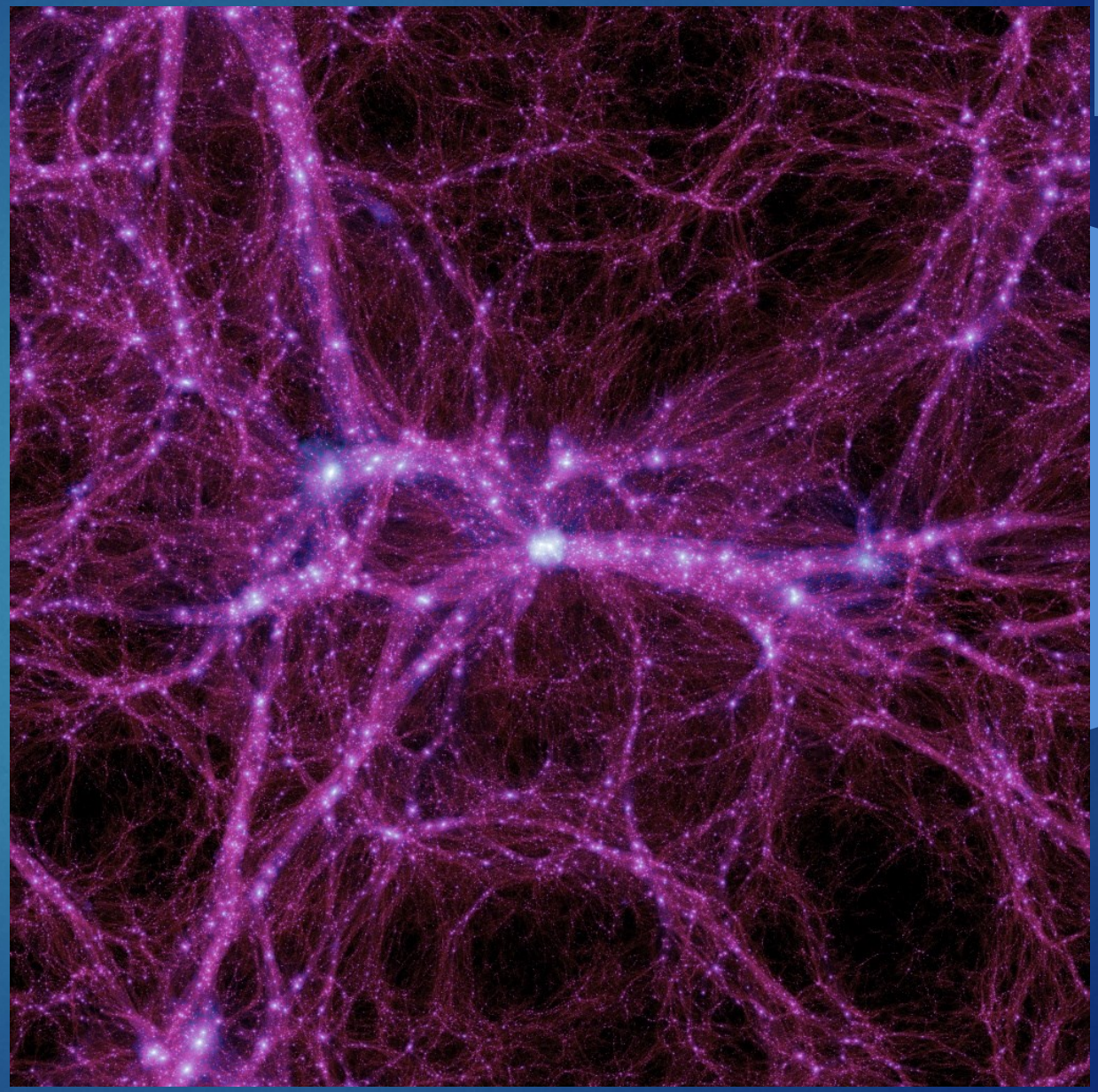
FileDB

ACCESSING (COSMOLOGICAL) SIMULATIONS FROM THE DATABASE

Millennium-II Simulation

- 100 Mpc/h
- 10^{10} particles
- 6.9 10^{6} M_{sun}/h
- ~10 million halos
- ~300GB/snapshot

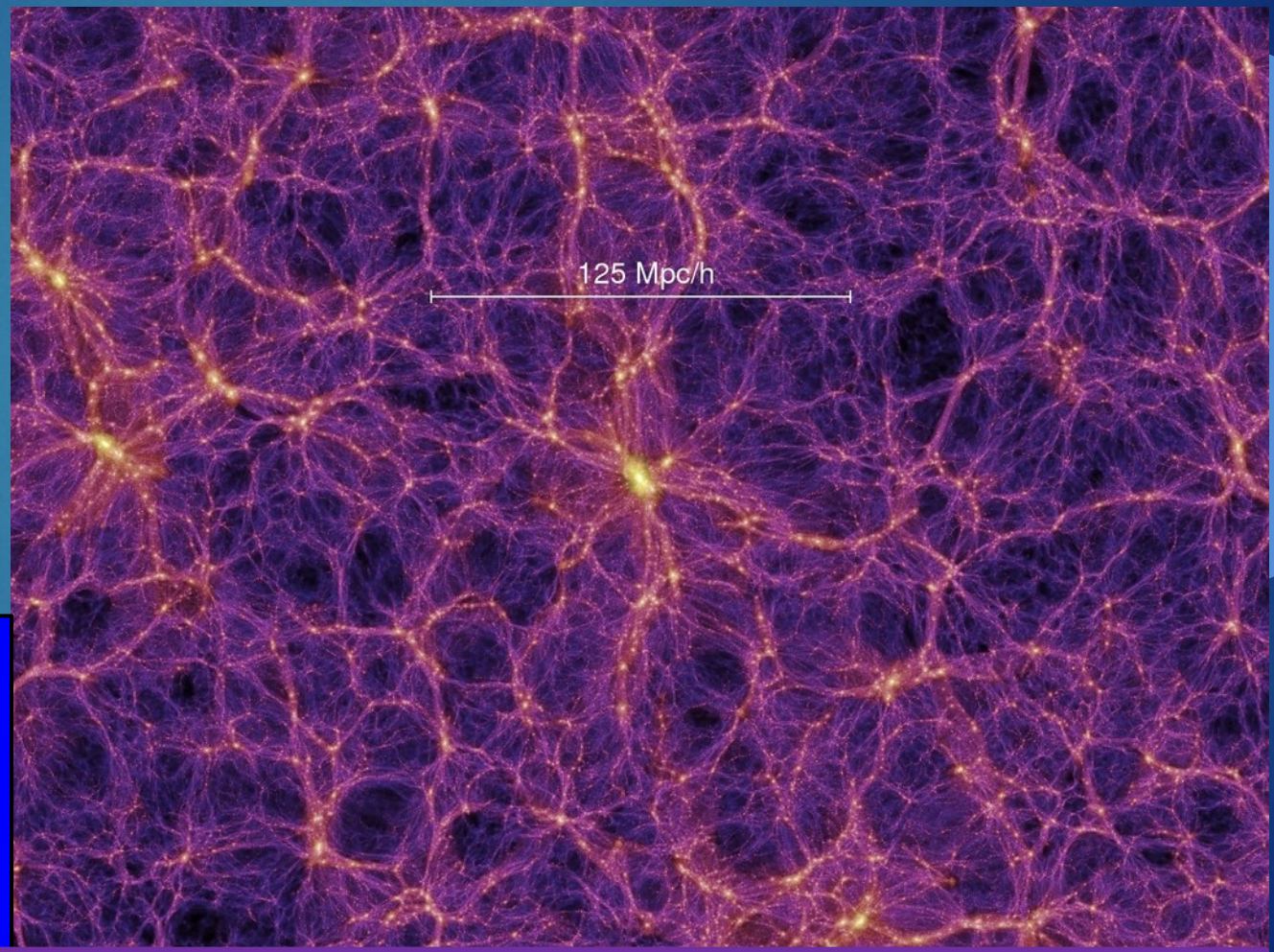
Boylan-Kolchin etal 2009



Millennium Simulation

- 500 Mpc/h
- 10^{10} particles
- 8.6 10^{8} M_{sun}/h
- ~18 million halos
- ~300GB/snaphot

Springel etal 2005

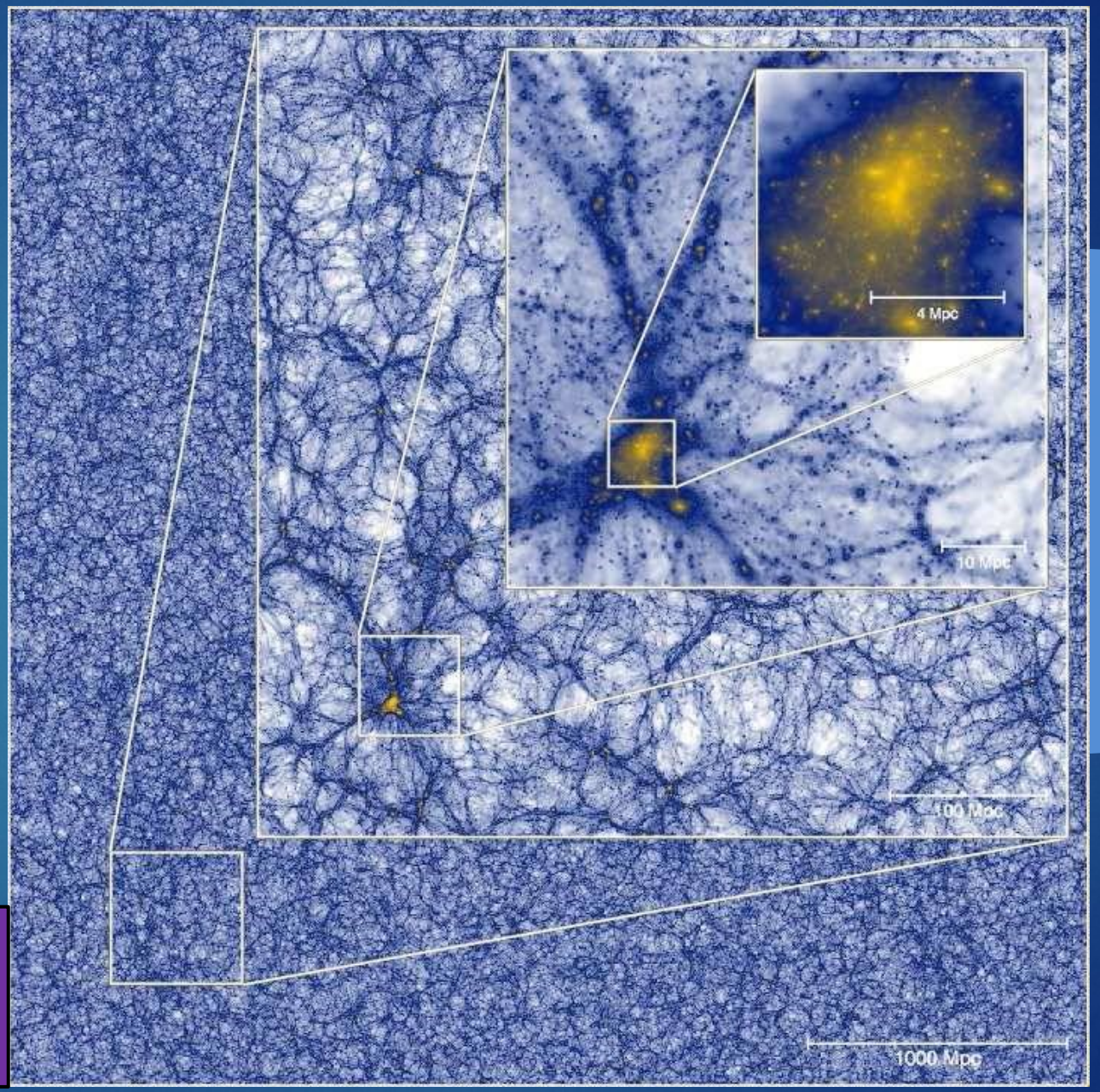


MRII

MR-XXL

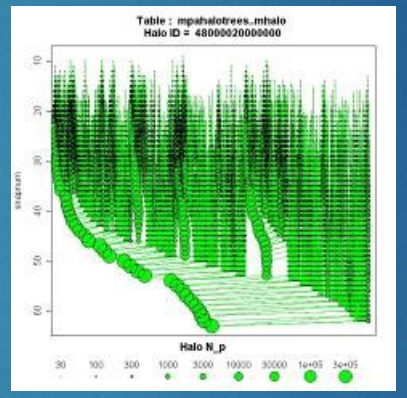
- 3Gpc/h
- $3x10^{11}$ particles
- 750 million halos /snapshot
- 9TB/snapshot

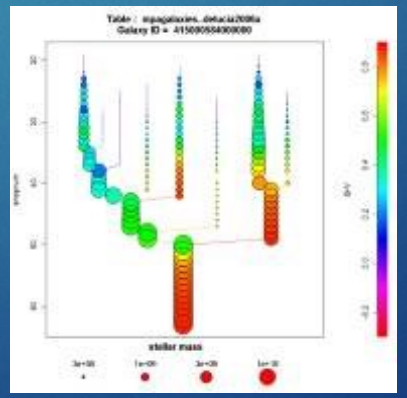
- browse

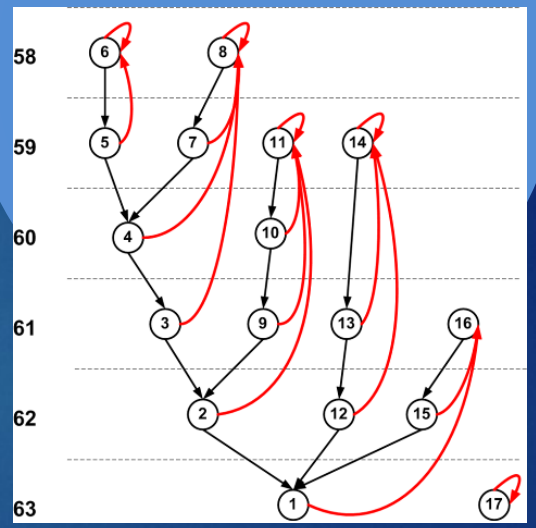


Millennium Database

- Particle clusters
- Merger trees
- SAM galaxies
- Light cones







Access to raw data

- 10-300 billion particles
- Too large to ingest in database
- Would still be nice to query directly

FileDB

- Accessing data outside of db …
- … from within db

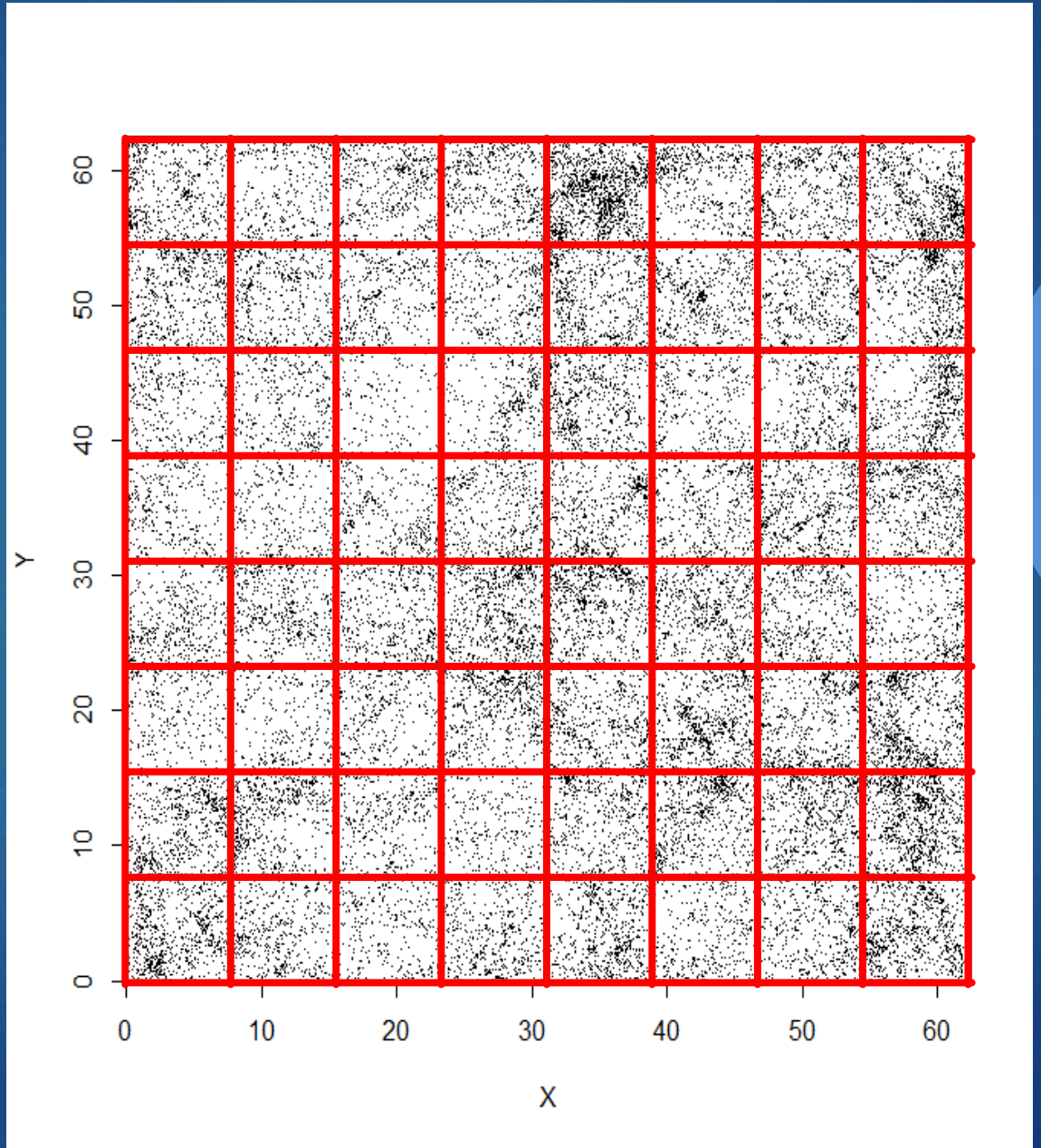
- Using custom user defined table-valued functions
- Written in C#, deployed in MS SQL Server database

```
SELECT *
  FROM MillenniumParticles(63,'SPHERE[0,0,0,10]')
```

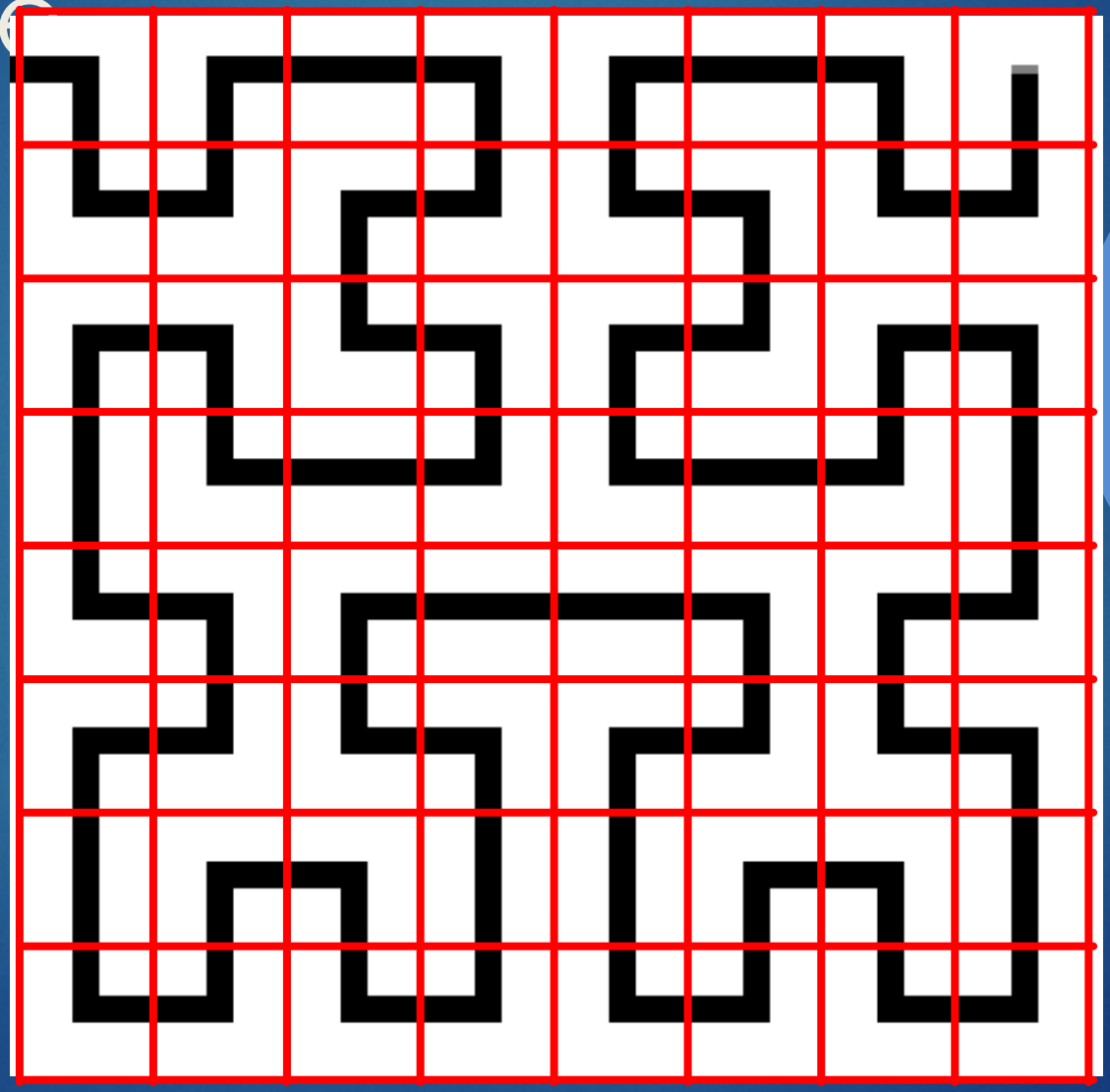
How to make this fast?

- Need 3D spatial index
- Use space-filling curve
  - Morton z-curve
  - Peano-Hilbert curve

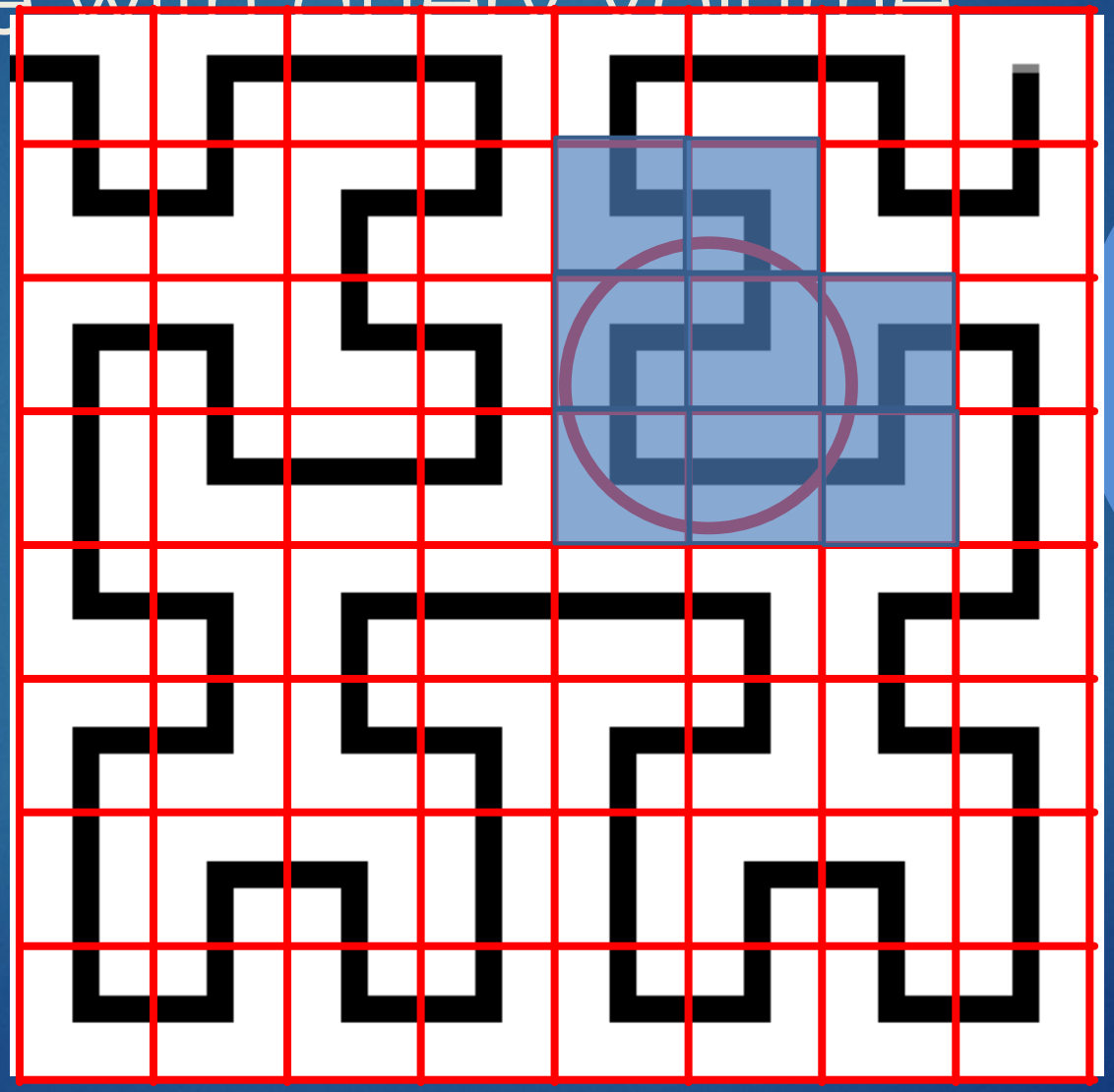
Divide simulation volume in regular grid



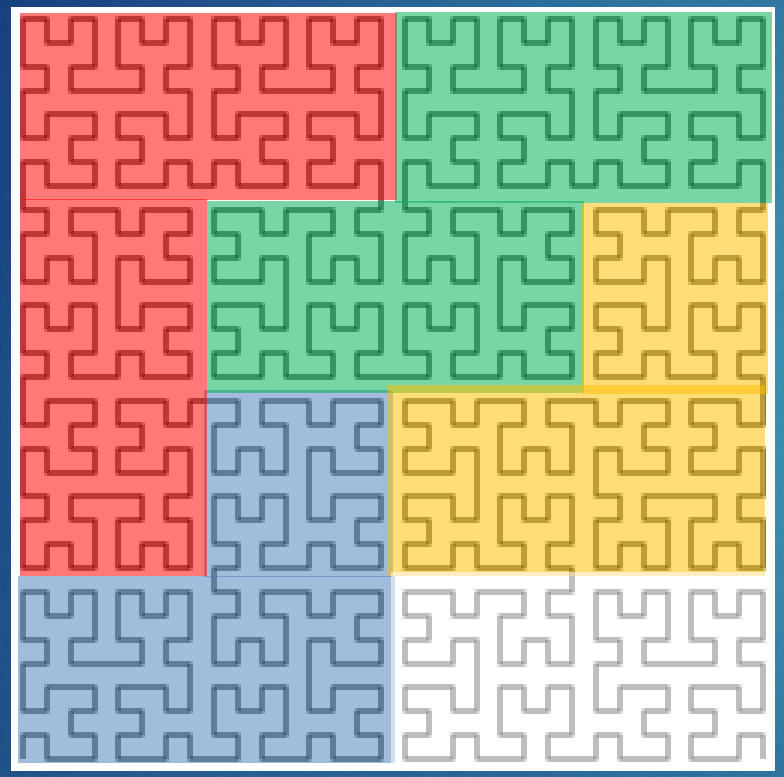
Index cells using space filling curve

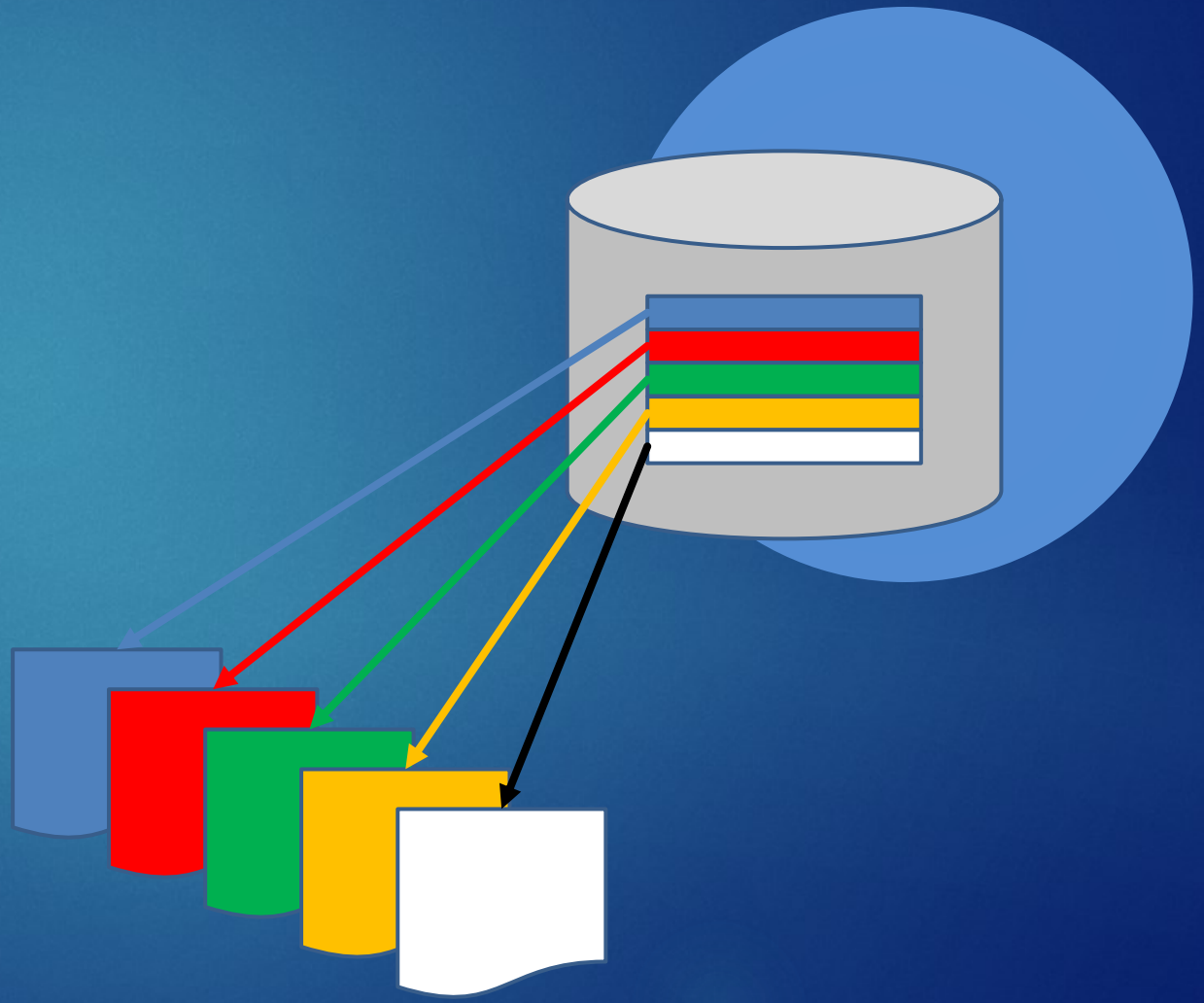


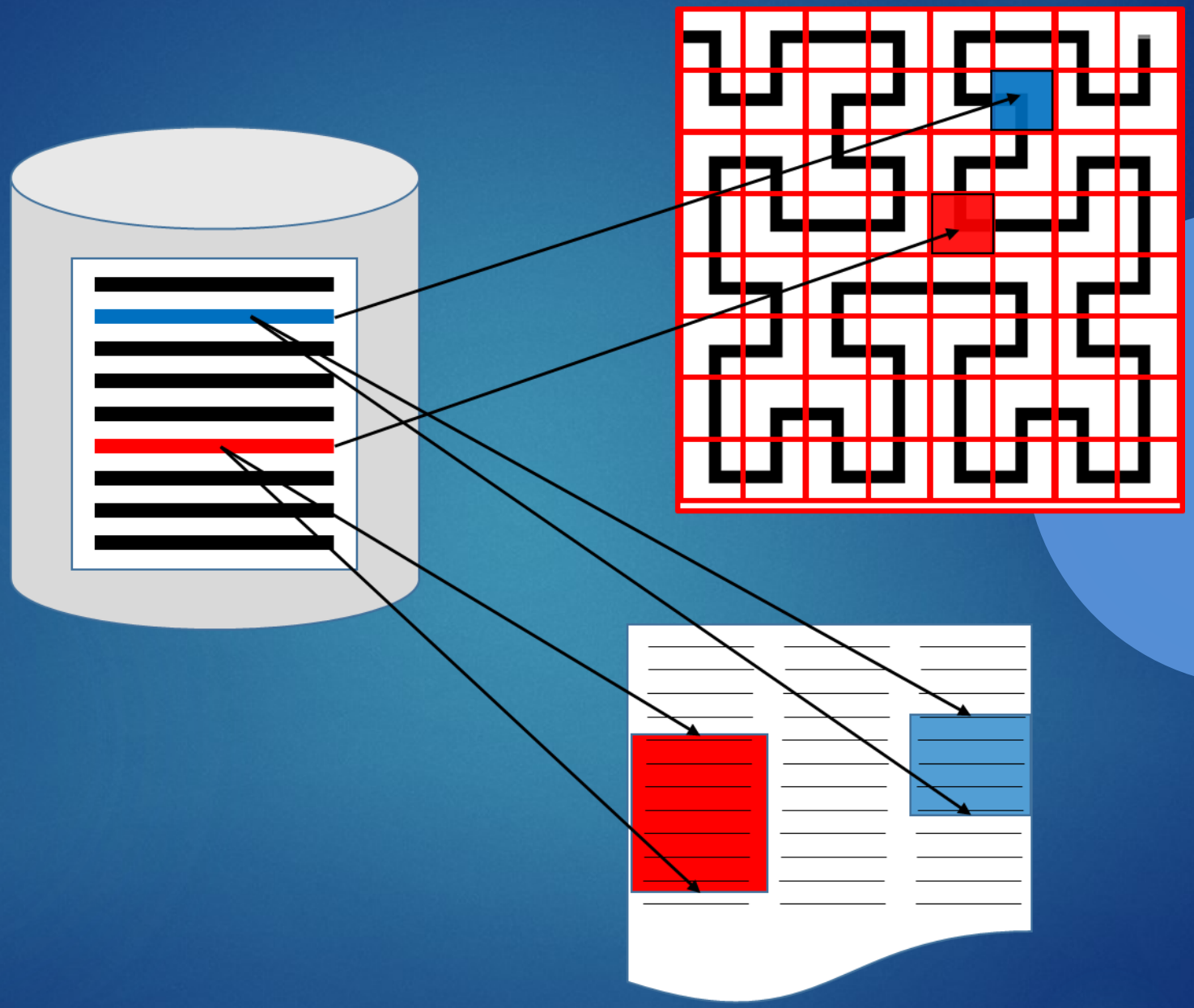
Calculate overlap space filling curve with query volume



Individual files store intervals on curve; indexed in B

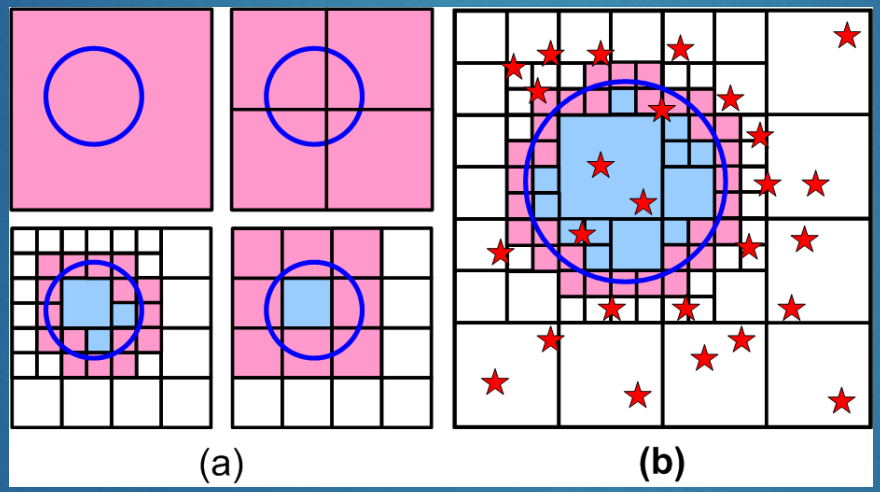


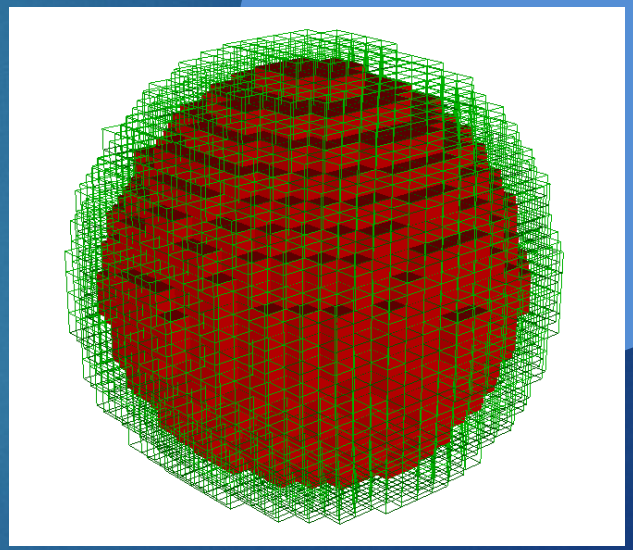




Query particles in sphere/box

- Divide simulation volume in regular grid
- Index using space filling curve
- Calculate overlap space filling curve with query volume





- Execute as table-valued function from database
- Technology: C# for SQLCLR table-valued-function

jupyter CosmoUC_5_PlotHalo Last Checkpoint: 11 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help Notebook saved Python 3

Code CellToolbar

Dark-matter Halos in a Cosmological Simulation

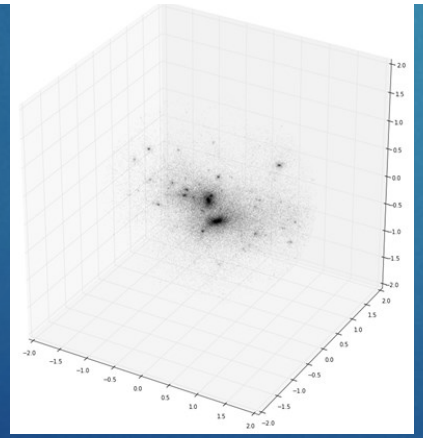
In [3]:
```
import SciServer.LoginPortal as Login
token = Login.getToken()
import SciServer.CasJobs
import pandas
import tables
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

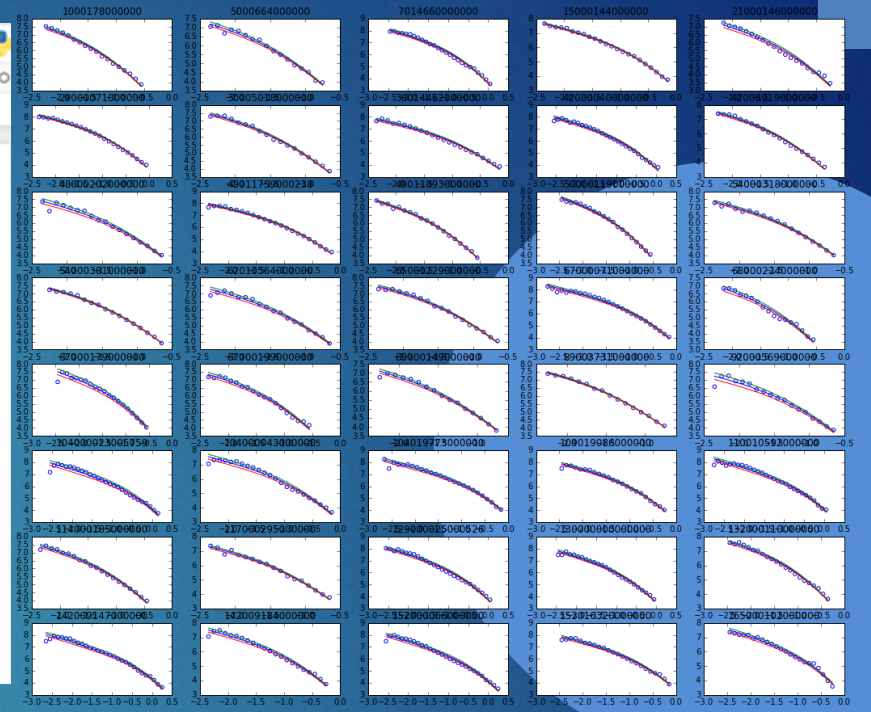
In [10]:
```
%%time
queryString = """
select top 100000 p.x-hh.x as x,p.y-hh.y as y,p.z-hh.z as z
from mpahalotrees.mr hh
   cross apply dbo.MillenniumParticles(hh.snapnum,
     dbo.Sphere::New(hh.x,hh.y,hh.z,3*hh.halfmassradius).ToString()) p
  where hh.haloid=84000007000000 order by newid()
    """
responseStream = SciServer.CasJobs.executeQuery(queryString, token=token,context="SimulationDB")
df = pandas.read_csv(responseStream, index_col=None)
```

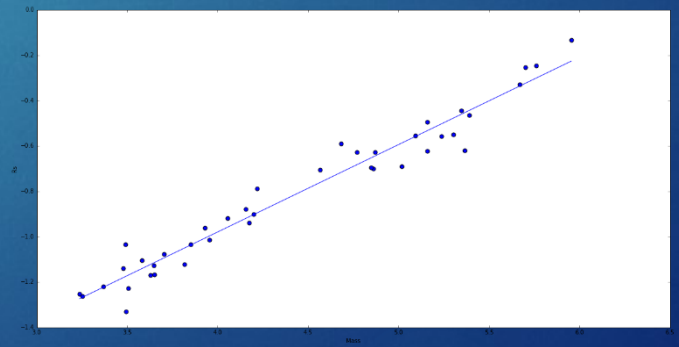
```
CPU times: user 351 ms, sys: 184 ms, total: 535 ms
Wall time: 5.27 s
```

In [13]:
```
fig = plt.figure(figsize=(15, 15))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df.x,df.y, df.z,s=0.001)
```

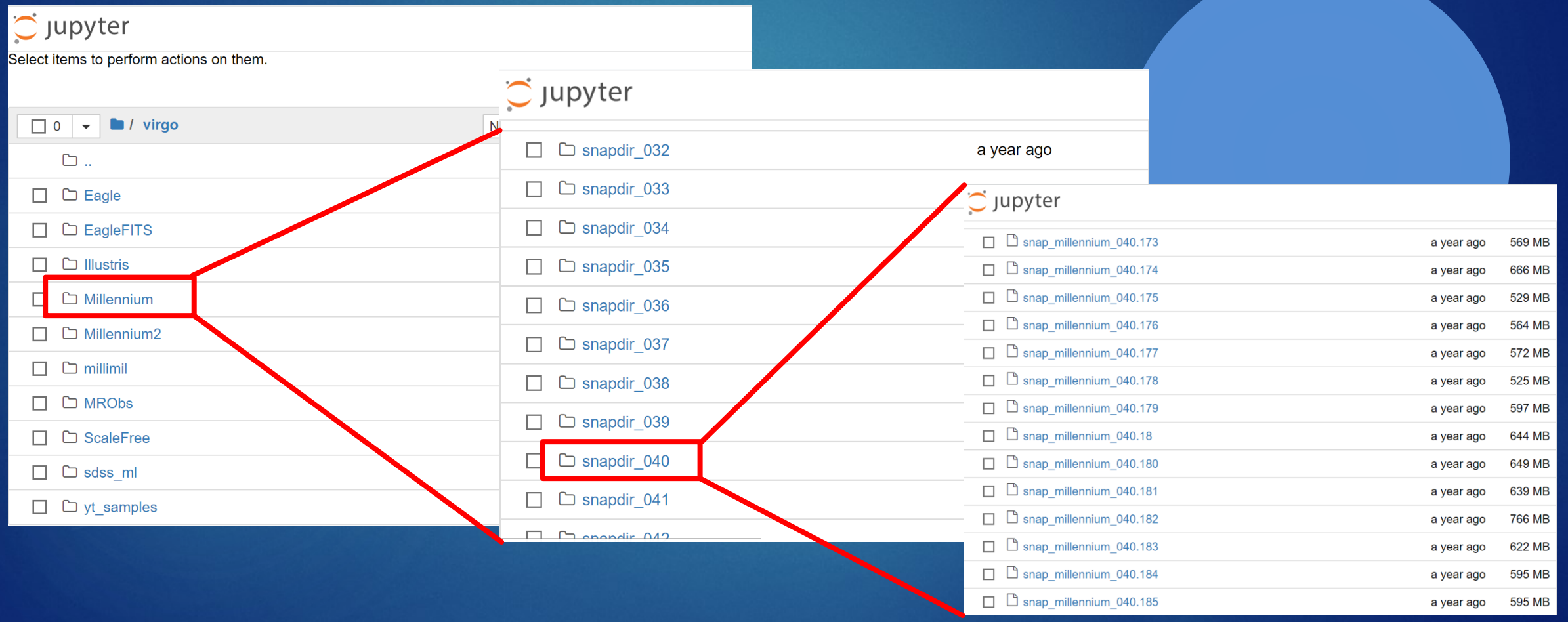
Out[13]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fe86428b9e8>





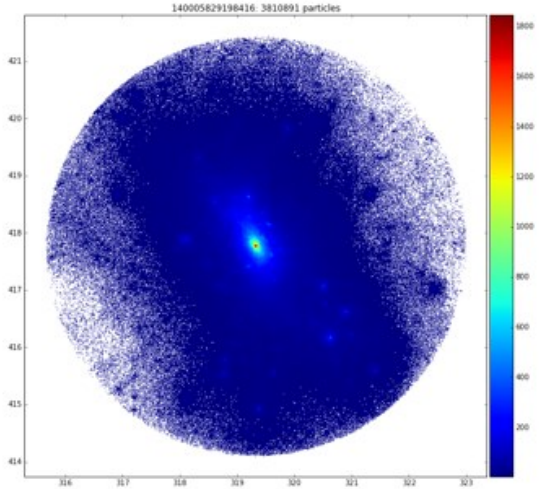


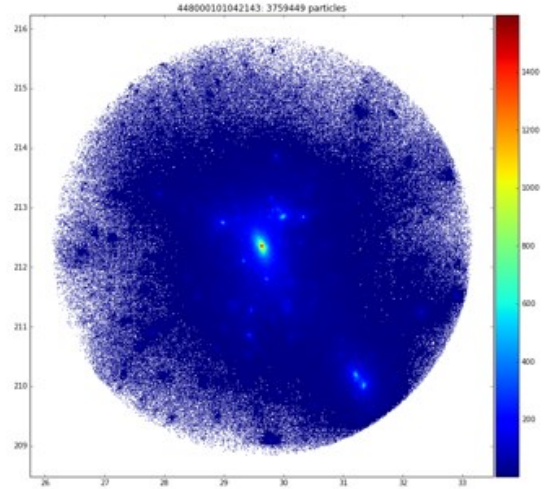
Also accessible from outside the database

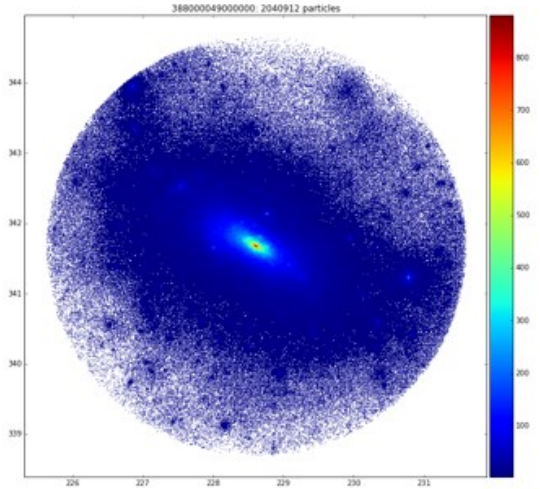


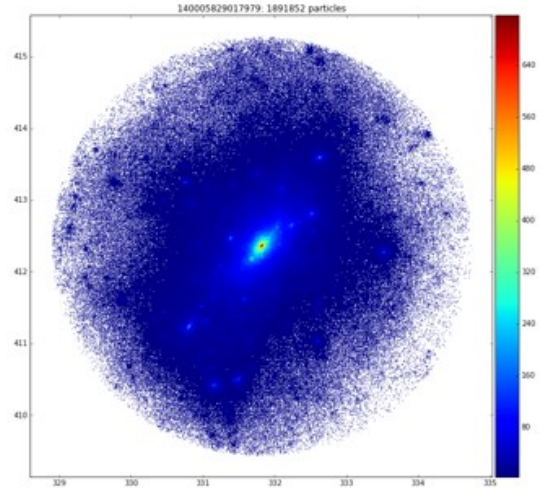
```
    ax.set_title(title)
    divider = make_axes_locatable(ax)
    # Append axes to the right of ax3, with 20% width of ax3
    cax = divider.append_axes("right", size="5%", pad=0.05)
    # Create colorbar in the appended axes
    # Tick locations can be set with the kwarg `ticks`
    # and the format of the ticklabels with kwarg `format`
    cbar = plt.colorbar(im, cax=cax)#, ticks=MultipleLocator(0.2), format="%.2f")
```

```
CPU times: user 17.4 s, sys: 3.87 s, total: 21.3 s
Wall time: 27 s
```









Thank you