High performance computing and numerical modeling

Volker Springel

Plan for my lectures

Lecture 1:	Collisional and collisionless N-body dynamics
Lecture 2:	Gravitational force calculation
Lecture 3:	Basic gas dynamics
Lecture 4:	Smoothed particle hydrodynamics
Lecture 5:	Eulerian hydrodynamics
Lecture 6:	Moving-mesh techniques
Lecture 7:	Towards high dynamic range
Lecture 8:	Parallelization techniques and current computing trends

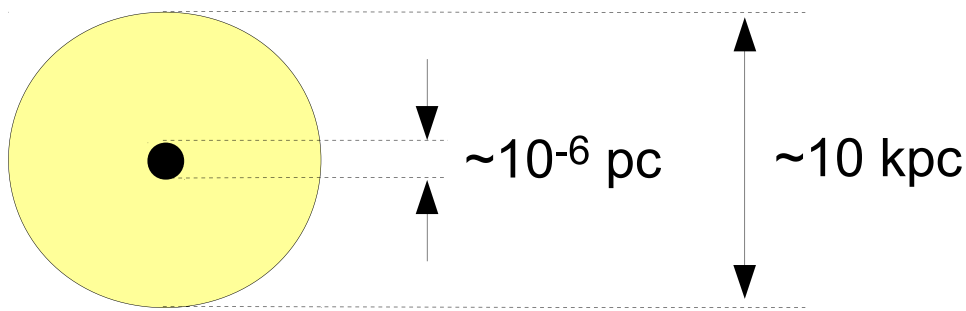
Heidelberg Institute for Theoretical Studies
HITS
Universität Heidelberg

43rd Saas Fee Course
Villars-Sur-Ollon, March 2013

Galaxy formation poses an enormous multi-scale physics problem

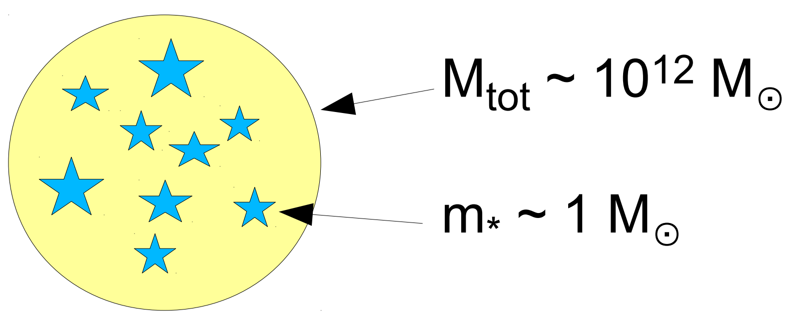
THE DYNAMIC RANGE CHALLENGE

A supermassive BH in a galaxy



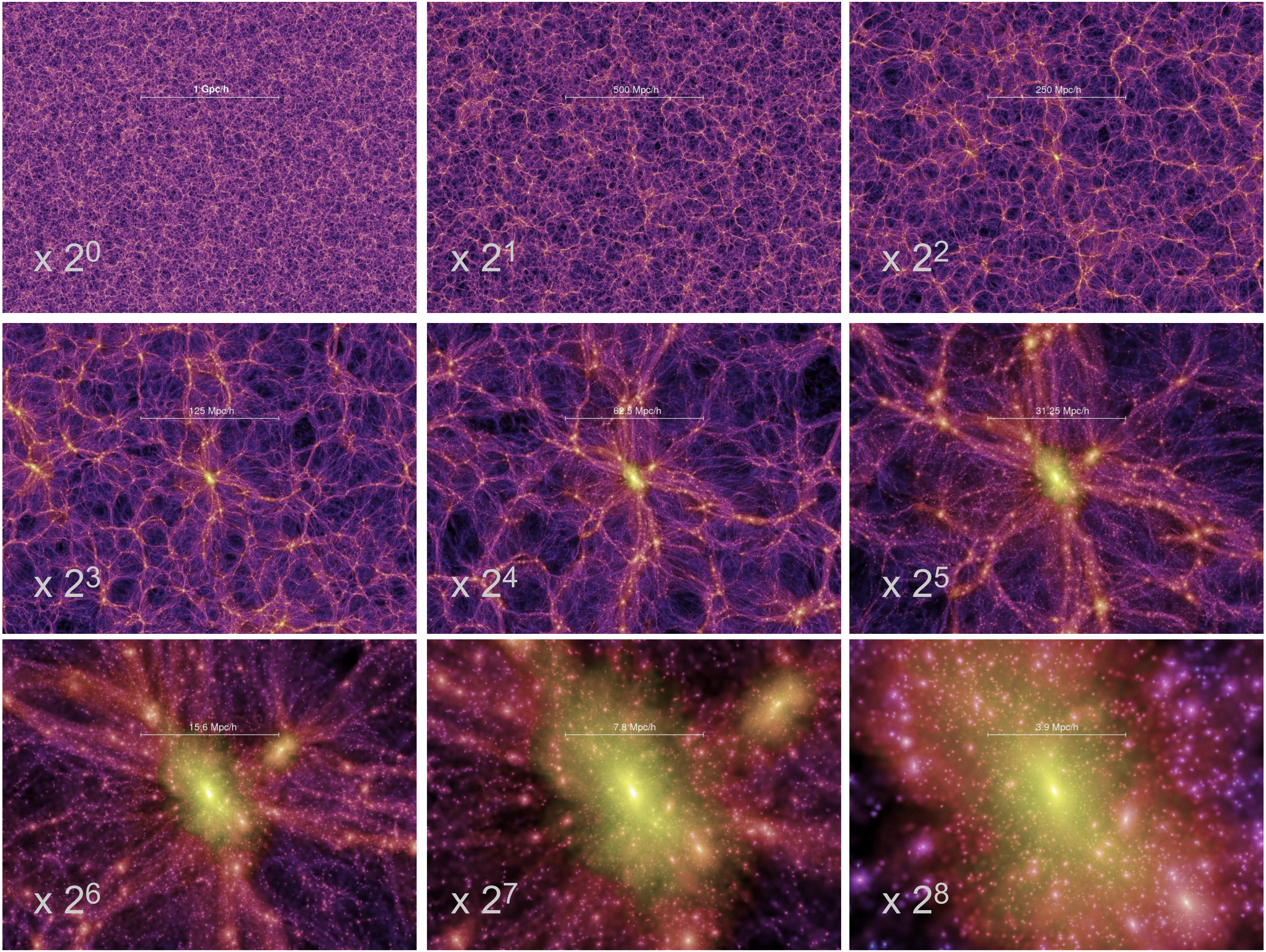
dynamic range of 10^{10}

Star formation in a normal galaxy

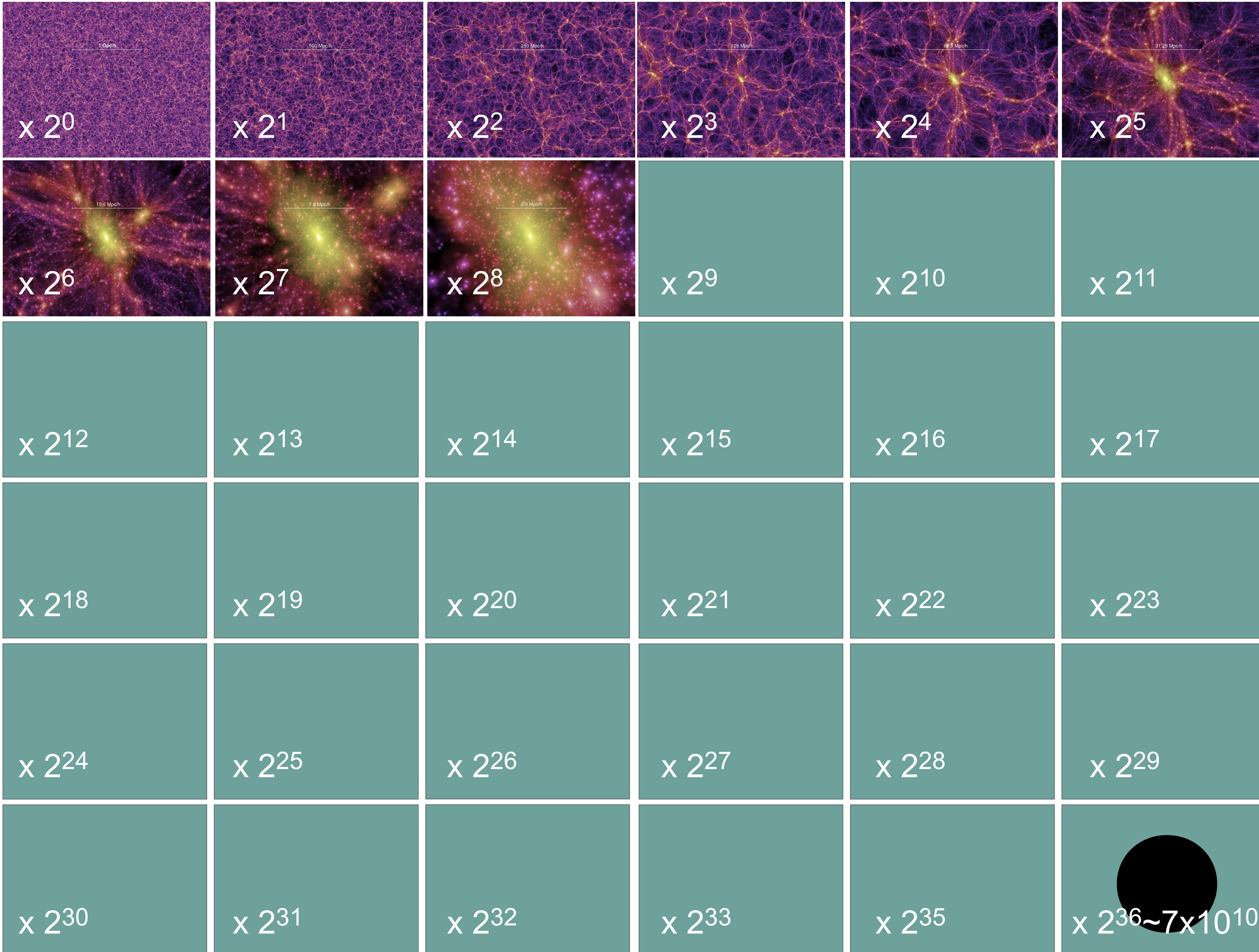


mass dynamic range of 10^{12}

- **Dynamic range prohibitively large for ab-initio calculations**
- **In addition: physics of star formation and AGN accretion only partially understood**



1 Gpc/h
x 2^0
500 Mpc/h
x 2^1
250 Mpc/h
x 2^2
125 Mpc/h
x 2^3
62.5 Mpc/h
x 2^4
31.25 Mpc/h
x 2^5
15.6 Mpc/h
x 2^6
7.8 Mpc/h
x 2^7
3.9 Mpc/h
x 2^8


x 2^0
x 2^1
x 2^2
x 2^3
x 2^4
x 2^5
x 2^6
x 2^7
x 2^8
x 2^9
x 2^10
x 2^11
x 2^12
x 2^13
x 2^14
x 2^15
x 2^16
x 2^17
x 2^18
x 2^19
x 2^20
x 2^21
x 2^22
x 2^23
x 2^24
x 2^25
x 2^26
x 2^27
x 2^28
x 2^29
x 2^30
x 2^31
x 2^32
x 2^33
x 2^35
x 2^36~7x10^10

Achieving high local resolution usually implies high dynamic range in space, time, and mass

THE DYNAMIC RANGE CHALLENGE OF GALAXY SIMULATIONS

- Assume we want to realize a 10 pc resolution using a uniform grid, for example in a 10 Mpc volume.
- This would require 10^{18} cells – a billion times more than a 1000^3 run, which is still a sizable simulation by today's standard.
- But actually, reducing the mesh size by a factor of 2 will also reduce the timestep by a factor of 2.
- So if you improve the linear dimension (of all cells) by a factor of 10, the computational cost goes up by a factor of $10^3 \times 10 = 10^4$.
- Going from a 1000^3 to a million^3 cells in a uniform grid then means a cost increase of 10^{12}.
- If computers keep getting faster at the current rate (a factor of 100 in 10 years), we merely have to wait 60 years for this.

Fortunately, high resolution is only required in a small fraction of the volume, making adaptive resolution techniques attractive

REALIZING HIGH SPATIAL DYNAMIC RANGE THROUGH ADAPTIVE RESOLUTION

Example: Suppose you want to have 10 pc resolution in the ISM of the Galaxy, but the rest of the galaxy (radius 200 kpc) can be coarser resolved.

With a uniform mesh you need:

$$\frac{4\pi}{3}\left(\frac{200\,\mathrm{kpc}}{10\,\mathrm{pc}}\right)^3 \simeq 3.4 \times 10^{13}$$

If you just fill the disk, say of radius 10 kpc and height 1 kpc, with high resolution you need:

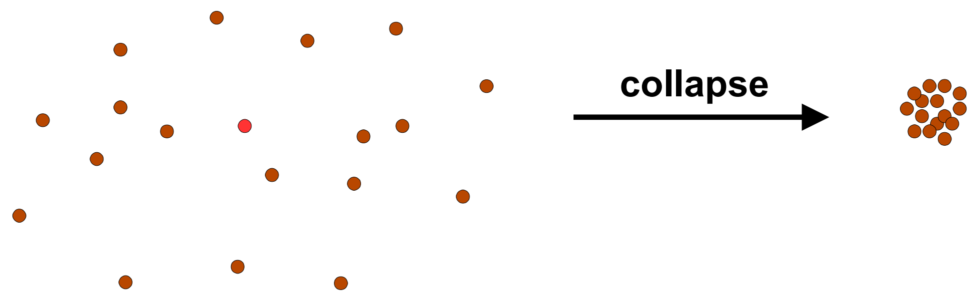
$$\frac{\pi(10\,\mathrm{kpc})^2 \times 1\,\mathrm{kpc}}{(10\,\mathrm{pc})^3} \simeq 3.1 \times 10^8$$

So adaptive spatial resolution is the way to go.

The Lagrangian character of SPH is automatically providing adaptive resolution that is very well suited for gravity-driven structure growth

DIFFERENT APPROACHES TO ADAPTIVE RESOLUTION

SPH:



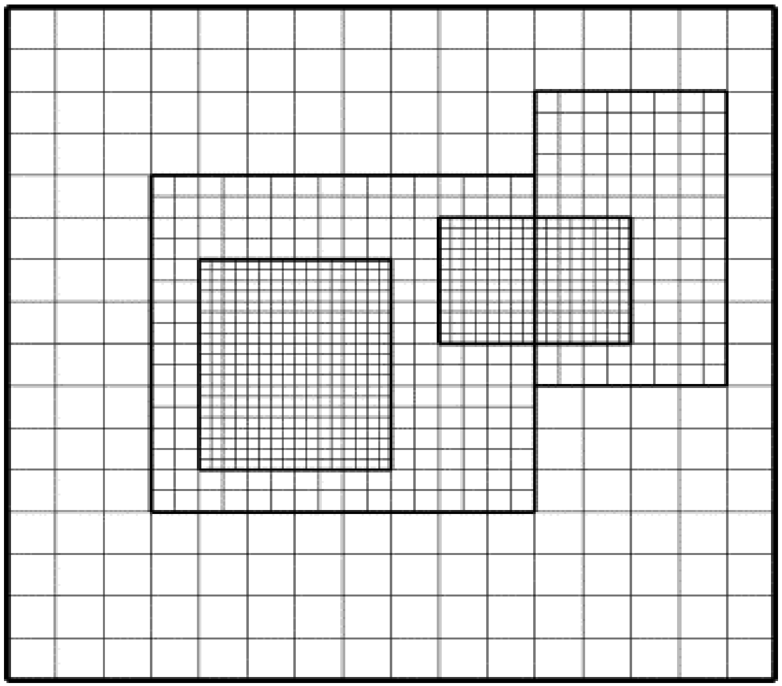
- Provided one puts enough particles initially into the region of interest, an adaptive resolution with constant mass resolution is automatically obtained.

- The downside is, resolution is difficult or impossible to change on the fly.

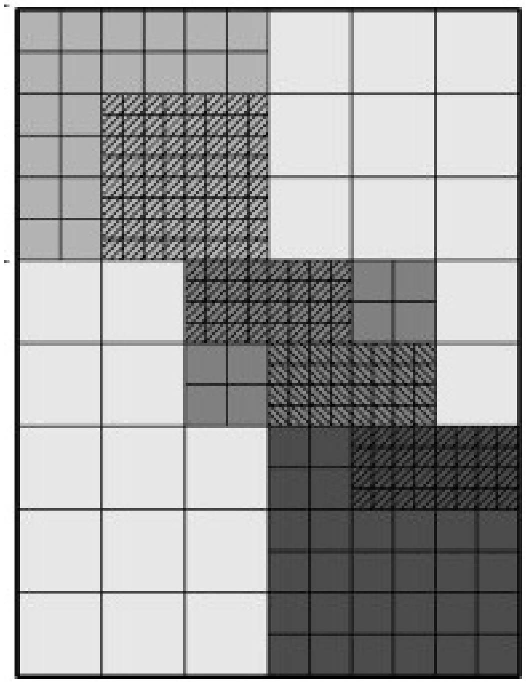
- Multi-mass technique do not work very well as the accuracy in regions where particles of different mass interact is poor.

Eulerian codes can employ **Adaptive Mesh Refinement** (AMR) to realize high dynamic range

DIFFERENT APPROACHES TO ADAPTIVE RESOLUTION



patch-based
refinement strategy
(e.g ENZO)



tree-based
refinement strategy
(e.g RAMSES)

Eulerian codes can employ **Adaptive Mesh Refinement** (AMR) to realize high dynamic range

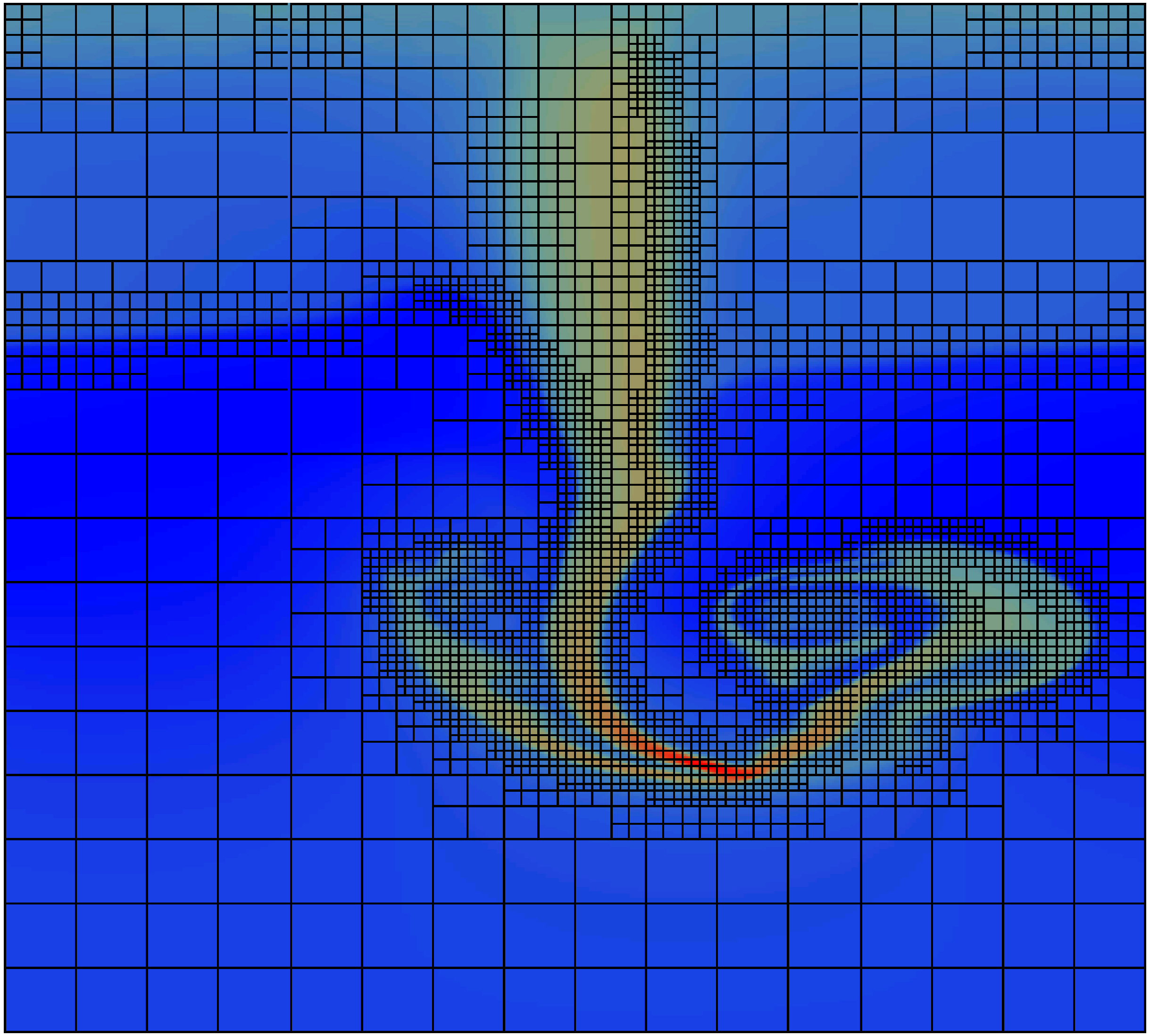
DIFFERENT APPROACHES TO ADAPTIVE RESOLUTION

AMR:

- Use a hierarchy of nested grids that allows in principle arbitrary dynamic range. Refinement criteria can be chosen almost arbitrarily.

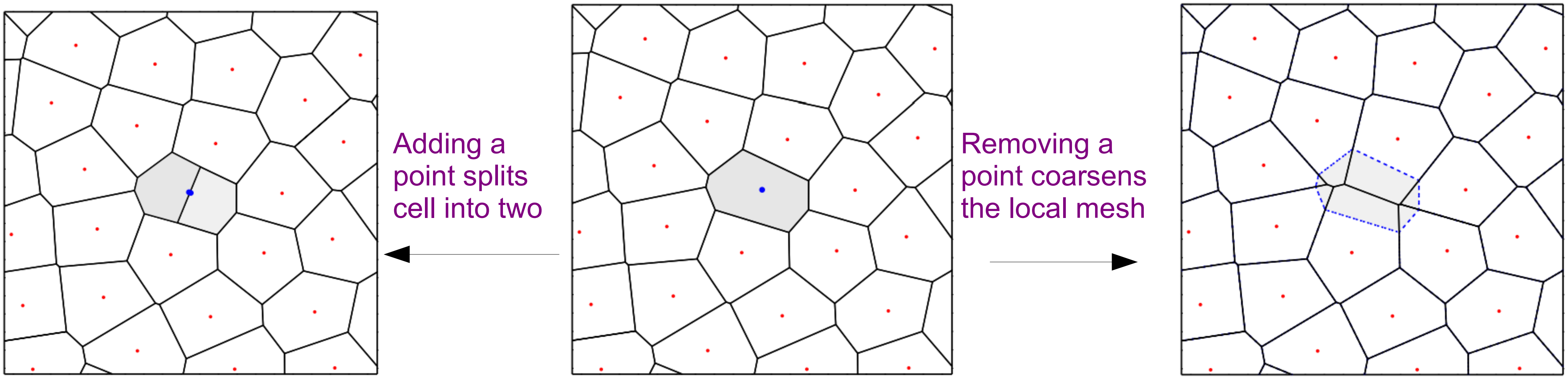
- Quick motion of a small high-resolution region requires however frequent changes of the mesh hierarchy.

- Accuracy at grid boundaries suffers and normally goes down to 1st order.



The moving-mesh approach is intermediate between SPH and AMR

DIFFERENT APPROACHES TO ADAPTIVE RESOLUTION



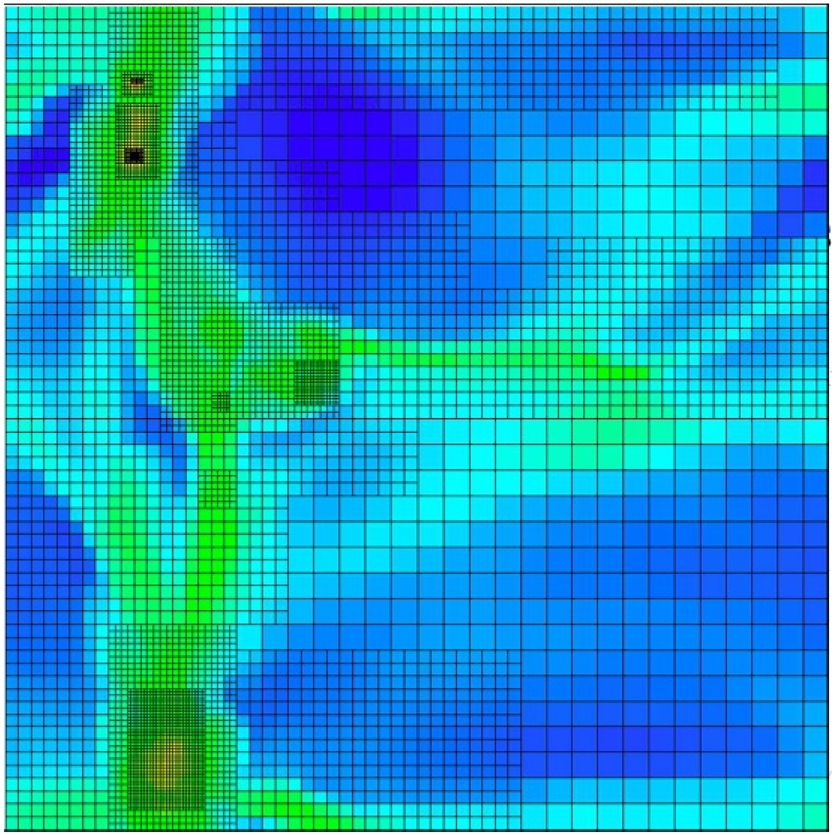
Moving Voronoi mesh:

- Similar to SPH, the method keeps the mass resolution approximately constant, independent of the clustering state.
- If desired, dynamic mesh refinements and de-refinements are however possible, similar to AMR.
- At any given time, only one mesh is tessellating the volume. The resolution changes gradually throughout space, in principal avoiding localized errors due to resolution changes.

Small spatial scales also imply short timesteps

INDIVIDUAL TIMESTEP INTEGRATION IS OFTEN IMPLEMENTED HIERARCHICALLY

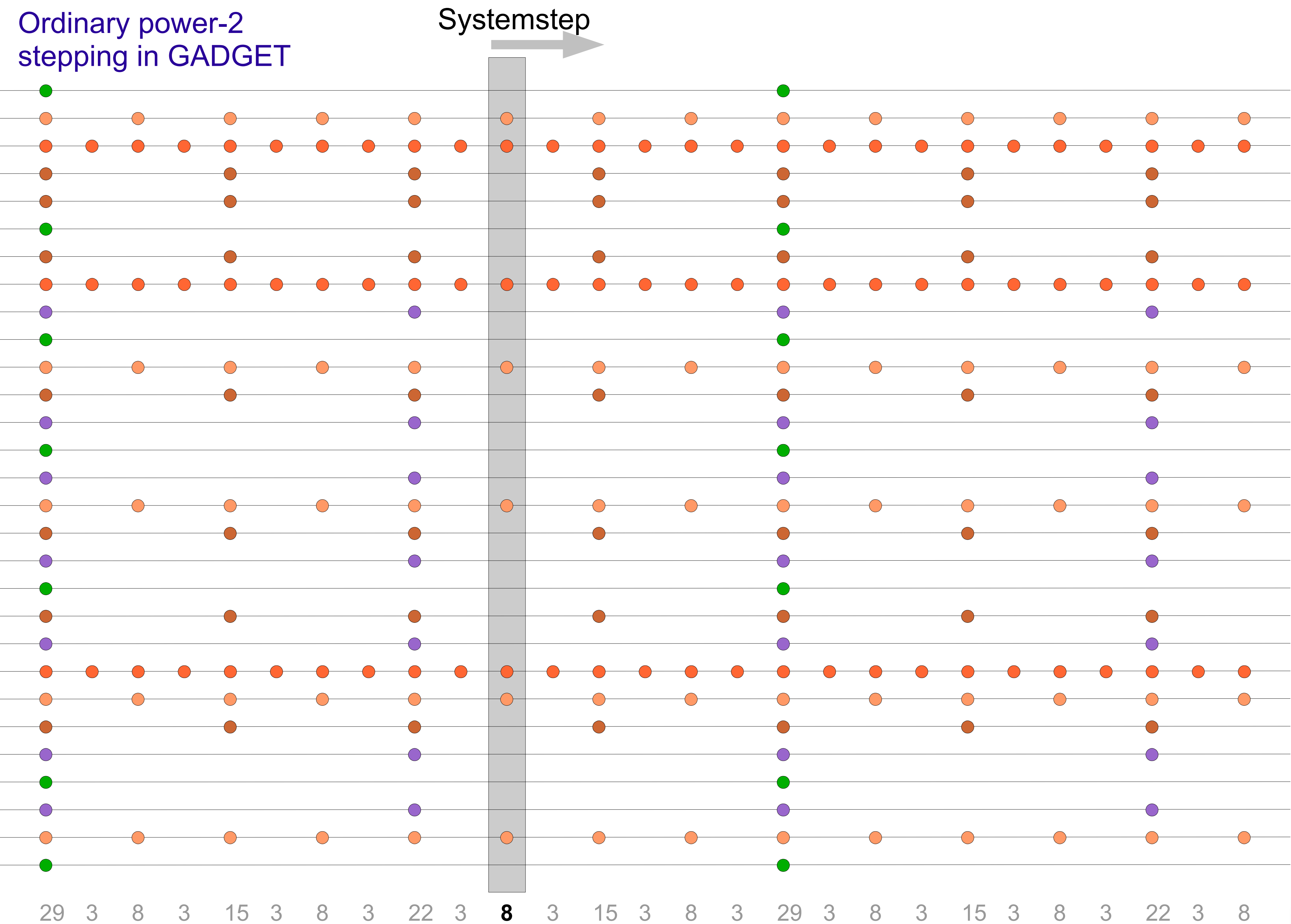
Timestep / Refinement Level	Particles/Cells on the timestep bin
$32 \times \Delta t$	$\sim 10^6$
$16 \times \Delta t$	$\sim 10^5$
$8 \times \Delta t$	$\sim 10^4$
$4 \times \Delta t$	$\sim 10^3$
$2 \times \Delta t$	$\sim 10^2$
Δt	~ 10



Greg Bryan

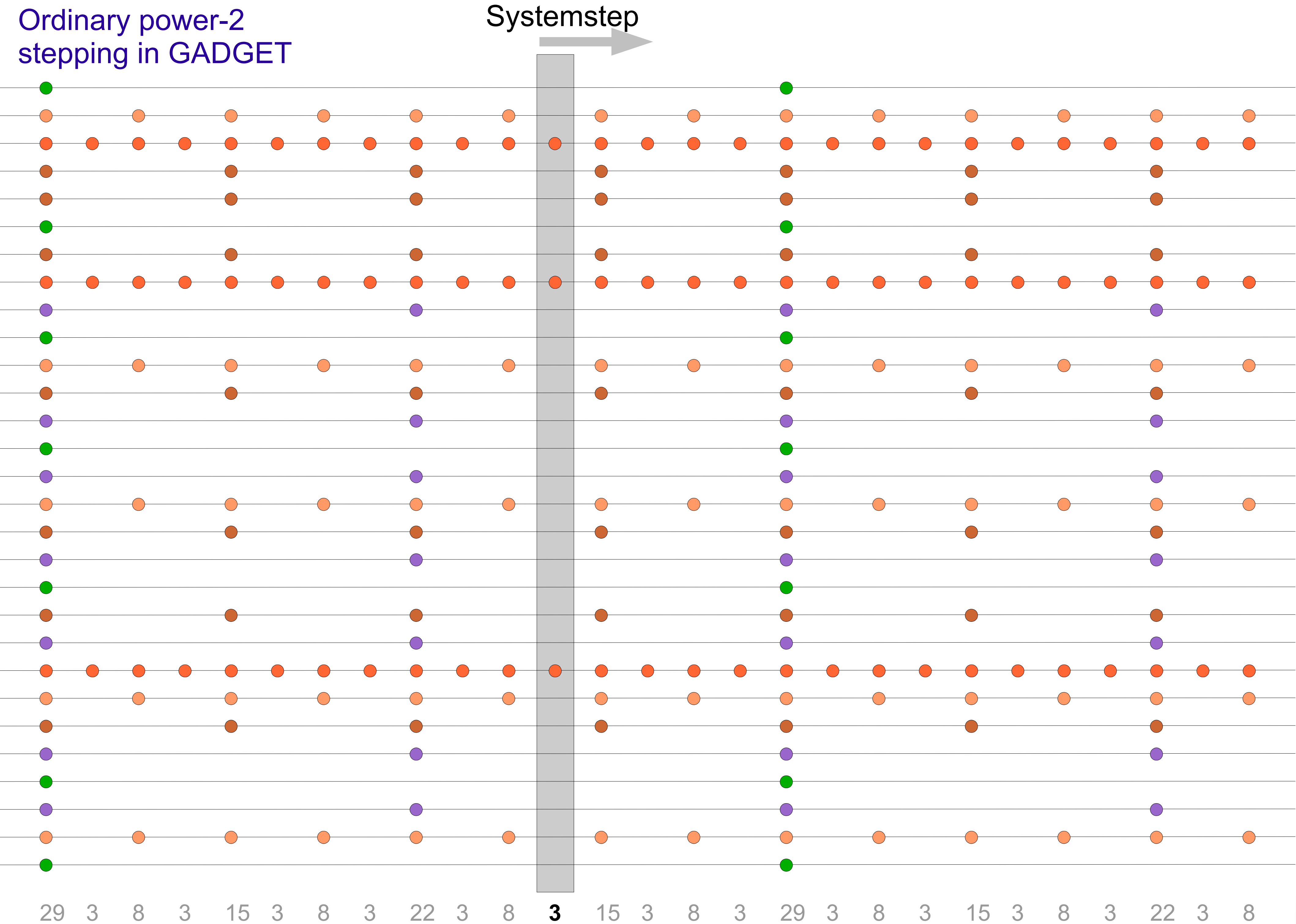
To simulate a certain timespan, you either need to advance every cell at every step (as in FLASH), or you advance only the finer meshes on shorter steps.

The individual stepping can be a factor 28.4 faster in this example.

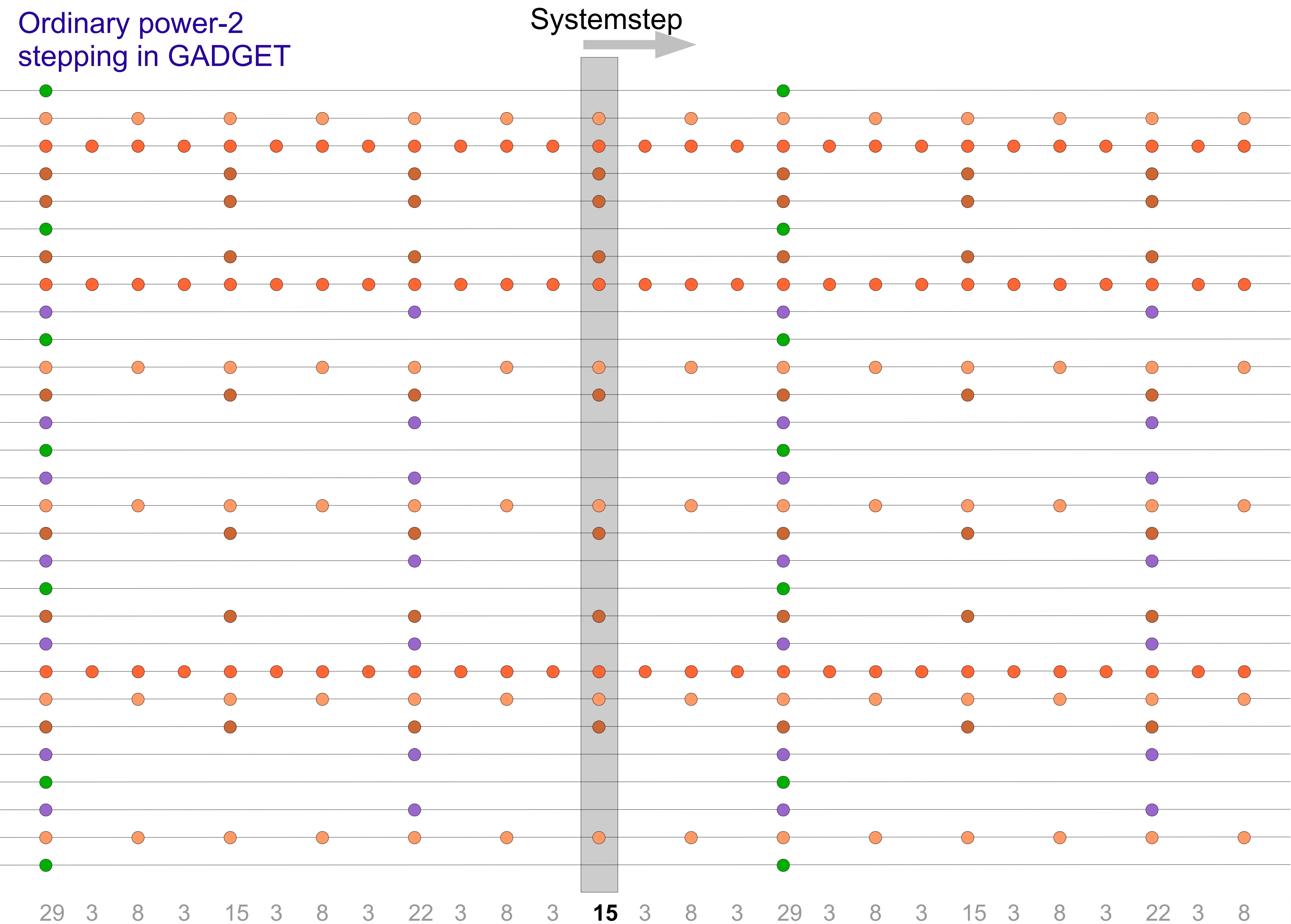


Ordinary power-2 stepping in GADGET
Systemstep
29 3 8 3 15 3 8 3 22 3 8 3 15 3 8 3 29 3 8 3 15 3 8 3 22 3 8

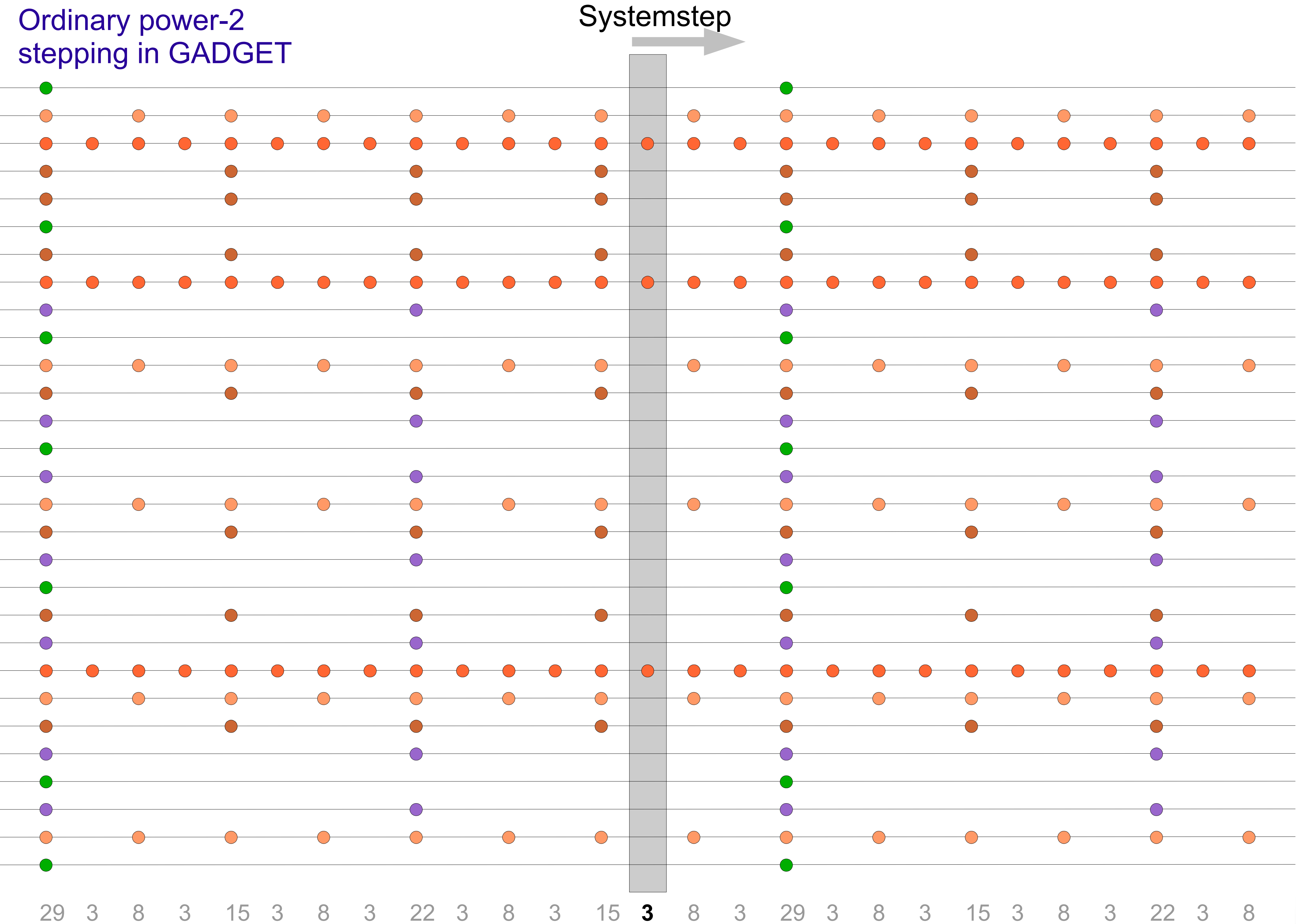
Ordinary power-2 stepping in GADGET



Ordinary power-2 stepping in GADGET



Ordinary power-2 stepping in GADGET



Use of "Divide and Conquer" for complicated PDE systems

OPERATOR SPLITTING TECHNIQUES

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = \sum_i S_i(\mathbf{U})$$

Right hand-side may describe physics such as radiative cooling, diffusion or chemistry.

Consider the general differential equation:

$$\frac{\partial u}{\partial t} = A(u) + B(u)$$

Suppose we can formulate solutions for A and B separately:

$$\alpha_t(u_0) \equiv \exp(tA)u_0$$

$$\beta_t(u_0) \equiv \exp(tB)u_0$$

Then the **Lie-split** approximate solution for the full system is:

$$u^{\mathrm{Lie}}(h) \simeq \beta_h(\alpha_h(u_0)) = \mathrm{e}^{hB}\,\mathrm{e}^{hA}\,u_0$$

The Strang**-split** approximate solution for the full system is given by:

$$u^{\mathrm{Strang}}(h) \simeq \mathrm{e}^{\frac{h}{2}A}\,\mathrm{e}^{hB}\,\mathrm{e}^{\frac{h}{2}A}\,u_0$$

How accurate are the operator-split timesteps?

$$\Delta u^{\mathrm{Lie}}(h) = u^{\mathrm{Lie}}(h) - u(h) = \left[\mathrm{e}^{hA}\mathrm{e}^{hB} - \mathrm{e}^{h(A+B)}\right] u_0$$

Taylor expand:

$$\Delta u^{\mathrm{Lie}}(h) = \left\{ \left(1 + hA + \frac{h^2}{2}A^2 + \ldots\right)\left(1 + hB + \frac{h^2}{2}B^2 + \ldots\right) - \left(1 + h(A+B) + \frac{h^2}{2}(A+B)^2\right) \right\} u_0$$

This gives for Lie:

$$\Delta u^{\mathrm{Lie}} = \frac{1}{2}[A,B]h^2 + \mathcal{O}(h^3)$$

With the help of the Baker-Campell-Hausdorff formula one finds for Strang:

$$\Delta u^{\mathrm{Strang}}(h) = \mathcal{O}(h^3)$$

This means we can split off the extra physics:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = S_{\mathrm{chem}}(\mathbf{U})$$

α →

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = 0$$

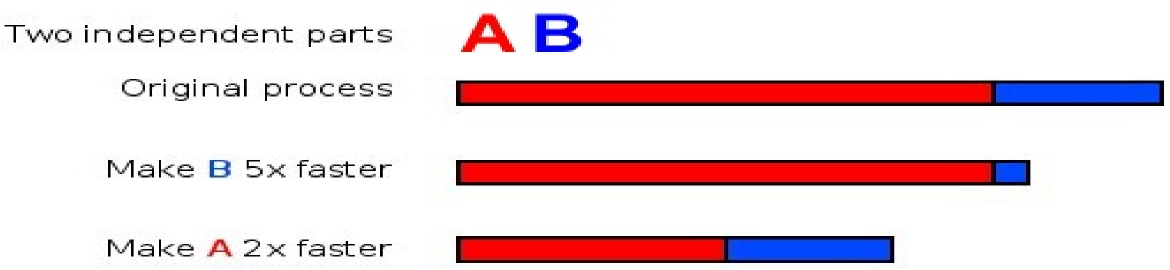
β →

$$\frac{\partial \mathbf{U}}{\partial t} = S_{\mathrm{chem}}(\mathbf{U})$$

Parallel computing: Scalability and its limitations

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

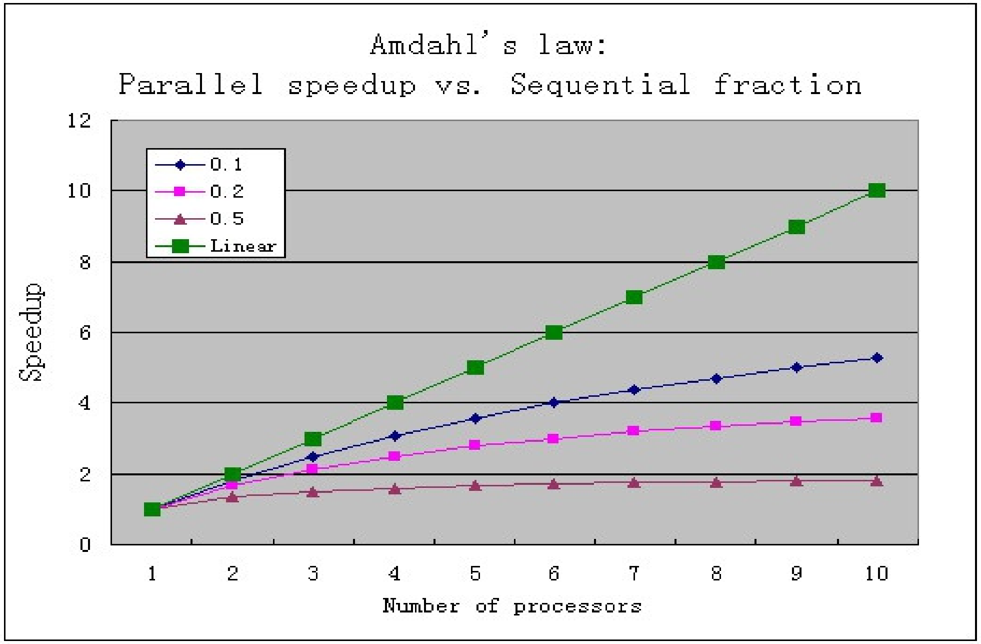
THE IMPLICATIONS OF A RESIDUAL SERIAL FRACTION



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

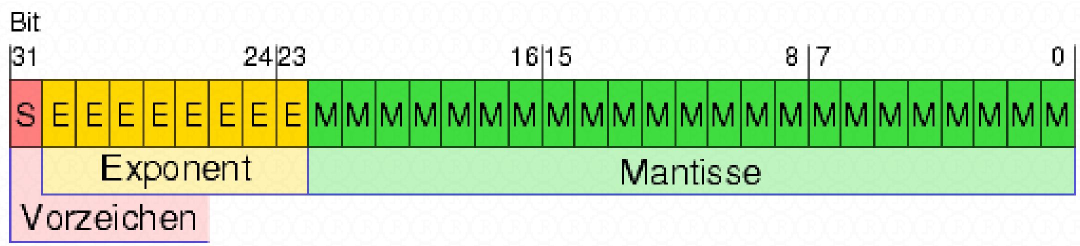
Issues of floating point accuracy

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



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

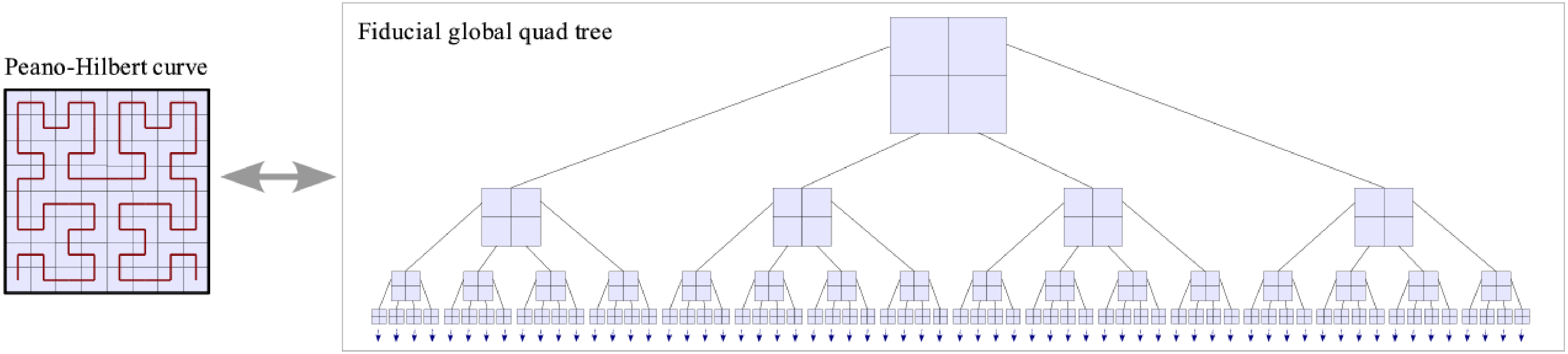
Mathematical operations regularly lead out of the space of the representable numbers. This results in **round-off** errors.

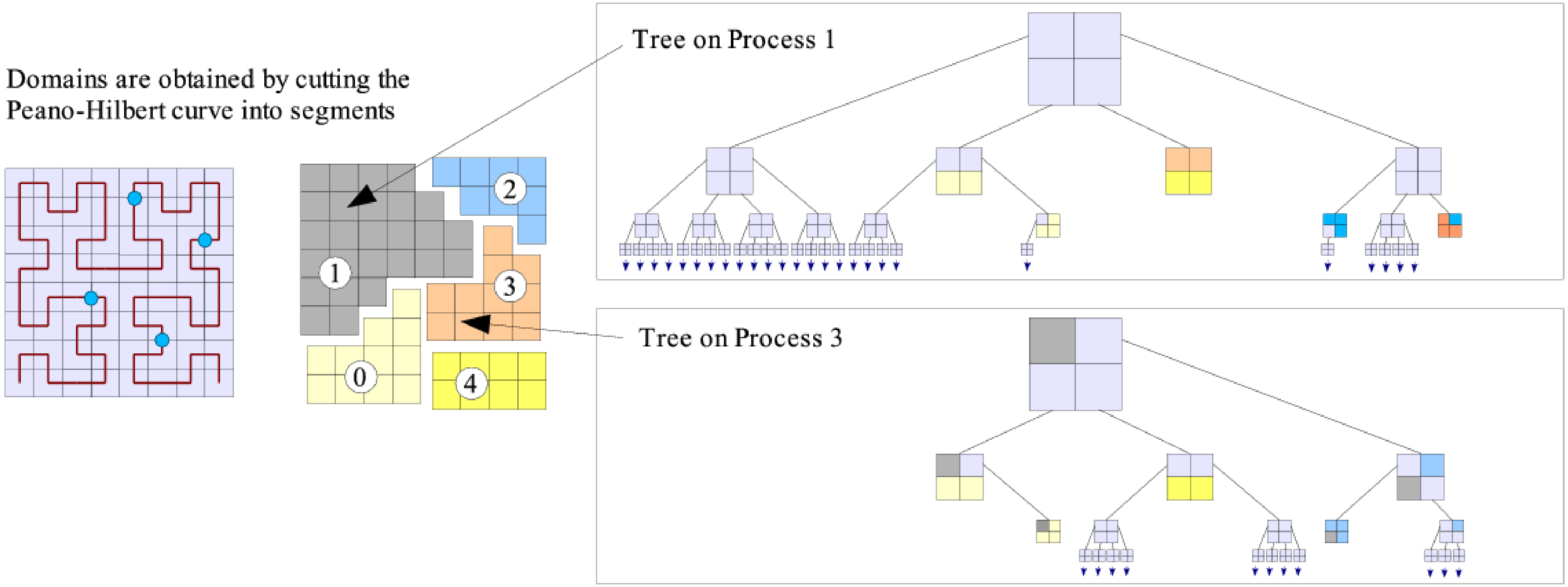
One result of this is that the law of associativity for simple additions doesn't hold on a computer.

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

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

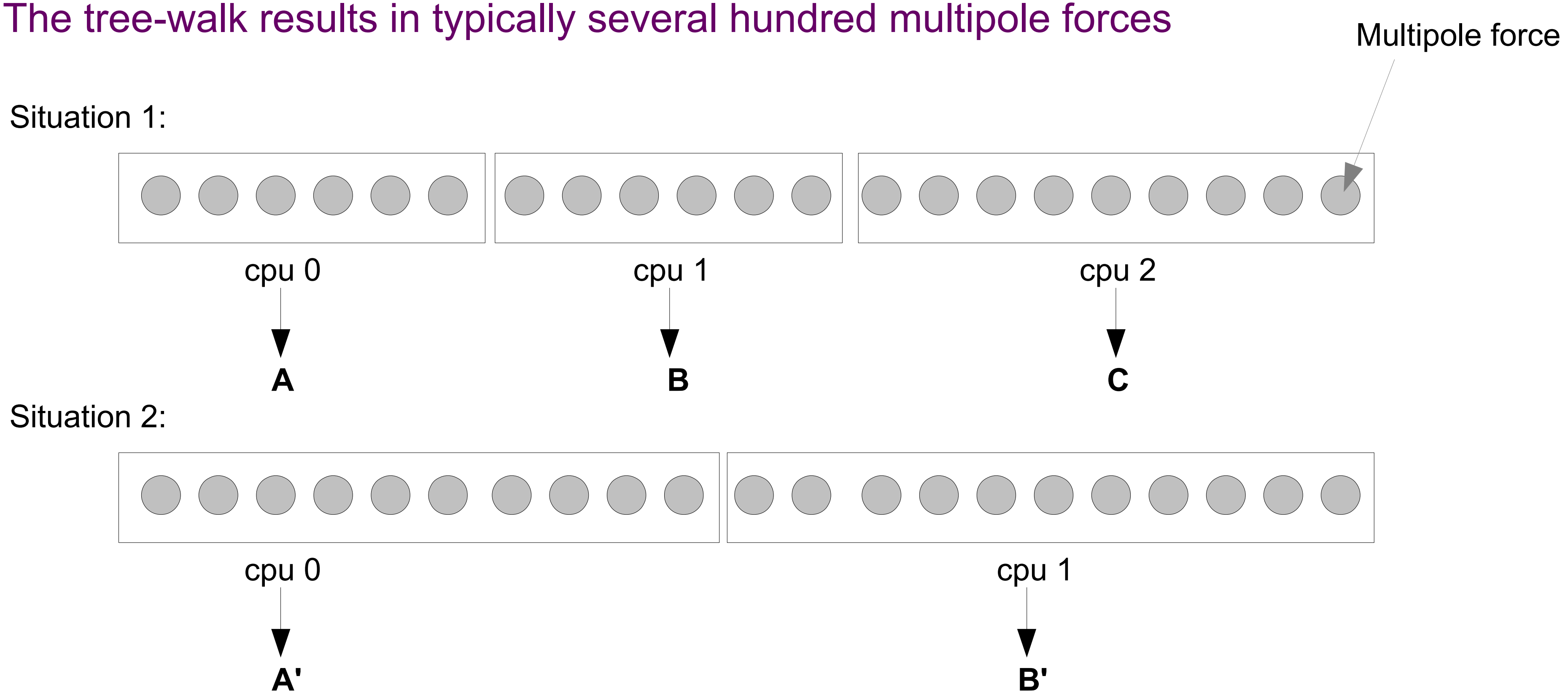




As a result of parallelization, the calculation of the force may be split to up onto different processors

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!

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

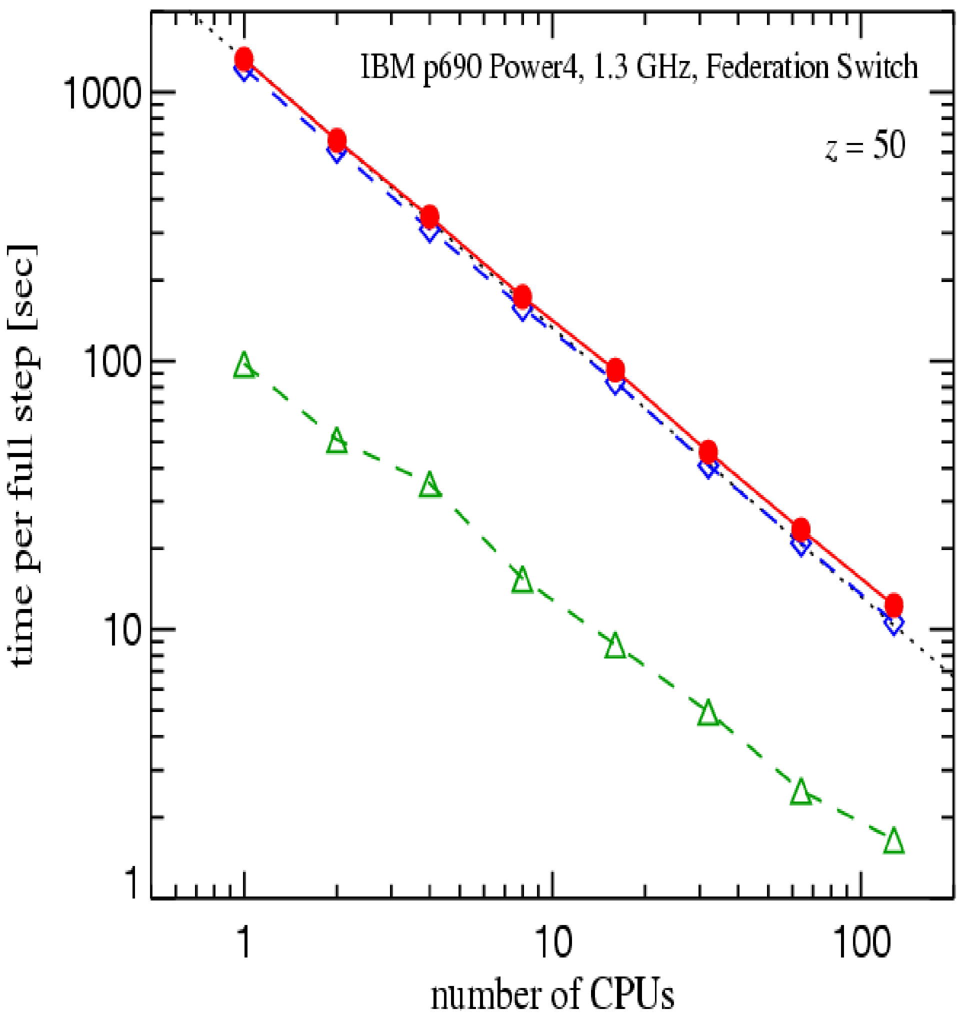
As a rule, scalability can be more easily retained in the weak scaling regime.

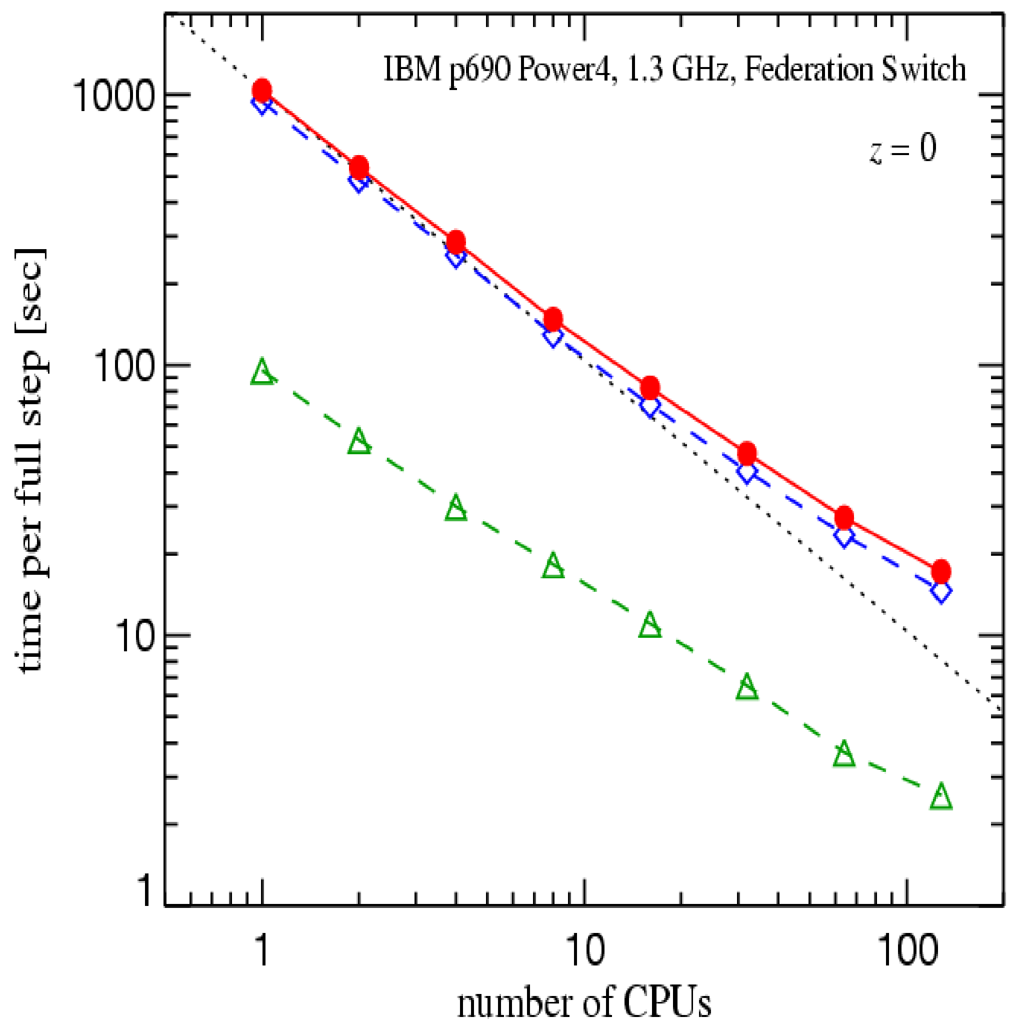
→ **In practice, it usually doesn't make sense to use a large number of processors for a (too) small problem size !**

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

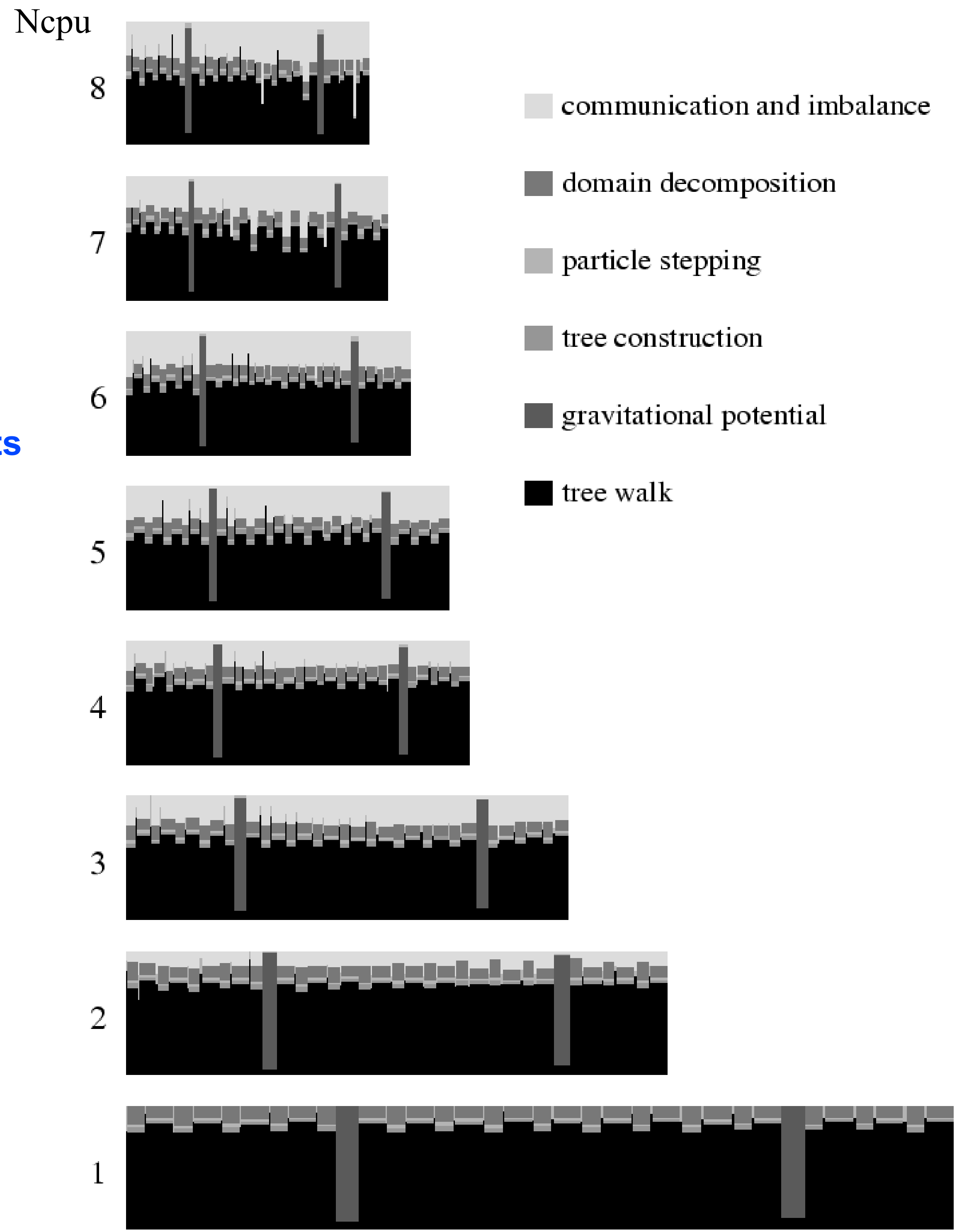




For small problem sizes or isolated galaxies, the scalability is limited

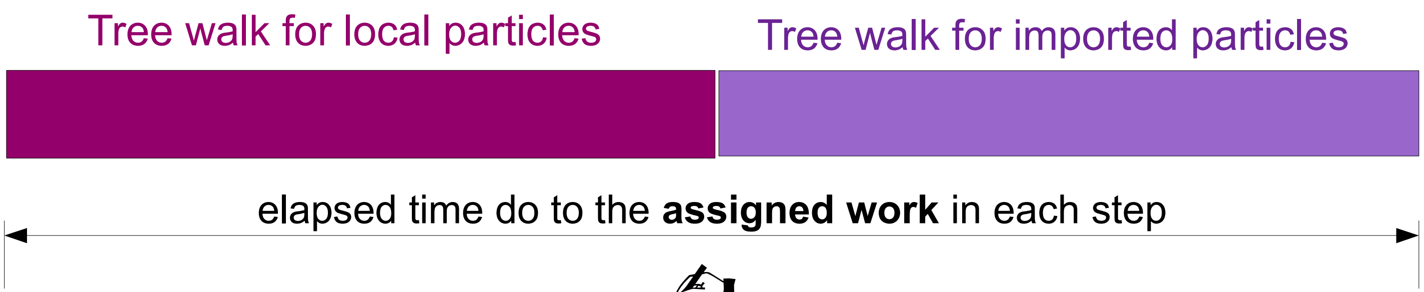
RESULTS FOR "STRONG SCALING" OF A GALAXY COLLISION SIMULATION

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

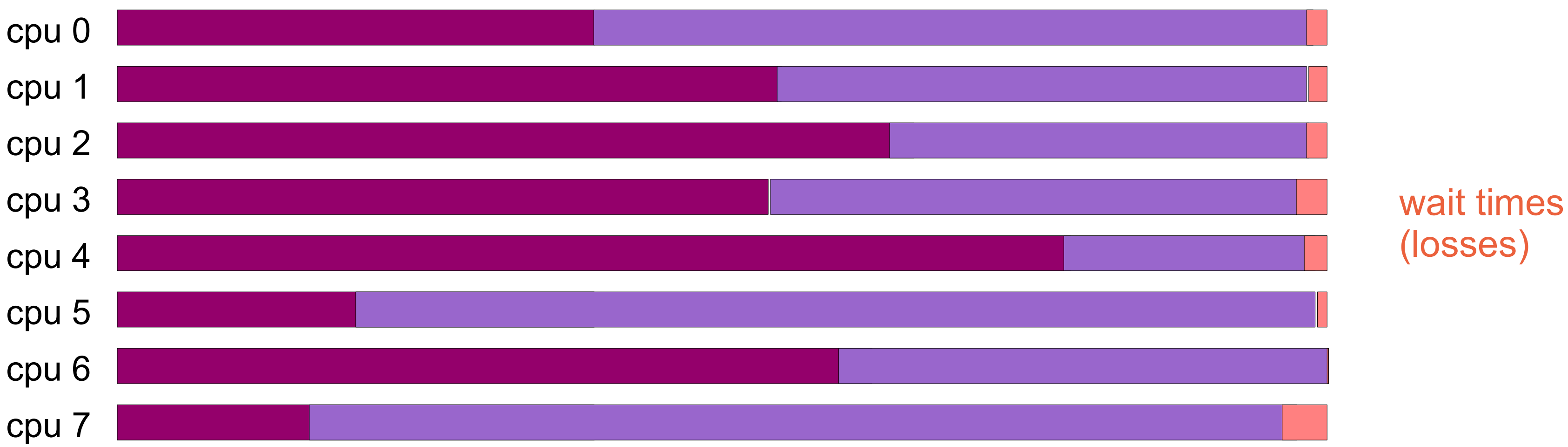


The cumulative execution time of the tree-walk on each processor can be measured and used to adjust the domain decomposition

BALANCING THE TOTAL WORK FOR EACH PROCESSOR



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

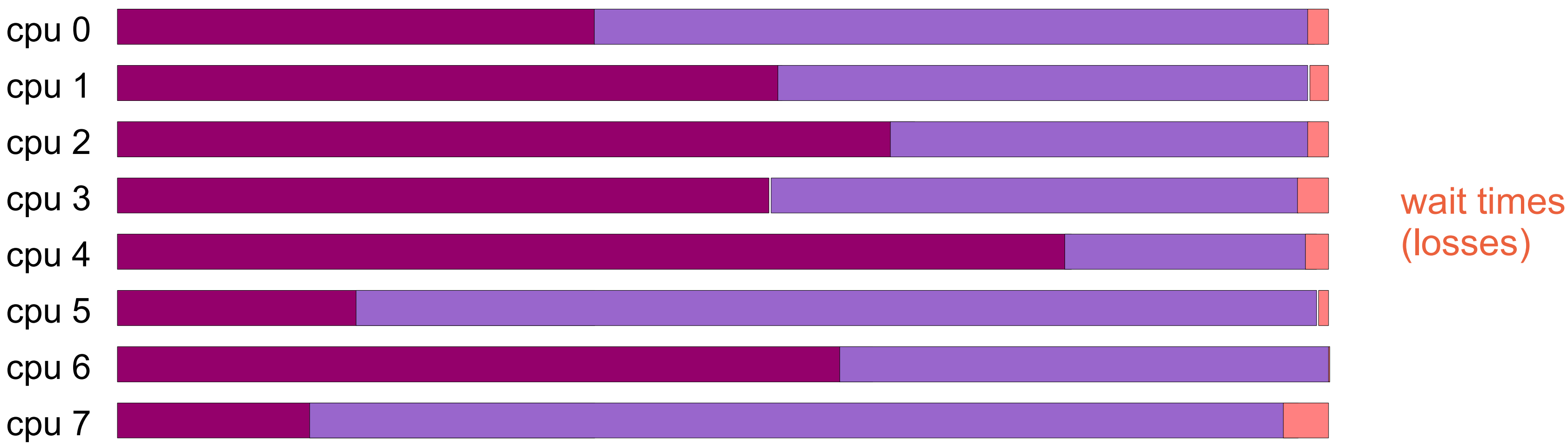


→ The total CPU-time for the tree-walks per step can be made roughly equal for each MPI task

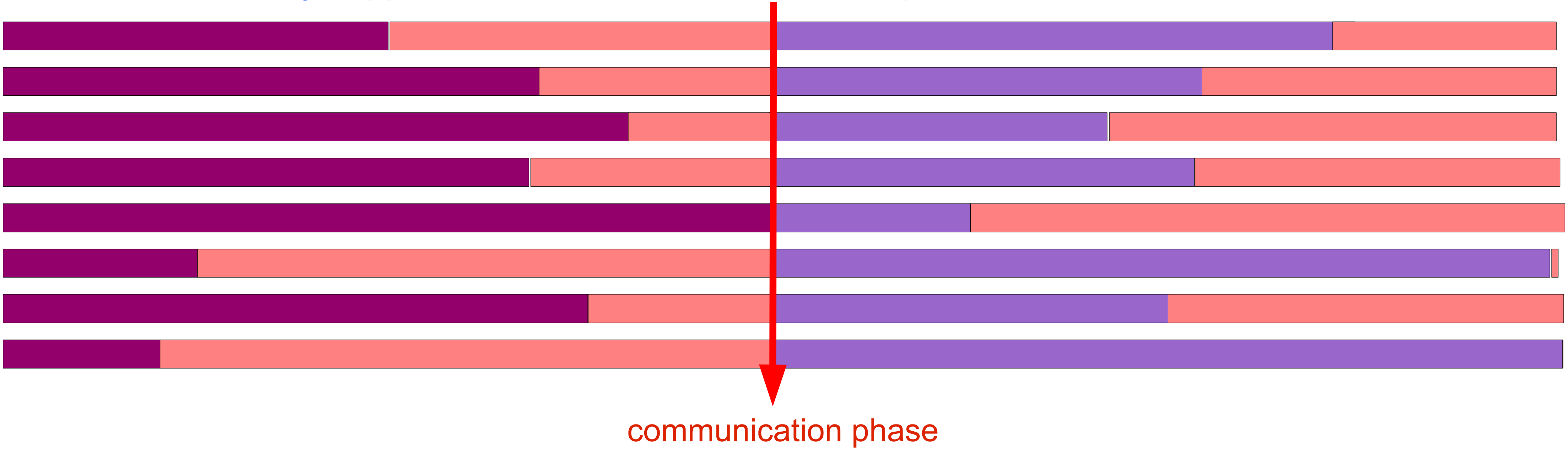
The communication between the two phases of a step introduces a synchronization point in GADGET2's standard communication scheme

LOSSES DUE TO IMBALANCE IN DIFFERENT COMMUNICATION PHASES

The situation after work-load balancing:



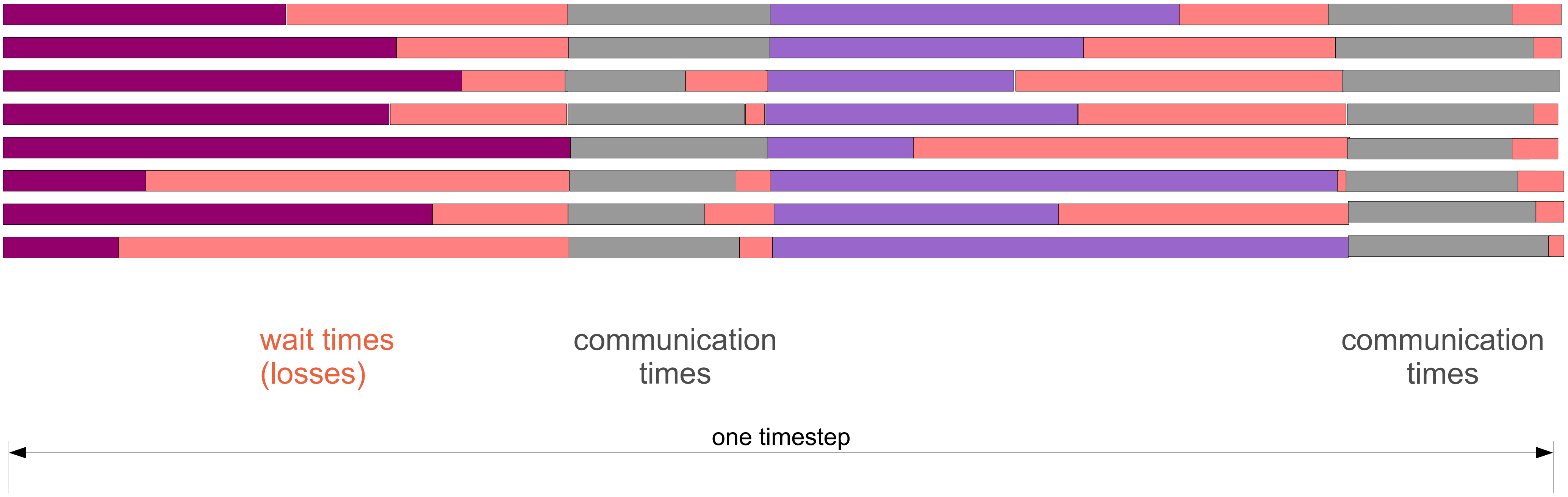
This is what actually happens once the communication step is accounted for:



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

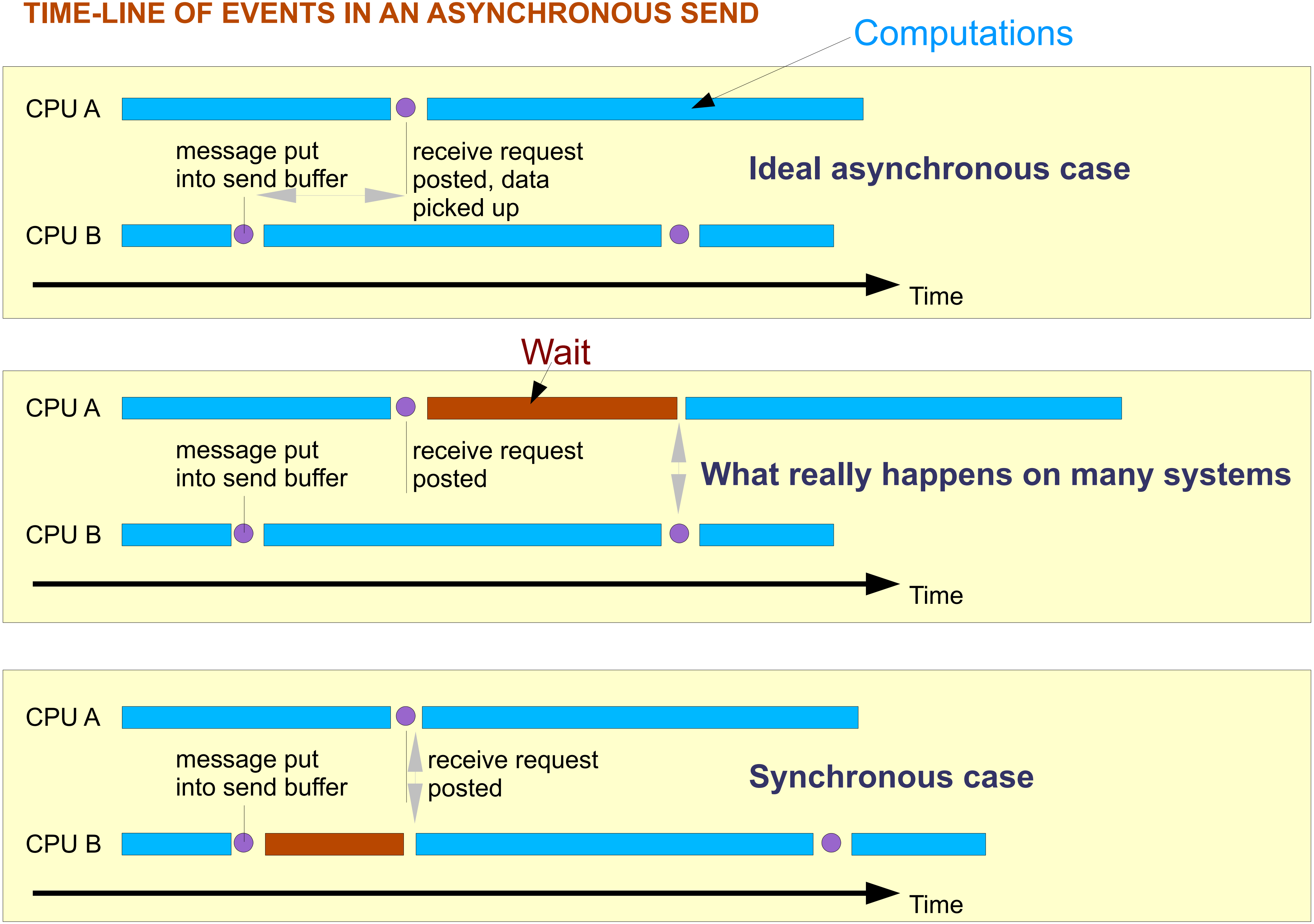
LOSSES DUE TO COMMUNICATION TIMES IN ONE GRAVITY STEP

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



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

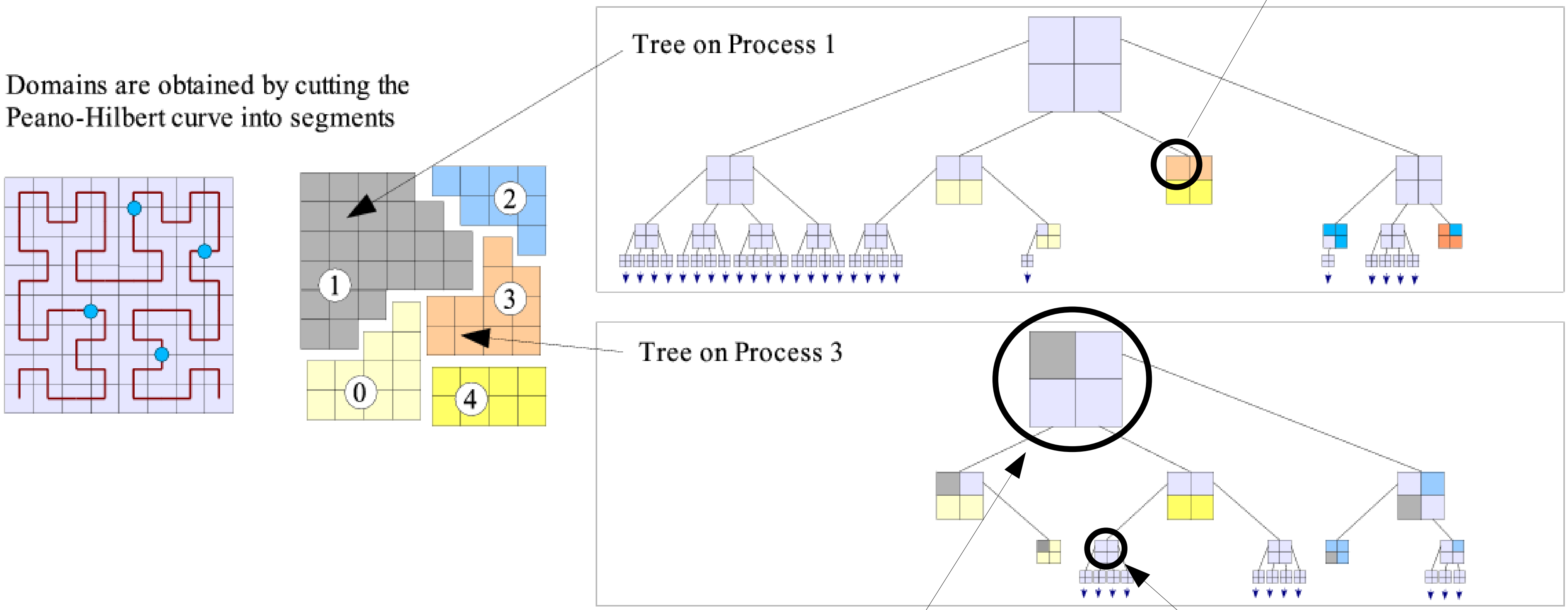


Reducing imbalance with a better domain decomposition

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

COMMUNICATION IN THE DISTRIBUTED TREE ALGORITHM

need to export to processor 3



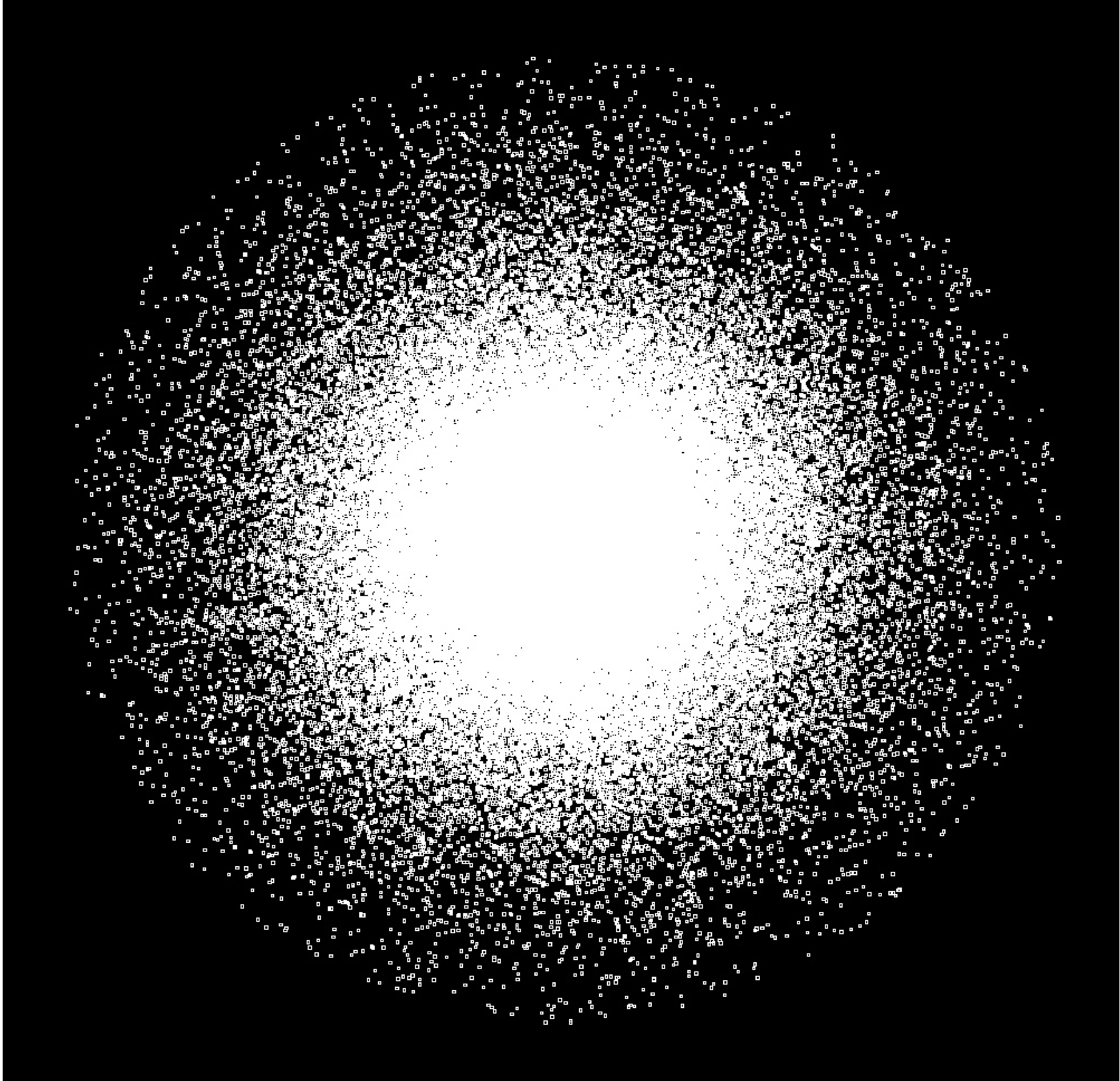
Evaluating opening criteria for top-level tree nodes multiple times can be eliminated. The work for tree walks (gravity and SPH neighbor search) becomes strictly independent of the number of processors.

Gadget2 starts to walk the tree for imported particles always at the root node

Gadget3 continues the tree walk at the right place for imported particles

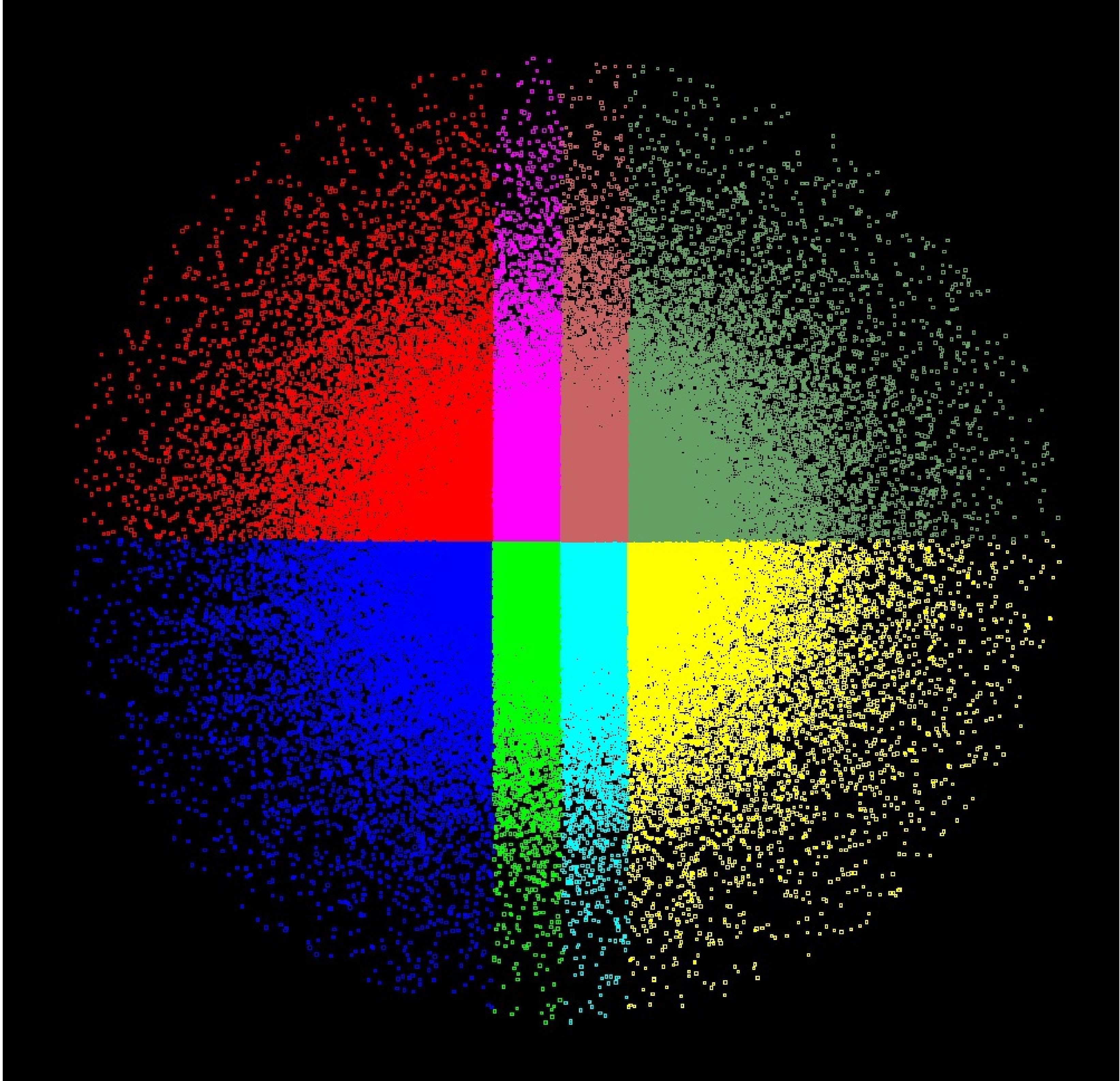
The inhomogeneous particle distribution and the different timesteps as a function of density make it challenging to find an optimum domain decomposition that balances work-load (and ideally memory-load)

PARTICLE DISTRIBUTION IN AN EXPONENTIAL DISK



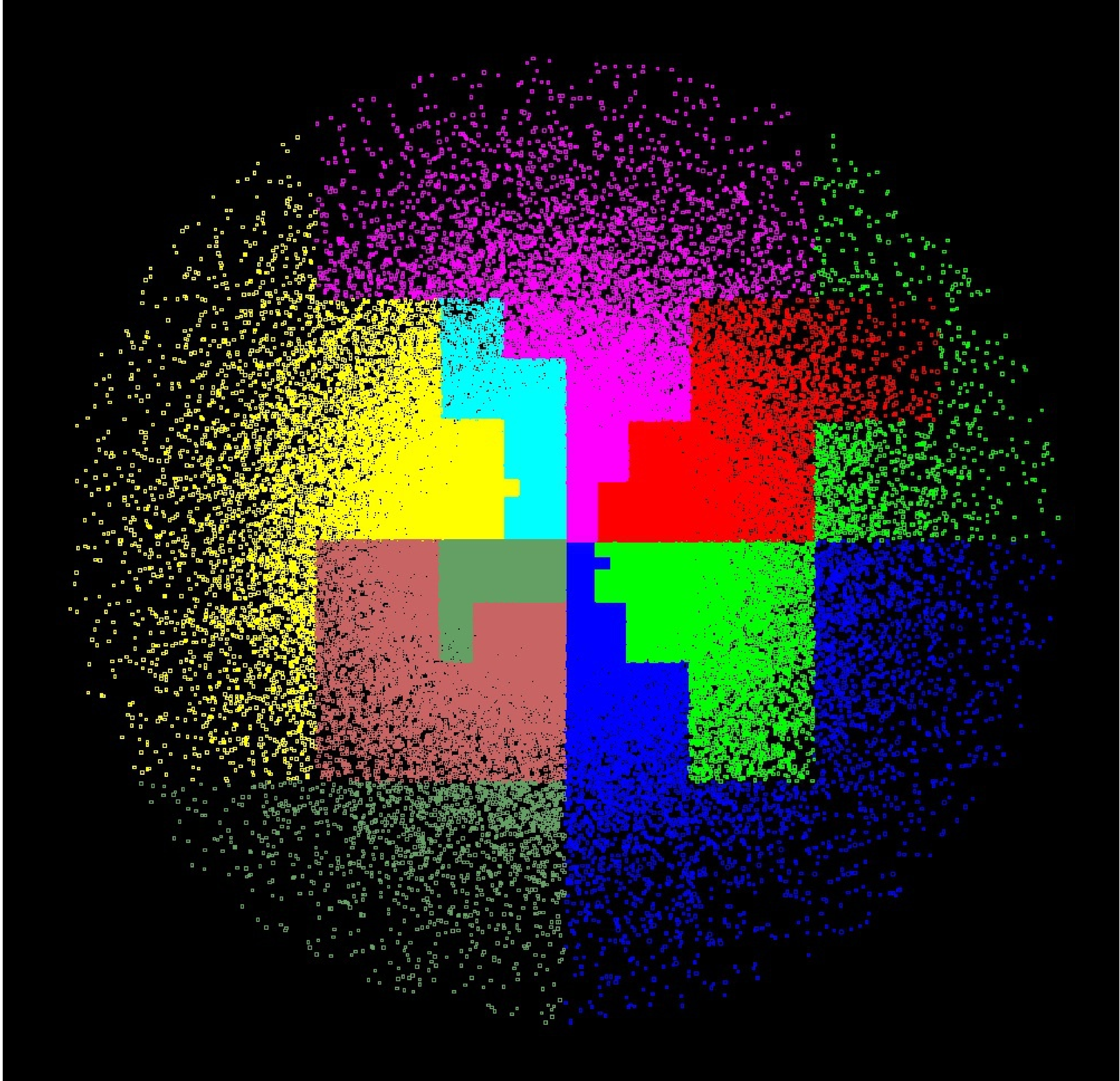
GADGET-1 used a simple orthogonal recursive bisection

EXAMPLE OF DOMAIN DECOMPOSITION IN GADGET-1



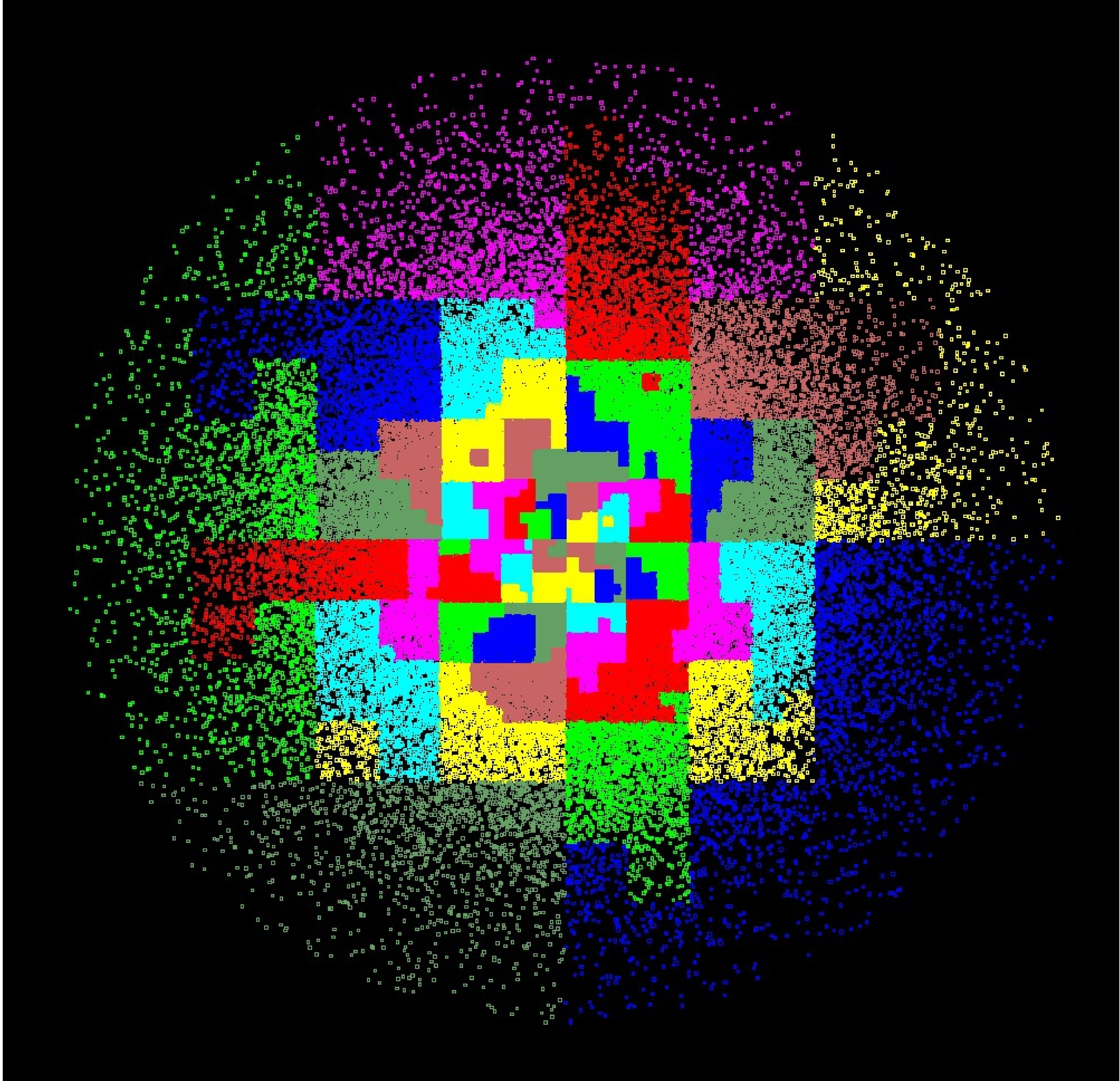
GADGET-2 uses a more flexible space-filling Peano-Hilbert curve

EXAMPLE OF DOMAIN DECOMPOSITION IN GADGET-2



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

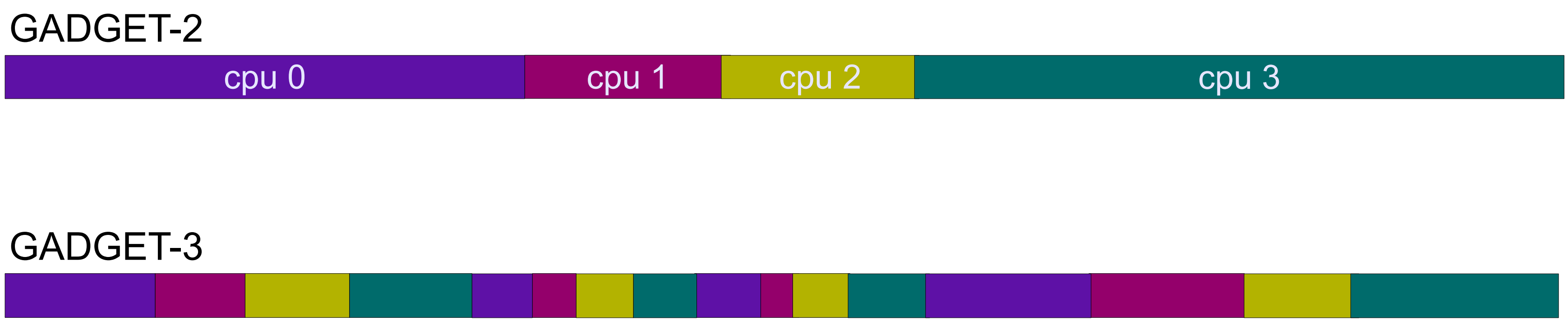
EXAMPLE OF DOMAIN DECOMPOSITION IN GADGET-3



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

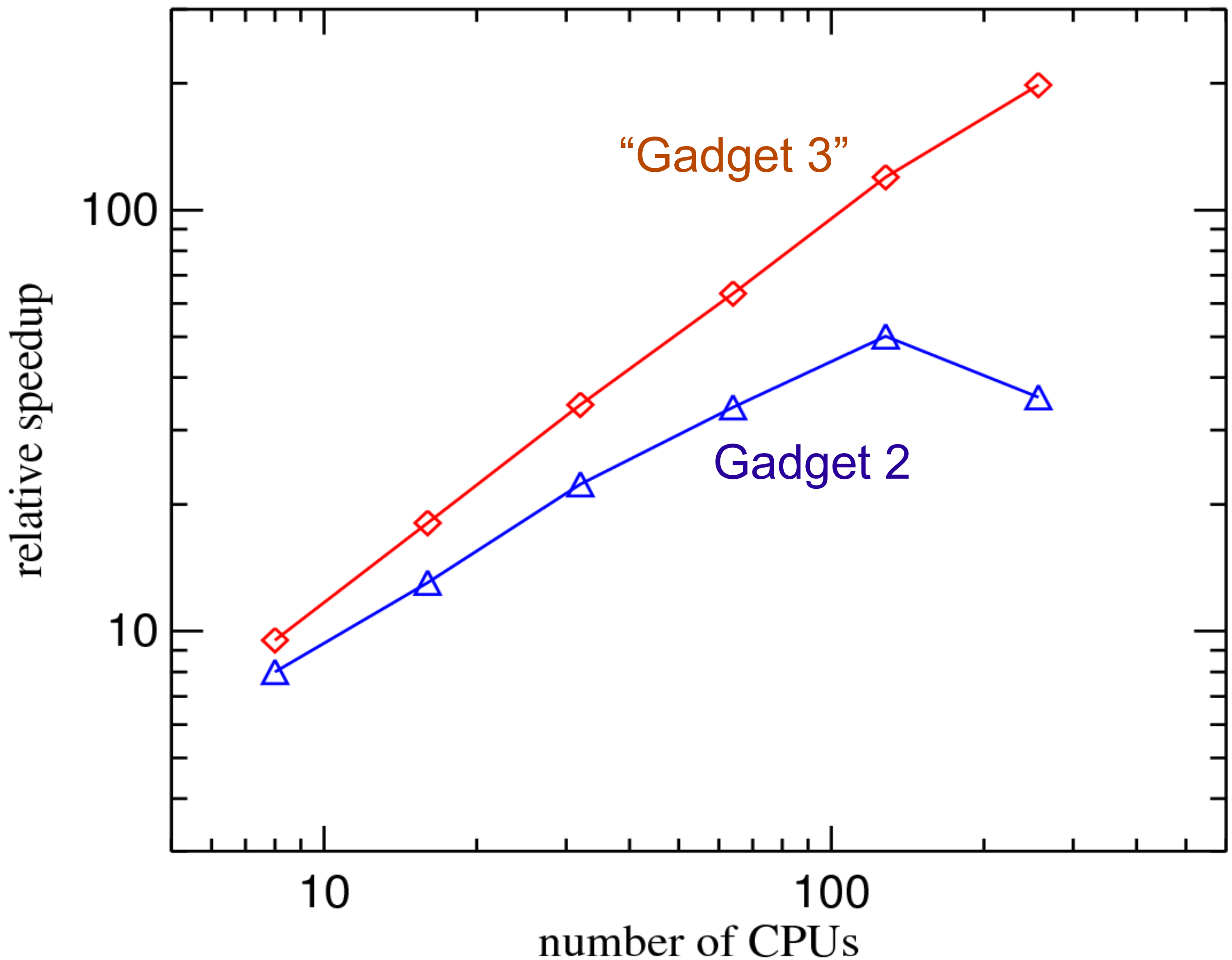


→ **But:**

- 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



Scaling of the AREPO code on Ranger

WEAK SCALING OF ALL CODE COMPONENTS