Zoomed initial conditions with Ginnungagap

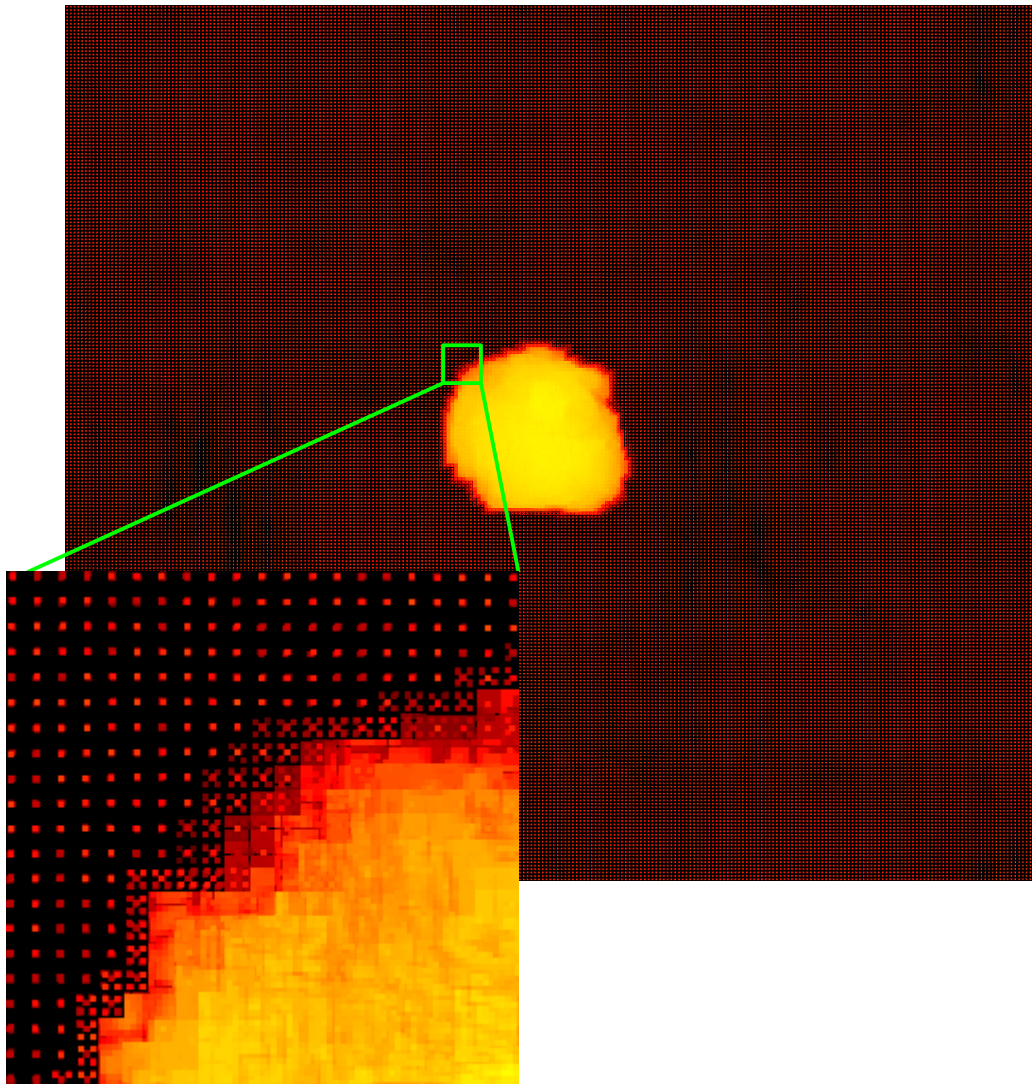
S. Pilipenko (LPI), G. Yepes (UAM)

Ginnungagap code

- Scalable and flexible generator of initial conditions with industry-style configuration
- Initially written by S. Knollmann
- Project home: https://code.google.com/p/ginnungagap/
- Ginnungagap reads or creates white noise fields and writes velocity fields in grafic or HDF5 format
- Tools: generateICs, realSpaceConstraints, estimateMemReq, ...
- The name means primordial chaos in the Nordic mythology

Zoomed Initial Conditions

- Ginnungagap already had the tools to scale *white noise* fields
- The approach:
  - produce several levels, in steps of 2 in resolution
  - produce velocity fields
  - combine particles from different levels according to *mask*



Requirements

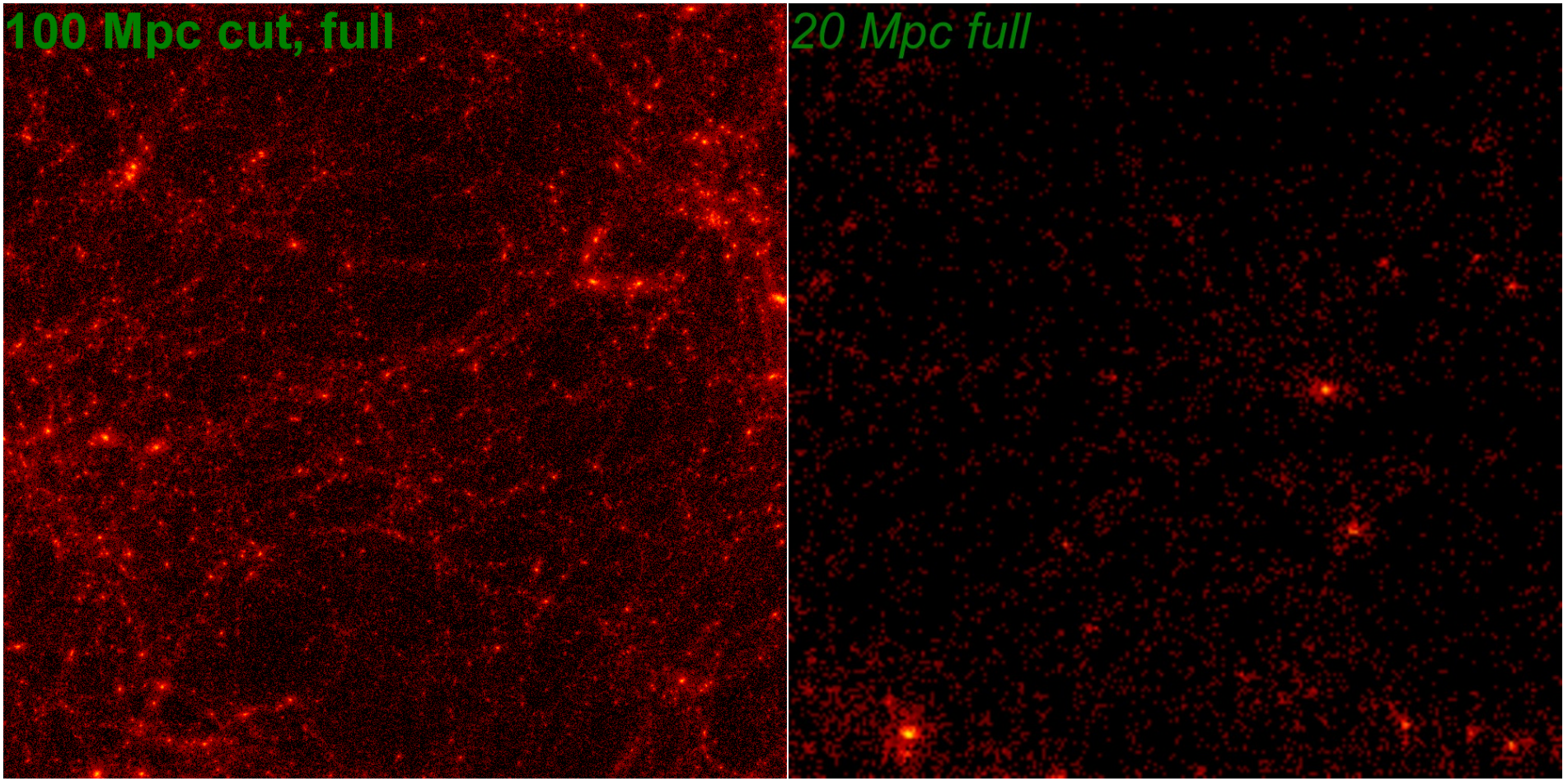
- Flexibility: any number of levels, all combinations of usage of GADGET particle types, with or without individual particle masses
- MPI & OpenMP parallelization
- Effective memory usage (no overheads)
- Effective usage of disk resources (avoid writing full velocity fields)
- Ease of use, exhaustive documentation
- Tool for preparing the mask

Tests

1) Run zoomed simulation and the full box with the resolution of the highest zoom level. As an example, one of new CLUES constrained 500/h Mpc boxes is used. Number of levels = 2. Highest resolution = 512^3. Zoom region is a R=100 Mpc/h sphere in *Lagrangian coordinates*.
2) Demonstrate that it is possible to run zoomed simulations suitable to search for the Local Group in these 500/h Mpc boxes. This requires particle mass ~10^9 M_{sun} and effective resolution of 2048^3.

Results – 2 levels

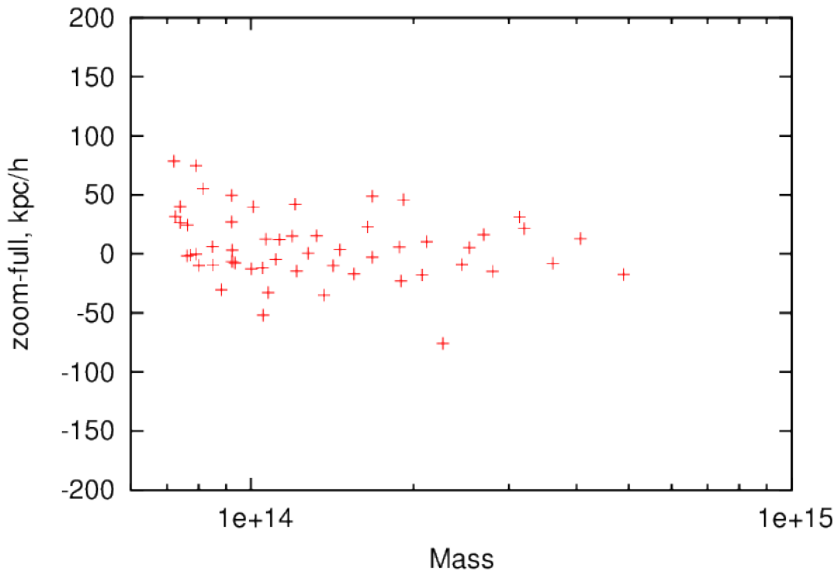
Sphere of D=200 Mpc was zoomed



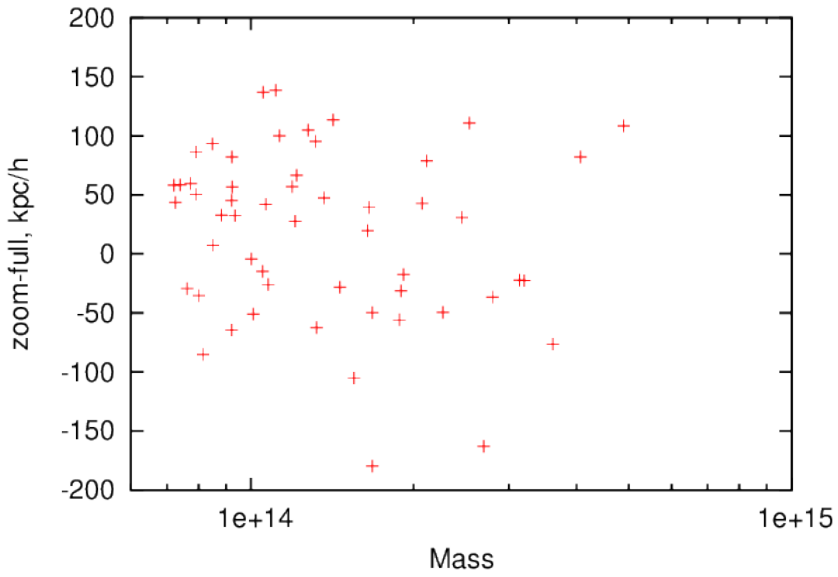
Results – 2 levels: coordinates

Halos were found using AHF and cross-identified by two methods (by particle IDs and by coordinates)

Using the same GADGET setup as the full box



Using -DPLACEHIGHRESREGION



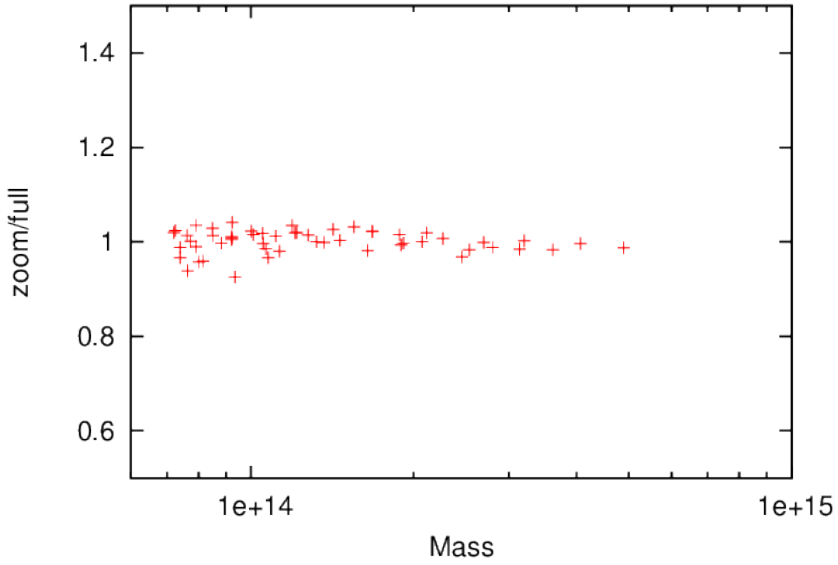
Softening length was ε=25 kpc/h

mean = 7 kpc/h
stddev = 29 kpc/h

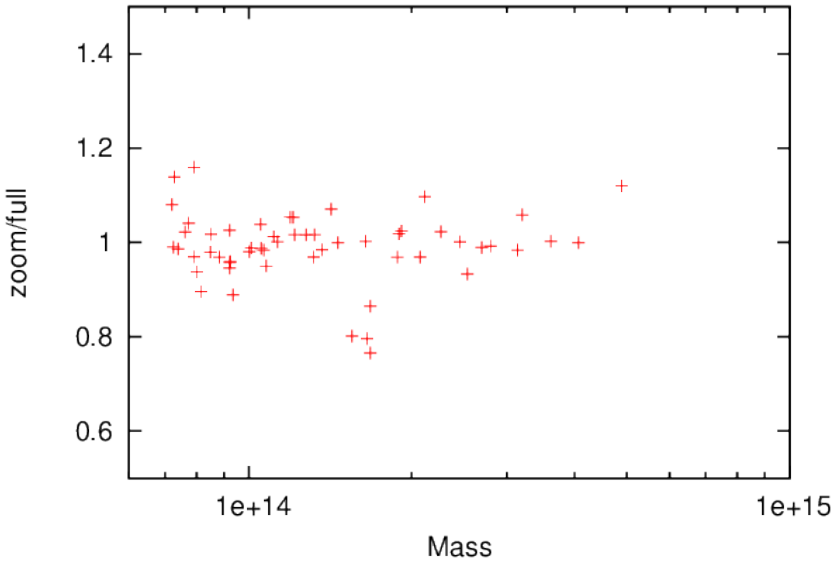
mean = 21 kpc/h
stddev = 77 kpc/h

Results – 2 levels: masses

Using the same GADGET setup as the full box



Using -DPLACEHIGHRESREGION

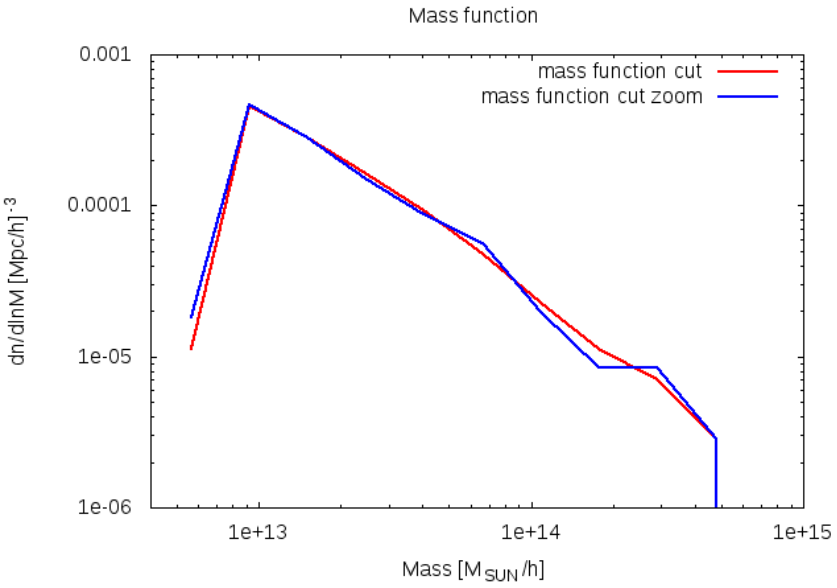


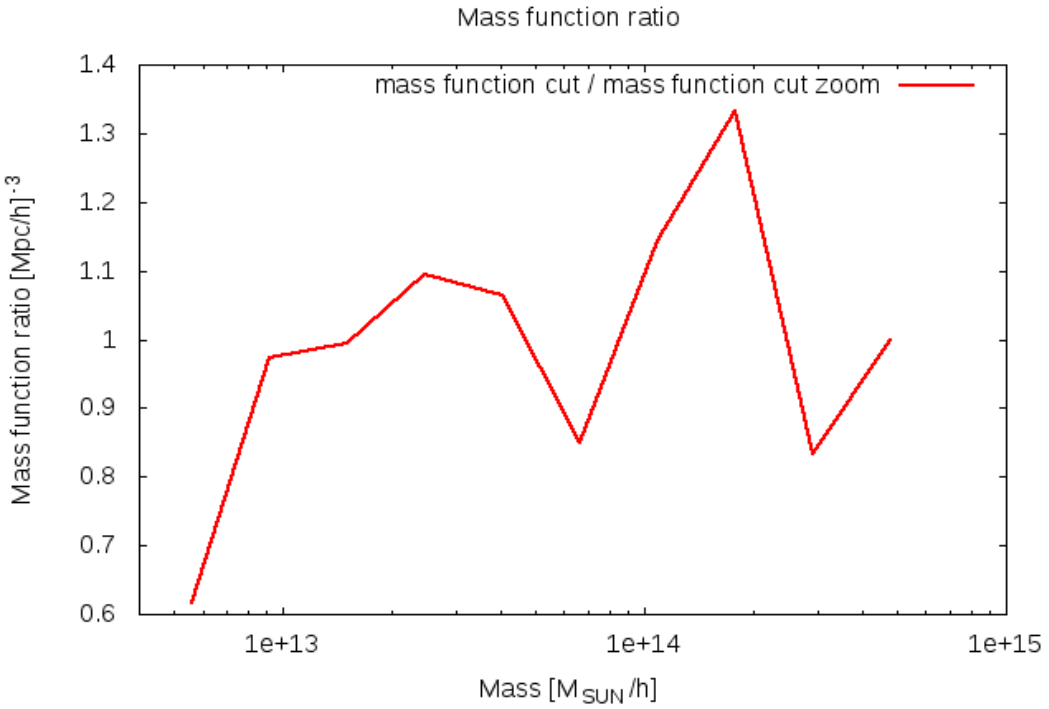
$\log(M_{zoom})-\log(M_{full})$:

mean = 2e-4
stddev = 0.02

mean = -0.04
stddev = 0.18

Results – 2 levels: mass functions

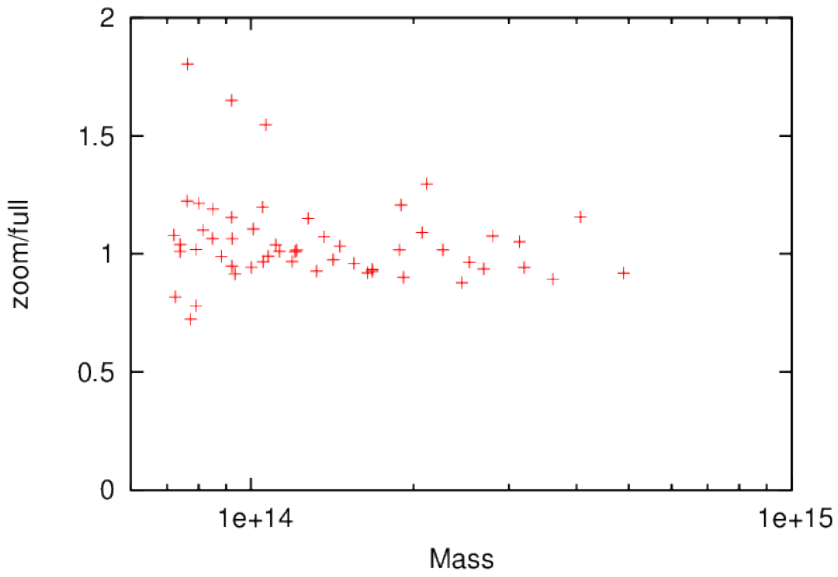




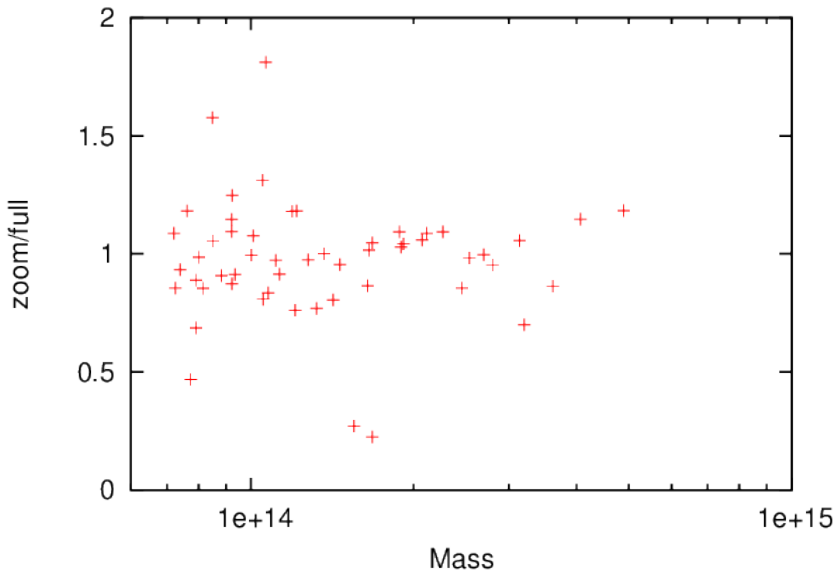
The Kolmogorov-Smirnov distance is 0.051,
probability 25.6% ≈ 1 σ
(Thanks to V. Yankelevich)

Results – 2 levels: concentrations

Using the same GADGET setup as the full box



Using -DPLACEHIGHRESREGION



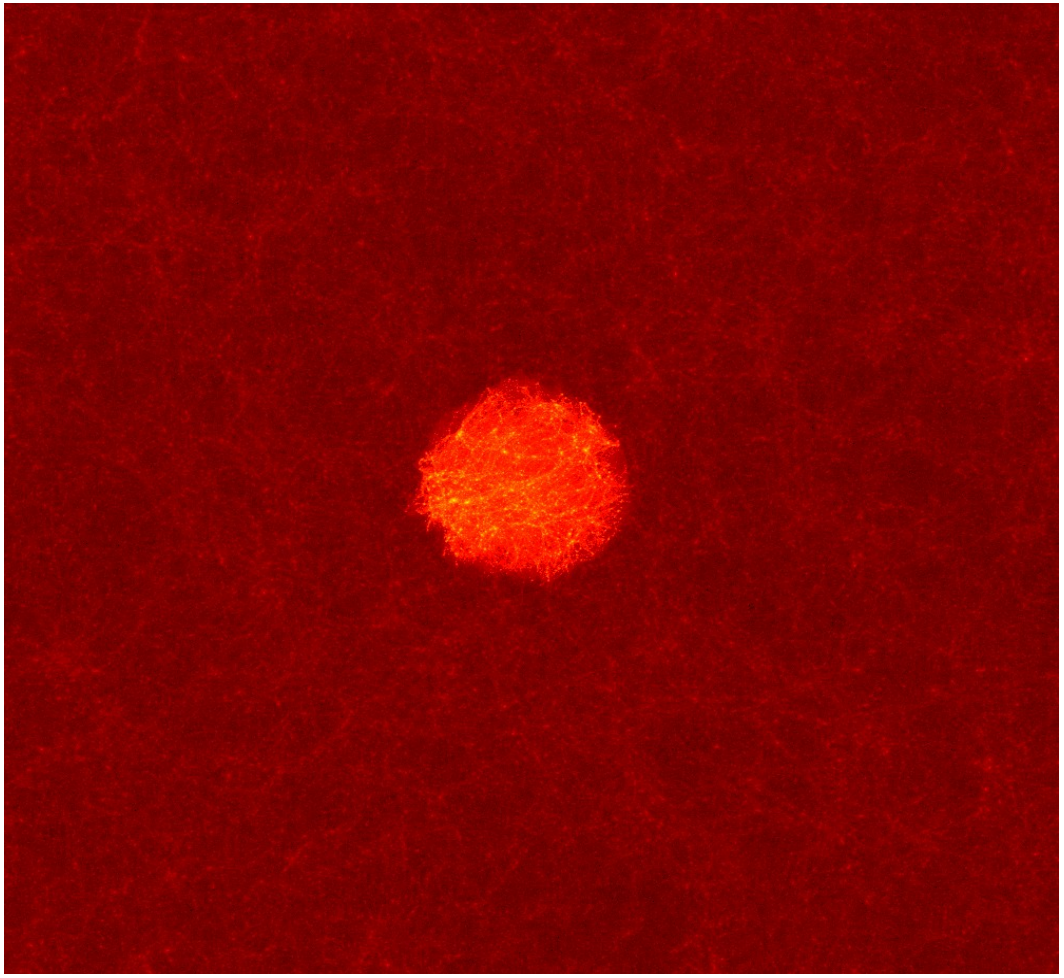
$\log(C_{zoom})-\log(C_{full})$:

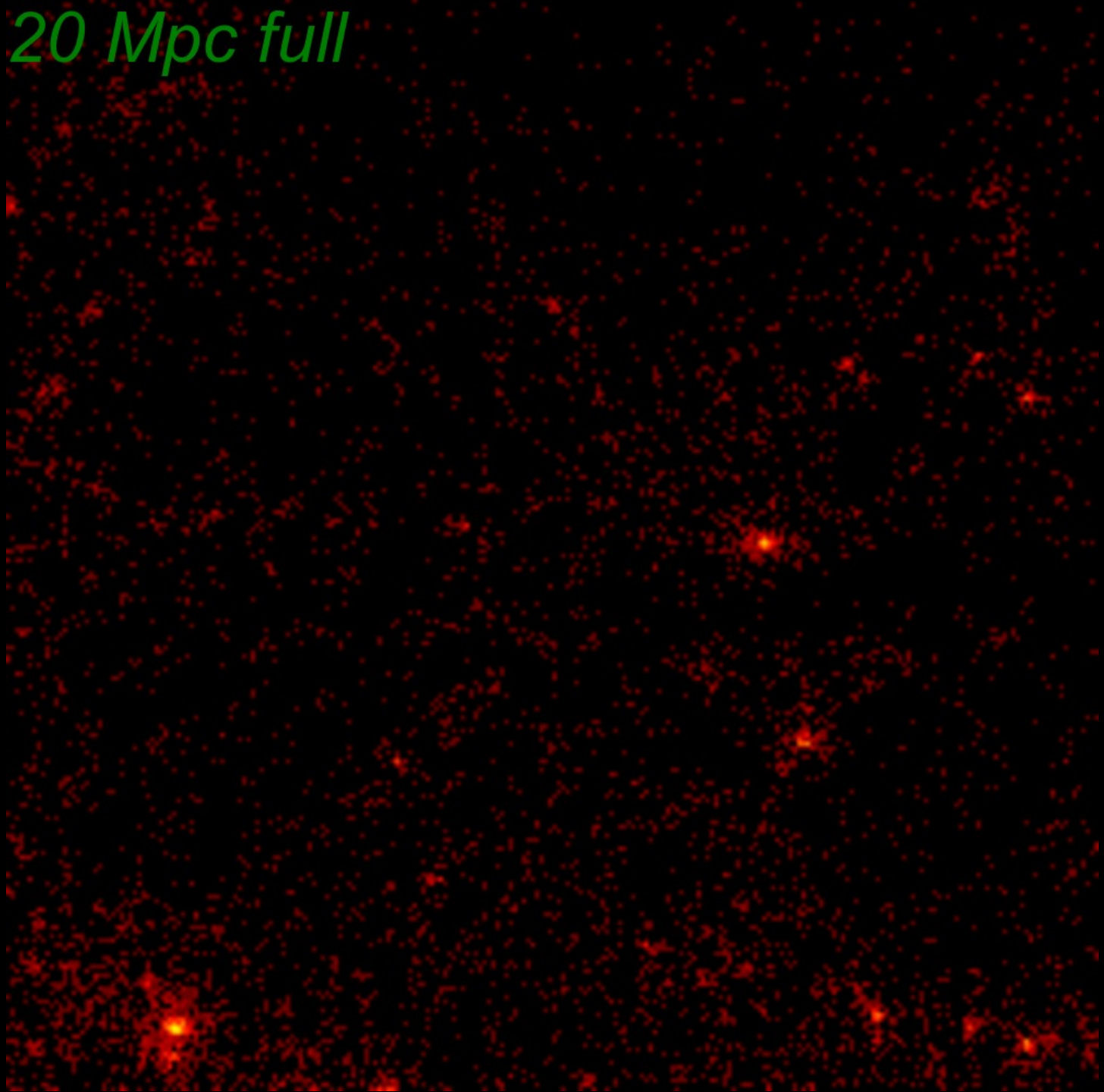
mean = 0.04
stddev = 0.16

mean = -0.07
stddev = 0.32

Results – 2048^3

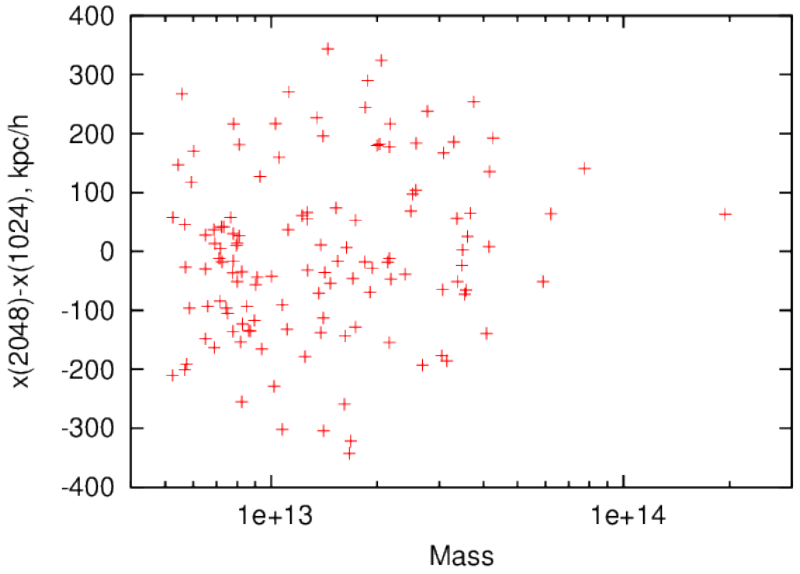
- Sphere of radius 44/h Mpc in *Eulerian coordinates* was simulated
- No haloes with M>10^14 within 2 Mpc from boundary
- ~1000 CPU-h




20 Mpc full

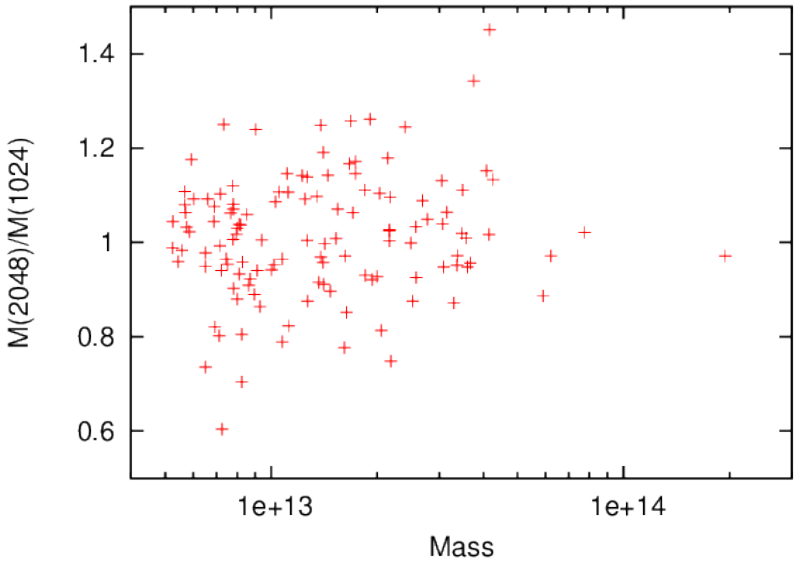
Results – 2048^3 vs 1024^3

Coordinates



mean -2.5 kpc/h
stddev 144 kpc/h

Masses



log(M1)-log(M2):

mean 0.015
stddev 0.15

Making zoomed initial conditions

- Obtaining modified ginnungagap:

```
git clone -b zoom https://code.google.com/p/ginnungagap/
```

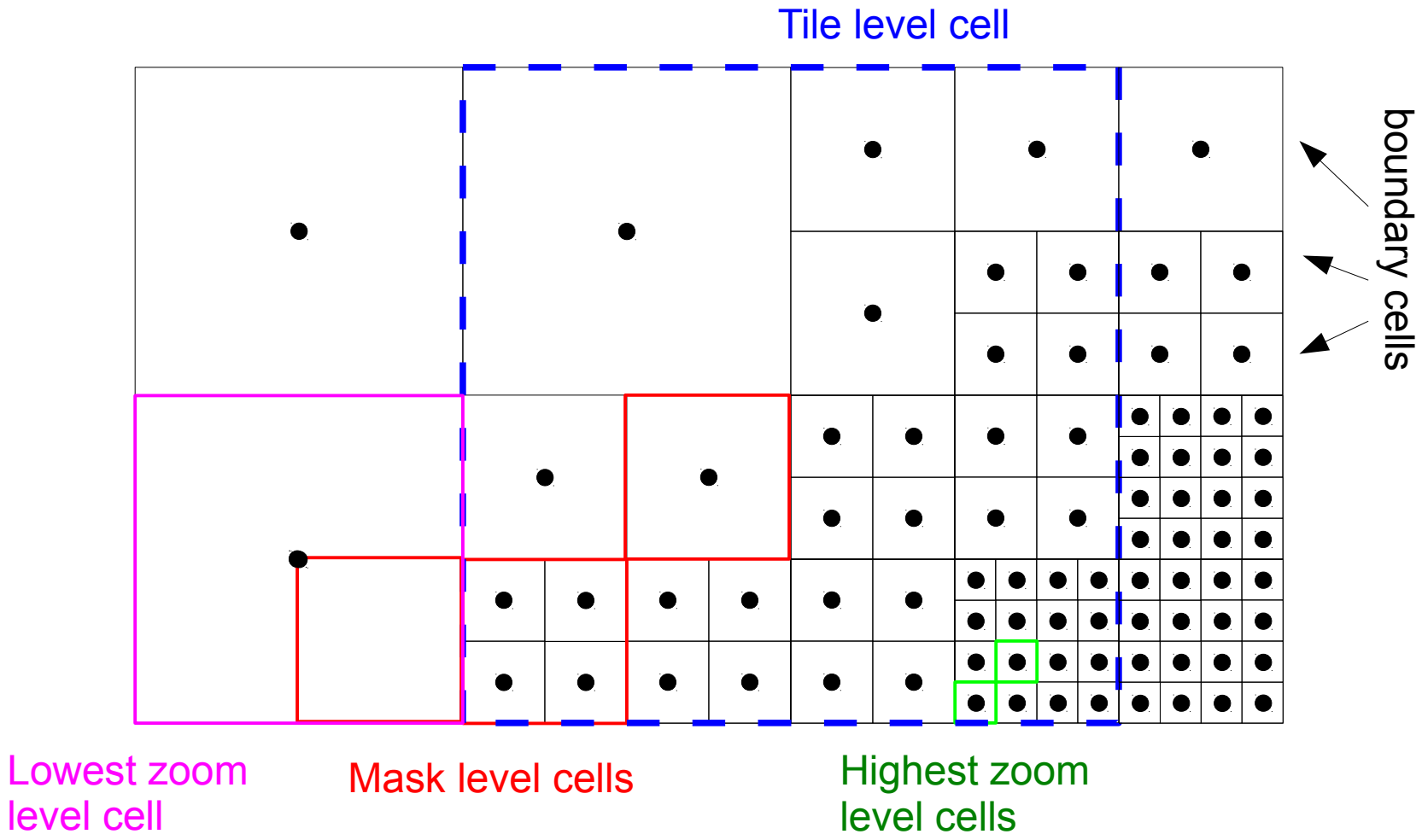
- make the documentation first: cd to doc/, make, cd to ref/ and see index.html
- Read the 'Related pages'

Making zoomed initial conditions

```
[GenerateICs]
ginnungagapSection = Ginnungagap
doGas = false
doLongIDs = true
bufferSection = Buffer
inputSection = GenicsInput
outputSection = GenicsOutput
cosmologySection = Cosmology
maskSection = Mask
hierarchySection = Hierarchy
zoomLevel = 9
typeForLevel7 = 4
typeForLevel8 = 3
typeForLevel9 = 2
typeForLevel10 = 1

[Mask]
maskLevel = 7
minLevel = 7
maxLevel = 10
tileLevel = 1
readerType = legacy
readerSection = Lare

[Hierarchy]
numLevels = 11
minDim1D = 2
factor = 2

[Lare]
hasHeader = false
fileName = lare.dat
ngrid = 256 256 256
```

```
[GenicsInput]
velxSection = GenicsInput_velx
velySection = GenicsInput_vely
velzSection = GenicsInput_velz

[GenicsInput_velx]
type = hdf5
path = ./
prefix = ic_1024
qualifier = _velx
suffix = .h5

[GenicsInput_vely]
type = hdf5
path = ./
prefix = ic_1024
qualifier = _vely
suffix = .h5

[GenicsInput_velz]
type = hdf5
path = ./
prefix = ic_1024
qualifier = _velz
suffix = .h5

[GenicsOutput]
numFilesForLevel7 = 1
numFilesForLevel8 = 1
numFilesForLevel9 = 1
numFilesForLevel10 = 1
prefix = gzlv
```

Looking inside GenerateICs: hierarchy, levels, masks...



The Hierarchy

- base level dimension
- number of levels
- step factor (usually 2)
- minimal zoom level
- maximal (highest) zoom level
- mask level

```
[Hierarchy]
numLevels = 7
minDim1D = 4
factor = 2

[Mask]
maskLevel = 4
minLevel = 3
maxLevel = 6
tileLevel = 0
```

In this example:
tile level dims = 4
min level dims = 4*2^3 = 32
mask level dims = 4*2^4 = 64
max level dims = 4*2^6 = 256

The Mask

- The initial mask (the Lagrangian region) is set on the **maskLevel.**
- It is read as a list of cells, in which particles for the **highest level** will be placed.
- Then, the mask is expanded to lower levels to make **boundaries**.

The tiles

- They are used for the effective memory usage
- Every process reads only one tile at once
- Smaller tiles = less memory used, but more disk operations
- Larger tiles = more memory used, less effective parallelization

Input & output files and parallelization

- Each zoom level is written in at least one GADGET file
- Each level requires a separate run of generateICs with its own **.ini** file because of:
  - input velocity fields for each level
  - how many processes to use
- Each MPI process should have an output file, so $N_{MPI} \leq N_{files}$ for each level
- The memory should have space for one full file for each MPI process
- If some zoom level has several output files, the volume is divided between files by **tiles**.

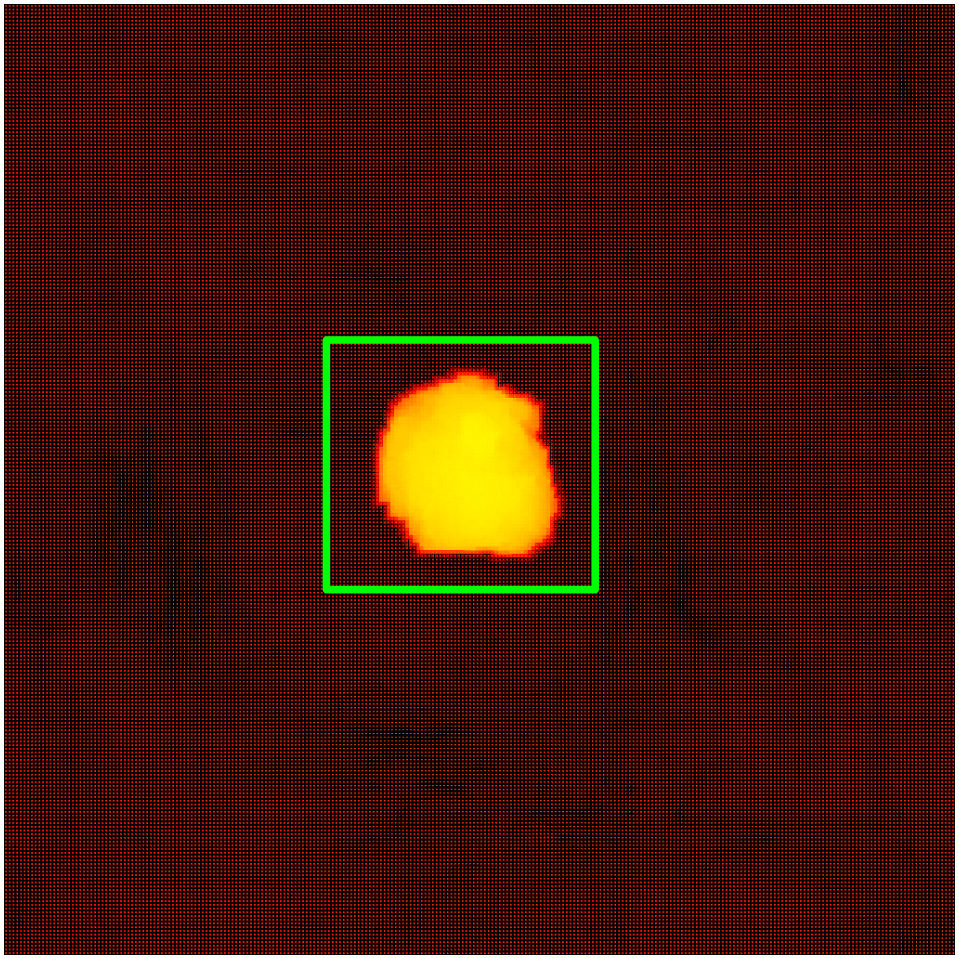
Controlling the particle types and masses

```
[GenerateICs]
...
typeForLevel3 = 3
typeForLevel4 = 3
typeForLevel5 = 2
typeForLevel6 = 1
```

- If two or more levels have the same type, they will be assigned particle masses for each particle. Otherwise massArr is used. Type 0 is gas, type 1 is halo, ...

```
[GenicsOutput]
numFilesForLevel3 = 1
numFilesForLevel4 = 1
numFilesForLevel5 = 1
numFilesForLevel6 = 2
```

Writing a specified region of velocity fields with ginnungagap

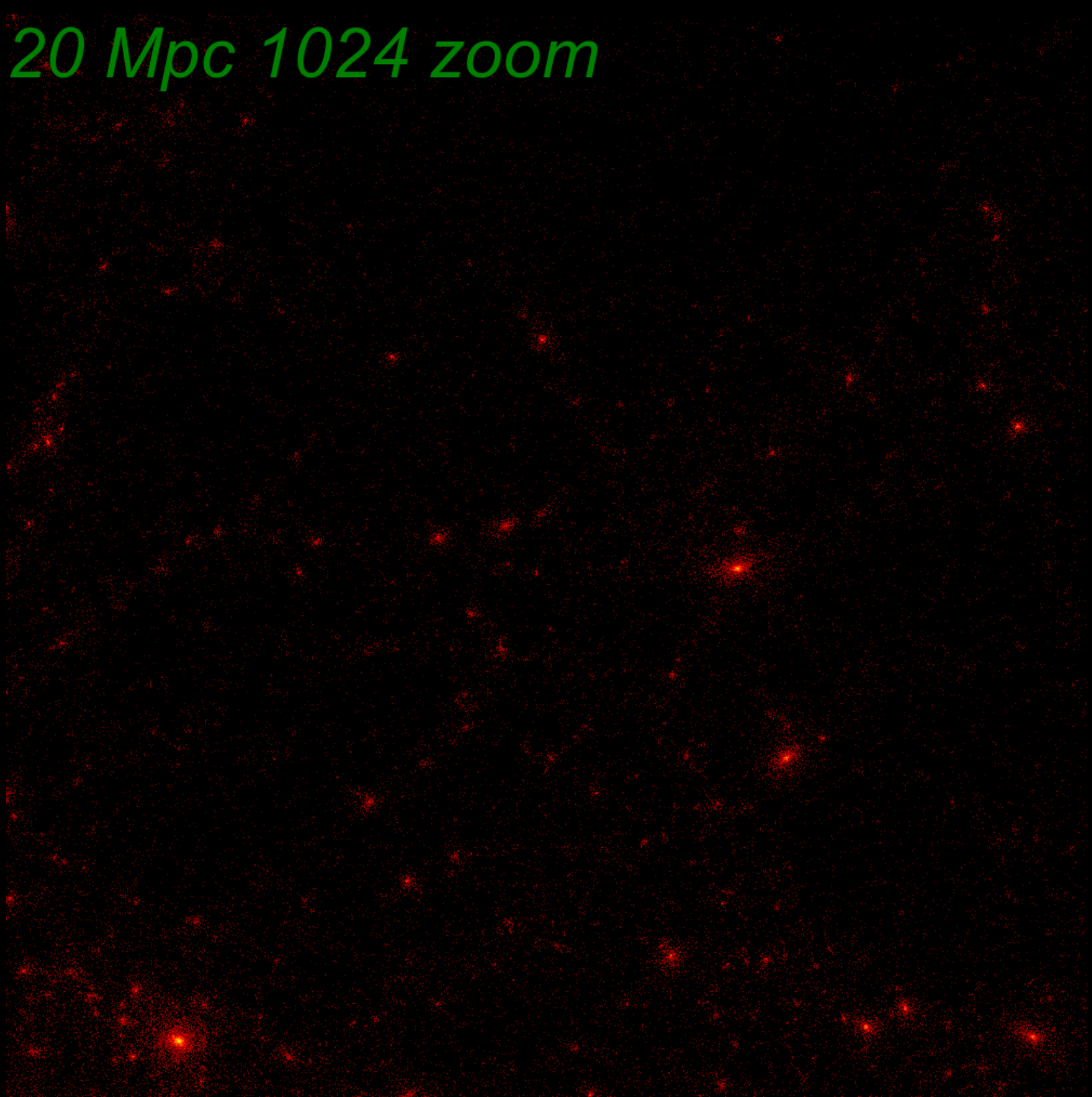


Writing a specified region of velocity fields with ginnungagap

- The disk space is needed to store white noise fields:

  4*(dim1D^3) bytes
- For velocities: +12*(dim1D^3) normally or a tiny fraction if we write only the essential part
- E.g. on SuperMUC, using 8 000 nodes and 16 Gb/node it should be possible to generate 16384^3 ICs.

```
doPatch = true
patchLo = 384 384 384
patchDims = 256 256 256
```


20 Mpc 1024 zoom

TODO

- region to write – the .ini file entry
- compatibility with previous versions .ini files
- test if this works with gas
- more examples in documentation