Function, the long-range force can also be computed for vaccuum boundaries using the FFT.
(Implemented in Gadget-2)

The maximum size of a TreePM simulation with *Lean*-GADGET-II is essentially memory bound

A HIGHLY MEMORY EFFICIENT VERSION OF GADGET-II

Particle Data	Tree storage	FFT workspace
44 bytes / particle	40 bytes / particle	24 bytes / mesh-cell

Not needed concurently!

Special code version ***Lean*-GADGET-II** needs: **84 bytes / particle** (Assuming 1.5 mesh-cells/particle)

Simulation Set-up:

- Particle number: 2160^3 = 10.077.696.000 = ~ 10^{10} particles
- Boxsize: L = 500 h^{-1} Mpc
- Particle mass: $m_p = 8.6 \times 10^8\ h^{-1}\ M_\odot$
- Spatial resolution: 5 h^{-1} kpc
- Size of FFT: 2560^3 = 16.777.216.000 = ~ 17 billion cells

Minimum memory requirement of simulation code

~840 GByte

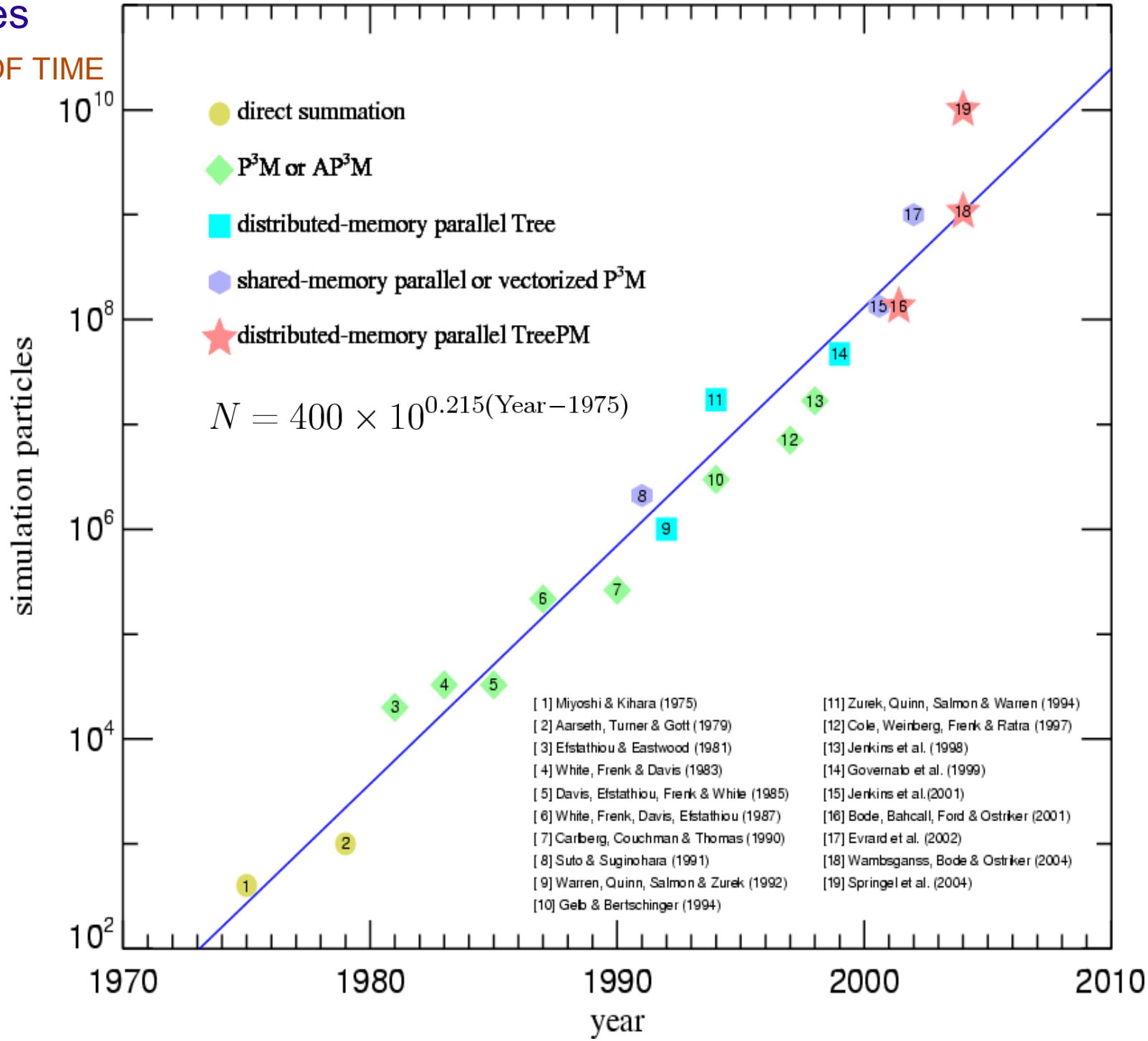
Compared to Hubble-Volume simulation:

- > 2000 times better mass resolution
- 10 times larger particle number
- 13 better spatial resolution

Cosmological N-body simulations have grown rapidly in size over the last three decades

"**N**" AS A FUNCTION OF TIME

- Computers double their speed every 18 months (Moore's law)
- N-body simulations have doubled their size every 16-17 months
- Recently, growth has accelerated further.
  The Millennium Run should have become possible in 2010 – we have done it in 2004 !



The simulation was run on the *Regatta* supercomputer of the RZG

REQUIRED RESSOURCES

1 TByte RAM needed

16 x 32-way Regatta Node
64 GByte RAM
512 CPU total

CPU time consumed

350.000 processor hours

- 28 days on 512 CPUs/16 nodes
- 38 years in serial
- ~ 6% of annual time on total Regatta system
- sustained average code performance (hardware counters) 400 Mflops/cpu
- 5 x 10^{17} floating point ops
- 11000 (adaptive) timesteps

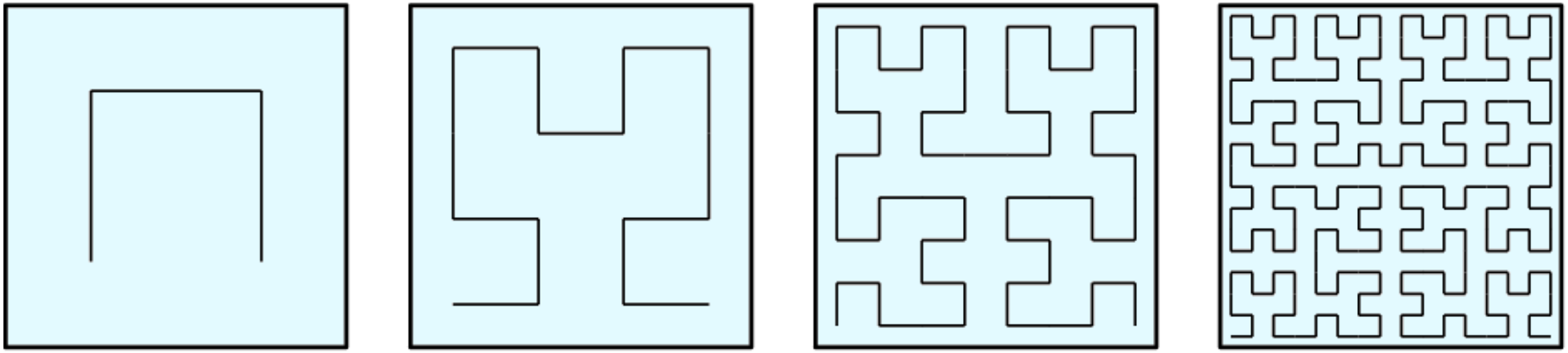
Organization of tree and domain decomposition

The tree-algorithm of Gadget-2 has been optimized for providing better memory locality

REDUCTION OF CACHE MISSES AND DOMAIN DECOMPOSITION

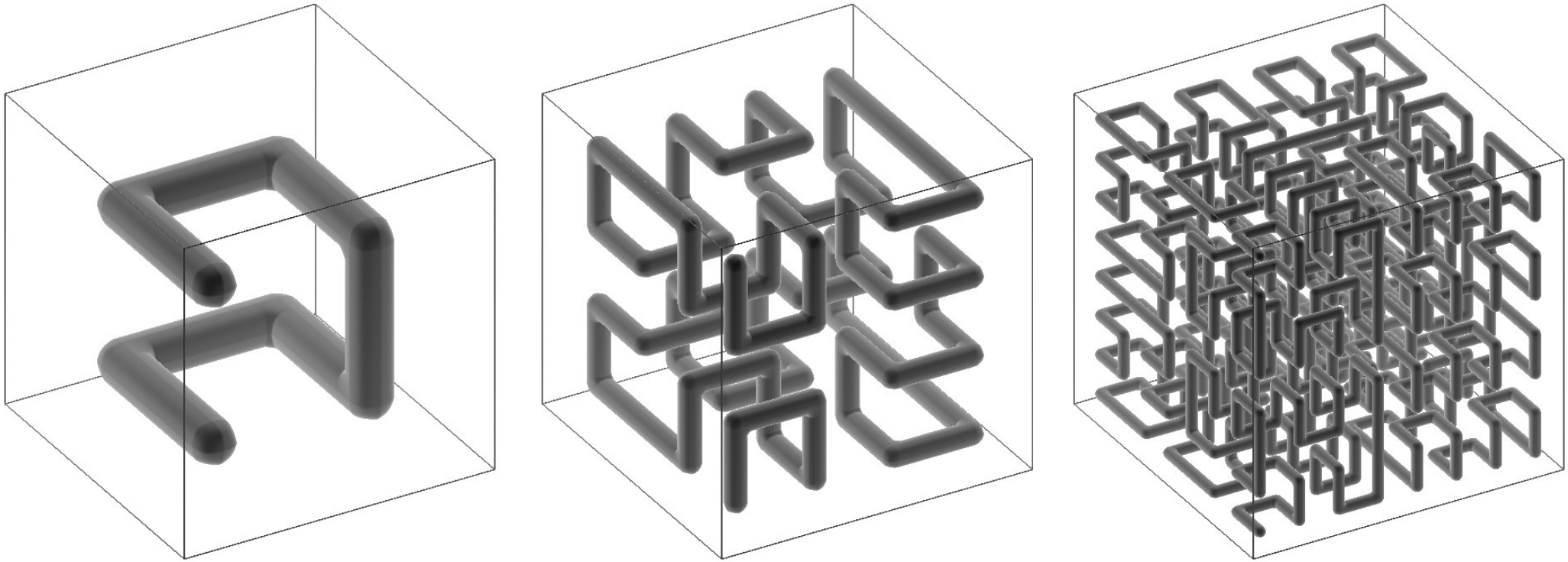
Idea: Order the particles along a space-filling curve

Hilbert's curve: A fractal that fills the square



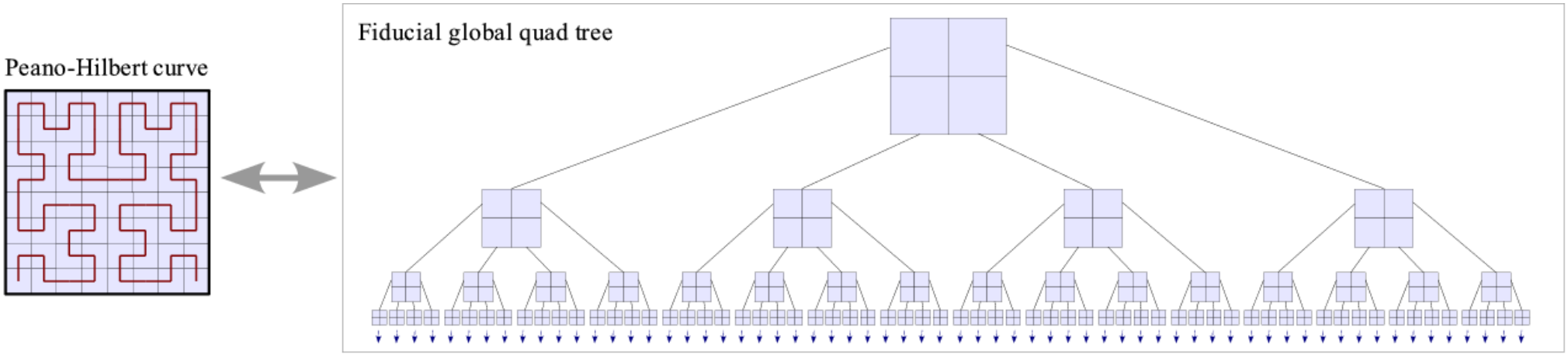
The space-filling Hilbert curve can be readily generalized to 3D

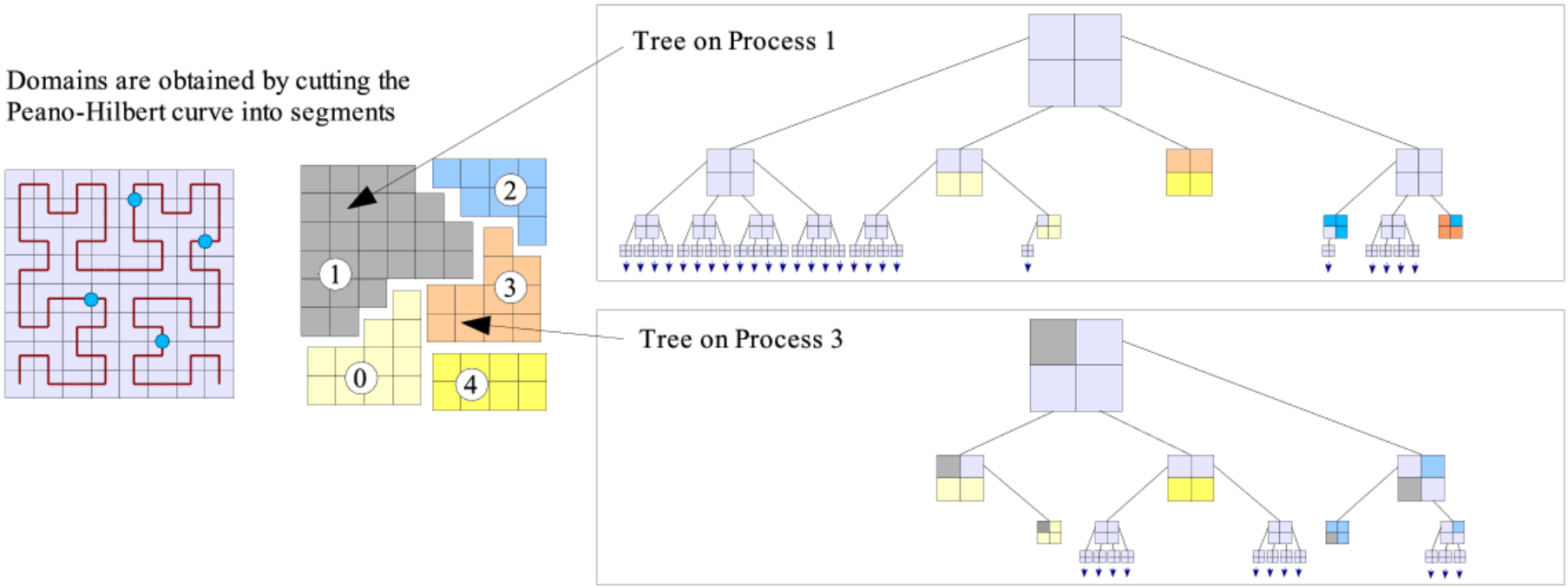
THE PEANO-HILBERT CURVE



A space-filling Peano-Hilbert curve is used in GADGET-2 for a novel domain-decomposition concept

HIERARCHICAL TREE ALGORITHMS





Overview of code options

GADGET2 is controlled both by compile-time options, and a parameterfile

OVERVIEW OF USAGE OF THE CODE

Requirements for compilation

- C-compiler
- make-utility (GNU-make)
- MPI-1.1 library
- GSL (GNU scientific library)
- FFTW ('Fastest Fourier Transform in the West')
- HDF5 library (optional)

Simulation settings and code parameters

- Makefile
- Parameterfile

Start of a simulation

- Start from initial conditions:
  `mpirun -np 32 ./Gadget2 param.txt`
- Continuation of run from a set of restart files
  `mpirun -np 32 ./Gadget2 param.txt 1`
- Start from a Gadget snapshot file
  `mpirun -np 32 ./Gadget2 param.txt 2`

P-GADGET2 is controlled both by compile-time options, and a parameterfile

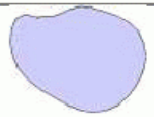
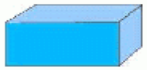
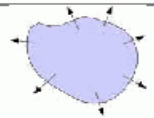
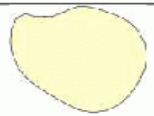
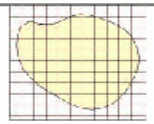
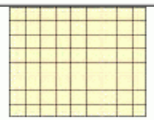
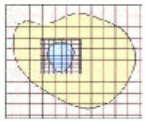
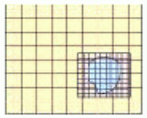
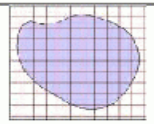
OVERVIEW OF CODE OPTIONS

There are **192** Makefile options by now...

```
#########################################################################
#  Look at end of file for a brief guide to the compile-time options. #
#########################################################################

#------------------------------------- Basic operation mode of code
OPT   +=  -DPERIODIC
OPT   +=  -DCOOLING
OPT   +=  -DSFR
#OPT   +=  -DUNEQUALSOFTENINGS


#------------------------------------- TreePM Options
OPT   +=  -DPMGRID=384
#OPT   +=  -DASMTH=1.25
#OPT   +=  -DRCUT=4.5
#OPT   +=  -DPLACEHIGHRESREGION=3
#OPT   +=  -DENLARGEREGION=1.2
#OPT   +=  -DONLY_PM
#OPT   +=  -DHPM
#OPT   +=  -DHPM_SMTH=1.5

#------------------------------------- Single/Double Precision
#OPT   +=  -DDOUBLEPRECISION
#OPT   +=  -DDOUBLEPRECISION_FFTW
#OPT   +=  -DFLTROUNDOFFREDUCTION    # enables round off reduction in particle sums
                                     # if DOUBLEPRECISION is set, these sums are done in 'long double'
                              # if single precision is used, they are done in 'double'
                              # This should in principle allow to make computations
                              # *exactly* invariant to different numbers of CPUs.

#OPT   +=  -DSOFTDOUBLEDOUBLE       # when this is set, a software implementation of
                              # 128bit double-double addition is used, implemented as a c++ class.
                              # Hence, this option requires compilation with a c++ compiler

#------------------------------------- SFR/feedback model

#OPT   +=  -DSOFTEREQS
OPT   +=  -DMOREPARAMS
```

GADGET2 supports different types of simulation set-ups

OVERVIEW OF TYPES OF SIMULATIONS POSSIBLE WITH GADGET

	Type of Simulation		Computational methods	Remarks
1	**Newtonian space**		Gravity: Tree, SPH (optional), vacuum boundary conditions	OmegaLambda should be set to zero
2	**Periodic long box**		No gravity, only SPH, periodic boundary conditions	NOGRAVITY needs to be set, LONG_X/Y/Z may be set to scale the dimensions of the box
3	**Cosmological, physical coordinates**		Gravity: Tree, SPH, vacuum boundaries	ComovingIntegrationOn set to zero
4	**Cosmological, comoving coordinates**		Gravity: Tree, SPH, vacuum boundaries	ComovingIntegrationOn set to one
5	**Cosmological, comoving periodic box**		Gravity: Tree with Ewald-correction, SPH, periodic boundaries	PERIODIC needs to be set
6	**Cosmological, comoving coordinates, TreePM**		Gravity: Tree with long range PM, SPH, vacuum boundaries	PMGRID needs to be set
7	**Cosmological, comoving periodic box, TreePM**		Gravity: Tree with long range PM, SPH, periodic boundaries	PERIODIC and PMGRID need to be set
8	**Cosmological, comoving coordinates, TreePM, Zoom**		Gravity: Tree with long-range and intermediate-range PM, SPH, vacuum boundaries	PMGRID and PLACEHIGHRESREGION need to be set
9	**Cosmological, periodic comoving box, TreePM, Zoom**		Gravity: Tree with long-range and intermediate-range PM, SPH, periodic boundaries	PERIODIC, PMGRID and PLACEHIGHRESREGION need to be set
10	**Newtonian space, TreePM**		Gravity: Tree with long-range PM, SPH, vacuum boundaries	PMGRID needs to be set

GADGET2 is controlled with a free-format ASCII parameterfile

EXAMPLE OF A PARAMETERFILE

Printed by Volker Springel


```
%  Relevant files

InitCondFile         ICs/galaxy_littleendian.dat
OutputDir            galaxy/

EnergyFile           energy.txt
InfoFile             info.txt
TimingsFile          timings.txt
CpuFile              cpu.txt

RestartFile          restart
SnapshotFileBase     snapshot

OutputListFilename   parameterfiles/output_list.txt


% CPU time -limit

TimeLimitCPU         36000  % = 10 hours
ResubmitOn           0
ResubmitCommand      my-scriptfile


% Code options

ICFormat                 1
SnapFormat               1
ComovingIntegrationOn    0

TypeOfTimestepCriterion  0
OutputListOn             0
PeriodicBoundariesOn     0


%  Caracteristics of run

TimeBegin            0.0        % Begin of the simulation
TimeMax              3.0        % End of the simulation

Omega0                 0
OmegaLambda            0
OmegaBaryon            0
HubbleParam          1.0
BoxSize                0


% Output frequency

TimeBetSnapshot            0.5
TimeOfFirstSnapshot        0

CpuTimeBetRestartFile      36000.0     % in seconds
TimeBetStatistics          0.05

NumFilesPerSnapshot        1
NumFilesWrittenInParallel  1


% Accuracy of time integration

ErrTolIntAccuracy      0.025

CourantFac             0.15

MaxSizeTimestep        0.01
MinSizeTimestep        0.0
```



```
% Tree algorithm, force accuracy, domain update frequency

ErrTolTheta            0.5
TypeOfOpeningCriterion 1
ErrTolForceAcc         0.005

TreeDomainUpdateFrequency    0.1
MaxRMSDisplacementFac        0.2


%  Further parameters of SPH

DesNumNgb              50
MaxNumNgbDeviation     2
ArtBulkViscConst       0.8
InitGasTemp            0           % always ignored if set to 0
MinGasTemp             0


% Memory allocation

PartAllocFactor       1.5
TreeAllocFactor       0.8
BufferSize            25           % in MByte


% System of units

UnitLength_in_cm         3.085678e21        ;  1.0 kpc
UnitMass_in_g            1.989e43           ;  1.0e10 solar masses
UnitVelocity_in_cm_per_s 1e5                ;  1 km/sec
GravityConstantInternal  0


% Softening lengths

MinGasHsmlFractional 0.25

SofteningGas               0
SofteningHalo              1.0
SofteningDisk              0.4
SofteningBulge             0
SofteningStars             0
SofteningBndry             0

SofteningGasMaxPhys        0
SofteningHaloMaxPhys       1.0
SofteningDiskMaxPhys       0.4
SofteningBulgeMaxPhys      0
SofteningStarsMaxPhys      0
SofteningBndryMaxPhys      0
```

GADGET2 is controlled with a free-format ASCII parameterfile

DETAILED LIST OF PARAMETERS - I

```
%  Relevant files

InitCondFile         ICs/galaxy_littleendian.dat
OutputDir            galaxy/

EnergyFile           energy.txt
InfoFile             info.txt
TimingsFile          timings.txt
CpuFile              cpu.txt

RestartFile          restart
SnapshotFileBase     snapshot

OutputListFilename   parameterfiles/output_list.txt
```

GADGET2 is controlled with a free-format ASCII parameterfile

DETAILED LIST OF PARAMETERS - II

```
% CPU time -limit

TimeLimitCPU          36000  % = 10 hours
ResubmitOn            0
ResubmitCommand       my-scriptfile


% Code options

ICFormat                  1
SnapFormat                1
ComovingIntegrationOn     0

TypeOfTimestepCriterion   0
OutputListOn              0
PeriodicBoundariesOn      0
```

GADGET2 is controlled with a free-format ASCII parameterfile

DETAILED LIST OF PARAMETERS - III

```
%  Caracteristics of run

TimeBegin           0.0        % Begin of the simulation
TimeMax             3.0        % End of the simulation

Omega0                0
OmegaLambda           0
OmegaBaryon           0
HubbleParam         1.0
BoxSize               0
```

GADGET2 is controlled with a free-format ASCII parameterfile

DETAILED LIST OF PARAMETERS - IV

```
% Output frequency

TimeBetSnapshot            0.5
TimeOfFirstSnapshot        0

CpuTimeBetRestartFile      36000.0    % in seconds
TimeBetStatistics          0.05

NumFilesPerSnapshot        1
NumFilesWrittenInParallel  1
```

GADGET2 is controlled with a free-format ASCII parameterfile

DETAILED LIST OF PARAMETERS - V

```
% Accuracy of time integration

ErrTolIntAccuracy       0.025

CourantFac              0.15

MaxSizeTimestep         0.01
MinSizeTimestep         0.0
```

GADGET2 is controlled with a free-format ASCII parameterfile

DETAILED LIST OF PARAMETERS - VI

```
% Tree algorithm, force accuracy, domain update frequency

ErrTolTheta             0.5
TypeOfOpeningCriterion 1
ErrTolForceAcc          0.005

TreeDomainUpdateFrequency    0.1
MaxRMSDisplacementFac        0.2


%  Further parameters of SPH

DesNumNgb               50
MaxNumNgbDeviation      2
ArtBulkViscConst        0.8
InitGasTemp             0           % always ignored if set to 0
MinGasTemp              0
```

GADGET2 is controlled with a free-format ASCII parameterfile

DETAILED LIST OF PARAMETERS - VII

```
% Memory allocation

PartAllocFactor       1.5
TreeAllocFactor       0.8
BufferSize            25          % in MByte


% System of units

UnitLength_in_cm         3.085678e21        ;  1.0 kpc
UnitMass_in_g            1.989e43           ;  1.0e10 solar masses
UnitVelocity_in_cm_per_s 1e5                ;  1 km/sec
GravityConstantInternal  0
```

GADGET2 is controlled with a free-format ASCII parameterfile

DETAILED LIST OF PARAMETERS - VIII

```
% Softening lengths

MinGasHsmlFractional 0.25

SofteningGas                0
SofteningHalo               1.0
SofteningDisk               0.4
SofteningBulge              0
SofteningStars              0
SofteningBndry              0

SofteningGasMaxPhys         0
SofteningHaloMaxPhys        1.0
SofteningDiskMaxPhys        0.4
SofteningBulgeMaxPhys       0
SofteningStarsMaxPhys       0
SofteningBndryMaxPhys       0
```

GADGET2's snapshot file format is a simple binary file with a block structure

BLOCKS IN GADGET2 SNAPSHOTS

#	Block ID	HDF5 identifier	Block content
1	HEAD		Header
2	POS	Coordinates	Positions
3	VEL	Velocities	Velocities
4	ID	ParticleIDs	Particle ID's
5	MASS	Masses	Masses (only for particle types with variable masses)
6	U	InternalEnergy	Internal energy per unit mass (only SPH particles)
7	RHO	Density	Density (only SPH particles)
8	HSML	SmoothingLength	SPH smoothing length h (only SPH particles)
9	POT	Potential	Gravitational potential of particles (only when enabled in makefile)
10	ACCE	Acceleration	Acceleration of particles (only when enabled in makefile)
11	ENDT	RateOfChangeOfEntropy	Rate of change of entropic function of SPH particles (only when enabled in makefile)
12	TSTP	TimeStep	Timestep of particles (only when enabled in makefile)

only used for I/O-format 2

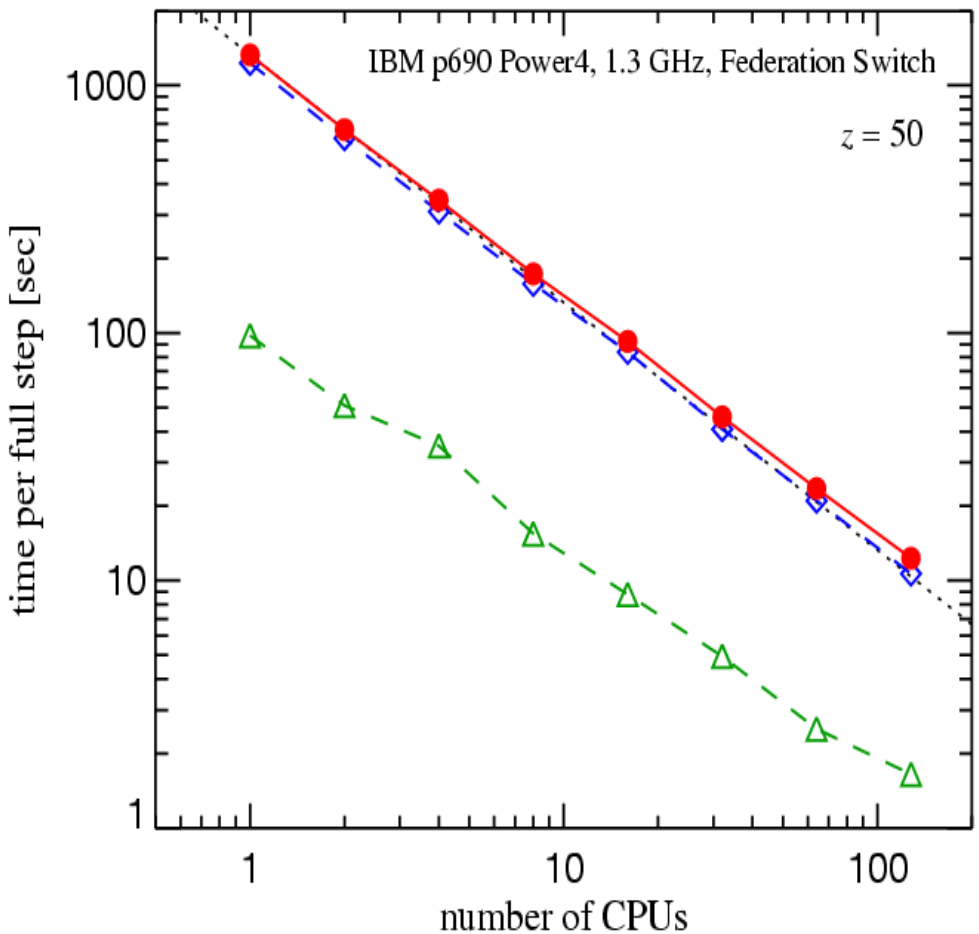
only used for I/O-format 3

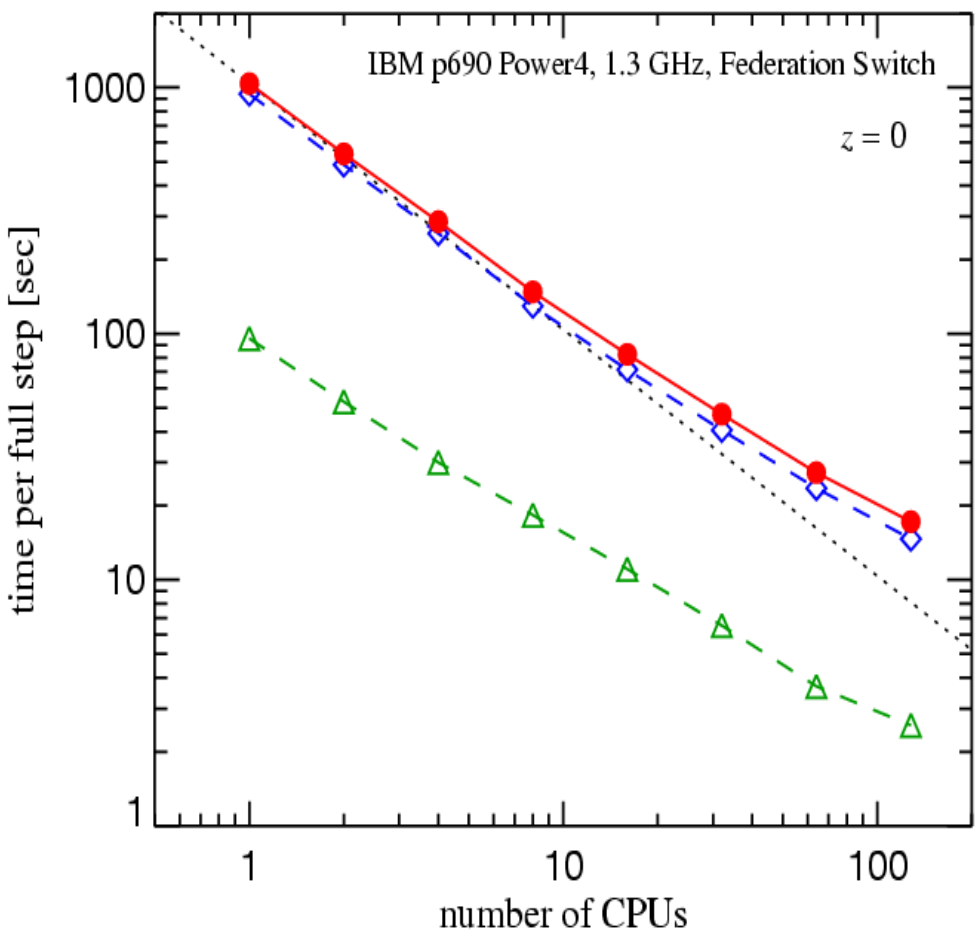
Scalability and its limitations

For fixed timesteps and large cosmological boxes, the scalability of the code is very good

RESULTS FOR A "STRONG SCALING" TEST (FIXED PROBLEM SIZE)

256^3 particles in a 50 h^{-1} Mpc box

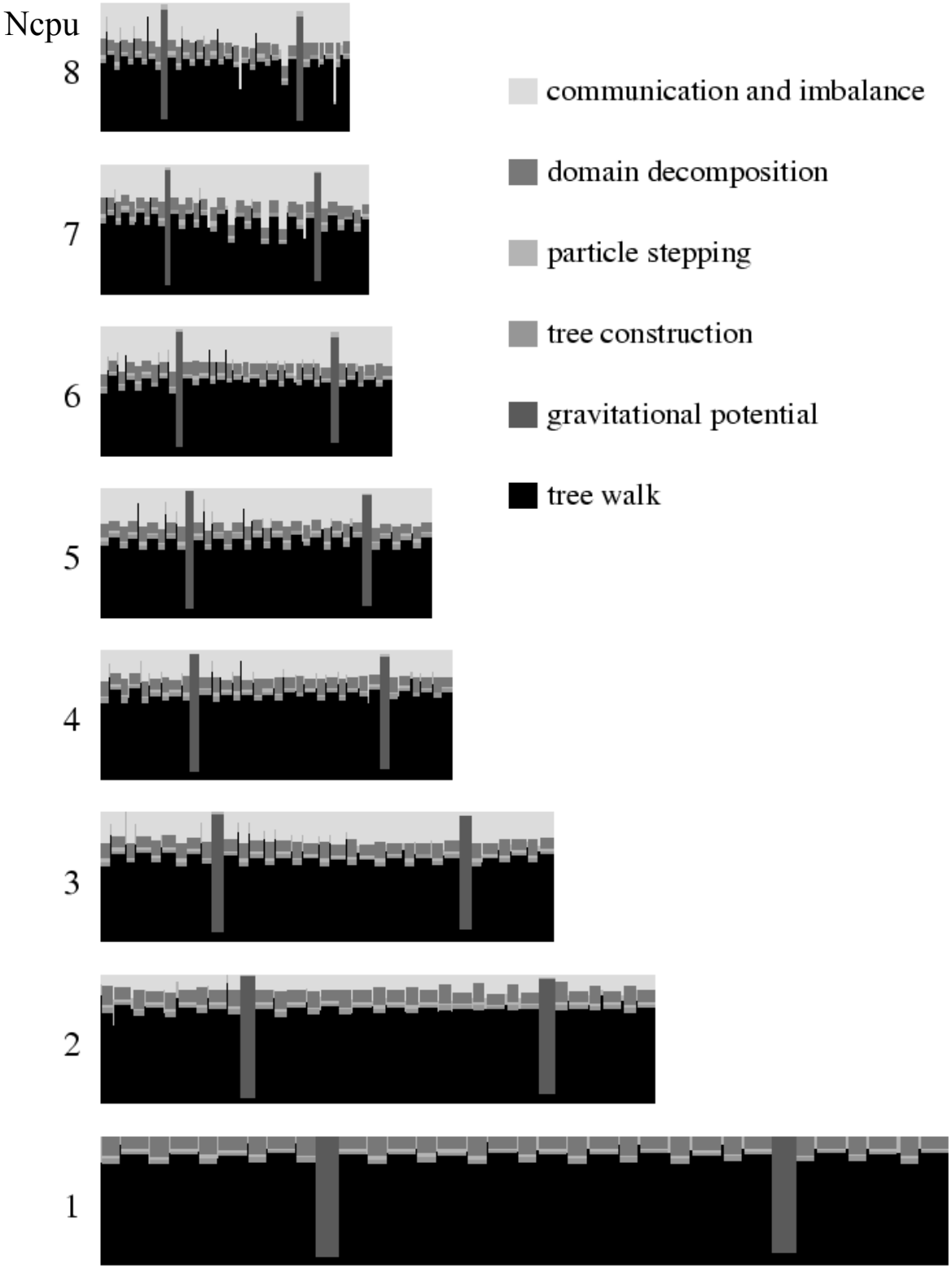




For small problem sizes or isolated galaxies, the scalability is limited

RESULTS FOR "STRONG SCALING" OF A GALAXY COLLISION SIMULATION

CPU consumption in different code parts as a function of processor number



In a parallel code, numerous sources of performance losses can limit scalability to large processor numbers

TROUBLING ASPECTS OF PARALLELIZATION

- **Incomplete parallelization**
  The residual serial part in an application limits the theoretical speed-up one can achieve with an arbitrarily large number of CPUs ('Ahmdahl's Law'), e.g. 5% serial code left, then parallel speed-up is at most a factor 20.

- **Parallelization overhead**
  The bookkeeping code necessary for non-trivial communication algorithms increases the total cost compared to a serial algorithm. Sometimes this extra cost increases with the number of processors used.

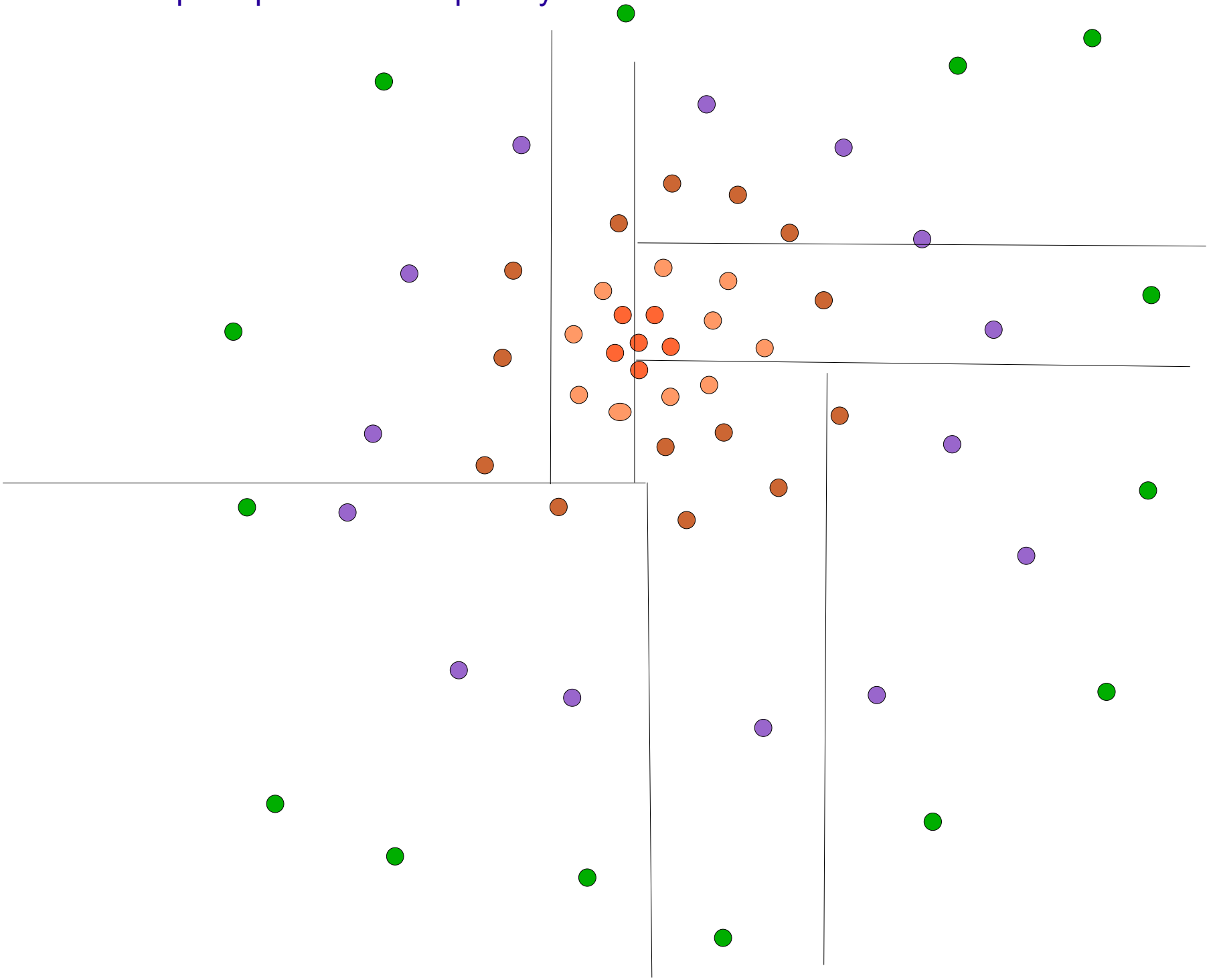
- **Communication times**
  The time spent in waiting for messages to be transmitted across the network (bandwith) and the time required for starting a communication request (latency).

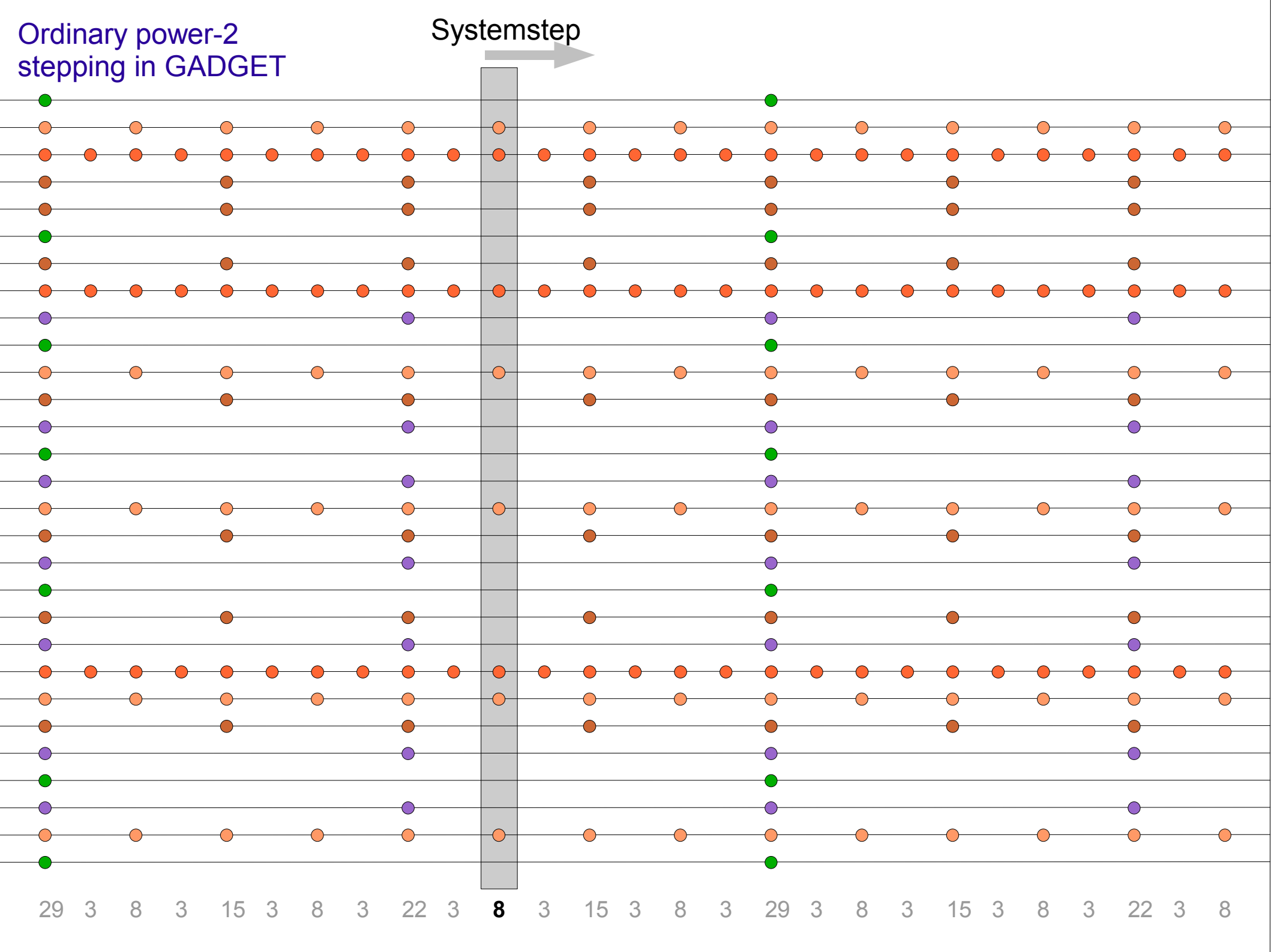
- **Wait times**
  Work-load imbalances will force the fastest CPU to idly wait for the slowest one.

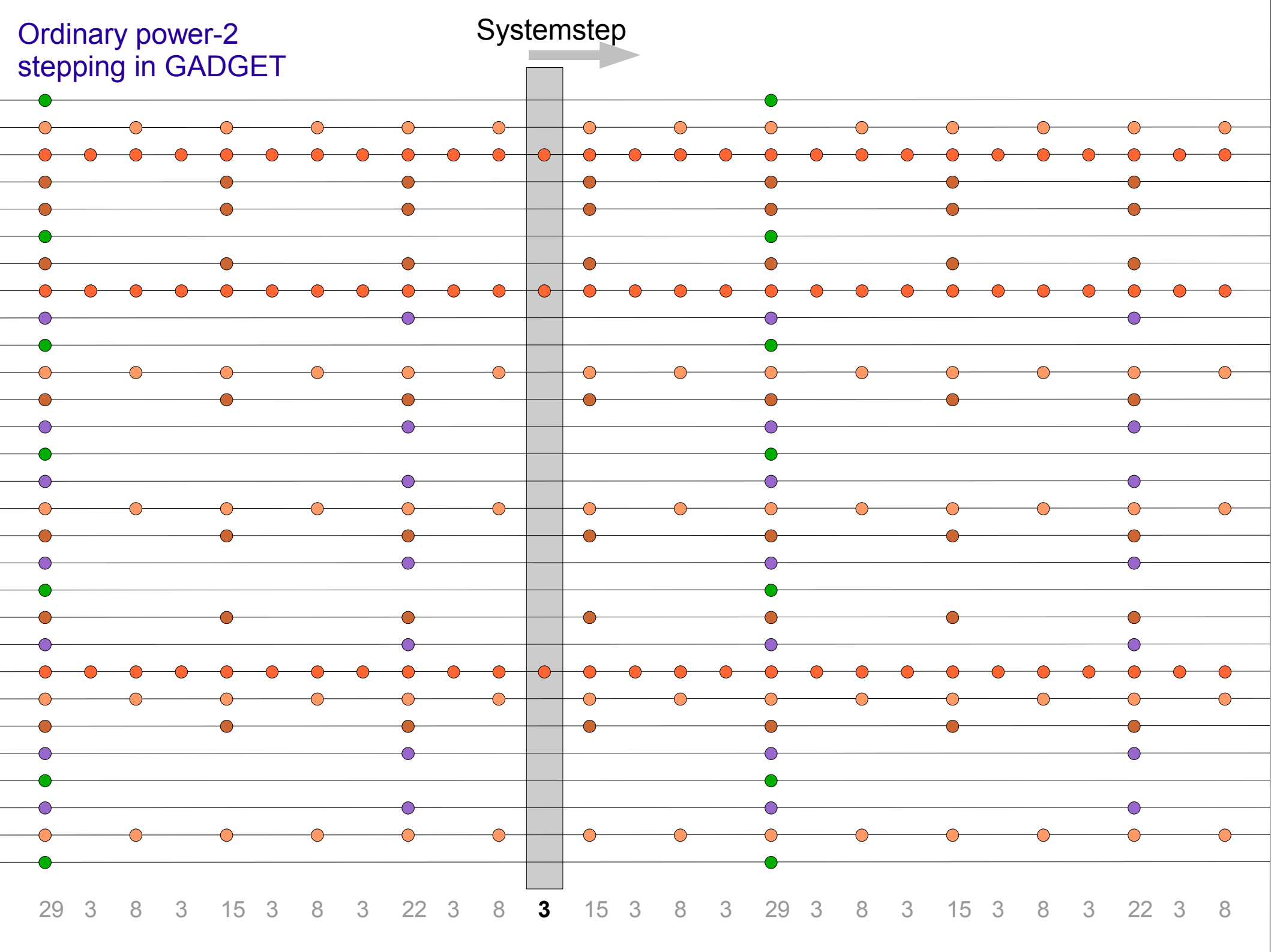
Strong scaling: Keep problem size fixed, but increase number of CPUs
Weak scaling: When number of CPUs is increased, also increase the problem size
As a rule, scalability can be more easily retained in the weak scaling regime.

→ **In practice, it usually doesn't make sense to use a large number of processors for a (too) small problem size !**

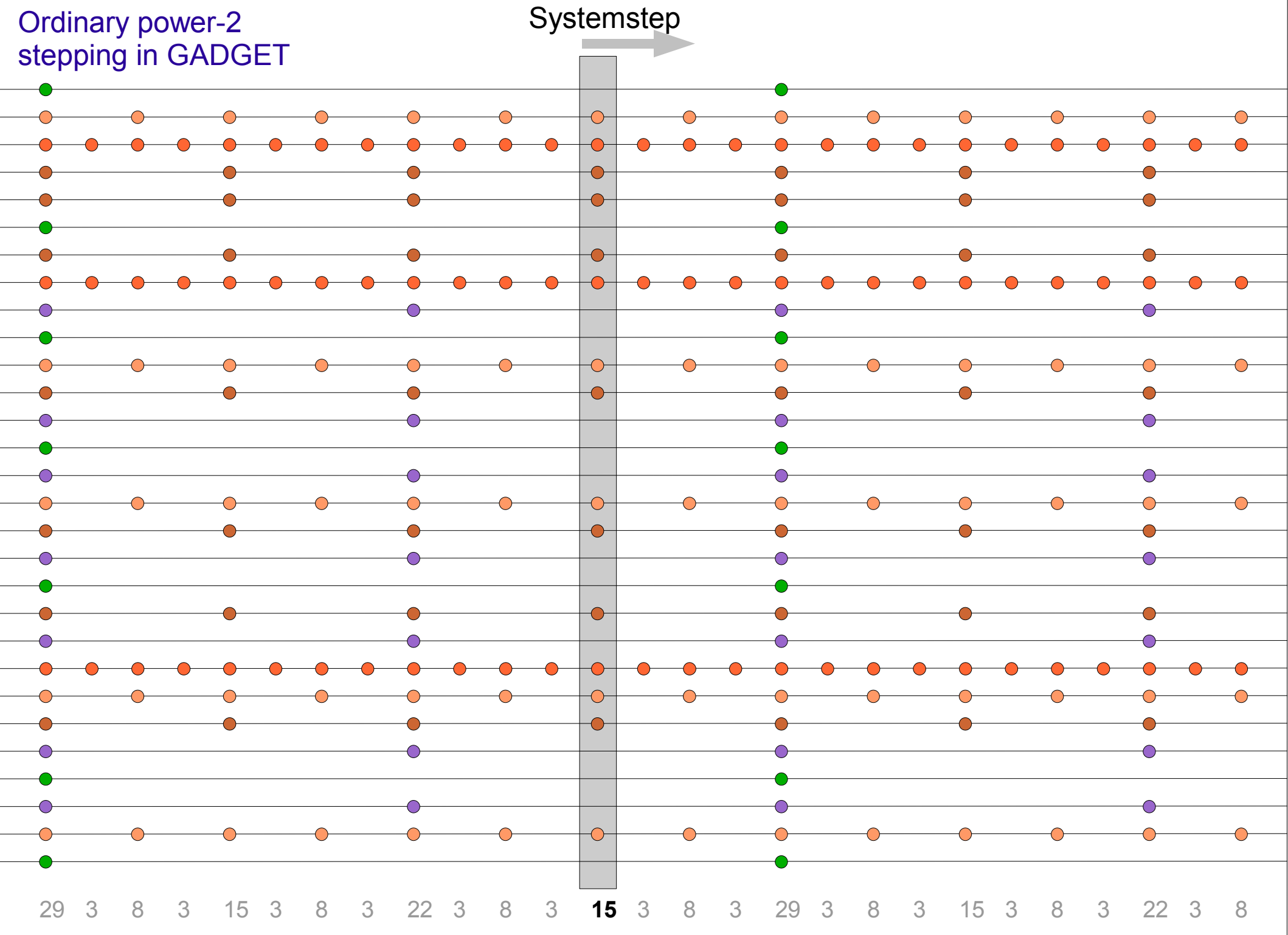
The time-steps of particles are spatially correlated

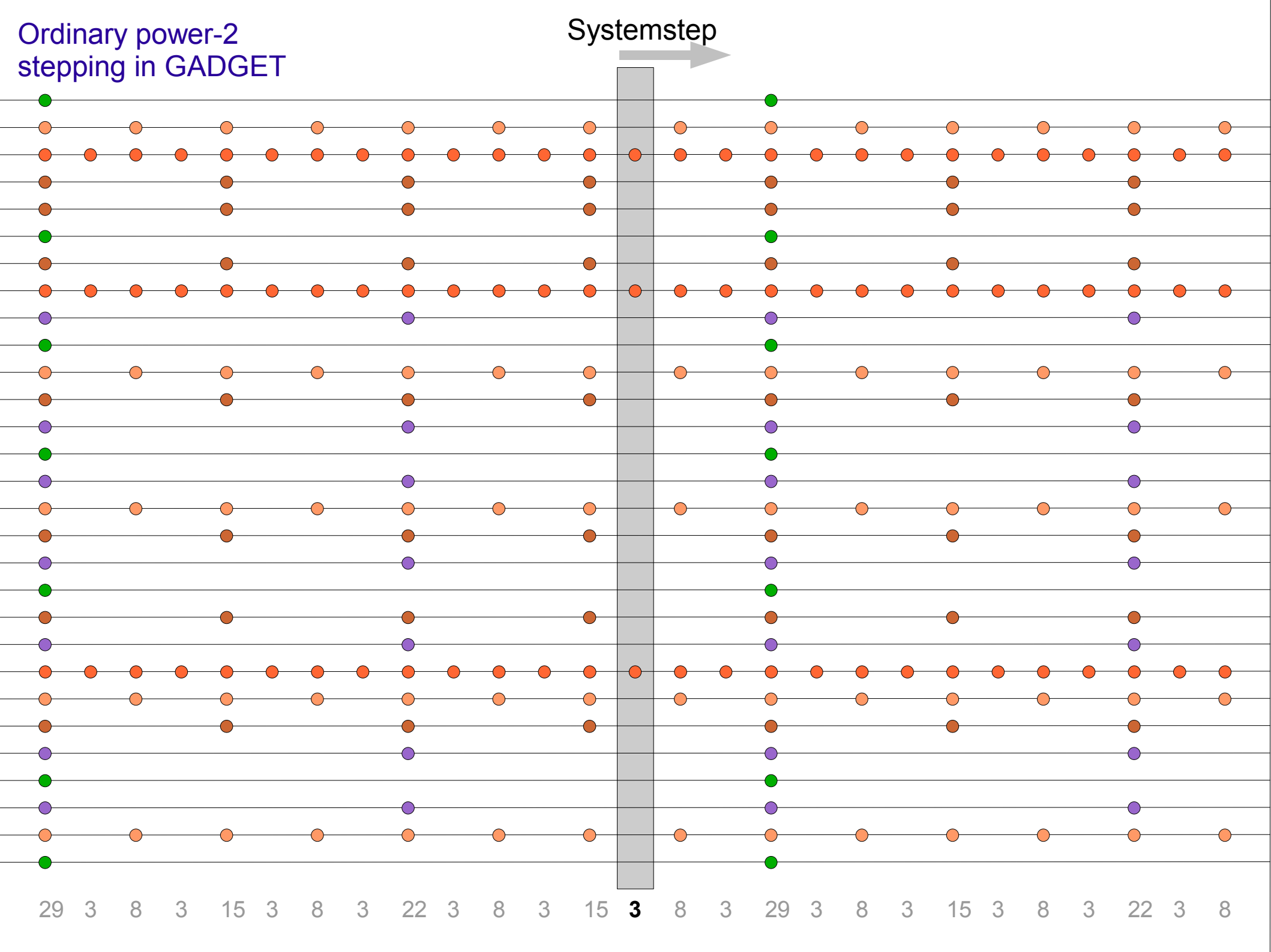


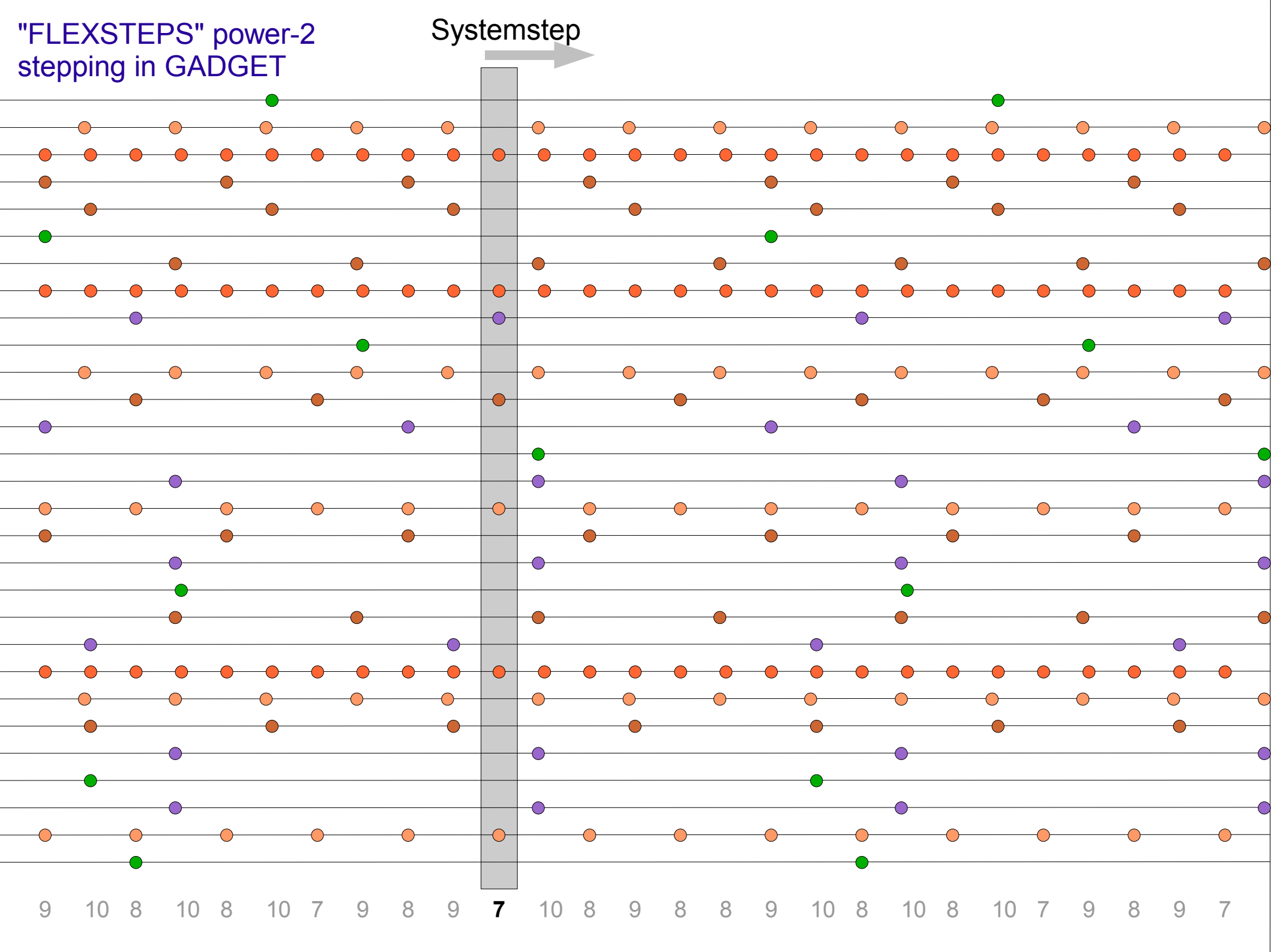

Ordinary power-2
stepping in GADGET
Systemstep
29 3 8 3 15 3 8 3 22 3 8 3 15 3 8 3 29 3 8 3 15 3 8 3 22 3 8


Ordinary power-2
stepping in GADGET
Systemstep
29 3 8 3 15 3 8 3 22 3 8 3 15 3 8 3 29 3 8 3 15 3 8 3 22 3 8

Ordinary power-2 stepping in GADGET

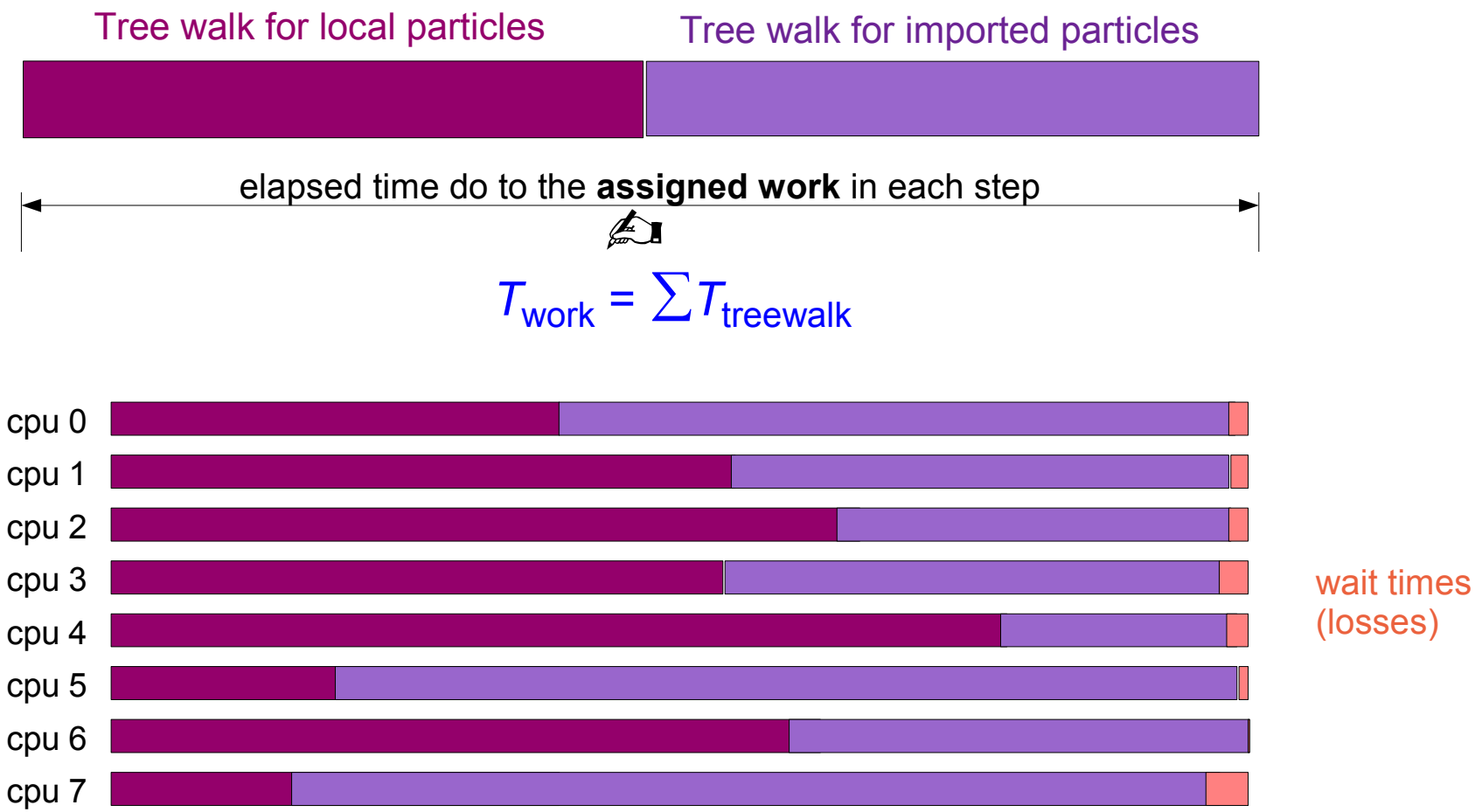



Ordinary power-2
stepping in GADGET
Systemstep
29 3 8 3 15 3 8 3 22 3 8 3 15 3 8 3 29 3 8 3 15 3 8 3 22 3 8


"FLEXSTEPS" power-2 stepping in GADGET
Systemstep
9 10 8 10 8 10 7 9 8 9 7 10 8 9 8 8 9 10 8 10 8 10 7 9 8 9 7

The cumulative execution time of the tree-walk on each processor is measured and used to adjust the domain decomposition

THE "CPUSPEEDADJUSTMENT" OPTION

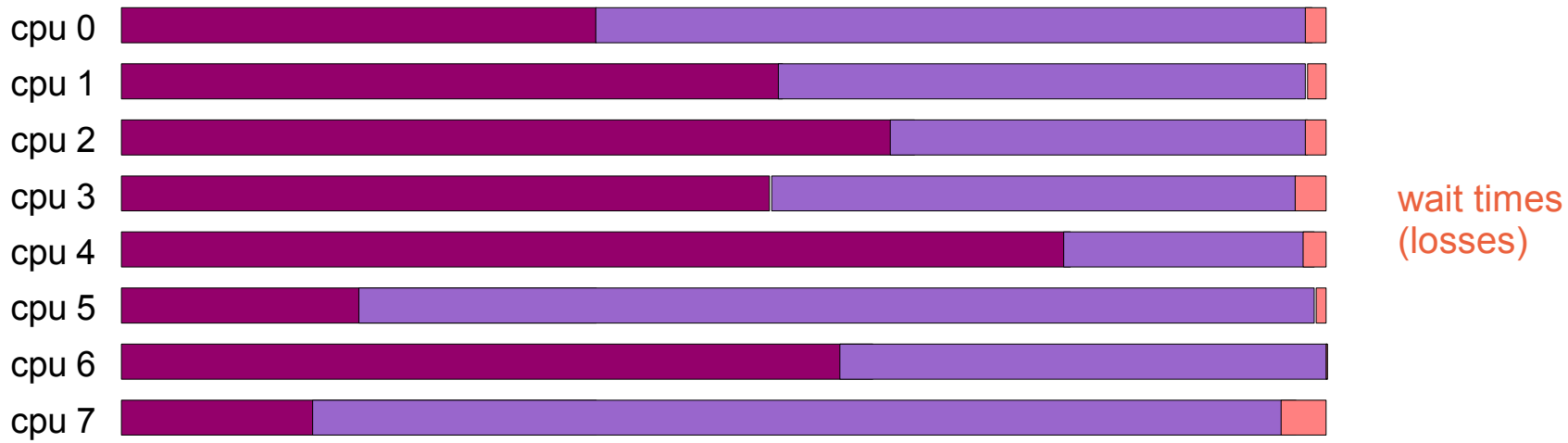


→ Together with shuffled timestep hierarchy, the total CPU-time for the tree-walks per step can be made roughly equal

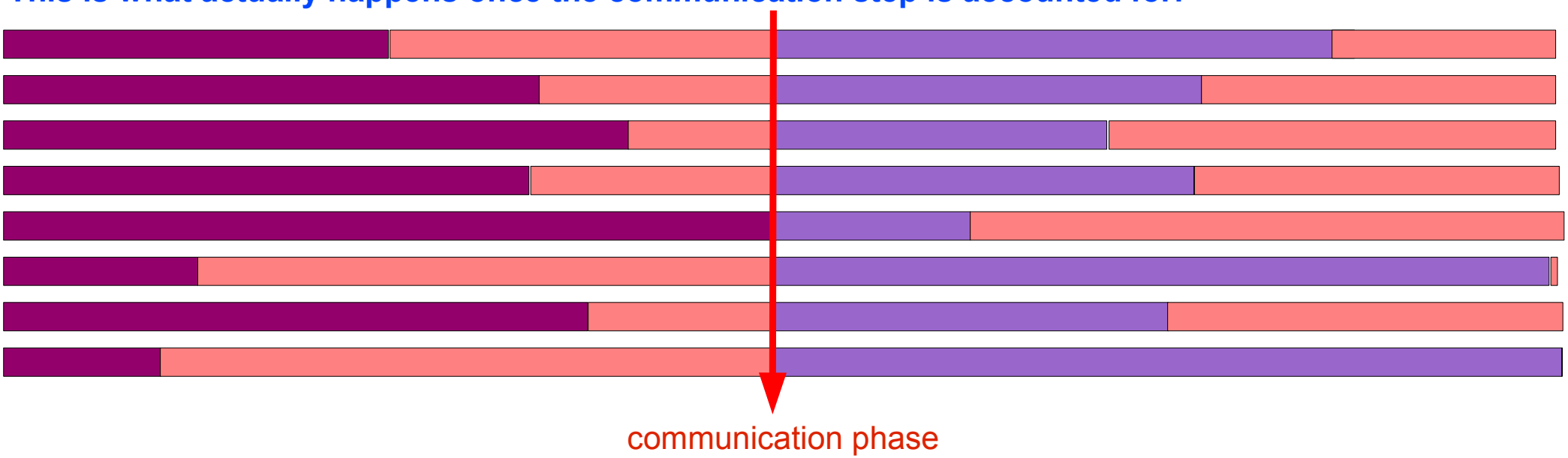
The communication between the two phases of a step introduces a synchronization point in GADGET2's standard communication scheme

LOSSES DUE TO IMBALANCE IN DIFFERENT COMMUNICATION PHASES

The situation after work-load balancing:



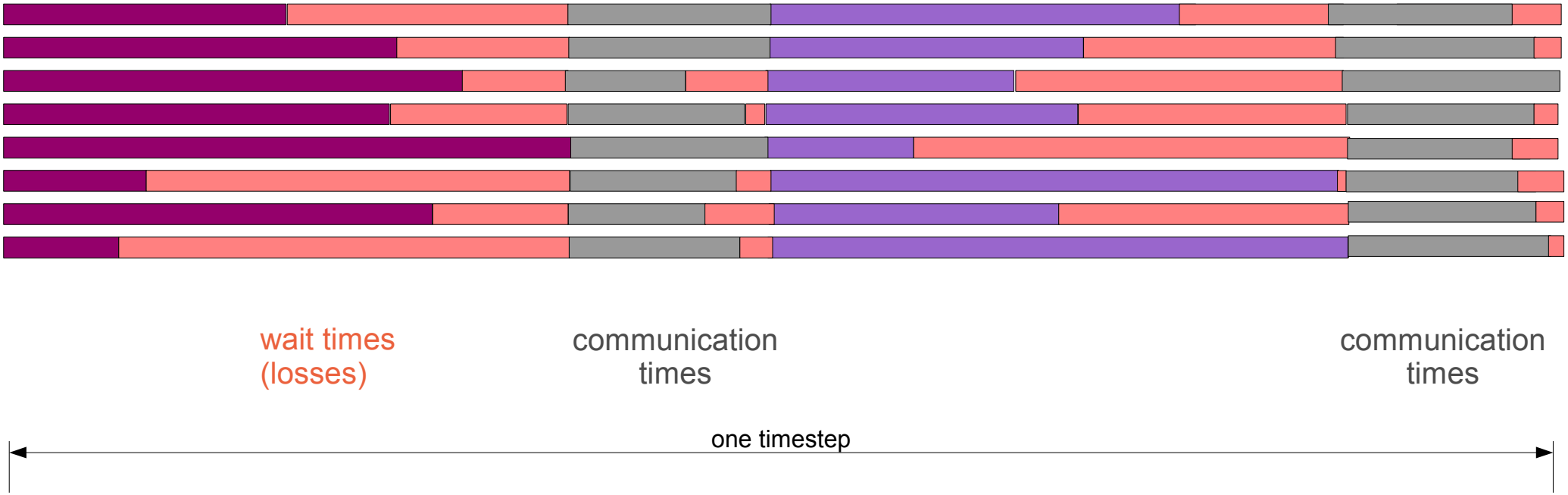
This is what actually happens once the communication step is accounted for:



The communication itself consumes some time and also induces additional wait times

LOSSES DUE TO COMMUNICATION TIMES IN ONE GRAVITY STEP

This is the real situation in GADGET-2....



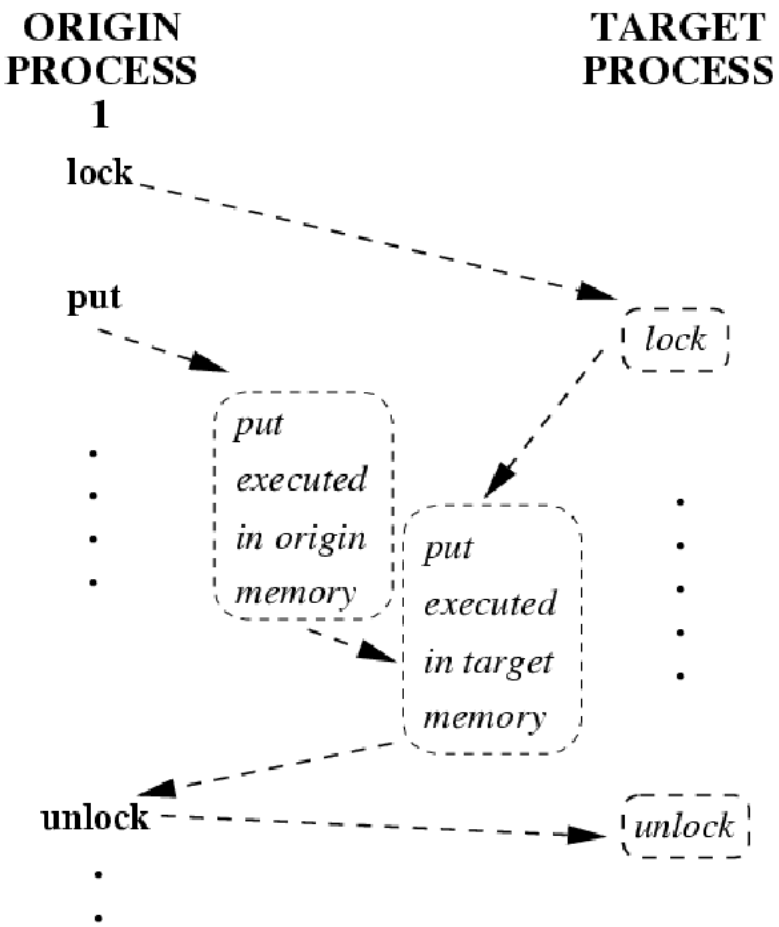
An improvement of scalability appears to require asynchronous communication

POSSIBLE OPTIONS FOR ASYNCHRONOUS COMMUNICATION

One-sided communication?

Available with MPI-2.... but:

- rather restrictive API
- complicated communication semantics
- active and **passive target** one-sided communications are supported, but both require explicit synchronisation calls
- progress of passive target mode may rely on MPI-calls of target (e.g. MPICH2)



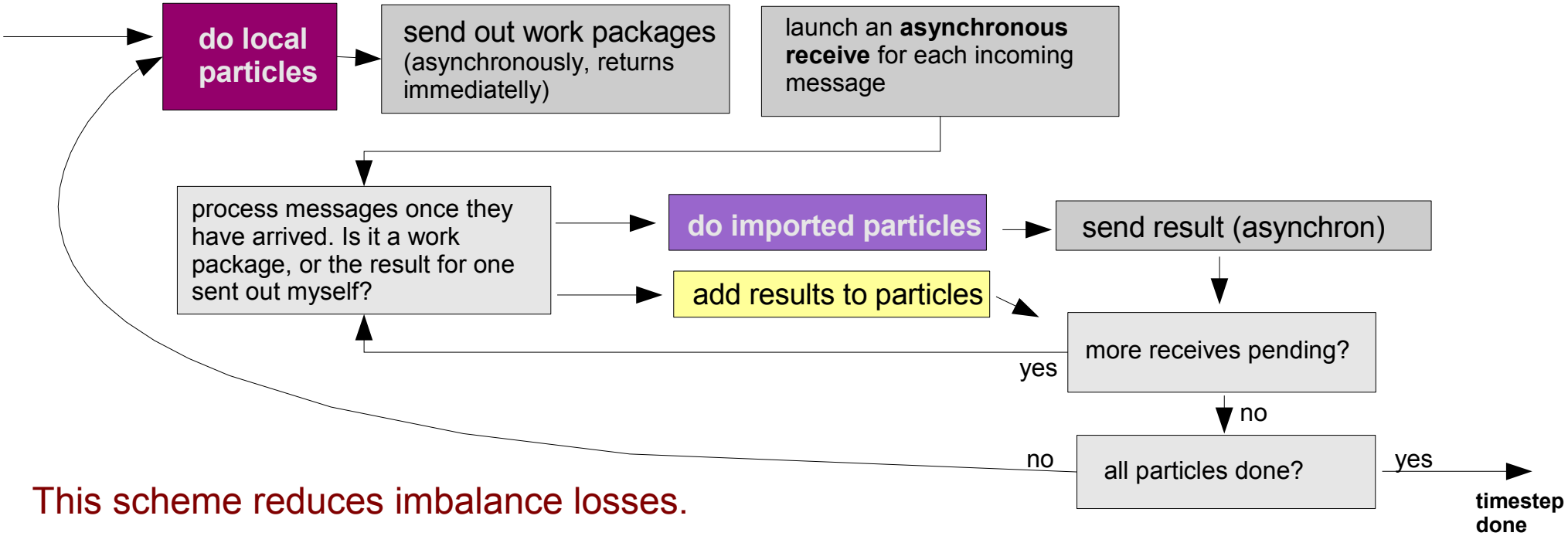
Use MPI's asynchronous two-sided communication?

Available with MPI-1

- use buffered sends (MPI_Bsend)
- use asynchronous receives with explicit checks for completion (MPI_Irecv)
- use MPI_Probe to test for incoming messages

Asynchronous communication and a pipelining approach can eliminate the mid-step imbalance losses in the gravity step

FLOW-CHART FOR ONE TIMESTEP IN NEW GADGET COMMUNICATION SCHEME



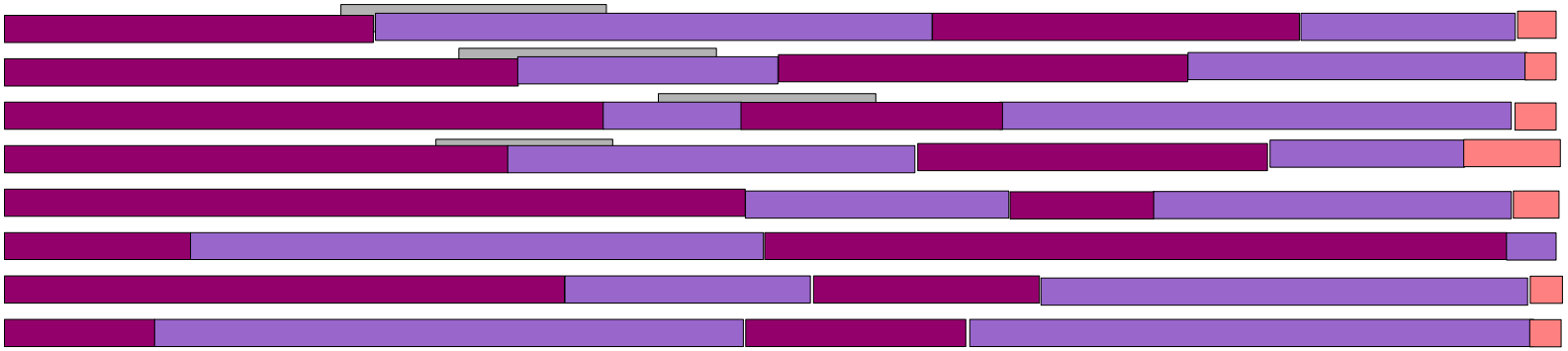
This scheme reduces imbalance losses.

It can also **overlap communication and computation.**

Overlap can be realized on:

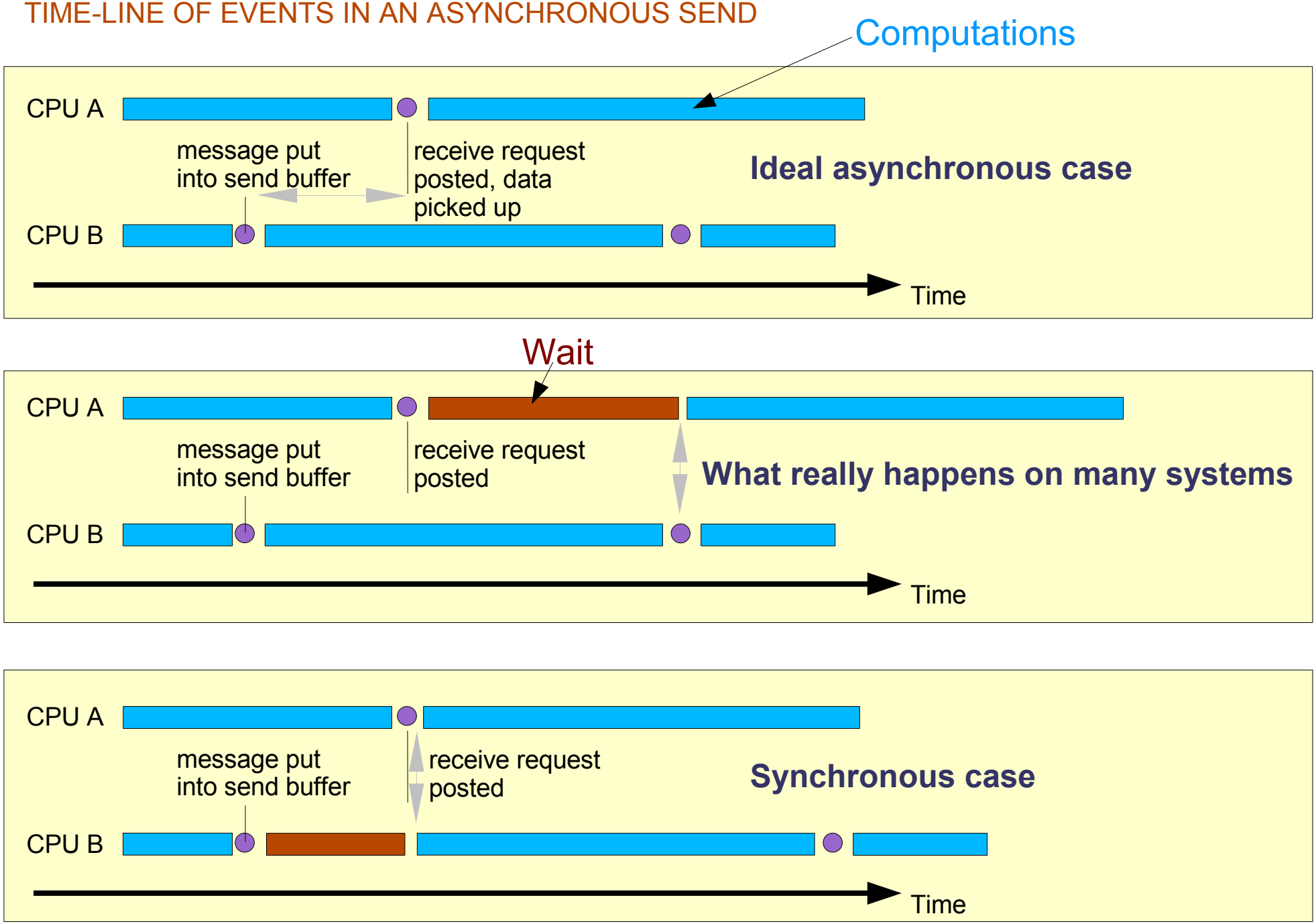
- IBM Power4
- **IBM Bluegene?**
- Infiniband Cluster (MVAPICH)
- SMP boxes
- Myrinet/Quadrics

New communication scheme:



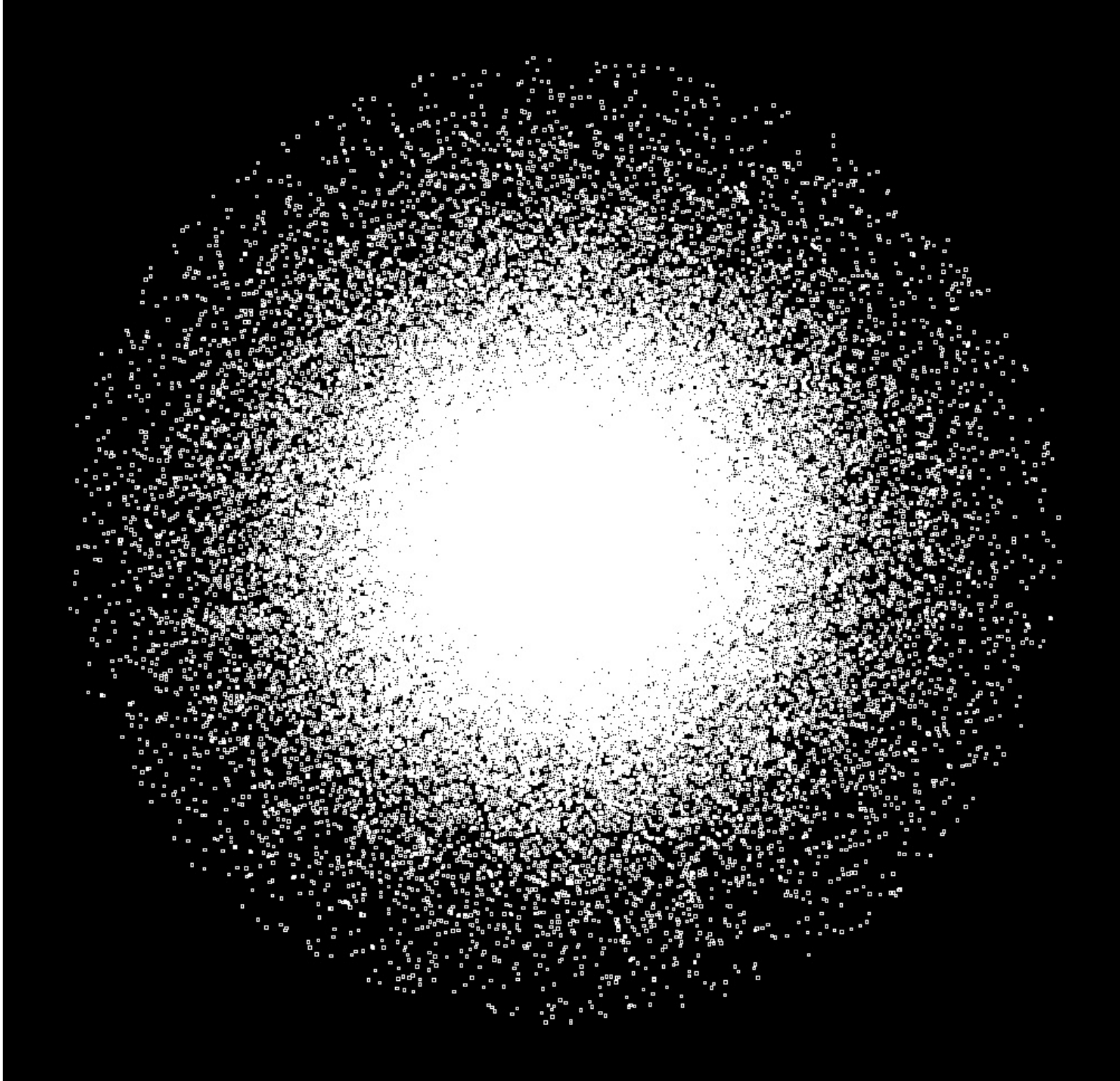
On many systems, asynchronous communication still requires a concurrent MPI call of the other process to ensure progress

TIME-LINE OF EVENTS IN AN ASYNCHRONOUS SEND



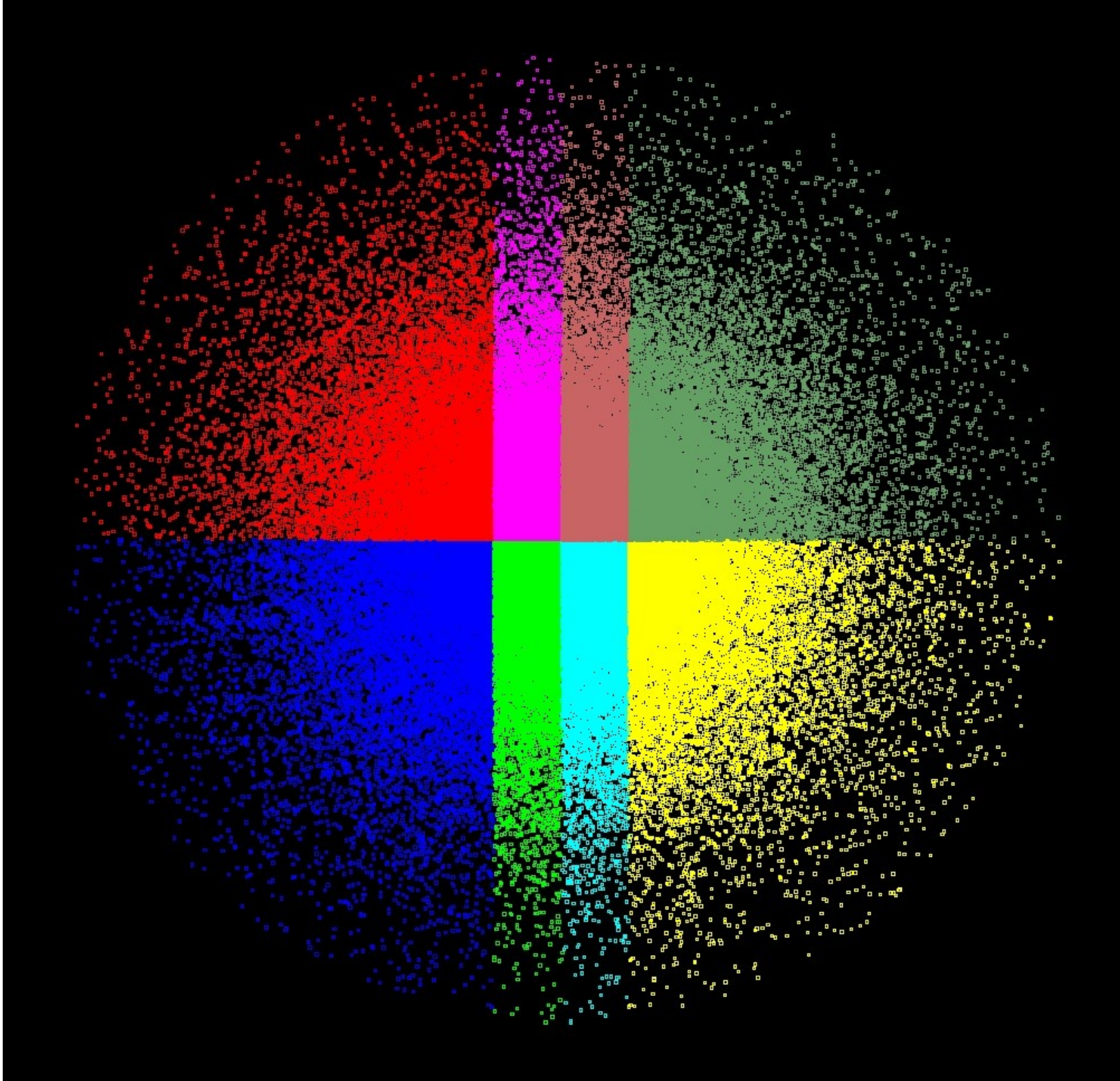
The inhomogeneous particle distribution and the different timesteps as a function of density make it challenging to find an optimum domain decomposition that balances work-load (and ideally memory-load)

PARTICLE DISTRIBUTION IN AN EXPONENTIAL DISK



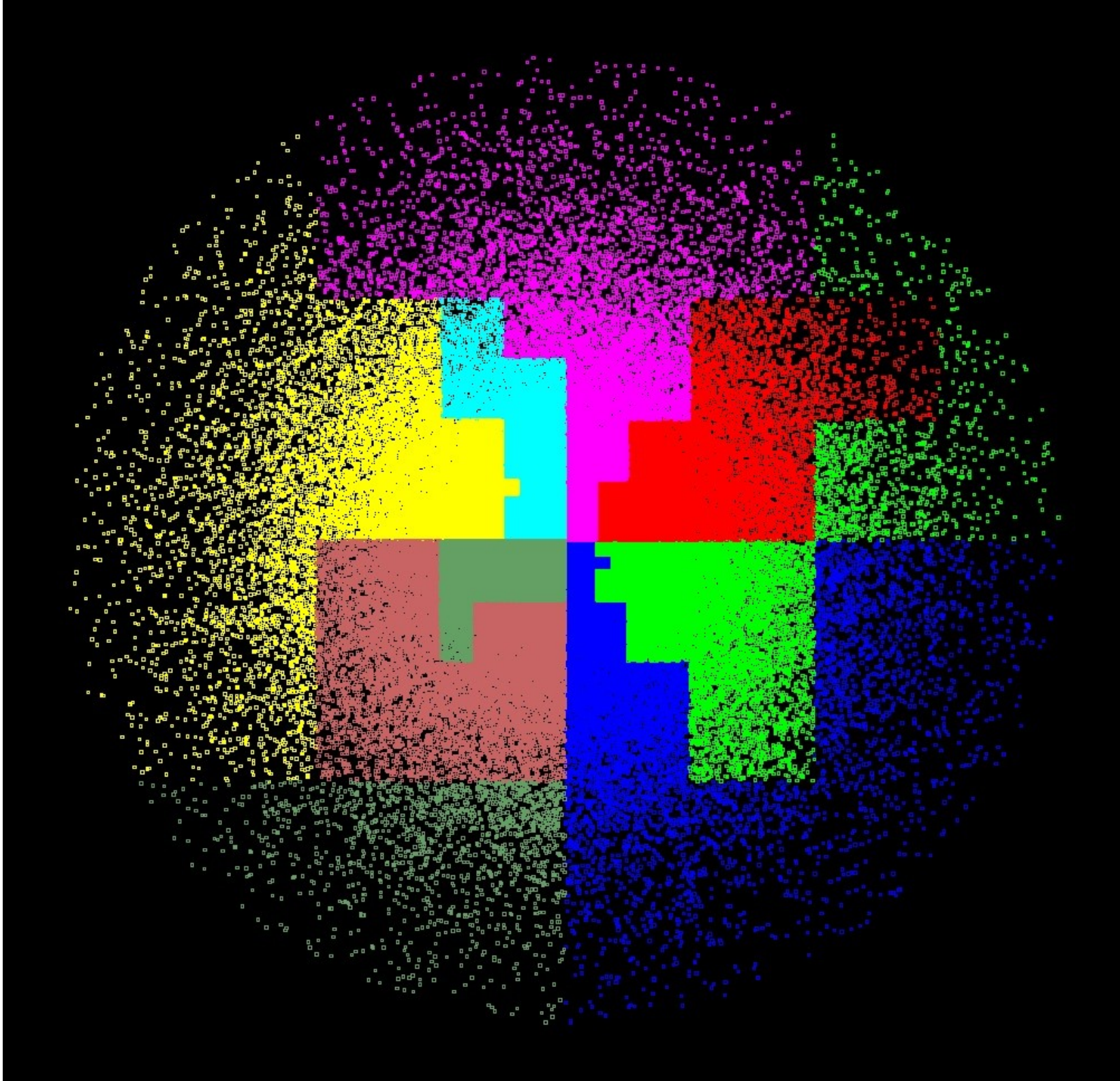
GADGET-1 used a simple orthogonal recursive bisection

EXAMPLE OF DOMAIN DECOMPOSITION IN GADGET-1



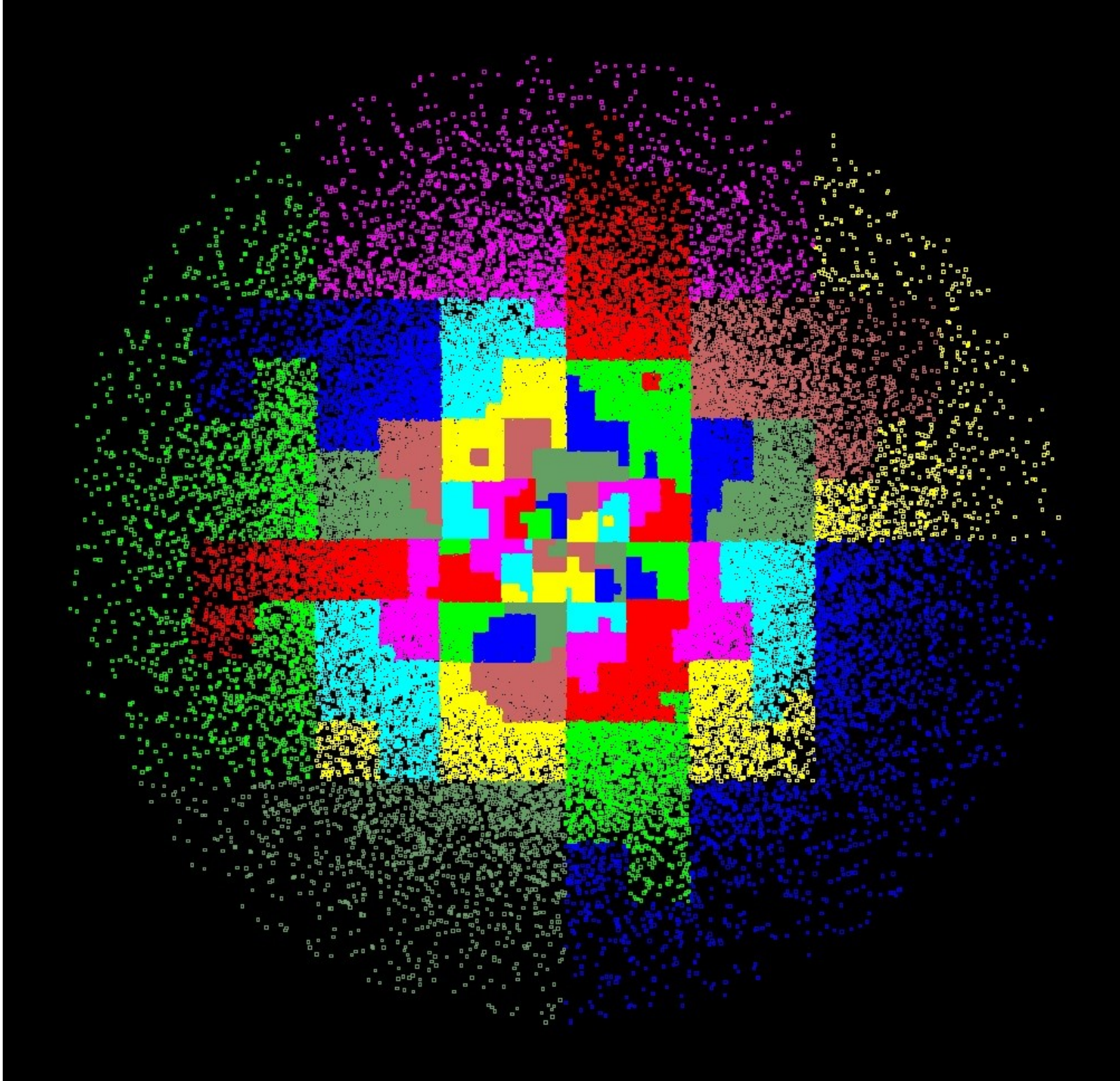
GADGET-2 uses a more flexible space-filling Peano-Hilbert curve

EXAMPLE OF DOMAIN DECOMPOSITION IN GADGET-2



GADGET-3 uses a space-filling Peano-Hilbert curve which is more flexible

EXAMPLE OF DOMAIN DECOMPOSITION IN GADGET-3



The new domain decomposition scheme can balance the work-load and the memory-load at the same time but requires more communication

THE SIMPLE IDEA BEHIND MULTI-DOMAINS

The domain decomposition partitions the space-filling curve through the volume

GADGET-2

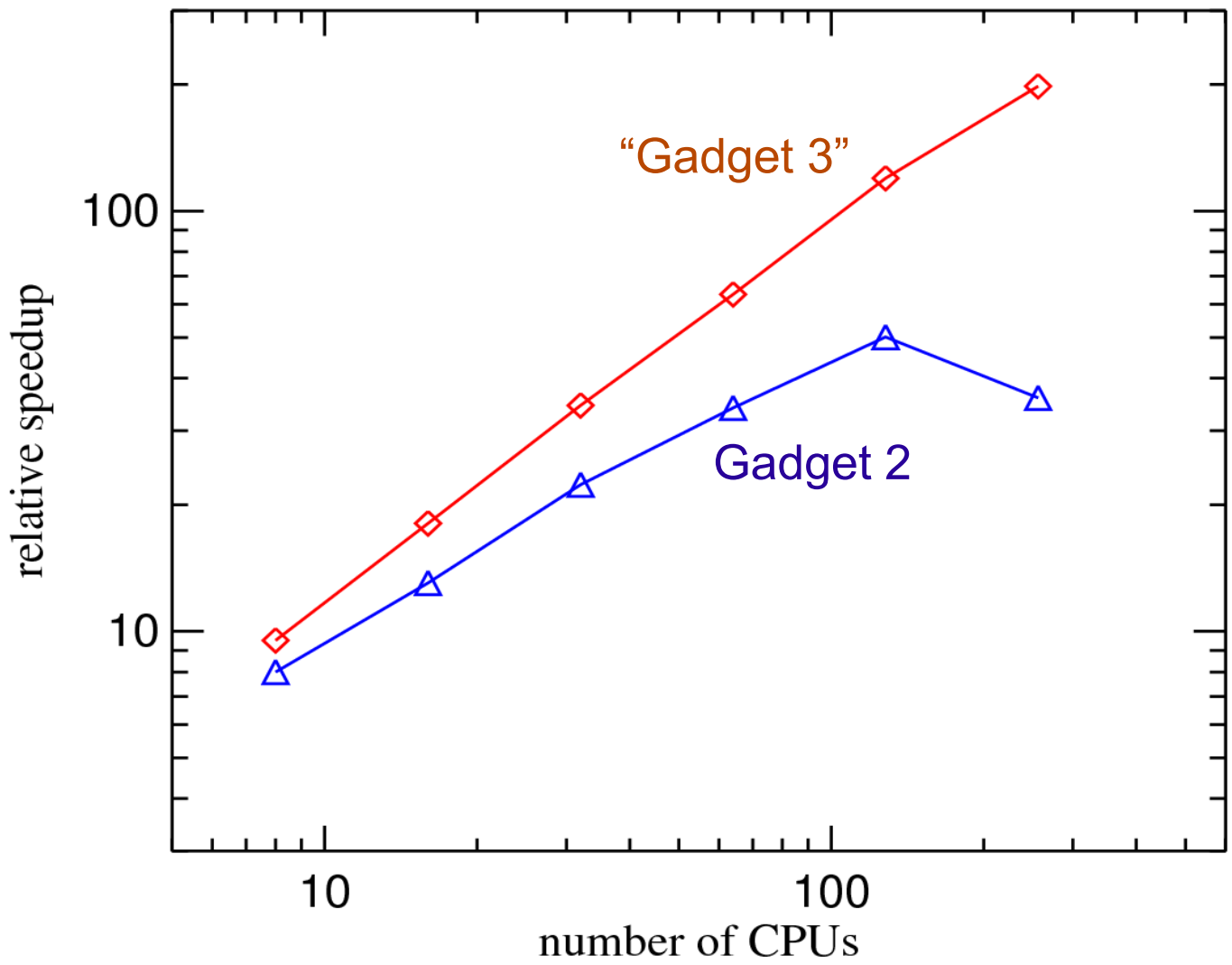
GADGET-3

But:

- Need a more efficicient domain decomposition code
- Need a tree-walk scheme that doesn't slow down if there are more domains
- Need a new communication strategy for the PM part of the code

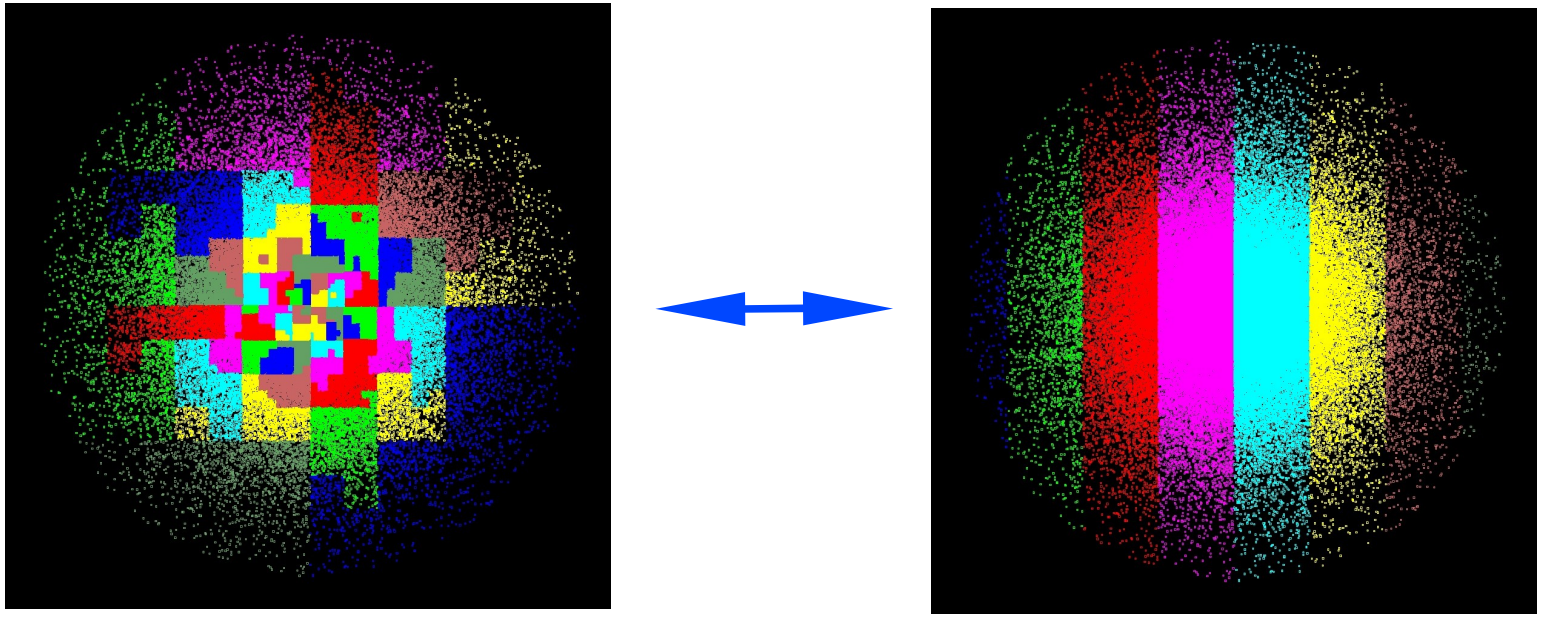
The new code scales substantially better for high-res zoom simulations of isolated halos

A STRONG SCALING TEST ON BLUEGENE OF A SMALL HIGH-RES HALO



Changing from the tree domain decomposition to the slab decomposition needed for the FFTs is a non-trivial problem

ACCOMDATING THE SLAB DECOMPOSITION

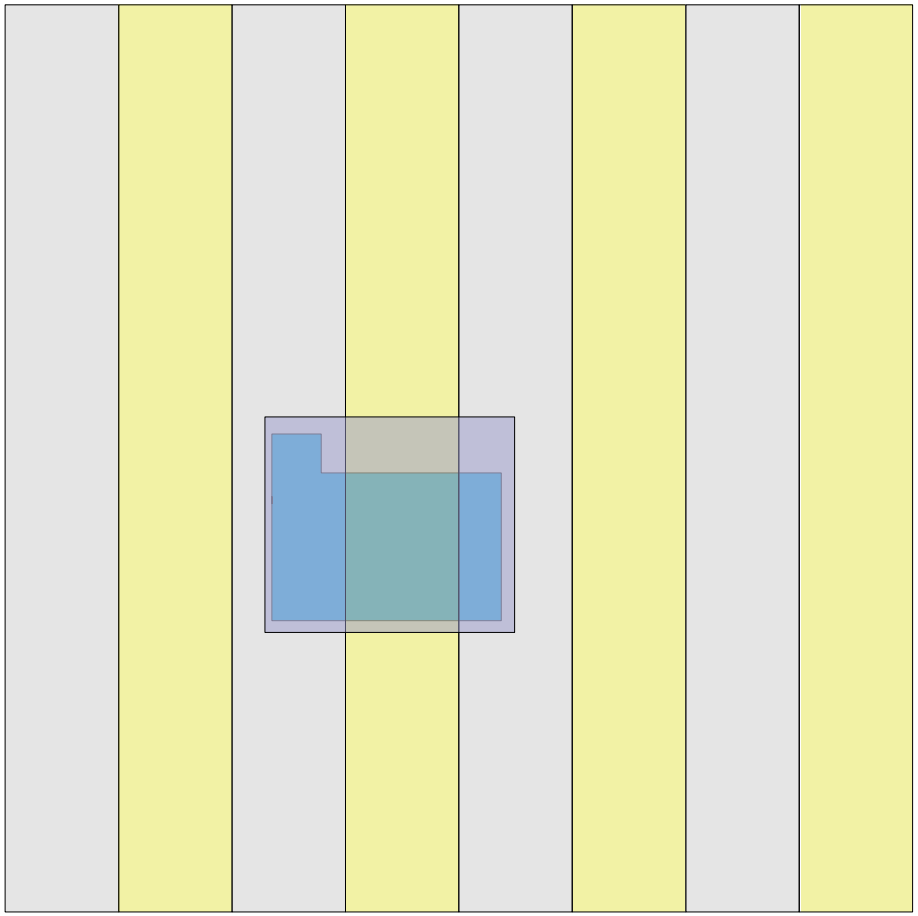


Simply swapping the particle set into a slab decomposition is in general not a good idea

- Memory-load can become hugely imbalanced (especially for zoom simulations)
- Work-load in binning and interpolating off the grid very imbalanced
- Ghost layers may require substantial memory if number of CPUs not very different from 1-d grid resolution

In GADGET2, a local mesh-patch is constructed that encloses the local domain

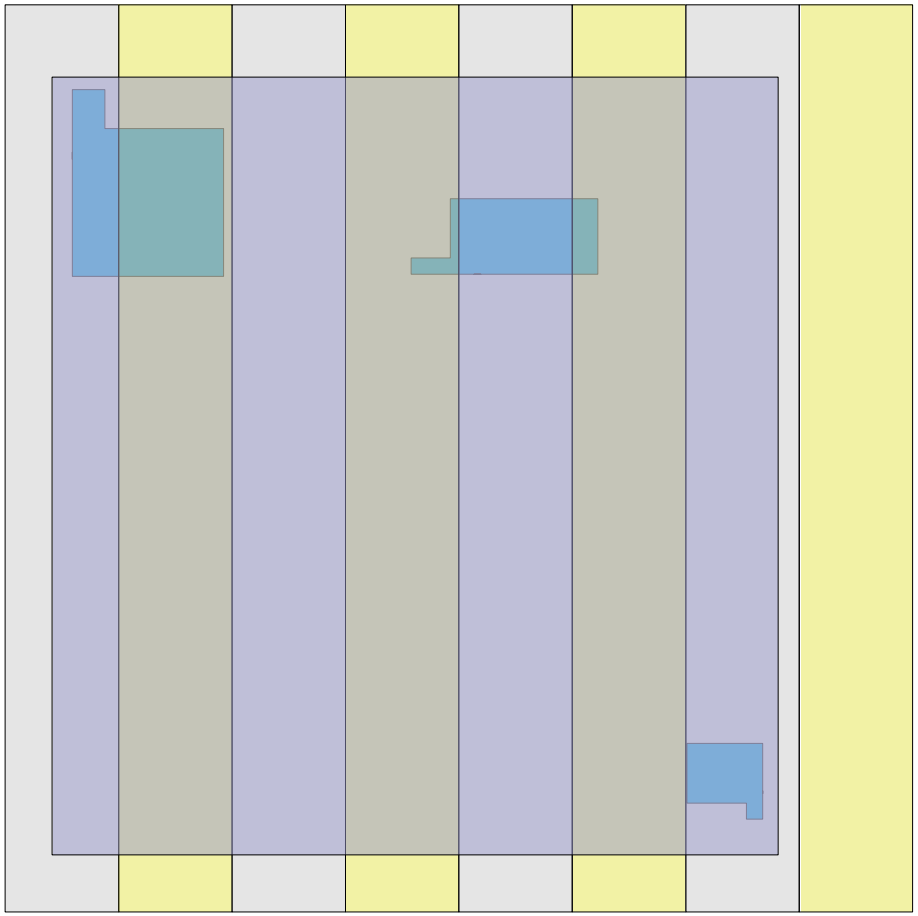
PM COMMUNICATION ALGORITHM IN GADGET-2



- Communication only occurs with subset of slabs that intersect local patch
- Memory requirement of PM algorithm independent of the number of CPUs used for a given PM mesh size (think slabs are no problem)

For multiple local domains, the enclosing rectangular patch quickly approaches the volume of the entire grid

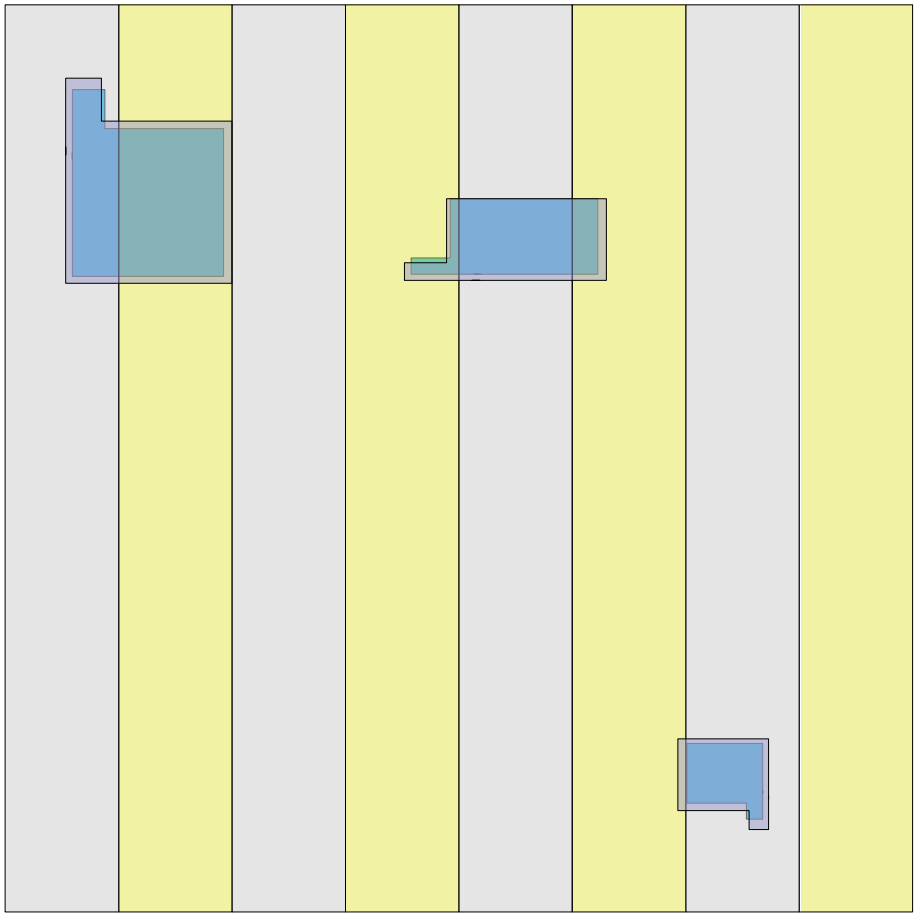
LOCAL RECTANGULAR MESH-PATCH FOR MULTIPLE DOMAINS



- This becomes quickly prohibitive in terms of memory consumption, as each processor effectively holds a pach of size of the whole PM grid

In the new approach, we tightly fit arbitrarily shaped mesh-patches to the local domains

LOCAL MESH-PATCHES FOR MULTIPLE DOMAINS

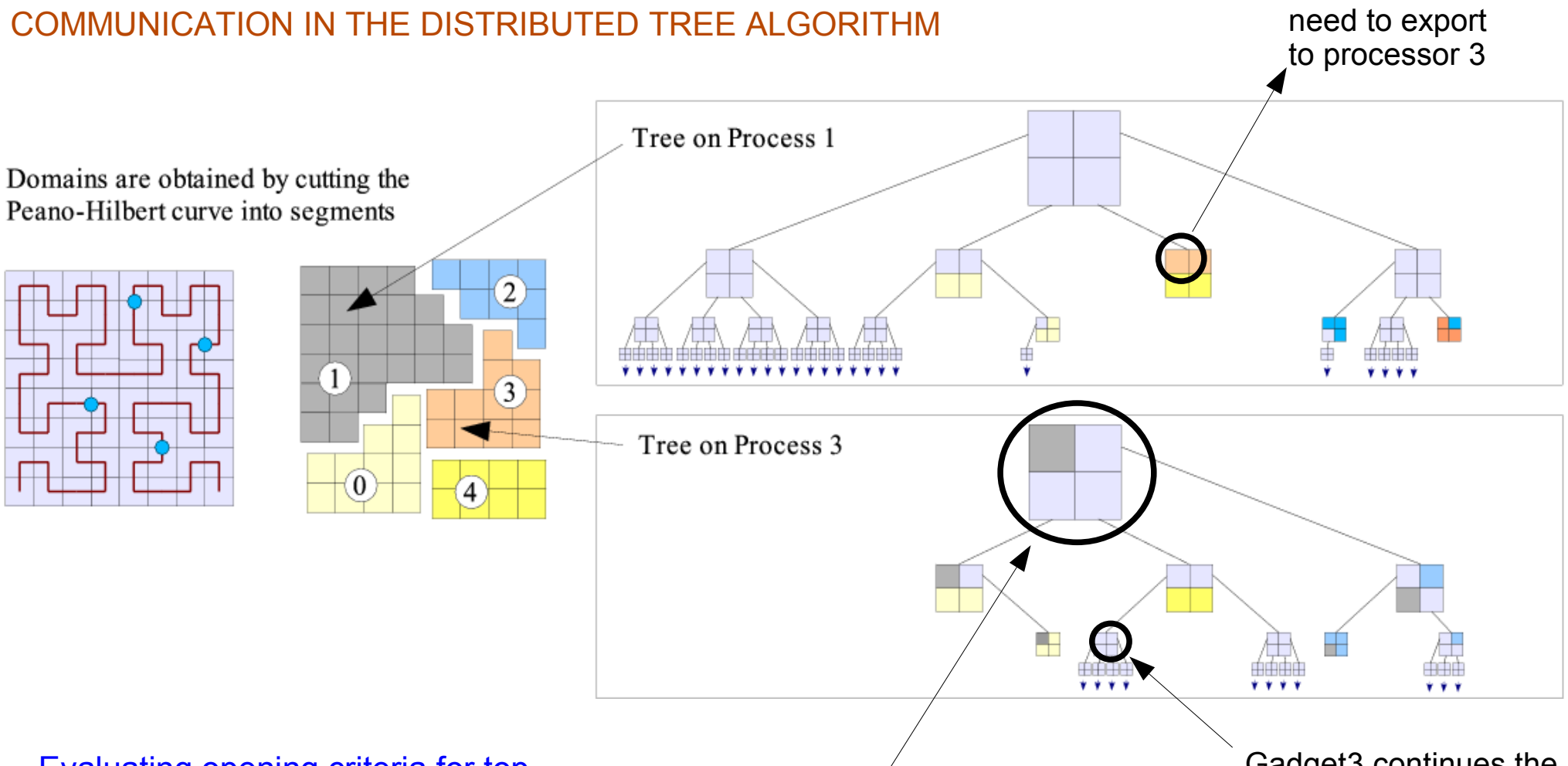


The arbitrarily shaped mesh patches are organized as a table with a value and an index into the full field. Only cells that are “touched” at least once are stored.

- Binning and interpolation part of the algorithm well balanced
- No superfluous storage needed, and storage requirements to good approximation independent of tree domain decomposition
- Since no ghost layers for finite differencing of the potential field are used, one additional global transposition of the potential is carried out

In the new code, exported particles know where to continue the tree walk on the *foreign* processor

COMMUNICATION IN THE DISTRIBUTED TREE ALGORITHM



Evaluating opening criteria for top-level tree nodes multiple times can be eliminated. The work for tree walks (gravity and SPH neighbor search) becomes strictly independent of the number of processors.

Code development in GADGET continues...

PRIMARY NEW FEATURES OF GADGET-3

- New domain decomposition for multiple domains, leading to better scalability of the code. Domain decomposition code itself is much faster for large processor numbers.
- Speed improvement of tree-walks by eliminating parallelization overhead. (required extensive rewrites of SPH and tree communication)
- Improved memory handling of code, reducing peak usage.
- Much more accurate and detailed internal accounting of CPU time consumption, including informative, human-readable output for every timestep.
- Speed improvements in neighbor search, tree construction and updates, and in generation of Peano-Hilbert keys
- New PM code which is work-load balanced even for zoom simulations.

The new version of the code can be quite a bit better than the old version...