PiTP Summer School 2009

Plan for my lectures

Volker Springel

Lecture 1 ▶ **Basics of collisionless dynamics and the N-body approach**

Lecture 2 ▶ **Gravitational solvers suitable for collisionless dynamics, parallelization**

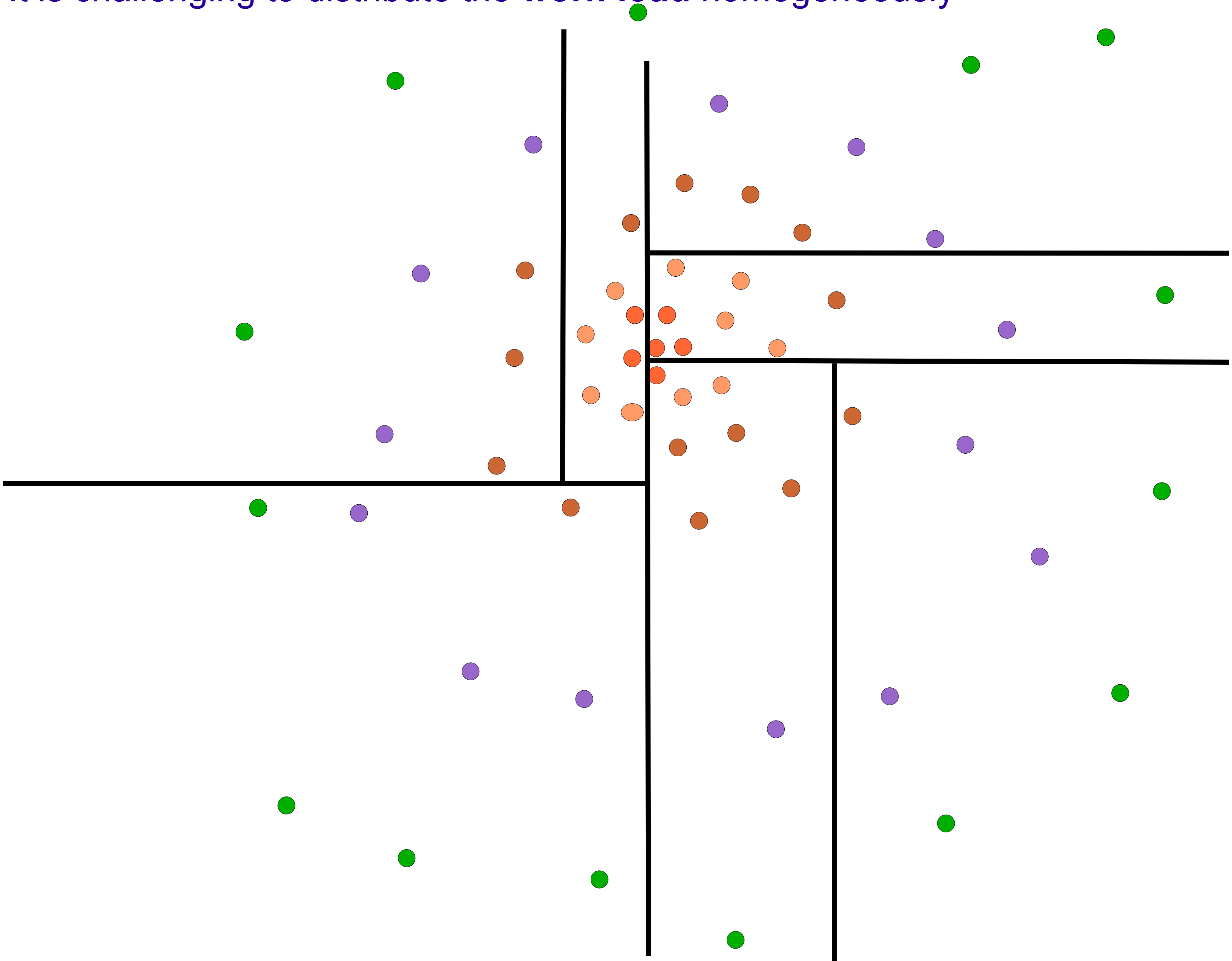
Lecture 3 ▶ **More parallelization, Introduction to smoothed particle hydrodynamics**

Lecture 4 ▶ **Algorithmic aspects of SPH, caveats, applications**

Lecture 5 ▶ **Comparison of SPH to finite volume methods, Moving-mesh hydrodynamics**

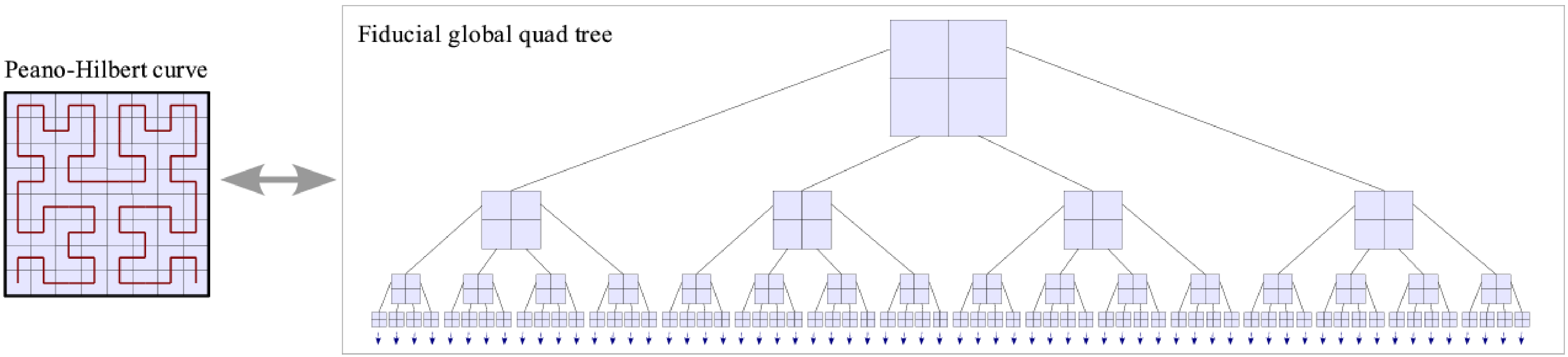
Parallel computing: Scalability and its limitations

It is challenging to distribute the **work-load** homogeneously

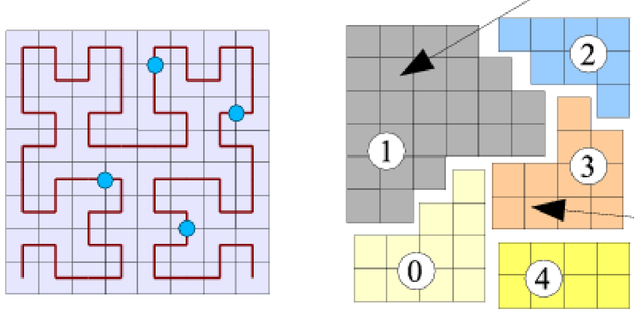


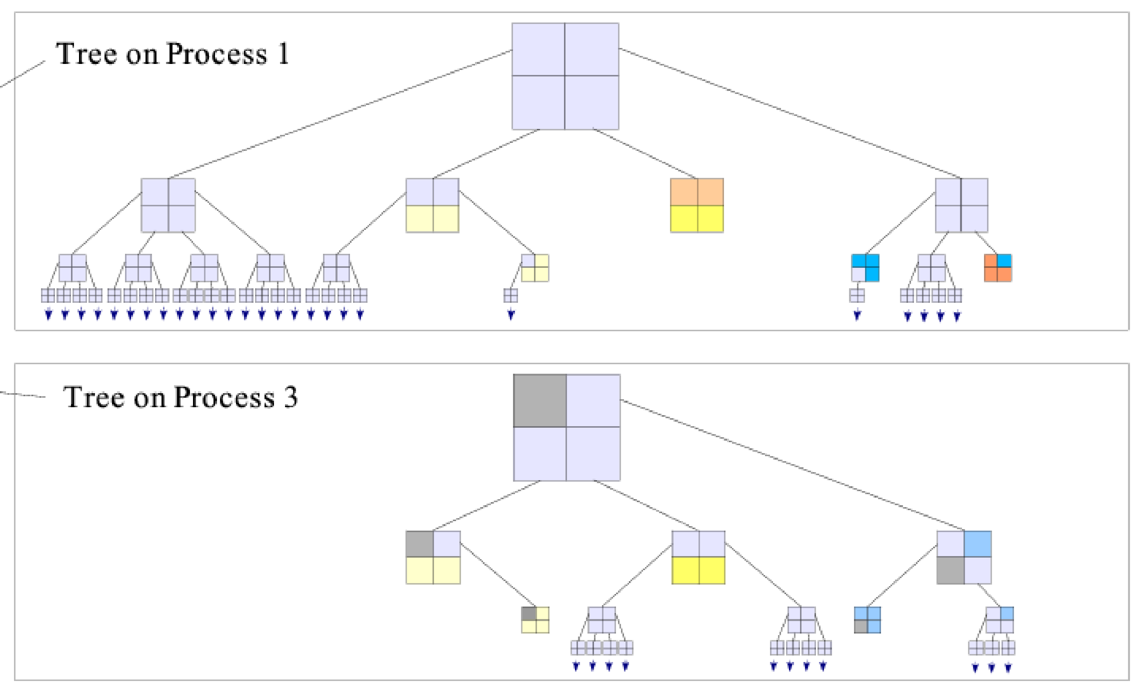
The space-filling Peano-Hilbert is used in GADGET-2 for the domain-decomposition

SPLITTING UP THE TREE FOR DIFFERENT PROCESSORS



Domains are obtained by cutting the Peano-Hilbert curve into segments





But: How well can the work-load split in practice?

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 arbritrarily 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

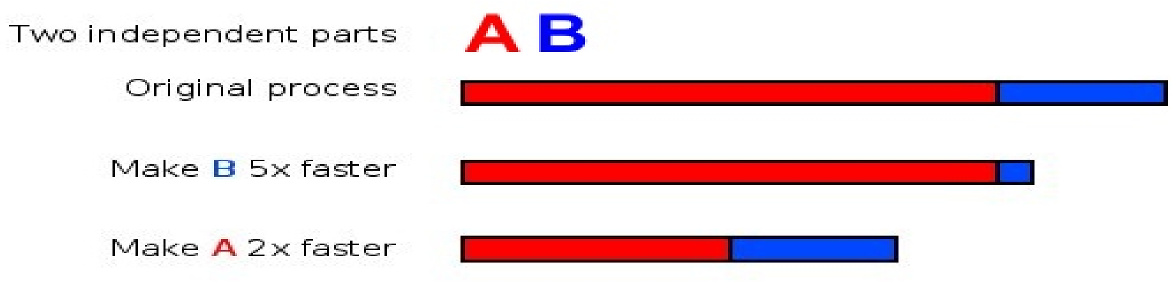
Weak scaling: When number of CPUs is increased, also increase the problem size

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 !**

Amdahl's law provides a fundamental limit for the speed-up that can be achieved in a parallel code

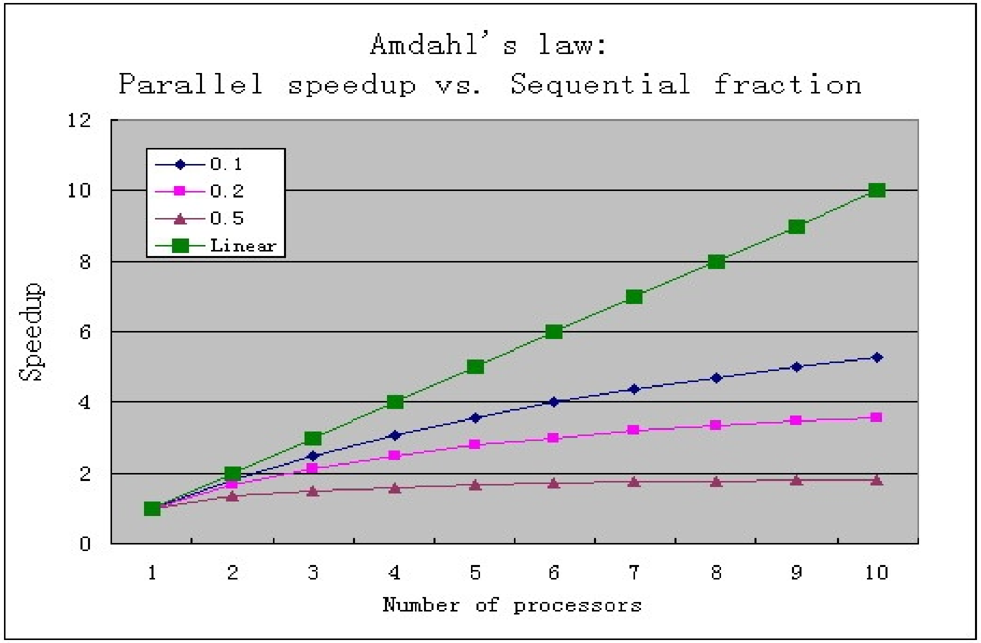
THE IMPLICATIONS OF A RESIDUAL SERIAL FRACTION



Speed up for serial fraction F on N processors:

$$\frac{1}{F + (1-F)/N}$$

Example: If F = 5%, then the speed up is at most 20, no matter how many processors are used!



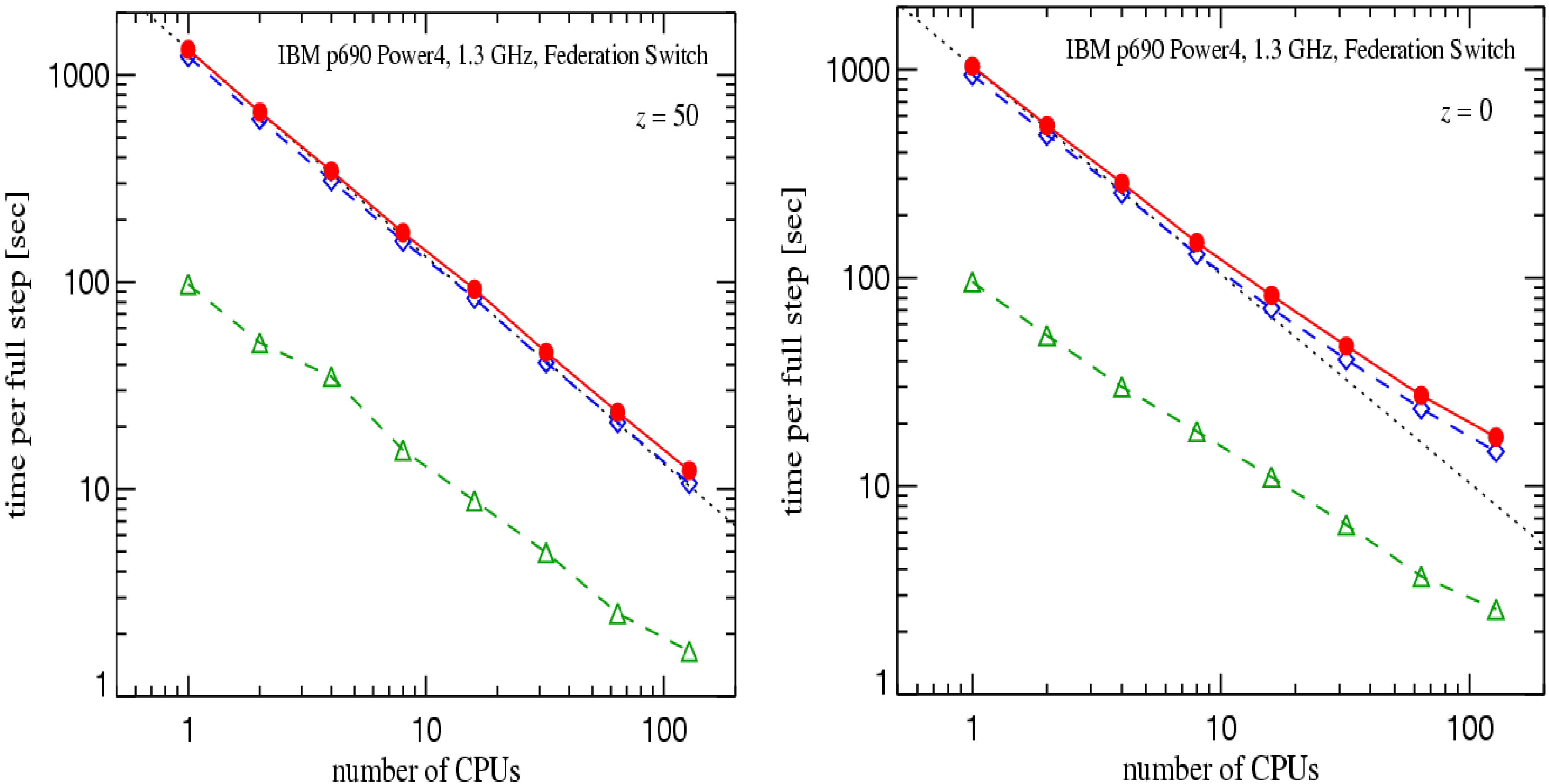
"The first 90% of the code accounts for the first 90% of the development time. The remaining 10% of the code account for the other 90% of the development time."

- Tom Cargill, Bell Labs

For fixed timesteps and large cosmological boxes, the scalability of the GADGET-2 code is not too bad

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

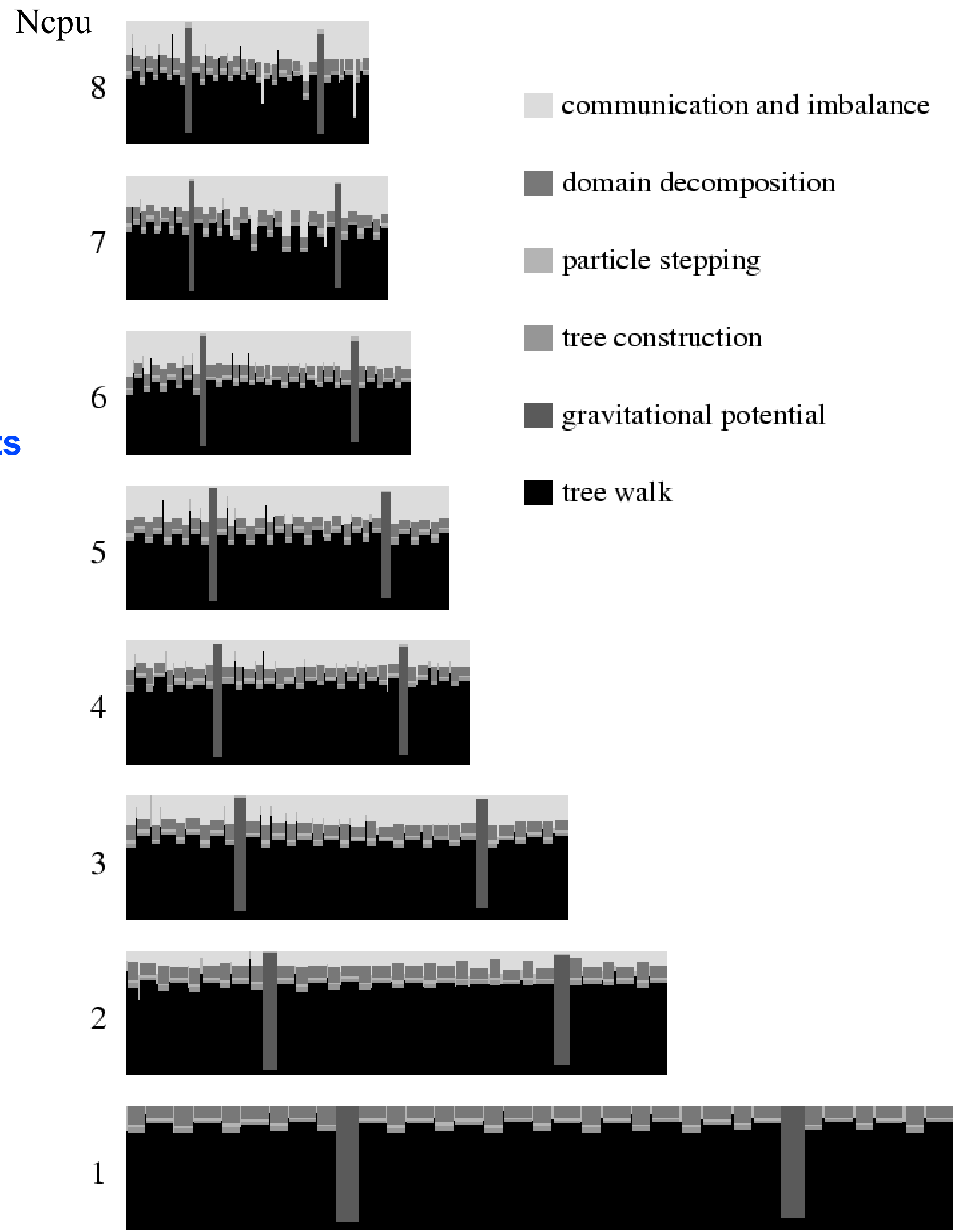
256^3 particles in a 50 h^{-1} Mpc box



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

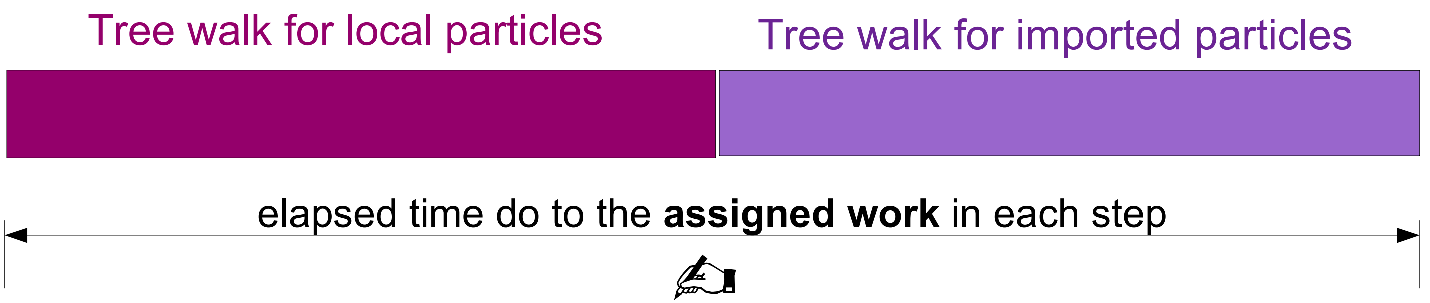
RESULTS FOR "STRONG SCALING" OF A GALAXY COLLISION SIMULATION

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

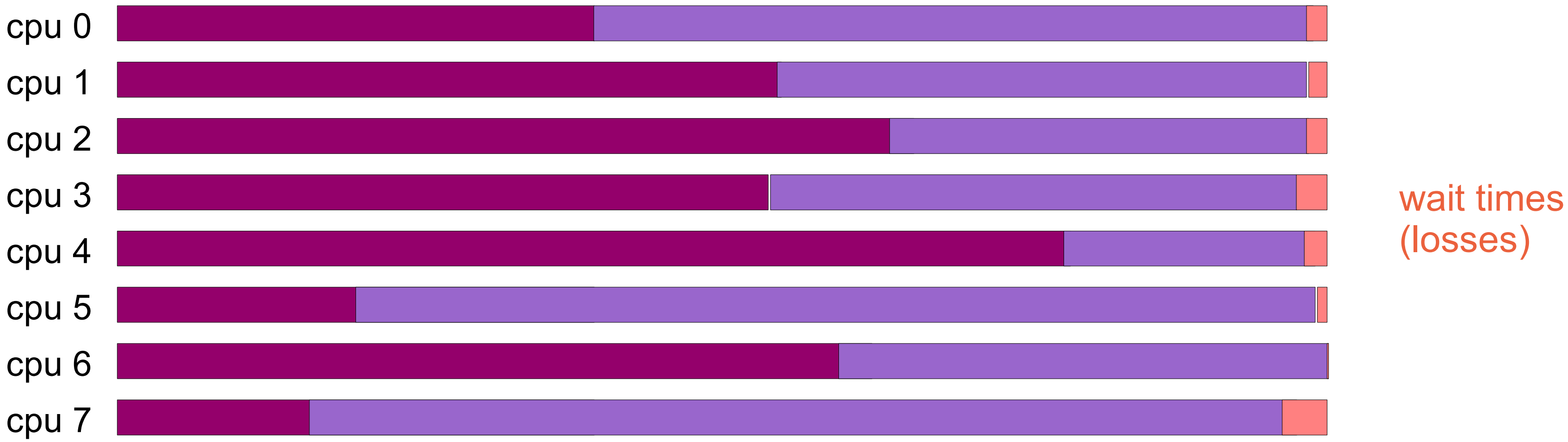


The cumulative execution time of the tree-walk on each processor can be measured and used to adjust the domain decomposition

BALANCING THE TOTAL WORK FOR EACH PROCESSOR



$$T_{\text{work}} = \sum T_{\text{treewalk}}$$

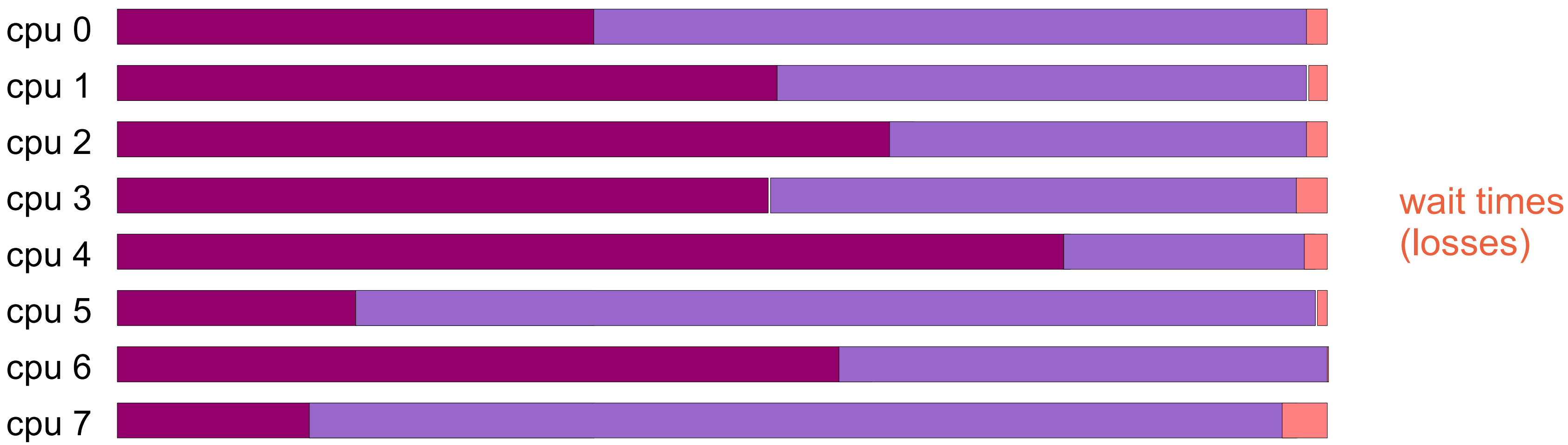


→ The total CPU-time for the tree-walks per step can be made roughly equal for each MPI task

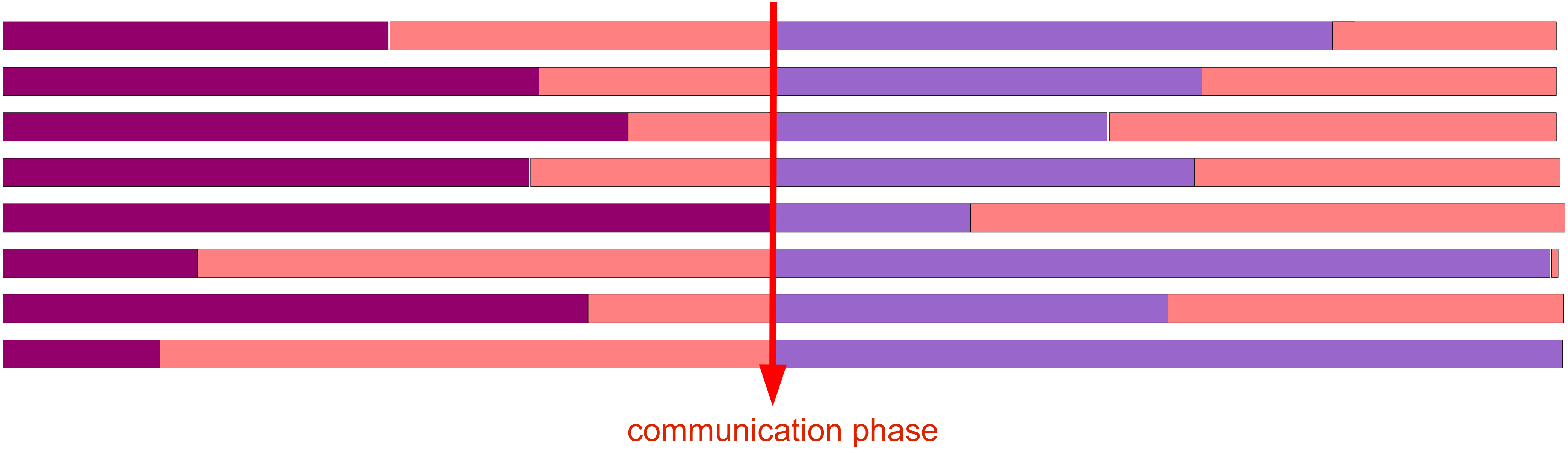
The communication between the two phases of a step introduces a synchronization point in GADGET2's standard communication scheme

LOSSES DUE TO IMBALANCE IN DIFFERENT COMMUNICATION PHASES

The situation after work-load balancing:



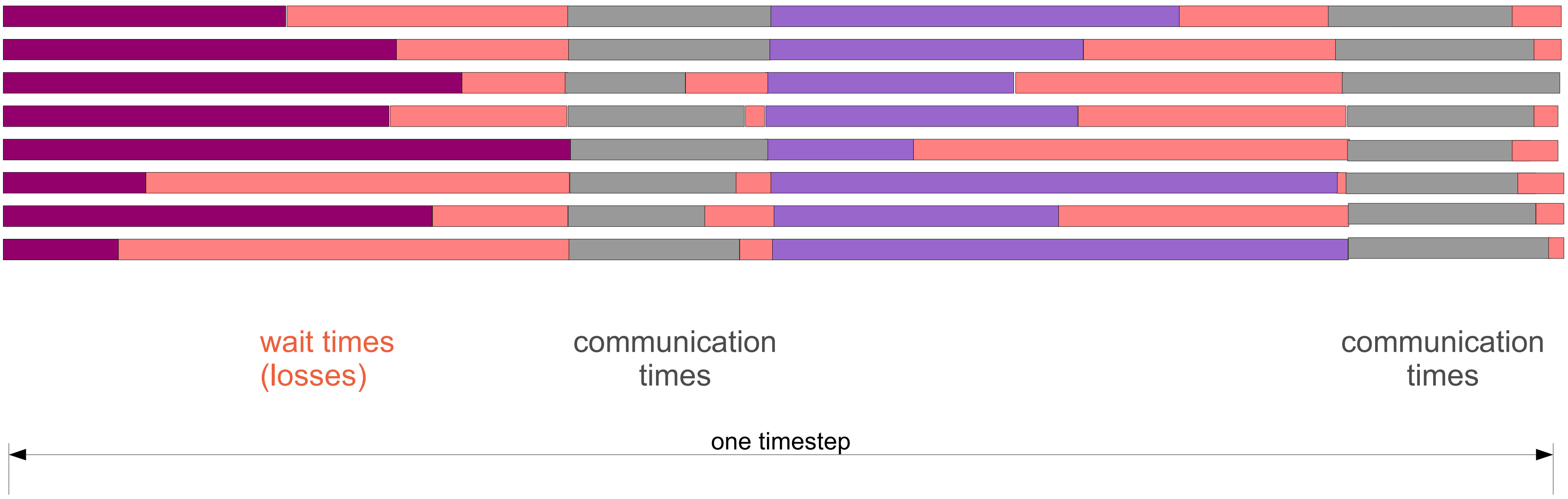
This is what actually happens once the communication step is accounted for:



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....



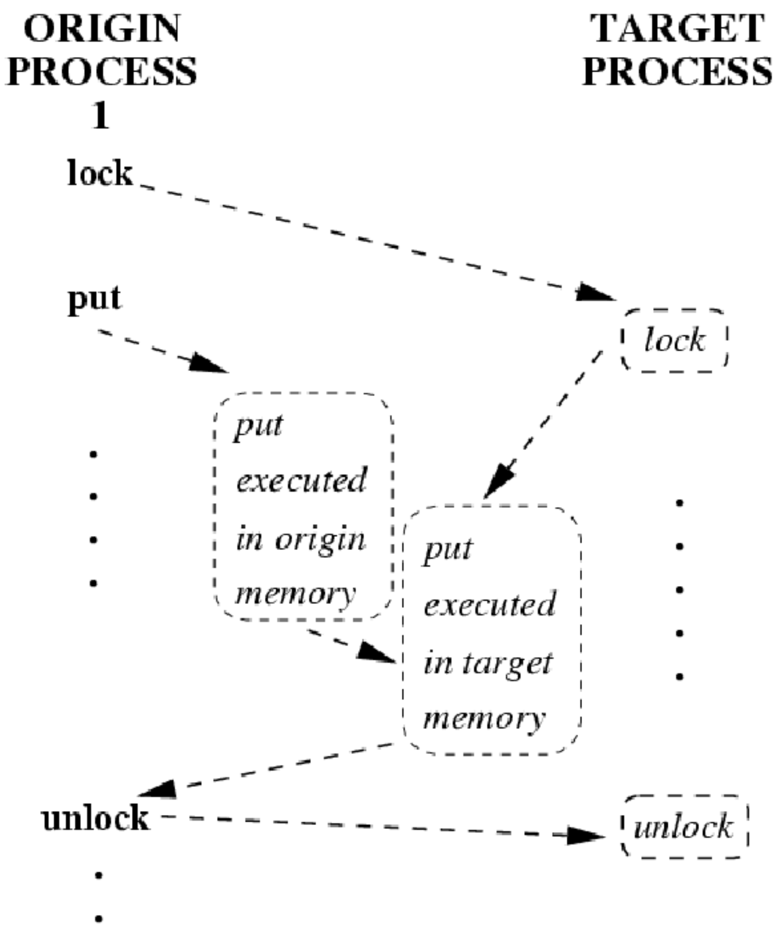
An improvement of scalability may be possible with asynchronous communication

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)



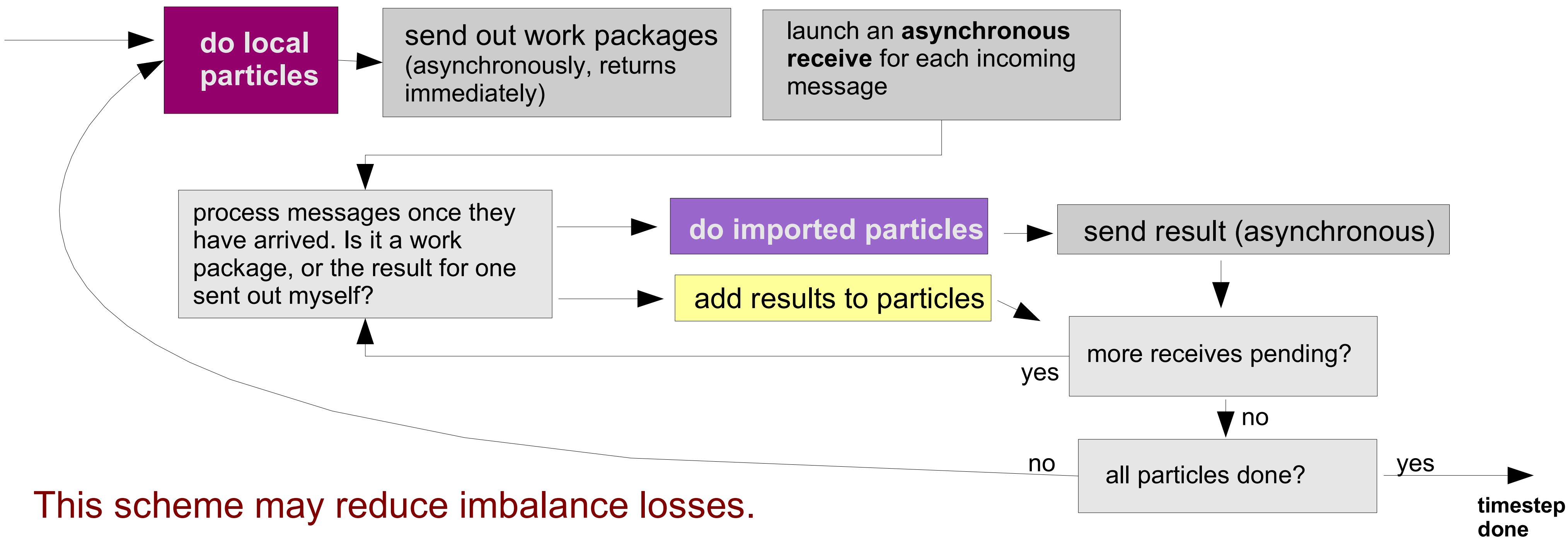
Use MPI's asynchronous two-sided communication?

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 could eliminate the mid-step imbalance losses in the gravity step

FLOW-CHART FOR ONE TIMESTEP IN AN ALTERNATIVE COMMUNICATION SCHEME



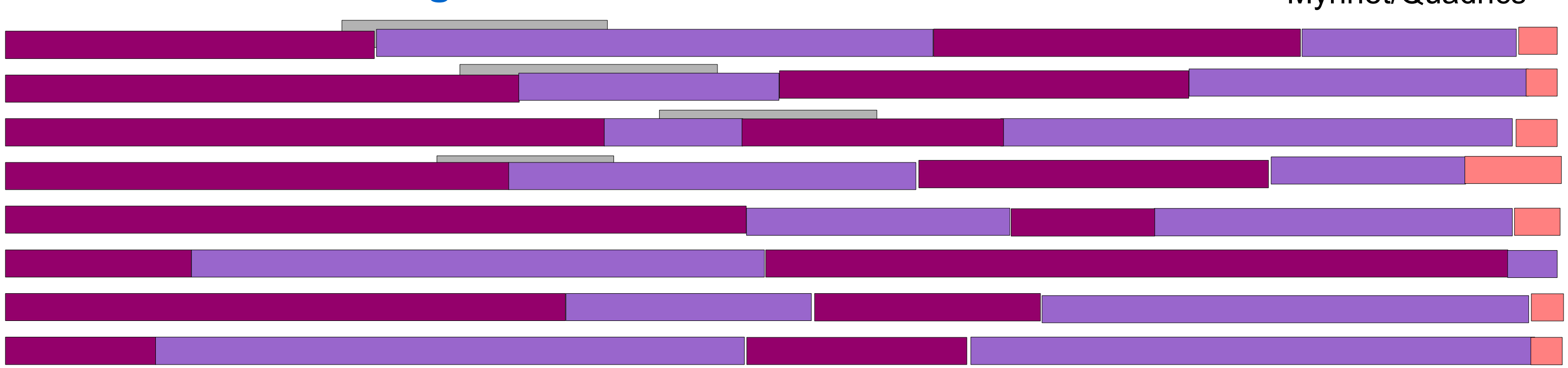
This scheme may reduce imbalance losses.

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

Overlap can be realized on:

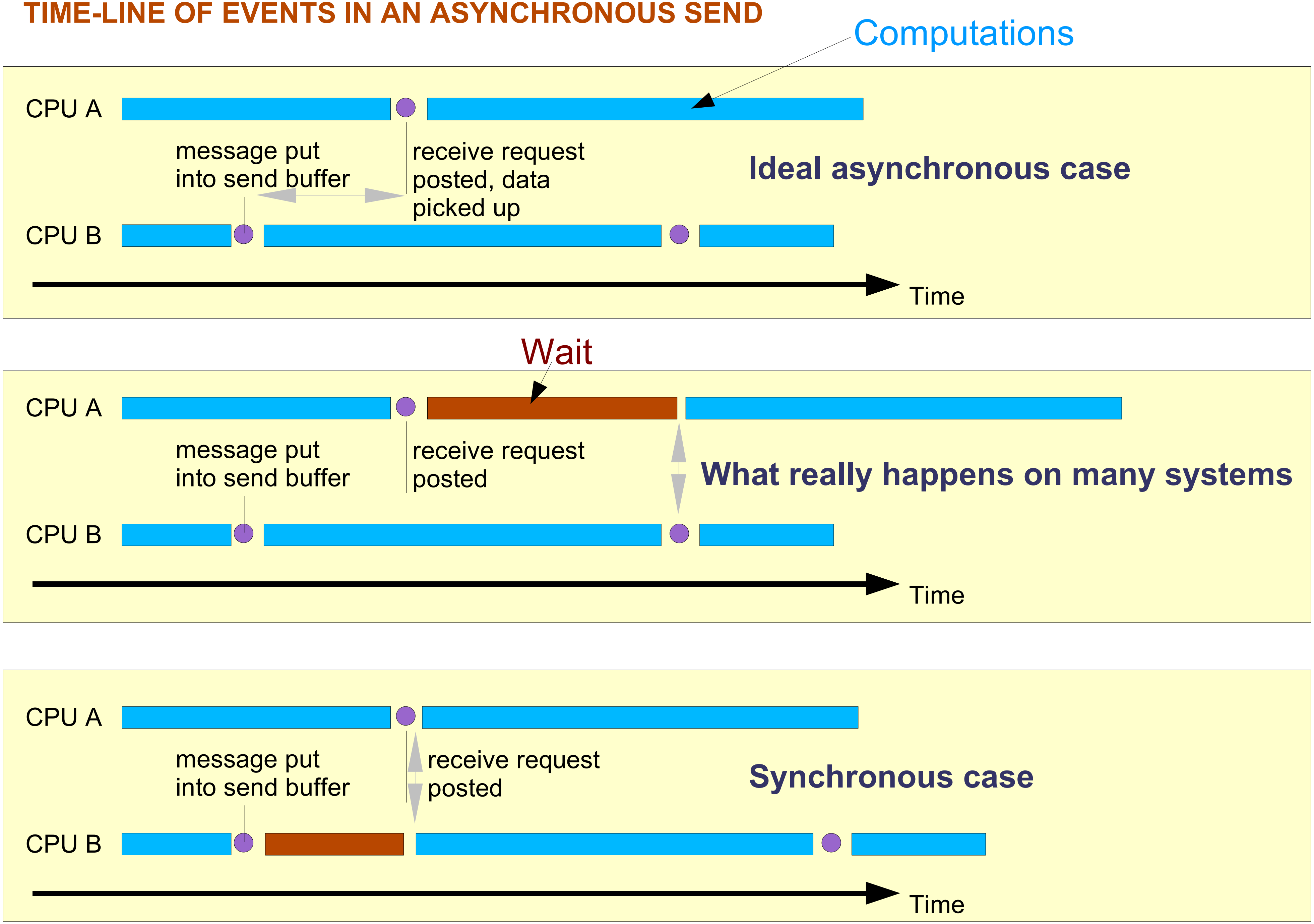
- IBM Power4
- IBM Bluegene
- Infiniband Cluster (MVAPICH)
- SMP boxes
- Myrinet/Quadrics

This scheme should give:



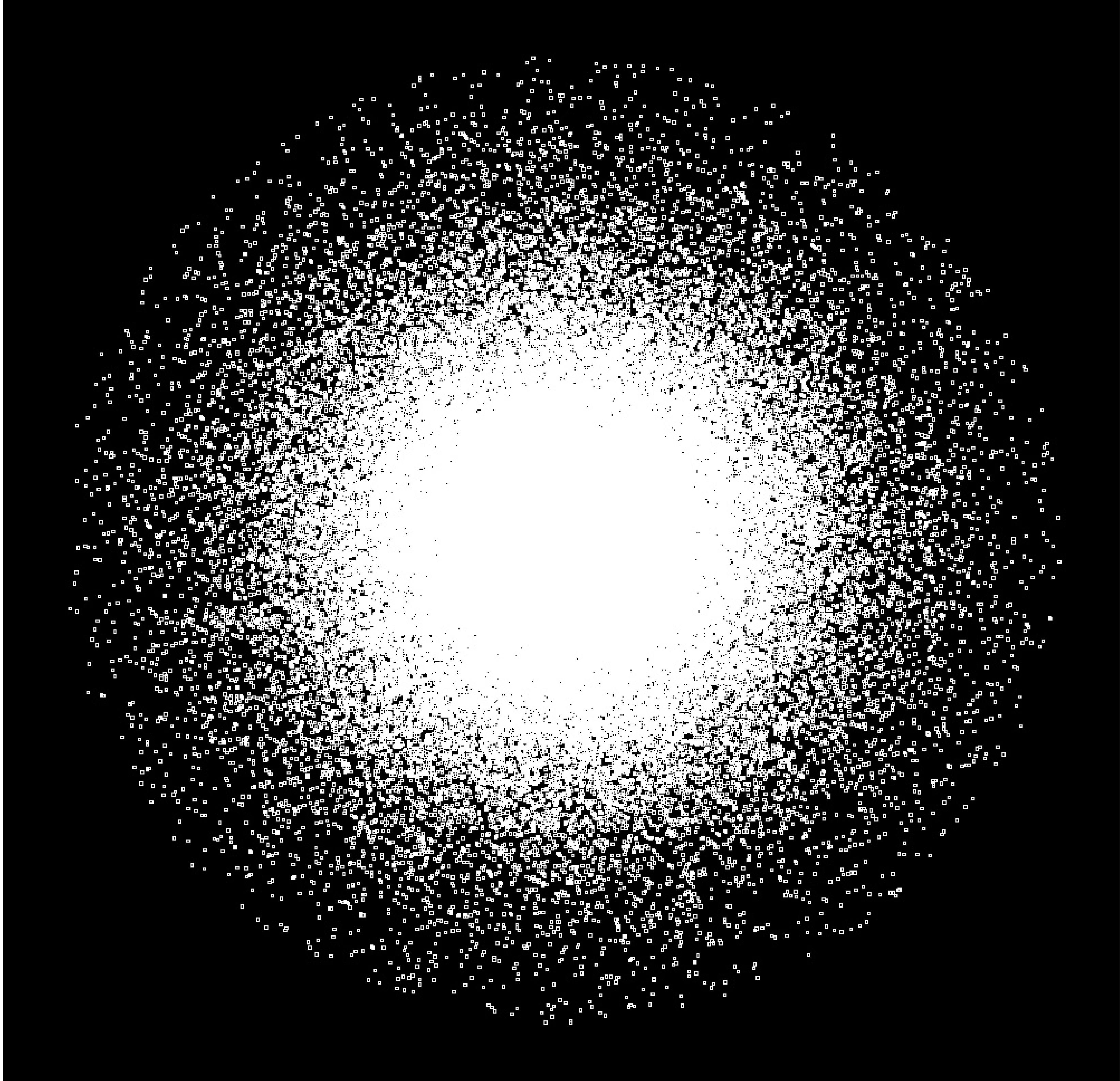
On many systems, asynchronous communication still requires a concurrent MPI call of the other process to ensure progress

TIME-LINE OF EVENTS IN AN ASYNCHRONOUS SEND



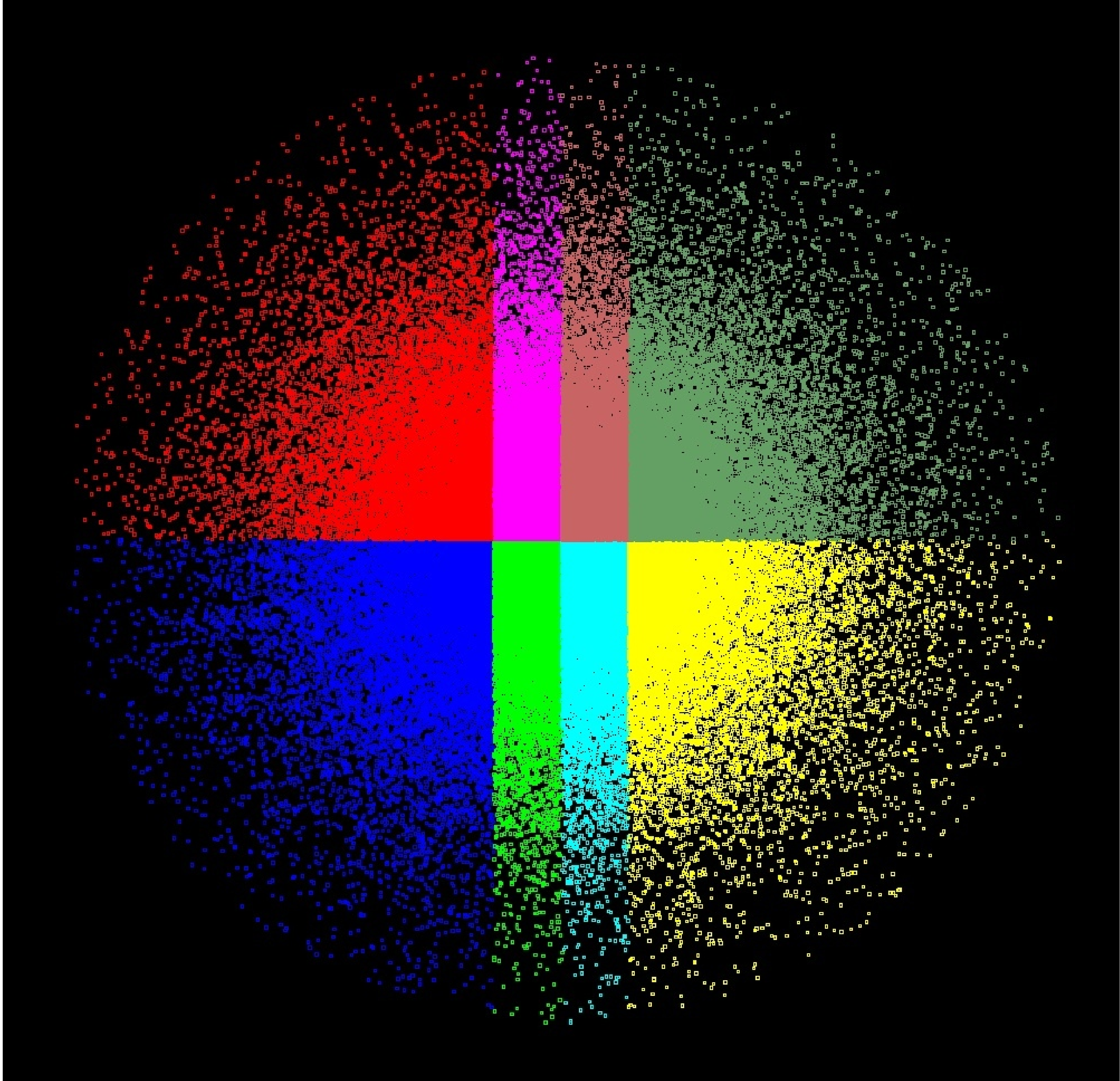
The inhomogeneous particle distribution and the different timesteps as a function of density make it challenging to find an optimum domain decomposition that balances work-load (and ideally memory-load)

PARTICLE DISTRIBUTION IN AN EXPONENTIAL DISK



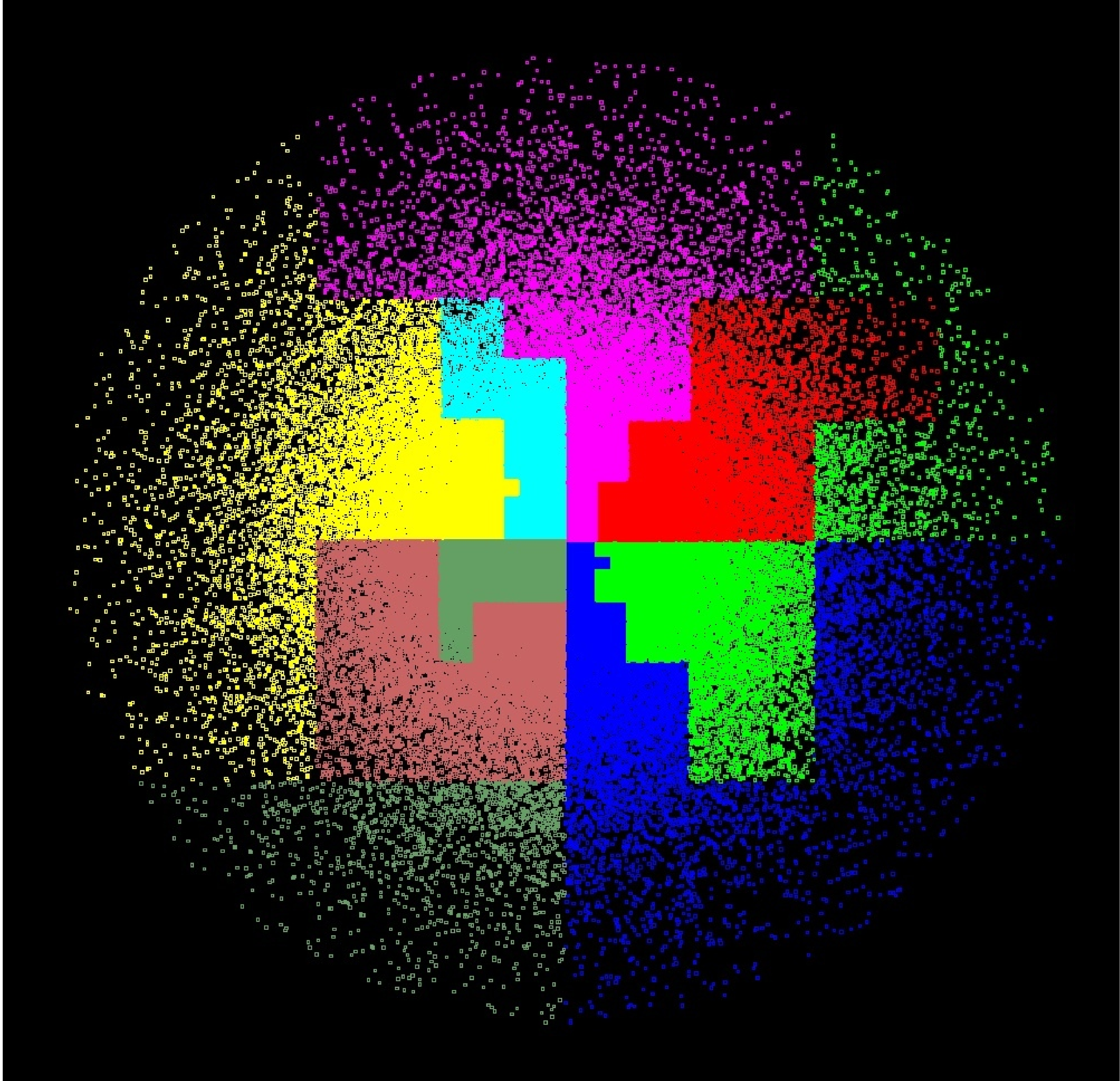
GADGET-1 used a simple orthogonal recursive bisection

EXAMPLE OF DOMAIN DECOMPOSITION IN GADGET-1



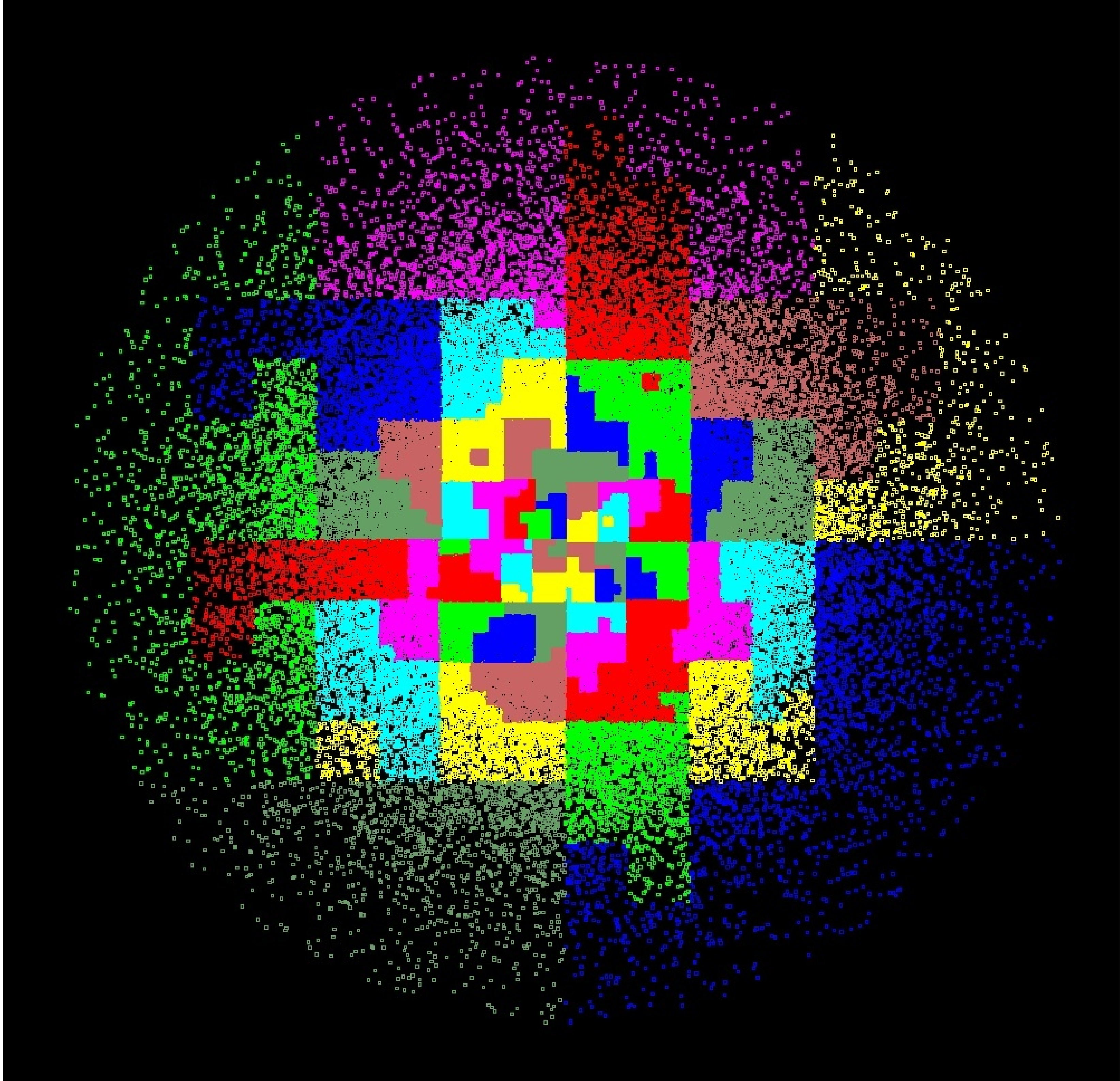
GADGET-2 uses a more flexible space-filling Peano-Hilbert curve

EXAMPLE OF DOMAIN DECOMPOSITION IN GADGET-2



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

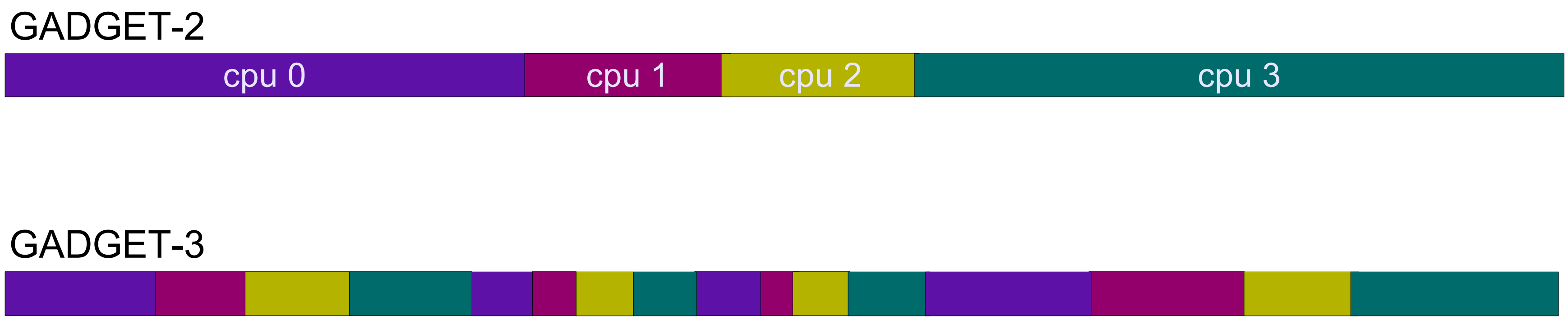
EXAMPLE OF DOMAIN DECOMPOSITION IN GADGET-3



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

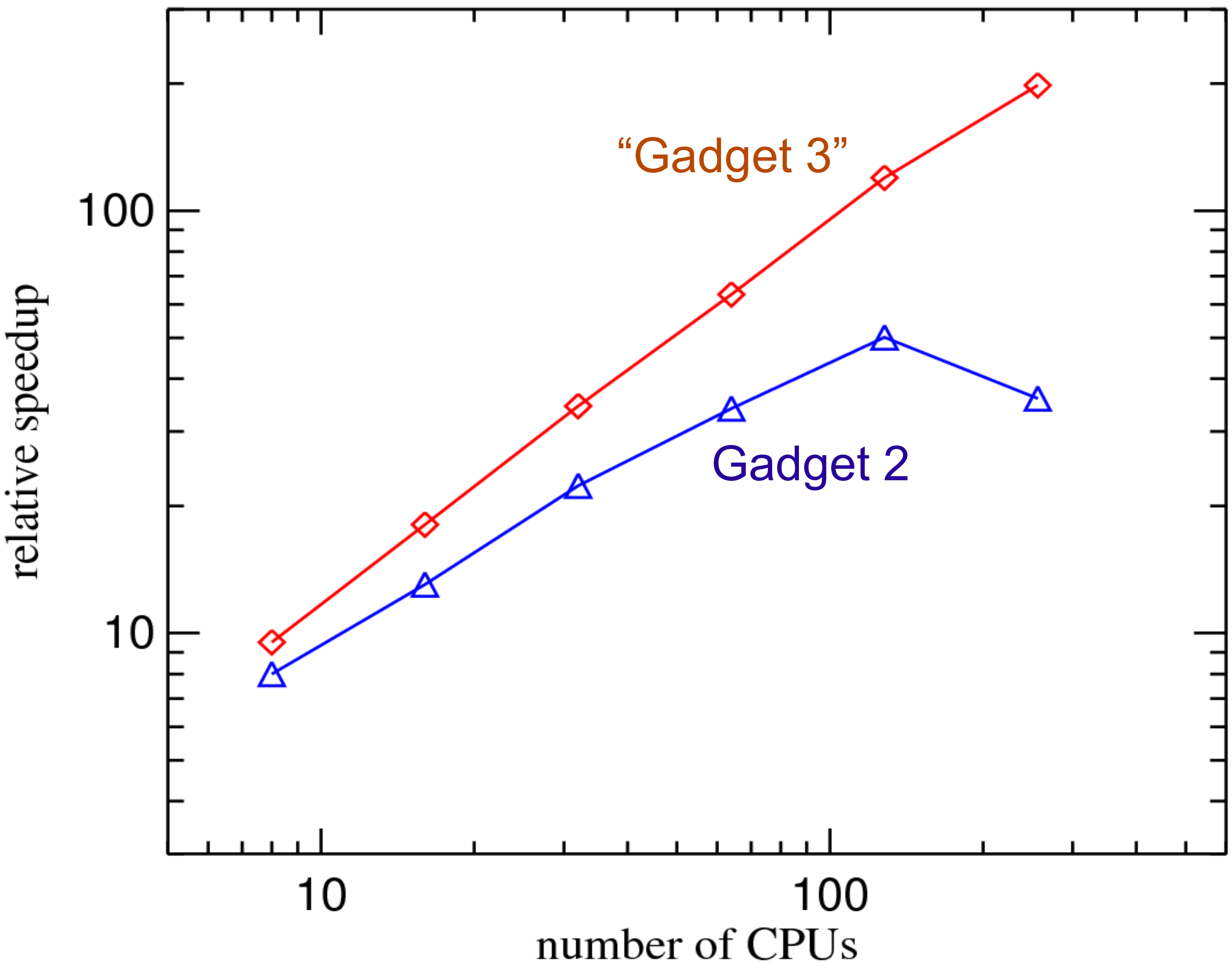


But:

- 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

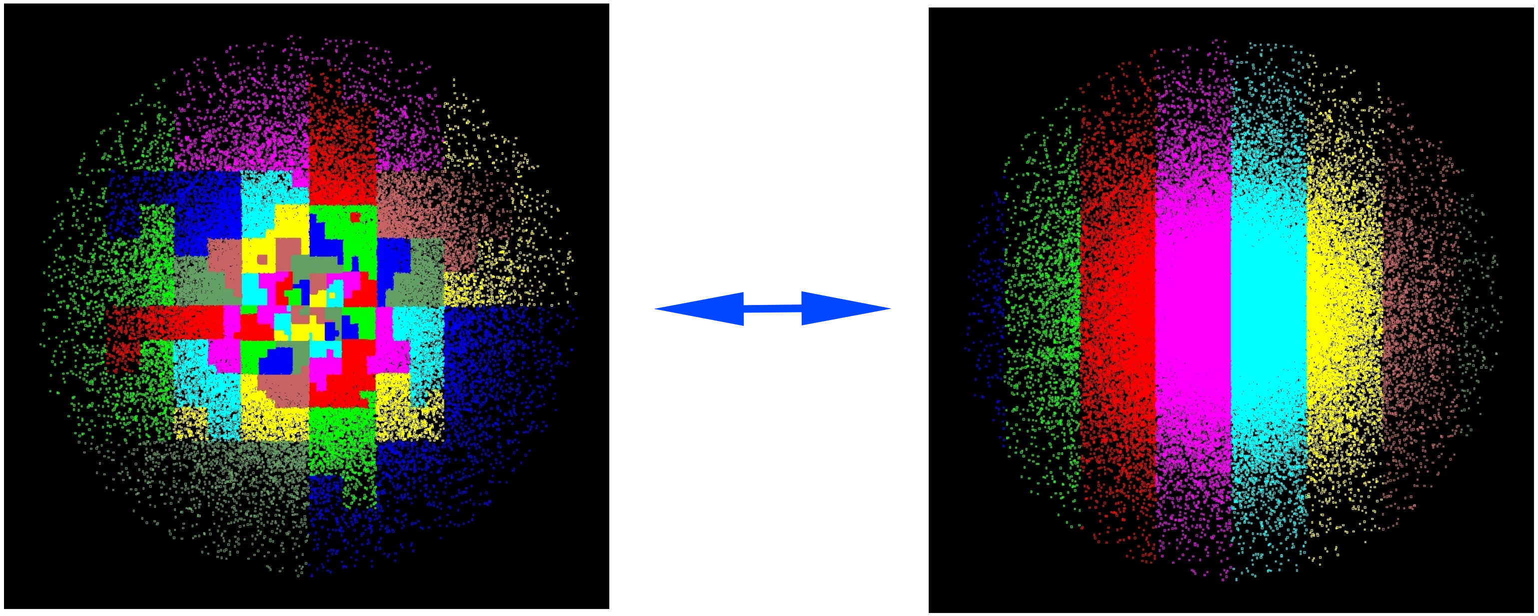
The new code scales substantially better for high-res zoom simulations of isolated halos

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

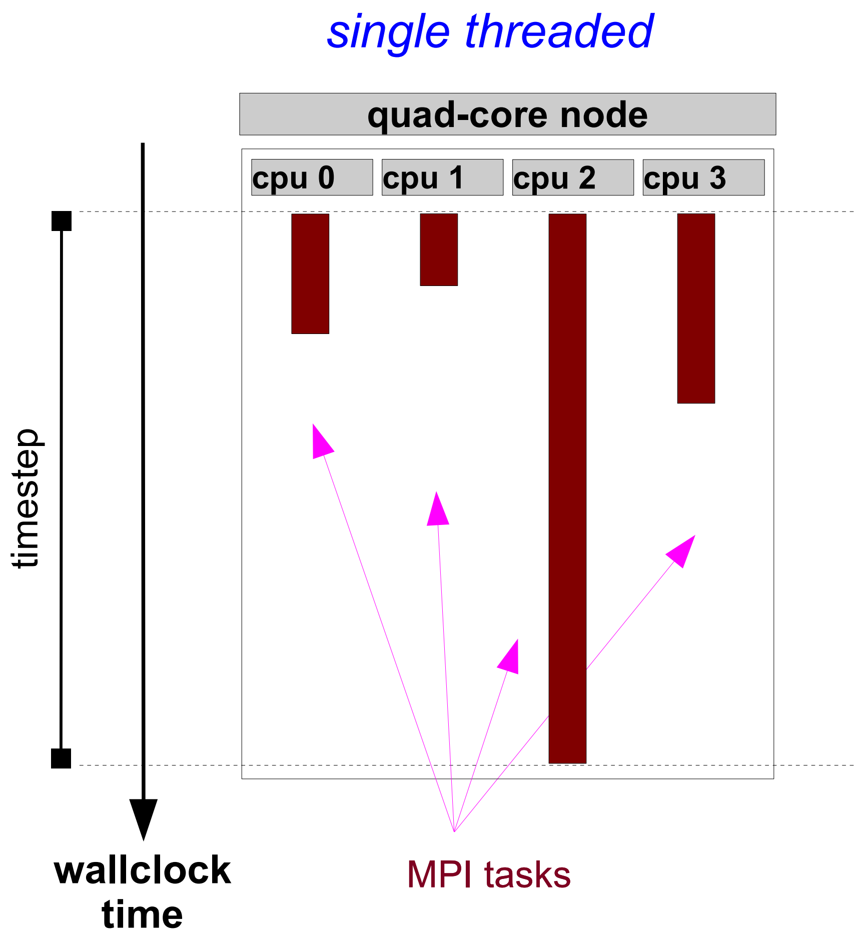


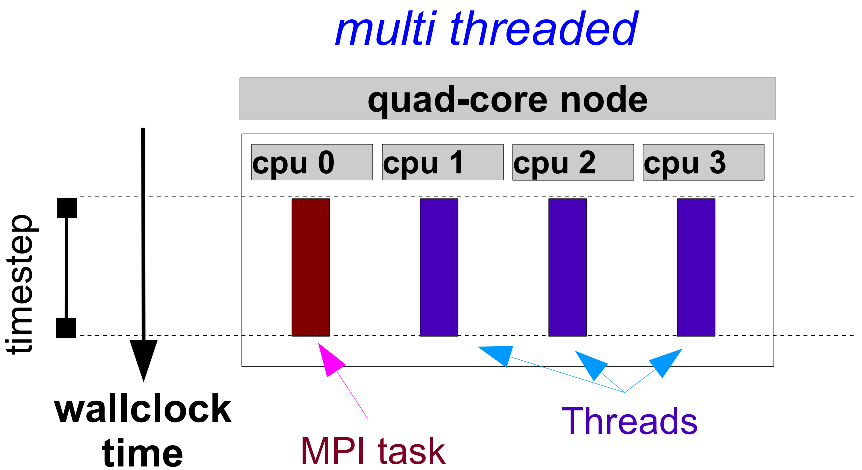
Simply swapping the particle set into a slab decomposition is in general not a good idea

- 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

Shared memory can be easily used for near perfect loop-level parallelism

USING MULTIPLE CORES WITH THREADS





- Threads are light-weight. Unlike processes, the creation/destruction takes almost no time.
- They inherit all global variables and resources (e.g. open file) from their parent process/thread.
- Mutual exclusion looks need to be used where needed to avoid race conditions.

How to get them?

- **POSIX/System-V Threads**
- **OpenMP**

GADGET-3 does now support multi-threading in combination with MPI

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.
- Mixed distributed/shared memory parallelism via MPI+Pthreads

Should be quite a bit better than the old version... and hopefully public reasonably soon.

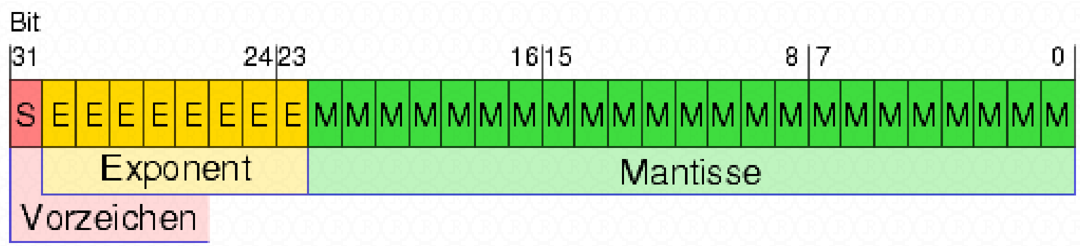
Issues of floating point accuracy

Parallelization may change the results of simulations

INTRICACIES OF FLOATING POINT ARITHMETIC

On a computer, real numbers are approximated by floating point numbers

a 32 bit float



$$v = (-1)^s \cdot (1{,}m_0 m_1 m_2 \ldots) \cdot 2^{e_0 e_1 e_2 \ldots - a}$$

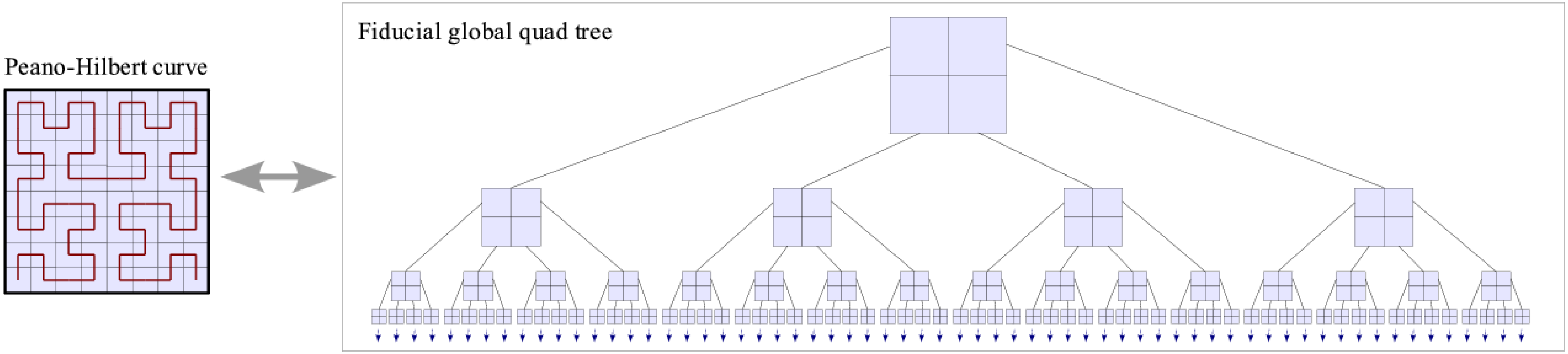
Mathematical operations regularly lead out of the space of the representable numbers. This results in **round-off** errors.

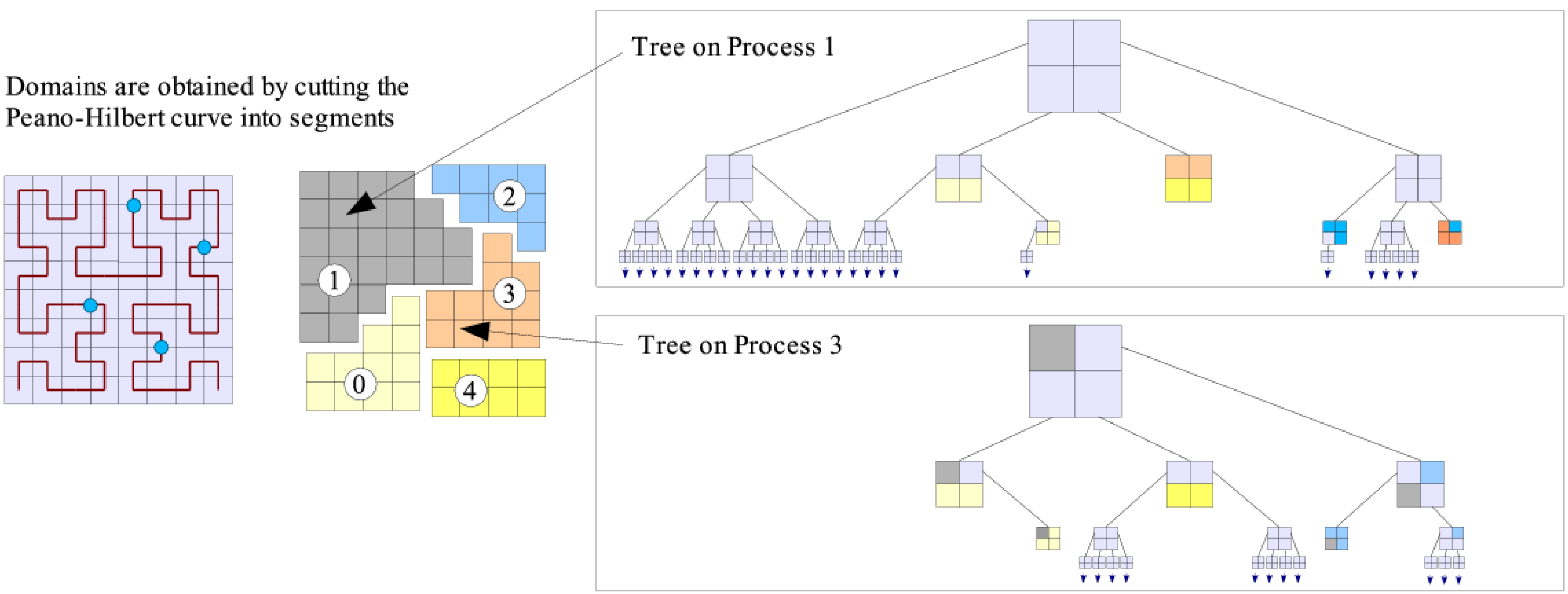
One result of this is that the law of associativity for simple additions doesn't hold on a computer.

$$A + (B + C) \neq (A + B) + C$$

In the parallelization scheme of GADGET-2, tree walks may be split up into parts that are carried out by different processors

HIERARCHICAL TREE ALGORITHMS

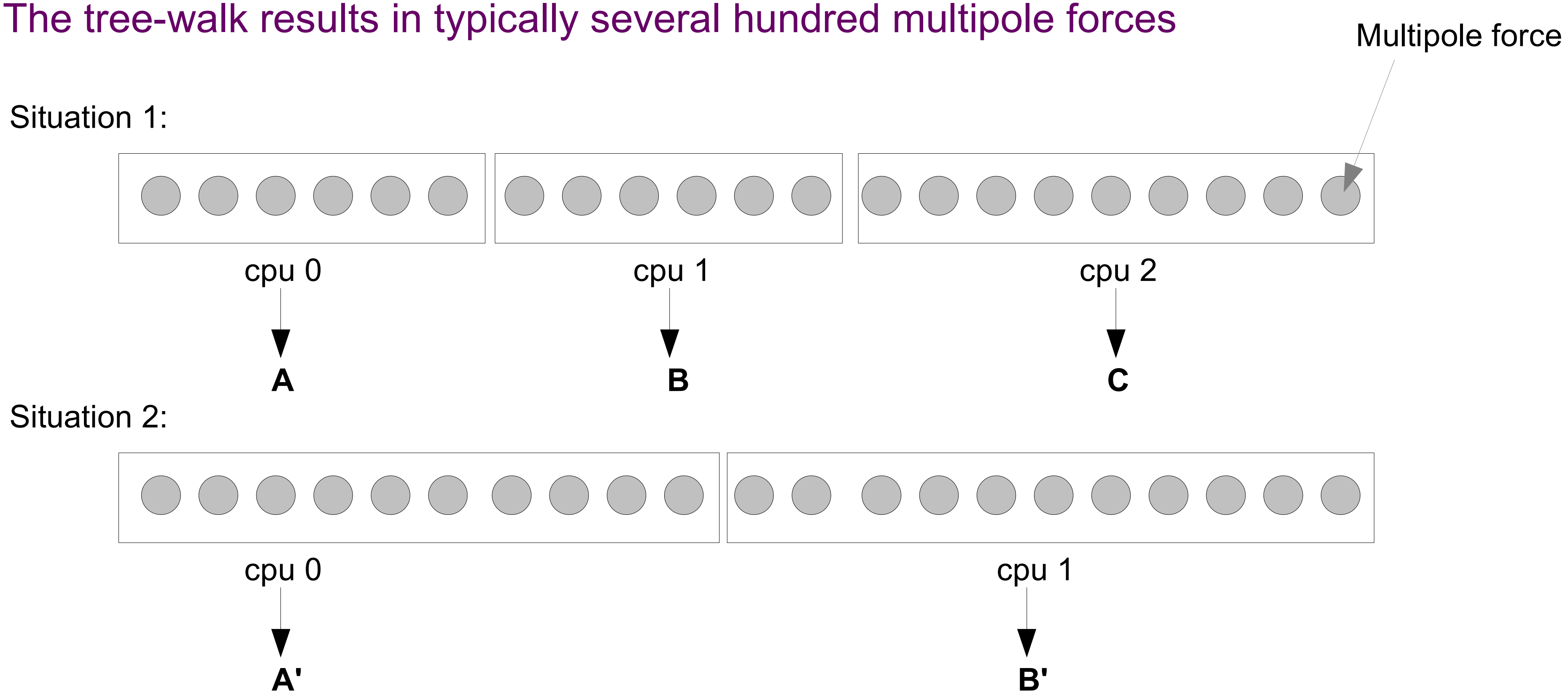




As a result of parallelization, the calculation of the force may be split to up onto different processors

THE FORCE SUM IN THE PARALLELIZED TREE ALGORITHM

The tree-walk results in typically several hundred multipole forces



When the domain decomposition is changed, round-off differences are introduced into the results

$$A + B + C \neq A' + B'$$

Consequences of round-off errors in collisionless systems

THE LIMITED RELEVANCE OF INDIVIDUAL PARTICLE ORBITS

As the systems are typically **chaotic**, small perturbations are quickly amplified.

- Since in tree codes the force errors *discontinuously* depend on the particle coordinates, small differences from round-off can be boosted in one step from machine epsilon to the order of the typical average force error.
- Changes in the number of processors modifies round-off errors in the forces of particles. Hence the final result of runs carried out on different numbers of processors may not be binary identical.
- Changing the compiler or its optimizer settings will also introduce differences in collisionless simulations.

Convergence in collisionless simulations can not be achieved on a particle-by-particle basis.

However, the **collective statistical properties** of the systems **do converge.**

Individual particles are noisy tracers of the dynamics!

Basics of SPH

The governing equations of an *ideal* gas can also be written in **Lagrangian form**

BASIC HYDRODYNAMICAL EQUATIONS

Euler equation:

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = -\frac{\nabla P}{\rho} - \nabla\Phi$$

Continuity equation:

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} + \rho\nabla\cdot\mathbf{v} = 0$$

First law of thermodynamics:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -\frac{P}{\rho}\nabla\cdot\mathbf{v} - \frac{\Lambda(u,\rho)}{\rho}$$

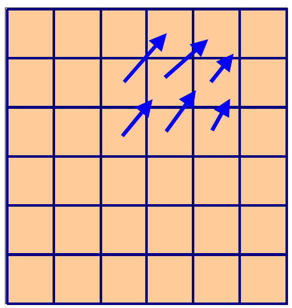
Equation of state of an ideal monoatomic gas:

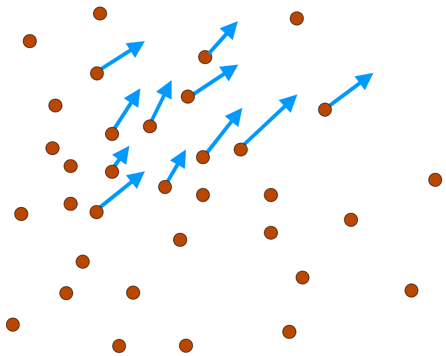
$$P = (\gamma - 1)\rho u\,, \qquad \gamma = 5/3$$

What is smoothed particle hydrodynamics?

DIFFERENT METHODS TO DISCRETIZE A FLUID

Eulerian	Lagrangian
discretize space	**discretize mass**
representation on a mesh (volume elements)	representation by fluid elements (particles)



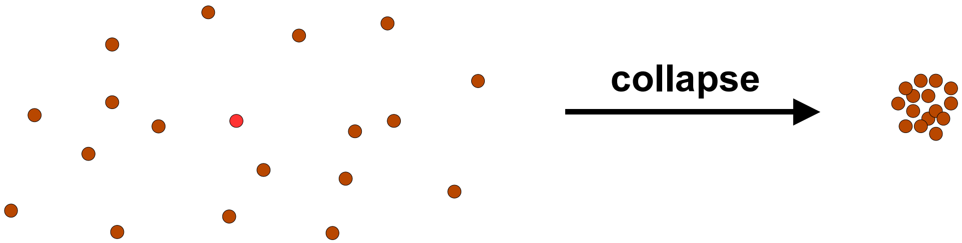


principle advantage:

high accuracy (shock capturing), low numerical viscosity

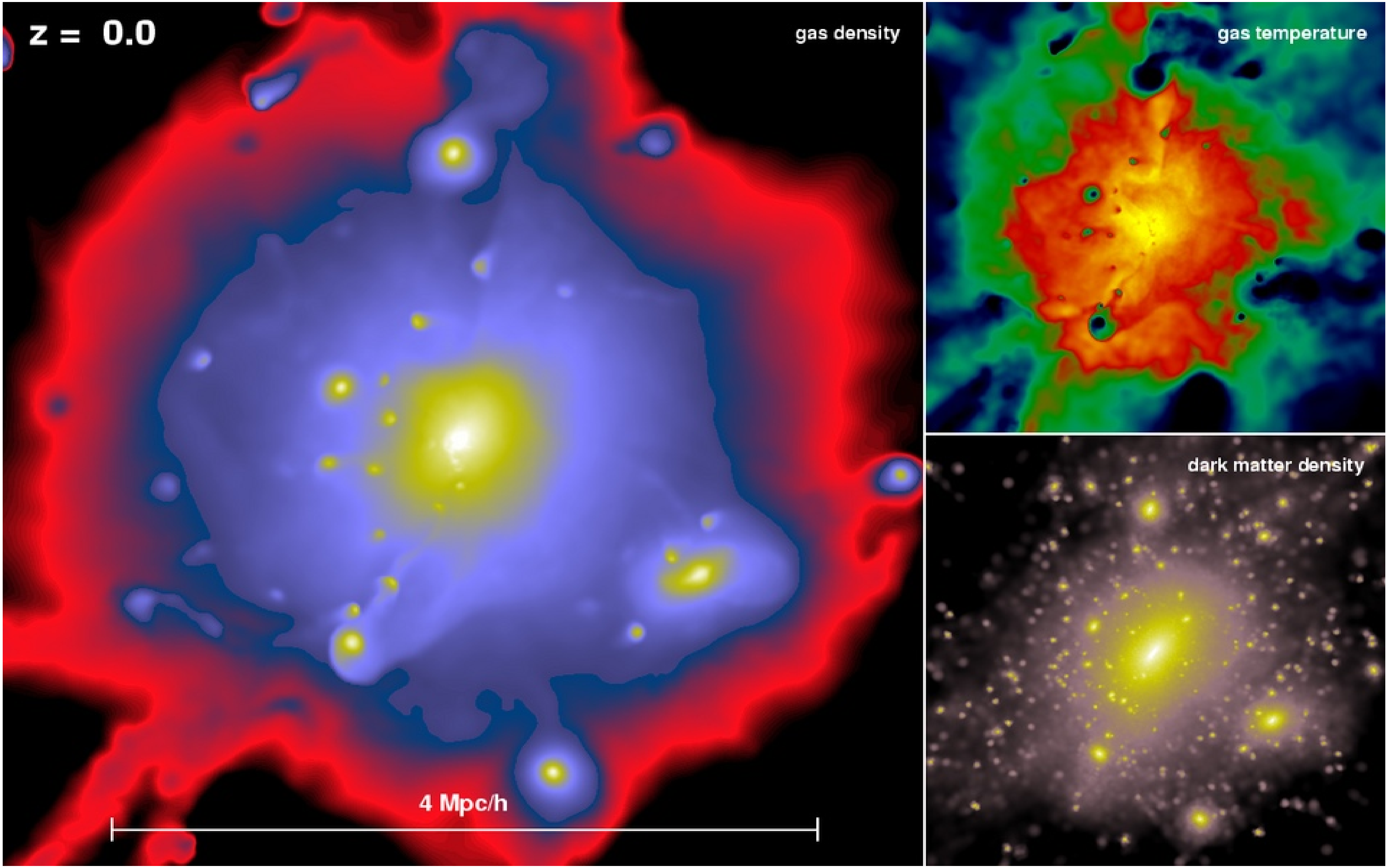
principle advantage:

resolutions adjusts automatically to the flow



SPH can be readily combined with collisionless simulations of dark matter

A SIMULATED CLUSTER WITH GAS

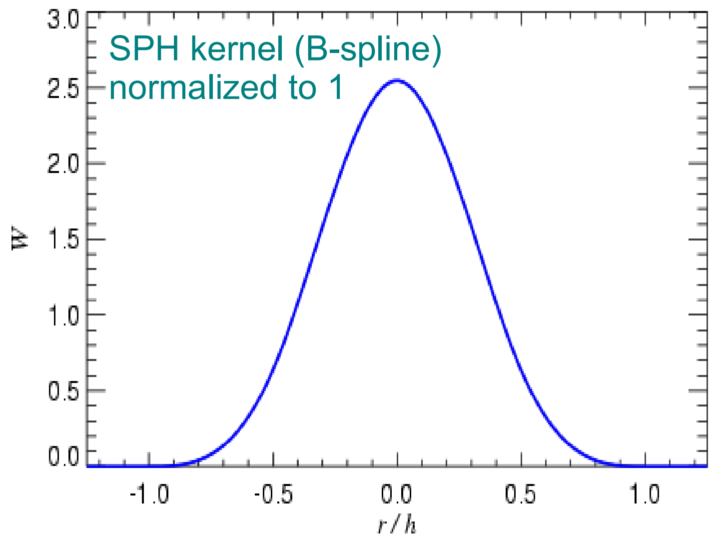


Kernel interpolation is used in smoothed particle hydrodynamics to build continuous fluid quantities from discrete tracer particles

DENSITY ESTIMATION IN SPH BY MEANS OF ADAPTIVE KERNEL ESTIMATION

Kernel interpolant of an arbitrary function:

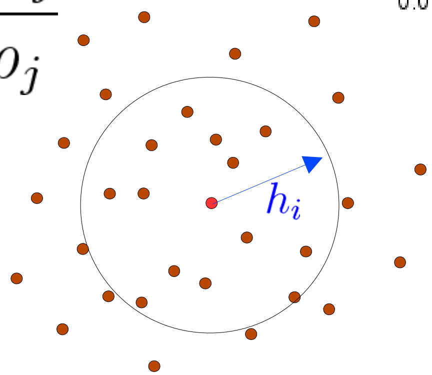
$$\langle A(\mathbf{r})\rangle = \int W(\mathbf{r}-\mathbf{r}', h)\, A(\mathbf{r}')\, \mathrm{d}^3 r'$$



If the function is only known at a set of discrete points, we approximate the integral as a sum, using the replacement:

$$\mathrm{d}^3 r' \mapsto \frac{m_j}{\rho_j}$$

$$\langle A_i\rangle = \sum_{j=1}^{N} \frac{m_j}{\rho_j} A_j\, W(\mathbf{r}_{ij}; h_i)$$



This leads to the SPH density estimate, for $A_i = \rho_i$

$$\rho_i = \sum_{j=1}^{N} m_j W(|\mathbf{r}_{ij}|, h_i)$$

→ **This can be differentiated !**

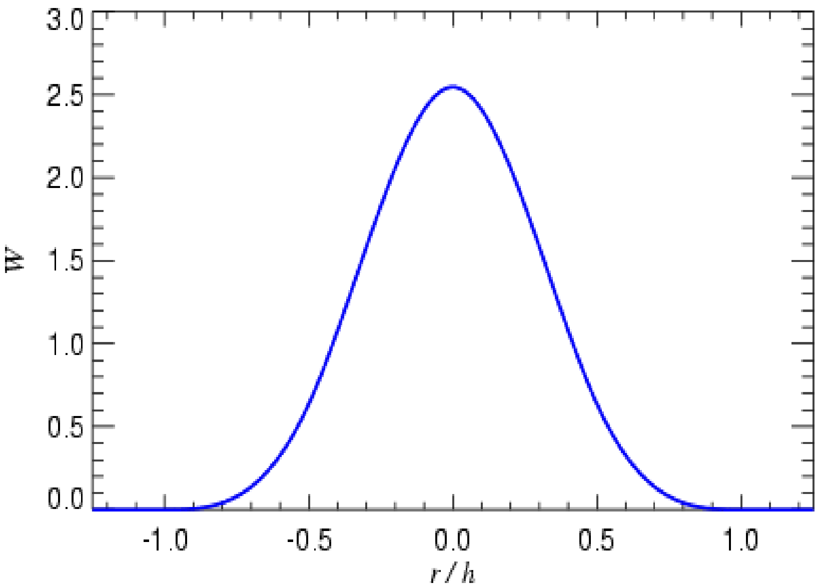
Good kernel shapes need to fulfill a number of constraints

CONDITIONS ON KERNELS

- Must be normalized to unity
- Compact support (otherwise N^2 bottleneck)
- High order of interpolation
- Spherical symmetry (for angular momentum conservation)

Nowadays, almost exclusively the cubic spline is used:

$$W(u) = \frac{8}{\pi}\begin{cases} 1-6u^2+6u^3, & 0 \le u \le \frac{1}{2}, \\ 2(1-u)^3, & \frac{1}{2} < u \le 1, \\ 0, & u > 1. \end{cases}$$



Kernel interpolants allow the construction of derivatives from a set of discrete tracer points

EXAMPLES FOR ESTIMATING THE VELOCITY DIVERGENCE

Smoothed estimate for the velocity field:

$$\langle \mathbf{v}_i \rangle = \sum_j \frac{m_j}{\rho_j} \mathbf{v}_j \, W(\mathbf{r}_i - \mathbf{r}_j)$$

Velocity divergence can now be readily estimated:

$$\nabla \cdot \mathbf{v} = \nabla \cdot \langle \mathbf{v}_i \rangle = \sum_j \frac{m_j}{\rho_j} \mathbf{v}_j \, \nabla_i W(\mathbf{r}_i - \mathbf{r}_j)$$

But alternative (and better) estimates are possible also:

Invoking the identity

$$\rho \nabla \cdot \mathbf{v} = \nabla \cdot (\rho \mathbf{v}) - \mathbf{v} \cdot \nabla \rho$$

one gets a “pair-wise” formula:

$$\rho_i (\nabla \cdot \mathbf{v})_i = \sum_j m_j (\mathbf{v}_j - \mathbf{v}_i) \, \nabla_i W(\mathbf{r}_i - \mathbf{r}_j)$$

Smoothed particle hydrodynamics is governed by a set of ordinary differential equations

BASIC EQUATIONS OF SMOOTHED PARTICLE HYDRODYNAMICS

Each particle carries either the energy or the entropy per unit mass as independent variable

Density estimate

$$\rho_i = \sum_{j=1}^{N} m_j W(|\mathbf{r}_{ij}|, h_i)$$

→ **Continuity equation automatically fulfilled.**

→ $$P_i = (\gamma - 1)\rho_i u_i$$

Euler equation

$$\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\sum_{j=1}^{N} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla_i \overline{W}_{ij}$$

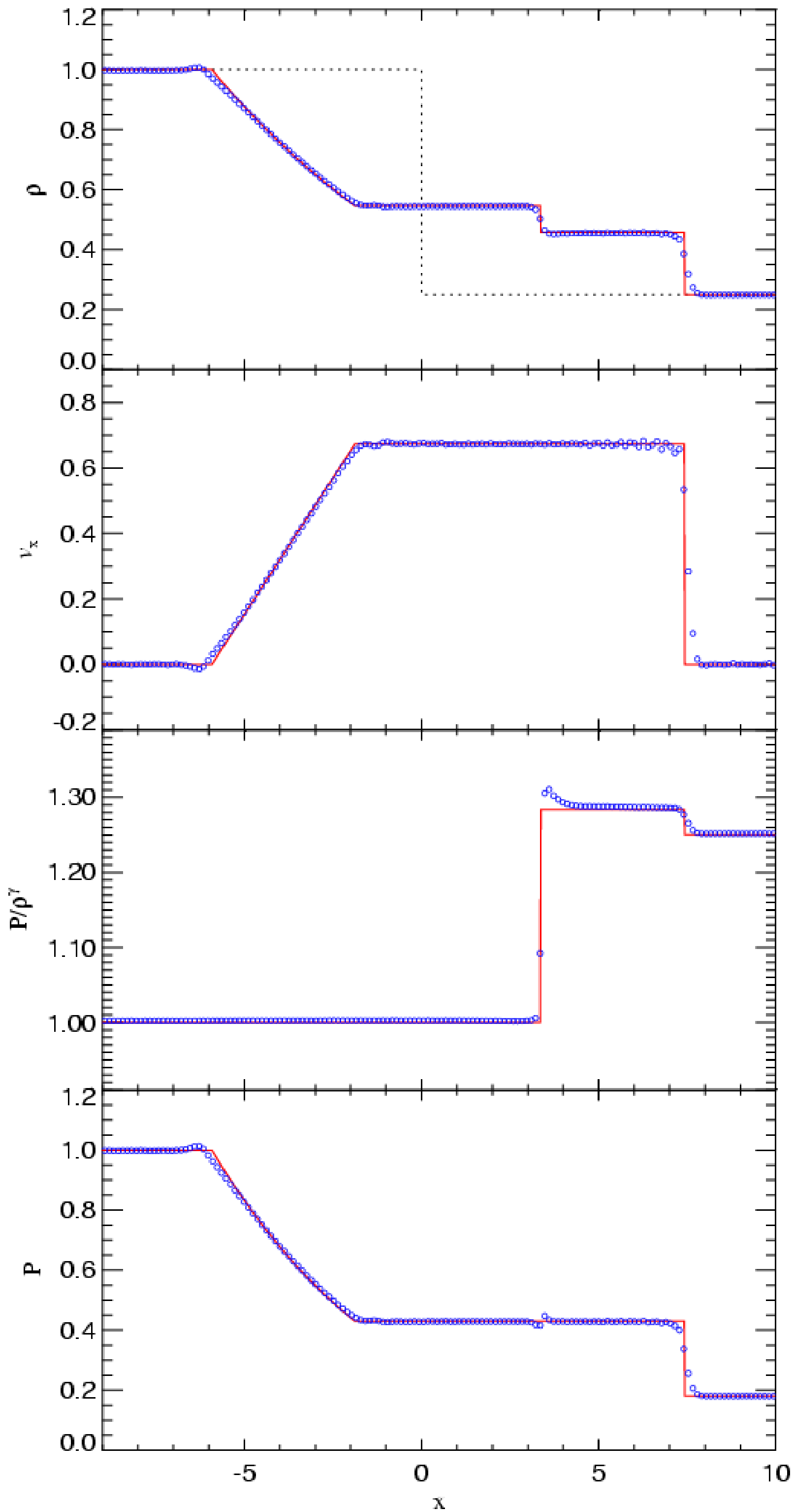
$+\Pi_{ij}$ **Artificial viscosity**

First law of thermodynamics

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = \frac{1}{2}\sum_{j=1}^{N} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \mathbf{v}_{ij} \cdot \nabla_i \overline{W}_{ij}$$

$+\Pi_{ij}$

Viscosity and shock capturing



An artificial viscosity needs to be introduced to capture shocks

SHOCK TUBE PROBLEM AND VISCOSITY

viscous force:

$$\left.\frac{d\mathbf{v}_i}{dt}\right|_{\rm visc} = -\sum_{j=1}^{N} m_j \Pi_{ij} \nabla_i \overline{W}_{ij}$$

parameterization of the artificial viscosity:

$$\Pi_{ij} = \begin{cases} -\frac{\alpha}{2}\frac{[c_i+c_j-3w_{ij}]w_{ij}}{\rho_{ij}} & \text{if } \mathbf{v}_{ij}\cdot\mathbf{r}_{ij} < 0 \\ 0 & \text{otherwise} \end{cases}$$

$$v_{ij}^{\rm sig} = c_i + c_j - 3w_{ij},$$

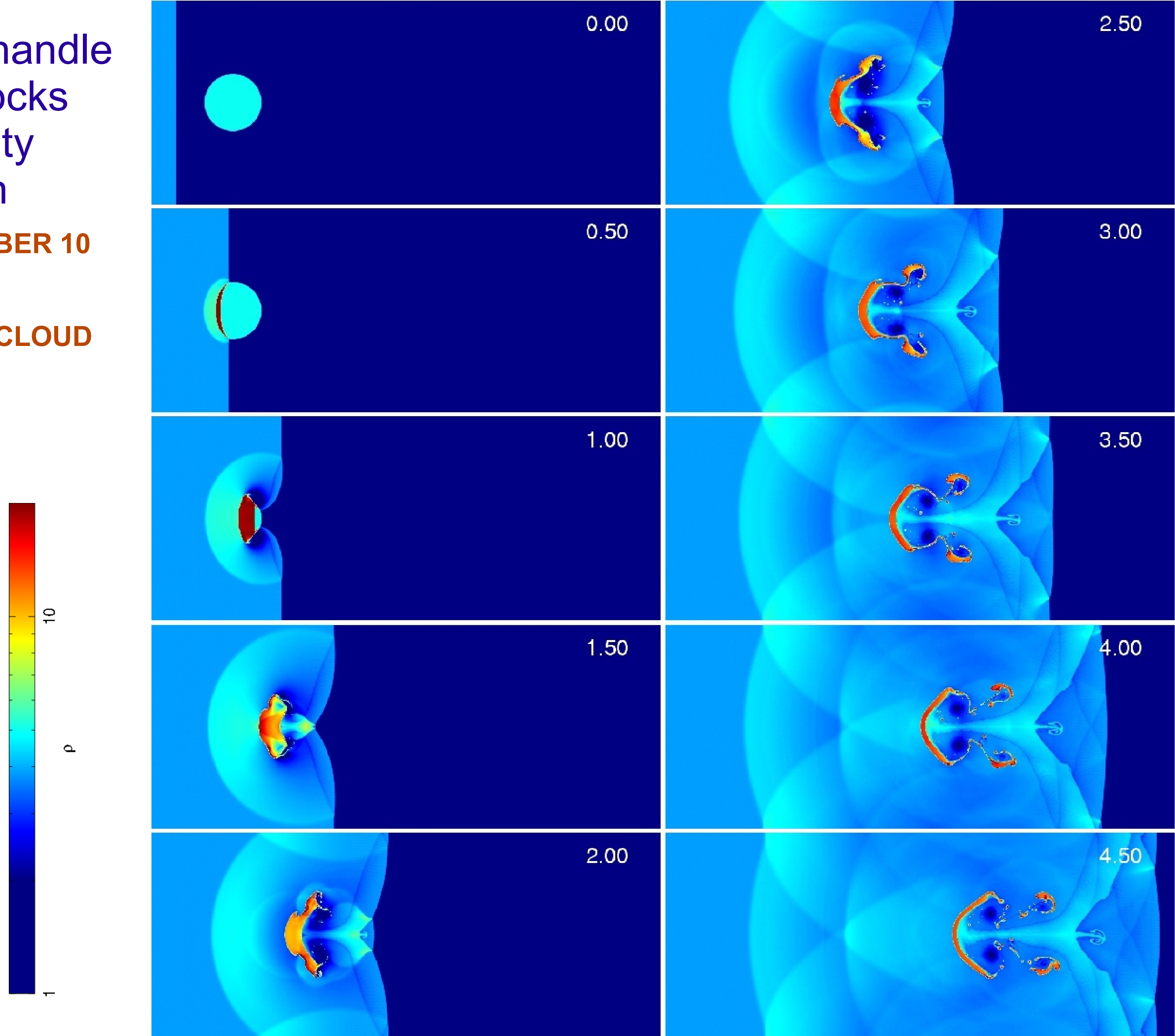
$$w_{ij} = \mathbf{v}_{ij}\cdot\mathbf{r}_{ij}/|\mathbf{r}_{ij}|$$

heat production rate:

$$\frac{du_i}{dt} = \frac{1}{2}\sum_{j=1}^{N} m_j \Pi_{ij} \mathbf{v}_{ij}\cdot\nabla_i \overline{W}_{ij}$$

SPH can handle strong shocks and vorticity generation

A MACH NUMBER 10 SHOCK THAT STRIKES AN OVERDENSE CLOUD



SPH accurately conserves all relevant conserved quantities in self-gravitating flows

SOME NICE PROPERTIES OF SPH

- ★ **Mass is conserved**
- ★ **Momentum is conserved**
- ★ **Total energy is conserved – also in the presence of self-gravity !**
- ★ **Angular momentum is conserved**
- ★ **Entropy is conserved – only produced by artificial viscosity, no entropy production due to mixing or advection**

Furthermore:

- ★ **High geometric flexibility**
- ★ **Easy incorporation of vacuum boundary conditions**
- ★ **No high Mach number problem**

Variational derivation of SPH

The traditional way to derive the SPH equations leaves room for many different formulations

SYMMETRIZATION CHOICES

Symmetrized kernel:

$$\overline{W}_{ij} = W\left(|\mathbf{r}_{ij}|, [h_i + h_j]/2\right)$$

$$\overline{W}_{ij} = \frac{1}{2}\left[W\left(|\mathbf{r}_{ij}|, h_i\right) + W\left(|\mathbf{r}_{ij}|, h_j\right)\right]$$

Symmetrization of pressure terms:

Using $\nabla P = 2\sqrt{P}\nabla\sqrt{P}$

$$\frac{1}{2}\left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}\right) \Longleftrightarrow \sqrt{\frac{P_i\, P_j}{\rho_i^2\, \rho_j^2}}$$

Is there a best choice?

For an adiabatic flow, temperature can be derived from the specific entropy

ENTROPY FORMALISM

Definition of an **entropic function**:

$$P_i = A_i \, \rho_i^{\gamma}$$

for an adiabtic flow:

$$A_i = A_i(s_i) = \mathrm{const.}$$

don't integrate the temperature, but infer it from:

$$u_i = \frac{A_i}{\gamma - 1} \rho^{\gamma - 1}$$

Use an artificial viscosity to generate entropy in shocks:

$$\frac{\mathrm{d}A_i}{\mathrm{d}t} = \frac{1}{2} \frac{\gamma - 1}{\rho_i^{\gamma - 1}} \sum_{j=1}^{N} m_j \Pi_{ij} \mathbf{v}_{ij} \cdot \nabla_i \overline{W}_{ij}$$

None of the adaptive SPH schemes conserves energy and entropy simultaneously

CONSERVATION LAW TROUBLES

Hernquist (1993):

If the **thermal energy** is **integrated**, **entropy** conservation can be **violated**...

If the **entropy** is **integrated**, total **energy** is **not** necessarily **conserved**...

The trouble is caused by varying smoothing lengths... ∇h -terms

Do we have to worry about this? **YES**

Can we do better? **YES**

A fully conservative formulation of SPH

Springel & Hernquist (2002)

DERIVATION

Lagrangian:

$$L(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\sum_{i=1}^{N} m_i \dot{\mathbf{r}}_i^2 - \frac{1}{\gamma - 1}\sum_{i=1}^{N} m_i A_i \rho_i^{\gamma-1}$$

$$\mathbf{q} = (\mathbf{r}_1, \ldots, \mathbf{r}_N, h_1, \ldots, h_N)$$

Constraints:

$$\phi_i(\mathbf{q}) \equiv \frac{4\pi}{3} h_i^3 \rho_i - M_{\mathrm{sph}} = 0$$

Equations of motion:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \sum_{j=1}^{N} \lambda_j \frac{\partial \phi_j}{\partial q_i}$$

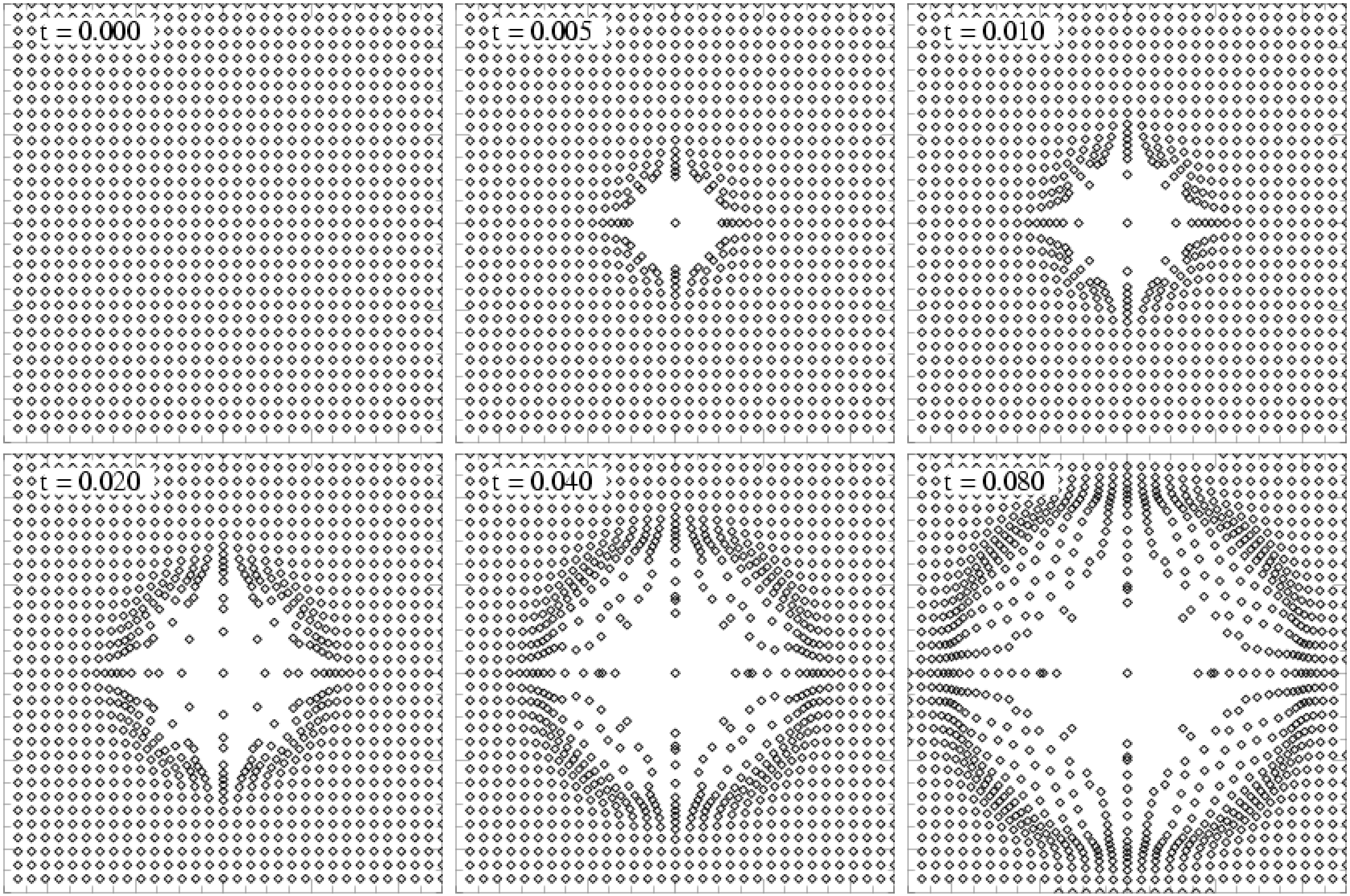
$$\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\sum_{j=1}^{N} m_j \left[f_i \frac{P_i}{\rho_i^2} \nabla_i W_{ij}(h_i) + f_j \frac{P_j}{\rho_j^2} \nabla_i W_{ij}(h_j) \right]$$

$$f_i = \left[1 + \frac{h_i}{3\rho_i} \frac{\partial \rho_i}{\partial h_i} \right]^{-1}$$

Does the entropy formulation give better results?

A point-explosion in three-dimensional SPH

TAYLOR-SEDOV BLAST

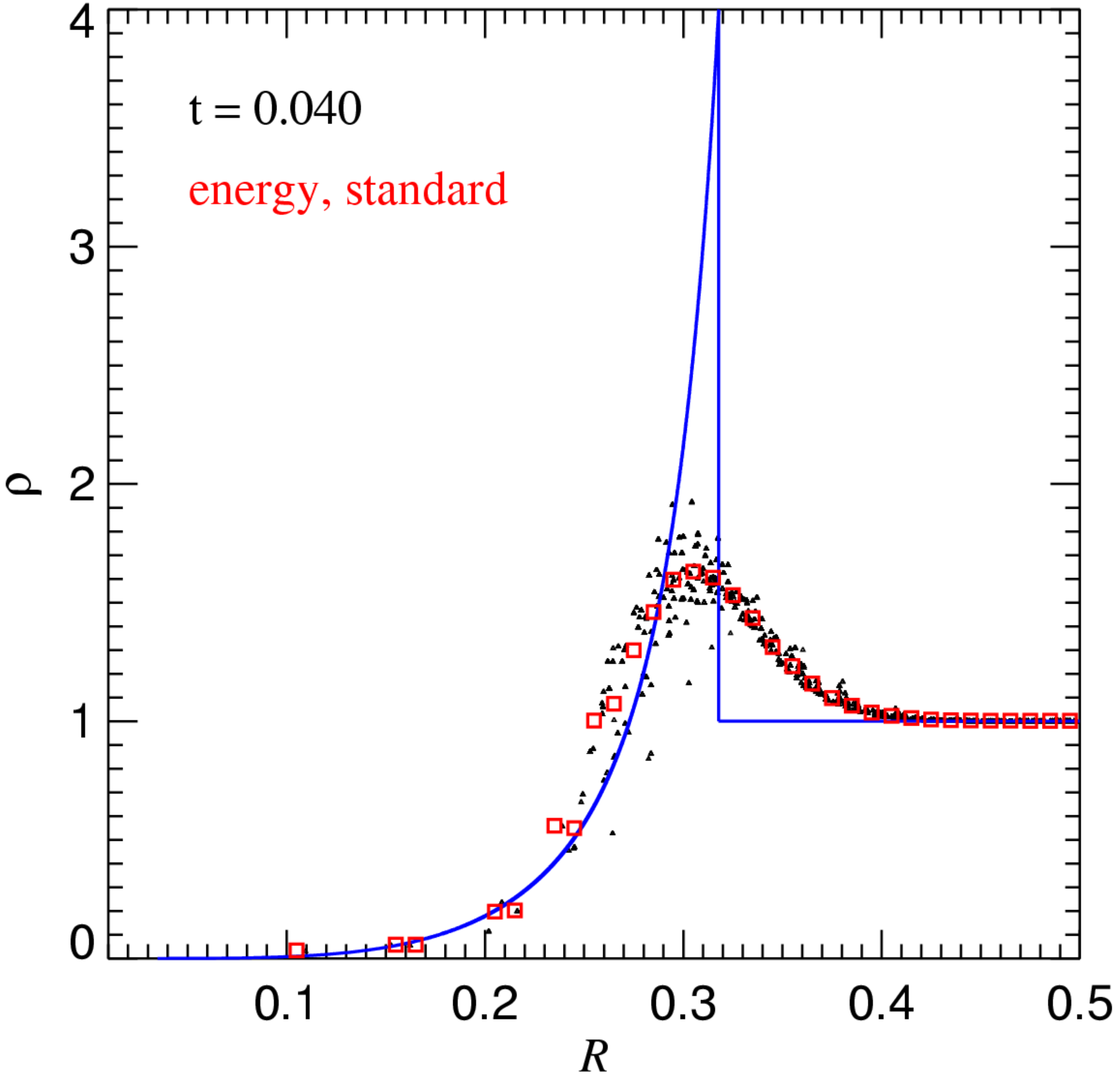


- Geometric formulation gives completely unphysical result (no explosion at all)
- Standard energy formulation produces severe error in total energy, but asymmetric form ok
- Standard entropy formulation ok, but energy fluctuates by several percent

There is a well-known similarity solution for strong point-like explosions

SEDOV-TAYLOR SOLUTIONS FOR **SMOOTHED** EXPLOSION ENERGY

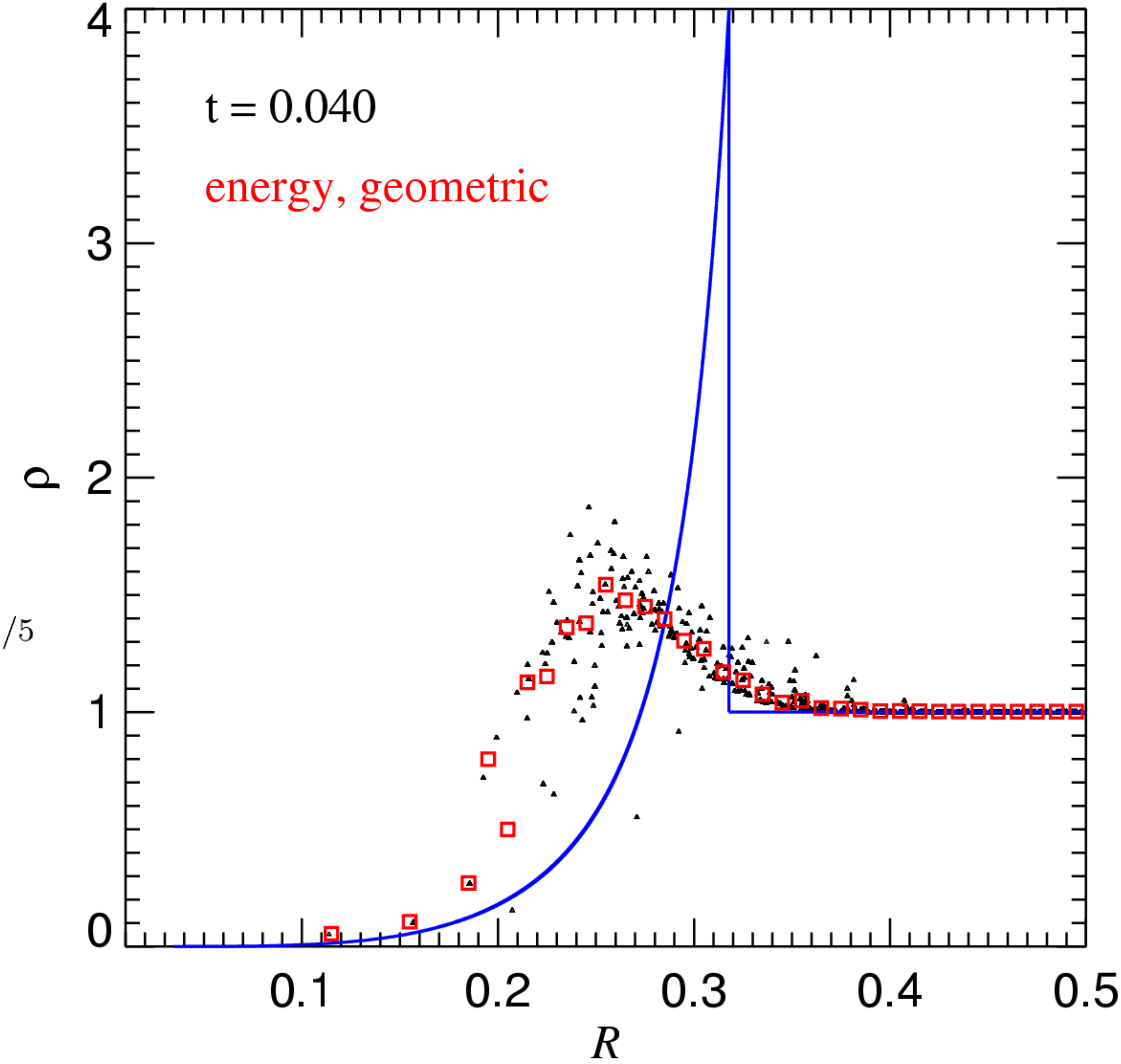
$$R(t) = \beta \left(\frac{E t^2}{\rho} \right)^{1/5}$$



There is a well-known similarity solution for strong point-like explosions

SEDOV-TAYLOR SOLUTIONS FOR **SMOOTHED** EXPLOSION ENERGY

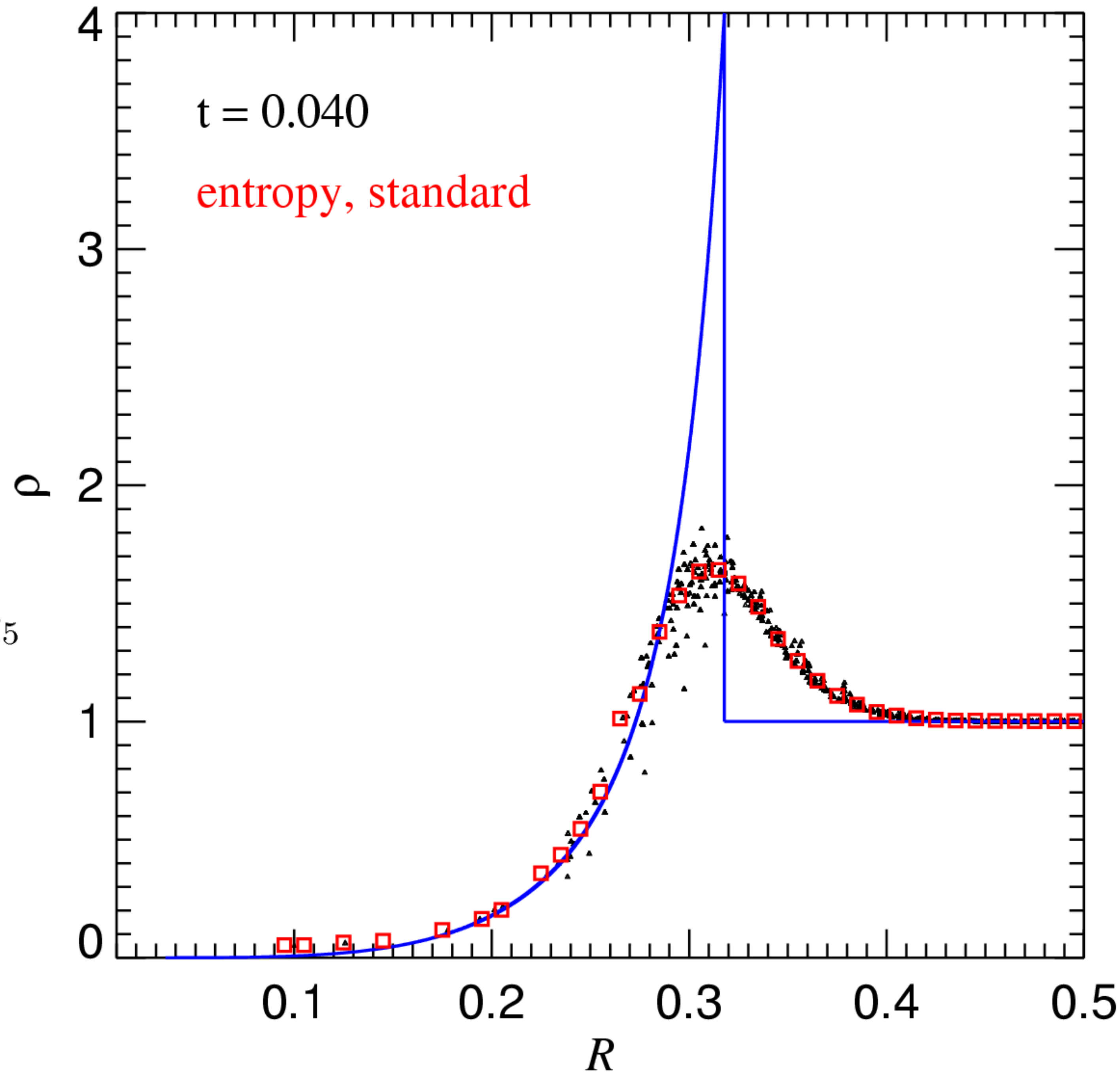
$$R(t) = \beta \left(\frac{E t^2}{\rho} \right)^{1/5}$$



There is a well-known similarity solution for strong point-like explosions

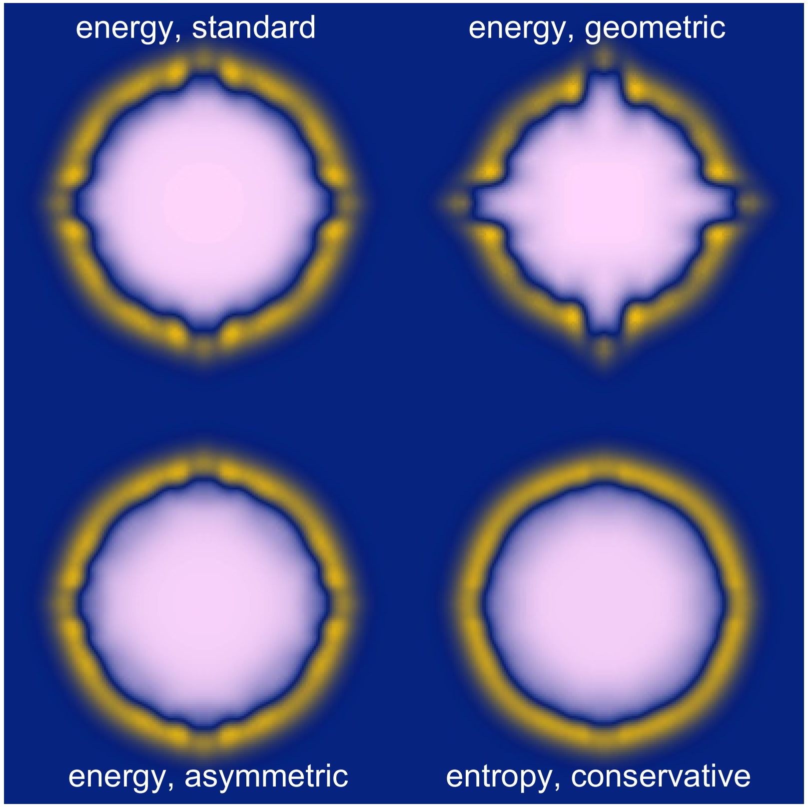
SEDOV-TAYLOR SOLUTIONS FOR **SMOOTHED** EXPLOSION ENERGY

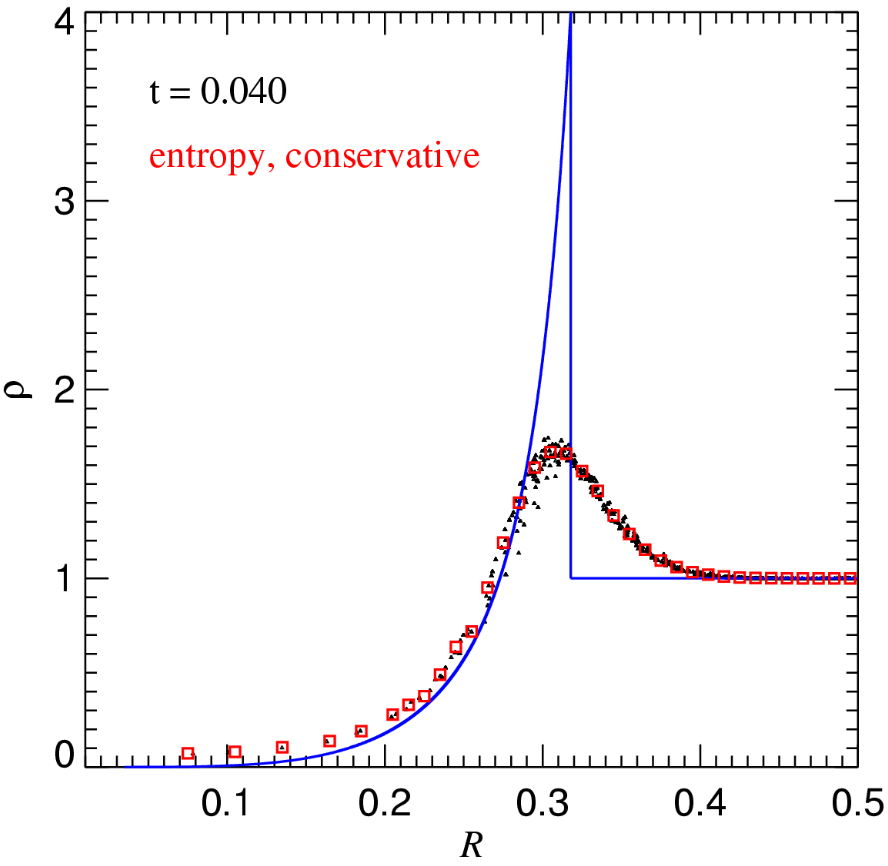
$$R(t) = \beta \left(\frac{E t^2}{\rho} \right)^{1/5}$$



The new conservative formulation gives better results for adiabtic flows

EXPLOSION PROBLEM

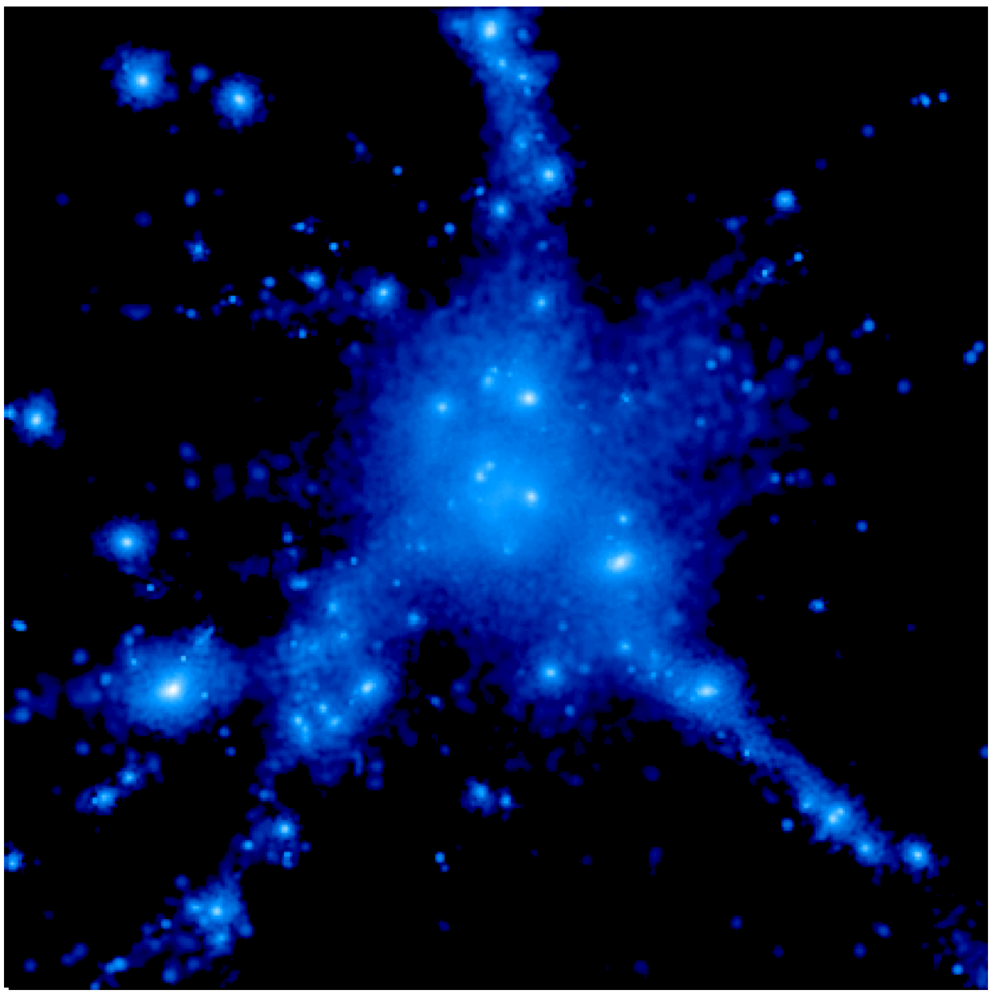




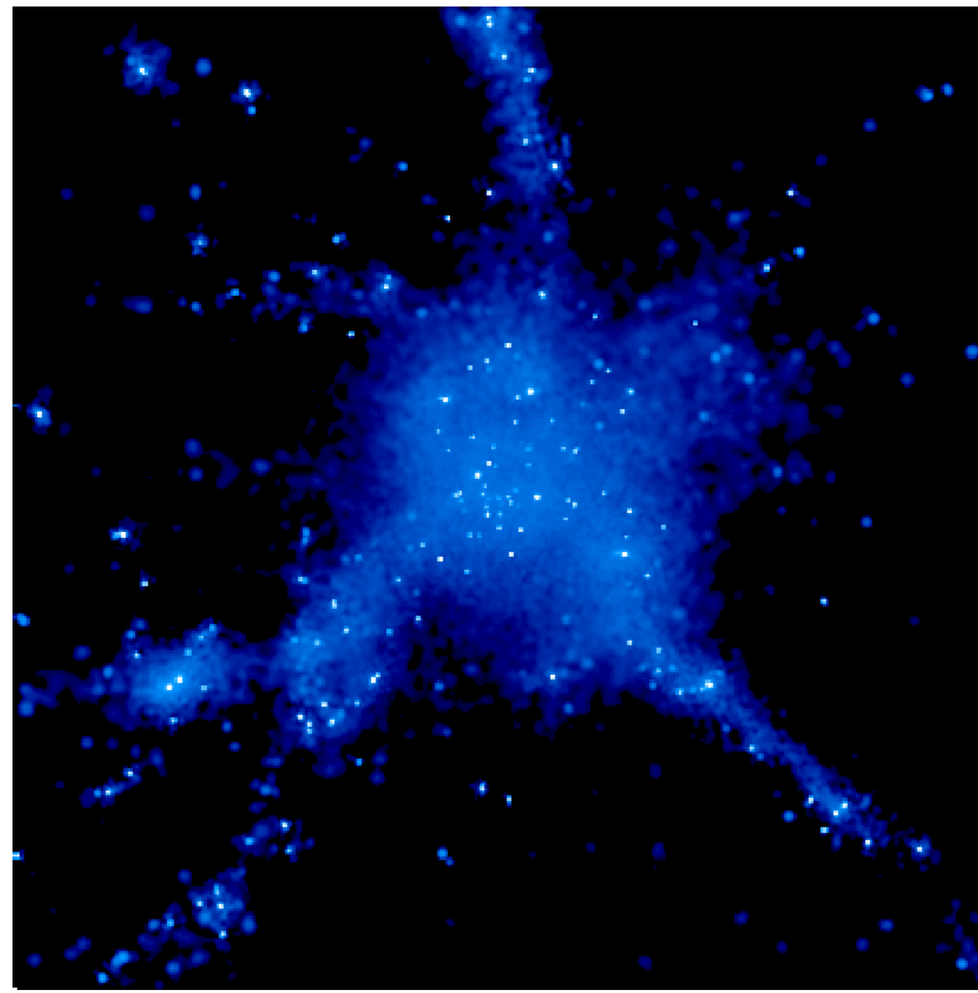
Cooling of gas is extremely efficient in high-resolution simulations of galaxy formation

CLUSTER RUNS WITH AND WITHOUT COOLING

SO_A (adiabatic)



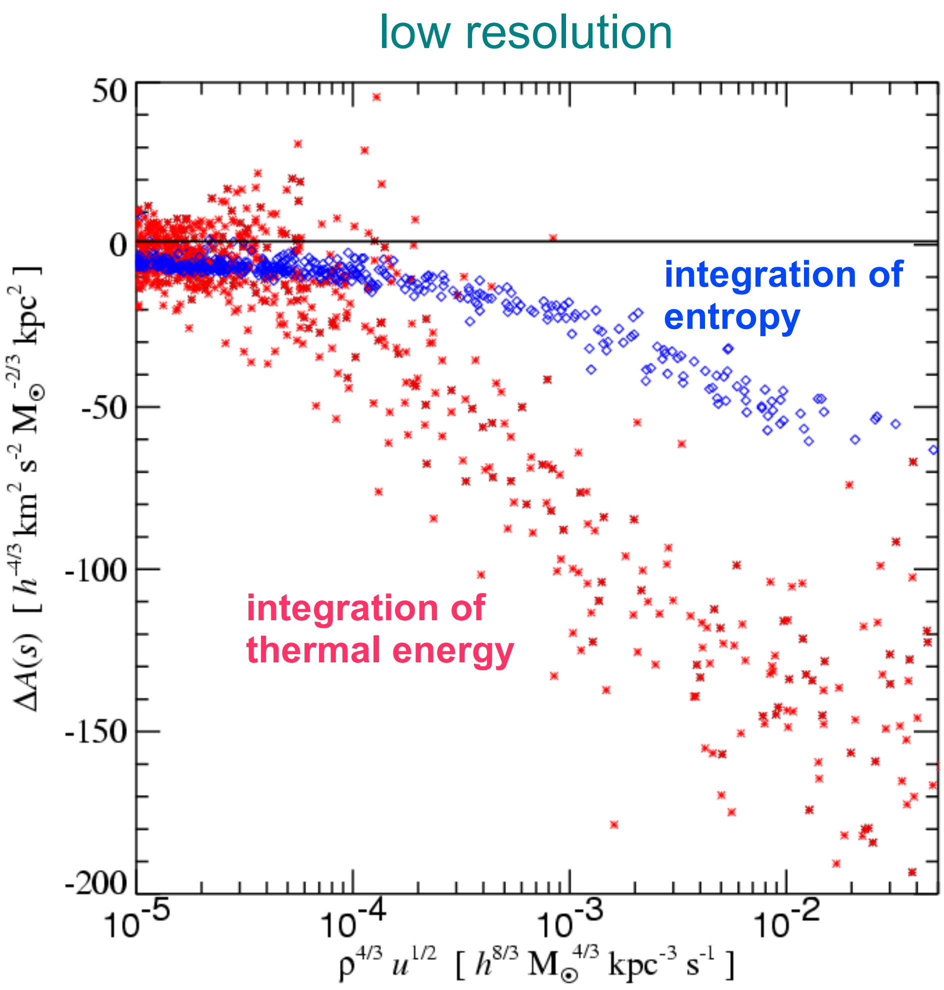
SO_C (cooling only)

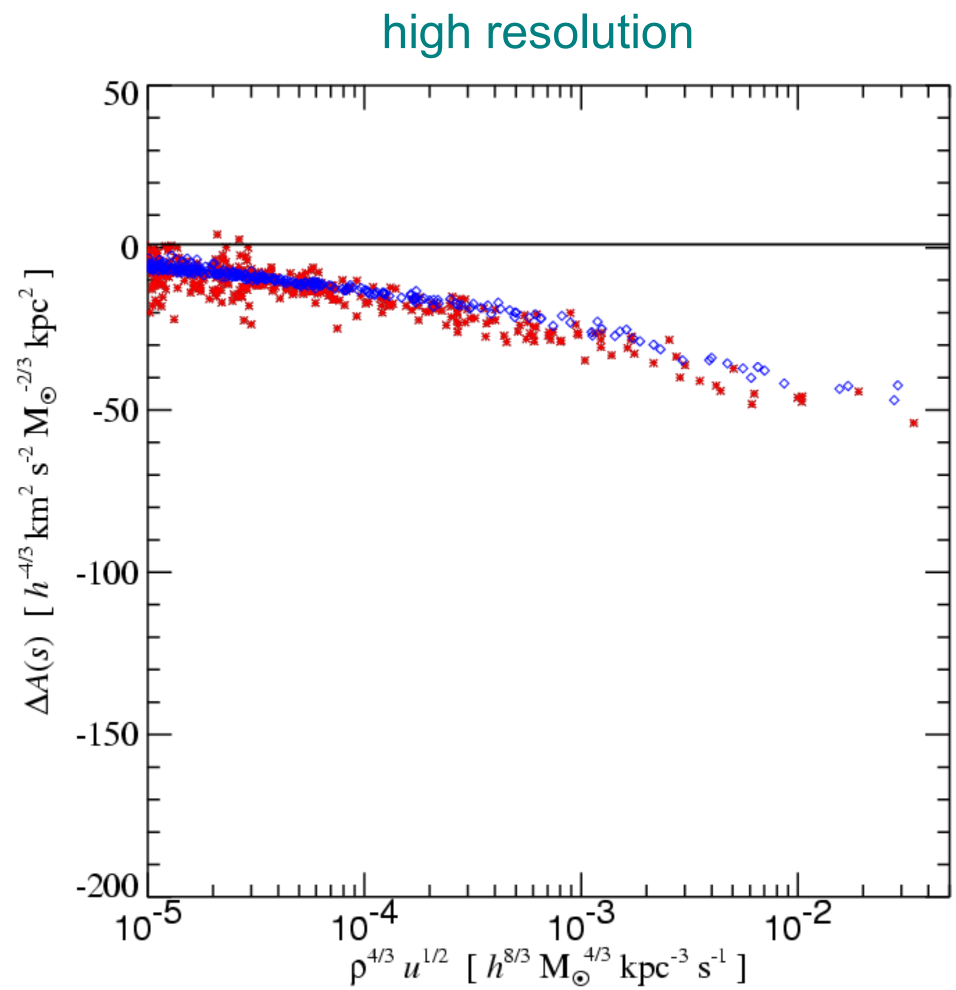


Yoshida, Stöhr, White & Springel (2001)

Fluid elements should lose entropy ***only*** by radiative cooling

DECLINE OF ENTROPY IN COOLING FLOW REGION





→ Entropy formulation is much less prone to overcooling when the resolution is poor

Neighbor search in SPH

RANGE SEARCHING WITH THE TREE

An efficient neighbor search is the most important factor that determines the speed of an SPH code

<u>But:</u> A simple search radius is not always sufficient, since for the hydro force we need to find all particles with

$$|\mathbf{r}_i - \mathbf{r}_j| < \max(h_i, h_j)$$

Solution: Store in each tree node the maximum h of all particles in the node.