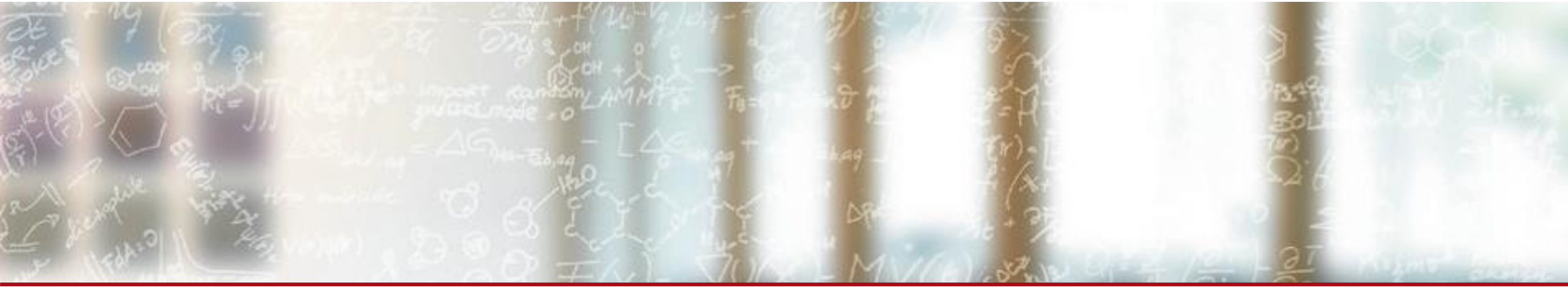
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Grid-less data visualization in ParaView/VTK

Jean M. Favre, CSCS
May 18, 2021

Presented via Zoom

Outline

- Present the Gadget2 and Tipsy plugins
- Present some SPH-specific filters
- Present query selectors

Data, Grids in VTK

- Cell Types
- Mesh Types

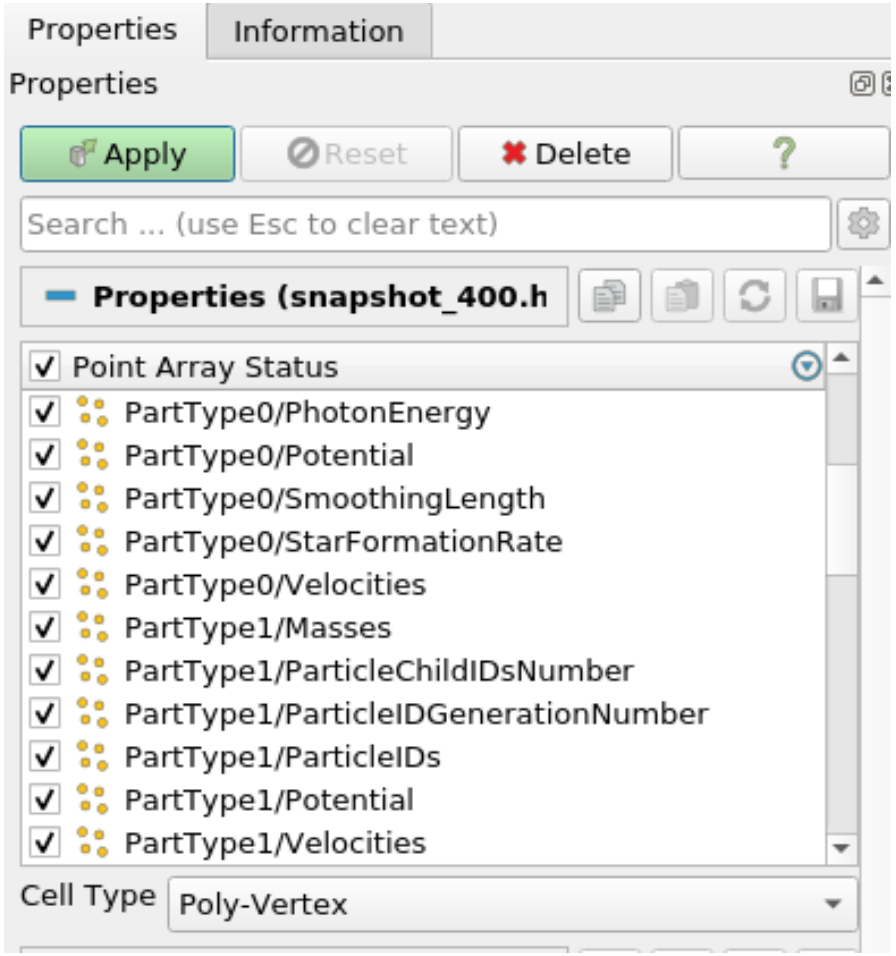
- The Gadget-2 and Tipsy plugins create a multi-block container with unstructured meshes made of Poly-Vertex (or Vertex) elements

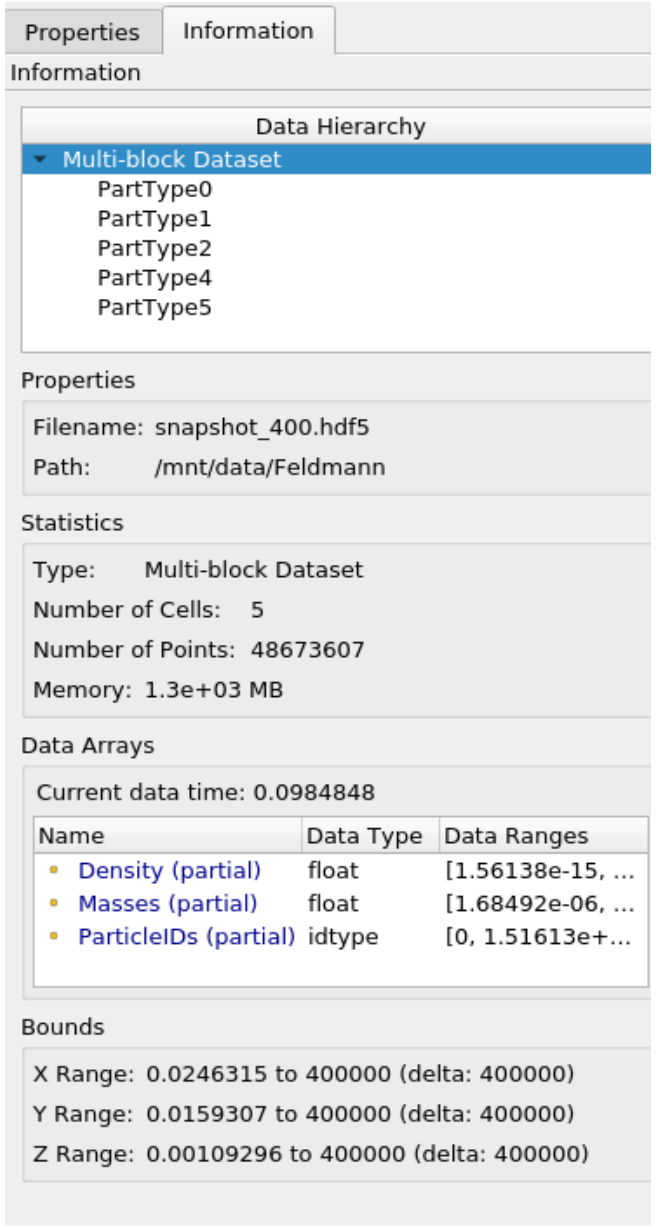
Gadget-2 HDF5 plugin

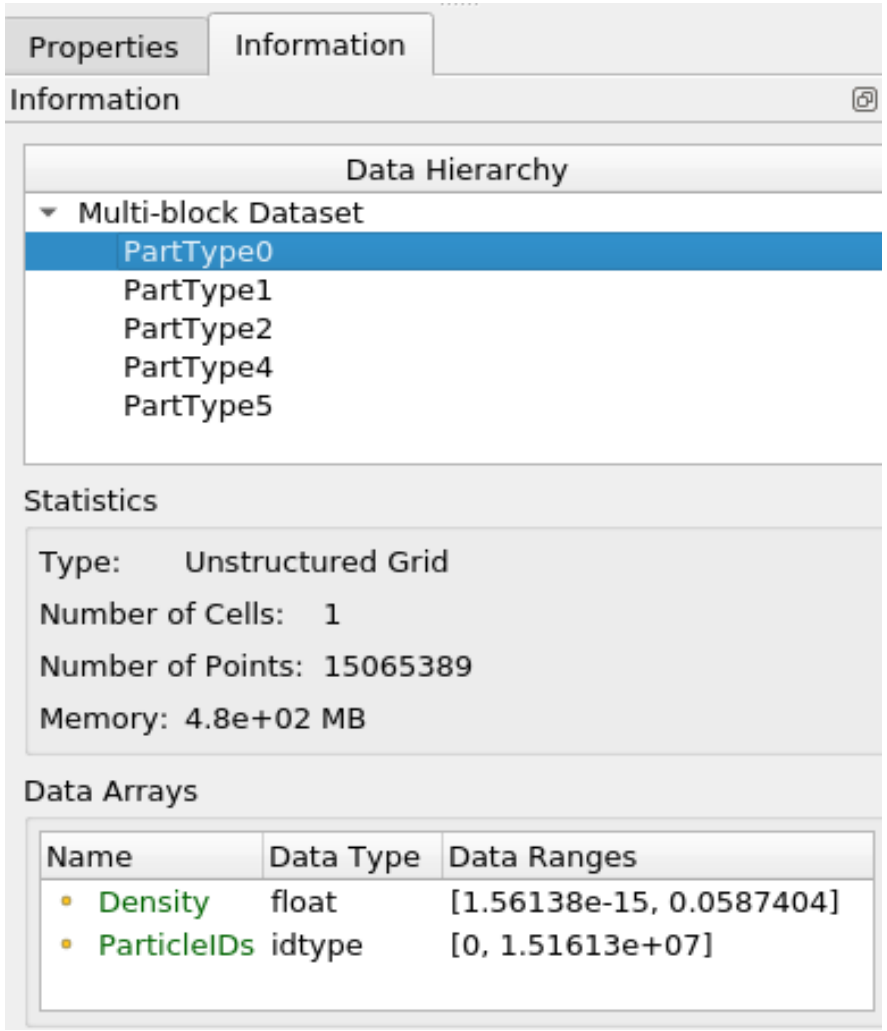
- Can read distributed snapshots (on a single processor, or with a parallel set of servers)
- Can read any variables (scalar, vector fields) for any Gadget-2 types

PartType0	PartType1	PartType2	PartType3	PartType4	PartType5
Gas	Halo	Disk	Bulge	Stars	Bndry

Gadget-2 HDF5 plugin







Gadget-2 HDF5 plugin

- Available as a shared library plugin on Piz Daint
- Available for both daint-gpu and daint-mc partitions

- Available as open source
  - Must compile ParaView from source on your desktop
  - git clone https://github.com/jfavre/ParaViewGadgetPlugin

LoadPlugin("/users/jfavre/Projects/Gadget/ParaViewPlugin/build59/lib64/paraview-5.9/plugins/pvGadgetReader/pvGadgetReader.so", ns=globals())

In batch mode

export PV_PLUGIN_PATH=/users/jfavre/Projects/Gadget/ParaViewPlugin/build59/lib64/paraview-5.9/plugins/pvGadgetReader

Gadget-2 HDF5 plugin on Piz Daint

- Does not require a client-side compilation
- Is available for all execution modes:
  - Client-server
  - Python-driven:
    - Batch-mode only
    - Interactive supercomputing (Jupyter Lab)

Launcher | Gadget_Jupyter-ParaView.ip

Code

```
# ParaView Gadget Reader Minimal Test
# Edited by Jean M. Favre, June 15, 2020
# tested with ParaView-OSMesa v 5.8
```

[1]:
```
from paraview.simple import *
from paraview.selection import *
```

[2]:
```
LoadPlugin("/users/jfavre/Projects/Gadget/v5.8/build/lib64/paraview-5.8/plugins/pvGadgetReader/pvGadgetReader.so", ns=globals())
```

[3]:
```
renderView1 = GetRenderView()

reader = GadgetSeriesReader(FileNames=['/scratch/snx3000/feldmann/MassiveFIRE2/B762_N1024_z6_TL00000_baryon_toz6/snapshot_358.hdf5
reader.PointArrayStatus = ['PartType0/Density', 'PartType4/Potential']
reader.UpdatePipeline()

#create a new 'Extract Block'
PartType4 = ExtractBlock(Input=reader)
PartType4.BlockIndices = [4]
PartType4.UpdatePipeline()

print("read      ", PartType4.PointData["Potential"].GetNumberOfTuples(), " particles")
```

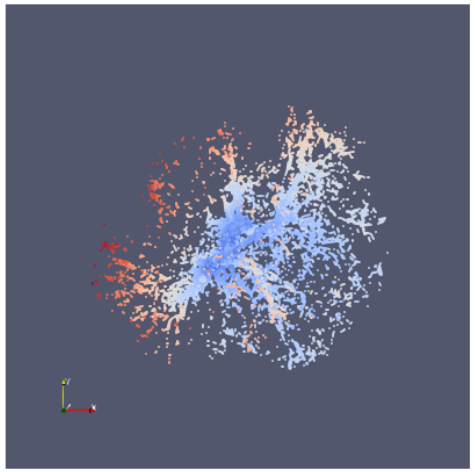
```
read       730950  particles
```

[4]:
```
partType4Display = Show(PartType4, renderView1, 'UnstructuredGridRepresentation')
potentialLUT = GetColorTransferFunction('Potential')
potentialLUT.RGBPoints = [-175255.328125, 0.231373, 0.298039, 0.752941, -154209.7890625, 0.865003, 0.865003, 0.865003, -133164.25,
potentialLUT.ScalarRangeInitialized = 1.0

# get opacity transfer function/opacity map for 'Potential'
potentialPWF = GetOpacityTransferFunction('Potential')
potentialPWF.Points = [-175255.328125, 0.0, 0.5, 0.0, -133164.25, 1.0, 0.5, 0.0]
potentialPWF.ScalarRangeInitialized = 1

# trace defaults for the display properties.
partType4Display.Representation = 'Points'
partType4Display.ColorArrayName = ['POINTS', 'Potential']
partType4Display.LookupTable = potentialLUT
```

[5]:
```
from ipyparaview.widgets import PVDisplay
disp = PVDisplay(GetActiveView())
w = display(disp)
```



Features

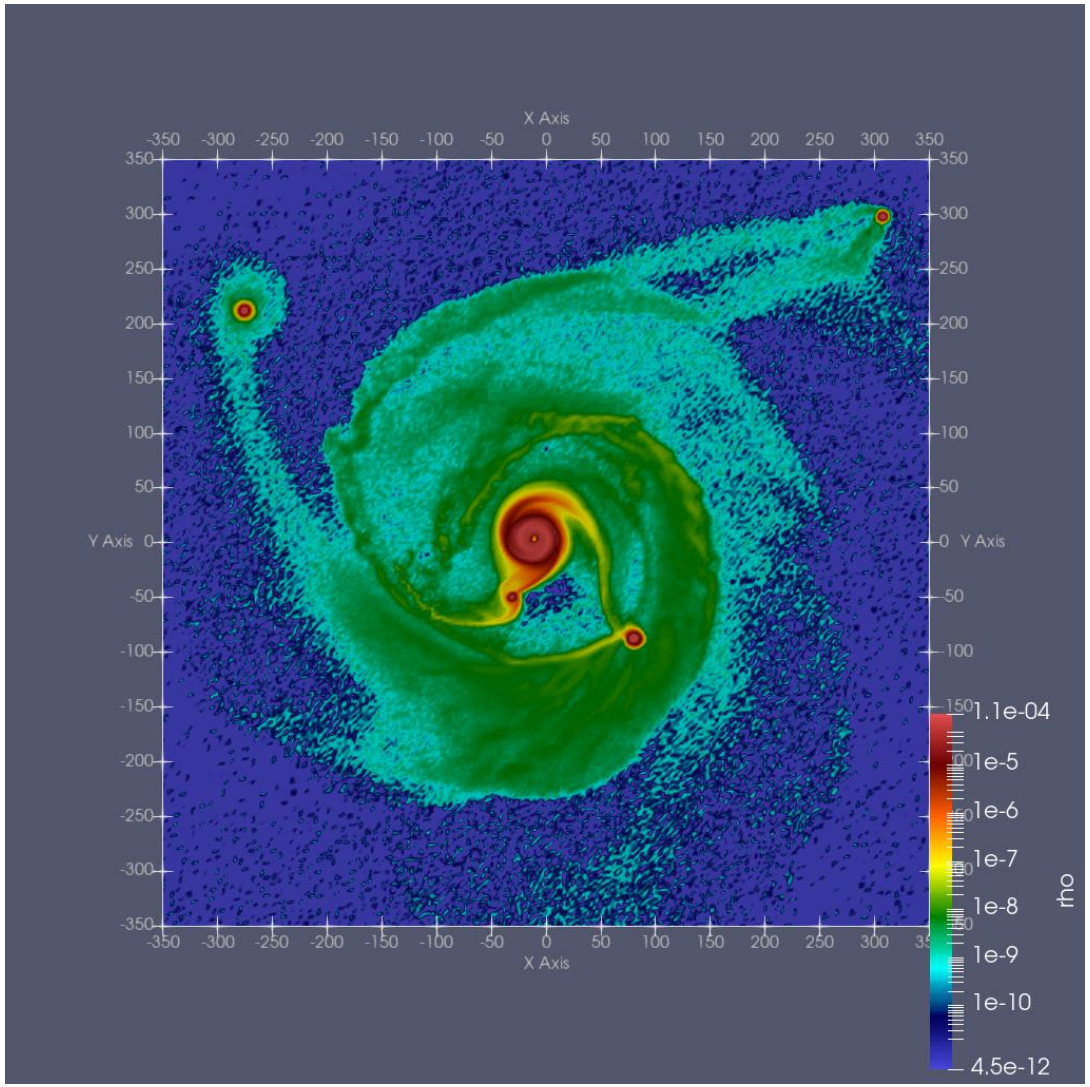
- SPH Interpolators
- Query-based Filtering

Remember SPLASH?

- Paper: by Daniel Price.

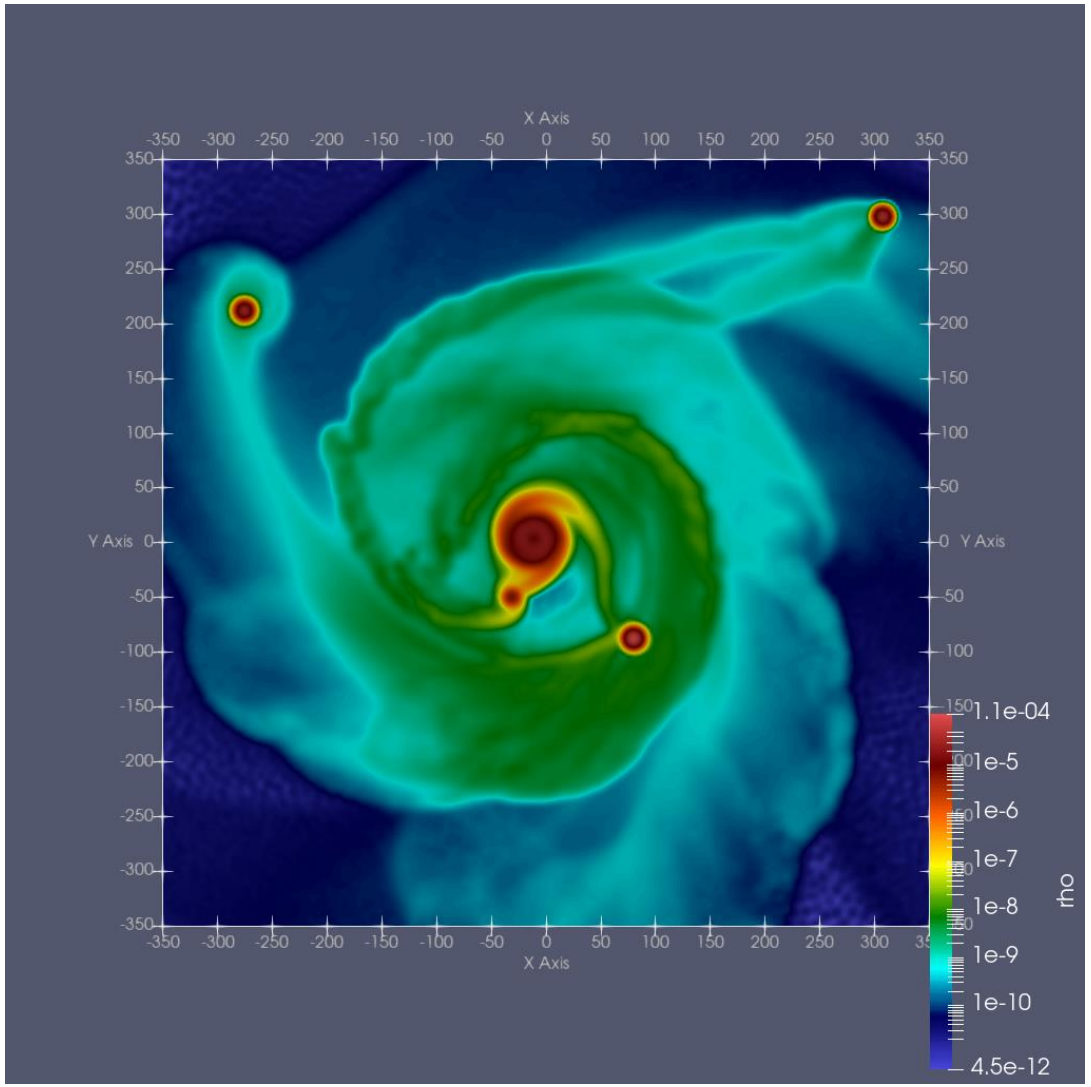
- D. Price was most likely the first person to state "*given that interpolation lies at the heart of SPH, consistency suggests use of the same interpolation algorithms as part of the visualization procedure*" (2007)

vtkSPHInterpolator, vtkSPH*Kernel



- Added in VTK in April 2016
- Available since ParaView 5.2
- The interpolators are available as a special category of Filters, providing the three basic sampling objects (line, slice, grid)
- The Point Sampling is multi-threaded with Intel TBB

vtkSPHInterpolator, vtkSPHQuinticKernel



- Use all particles within a cut-off sphere with a fixed kernel size / specified smoothing length h
- The cutoff distance (sphere around an interpolated point) is a function of the SPH kernel. A quintic kernel has cutoff distance 3*h.
- The current implementation uses a gather method. For each point to be interpolated, the basis neighbors around the point are retrieved. The provided kernel is then invoked to perform the interpolation.

Available as a Jupyter Lab notebook example

See ParaView/ParaView-SMP-Parallel.ipynb

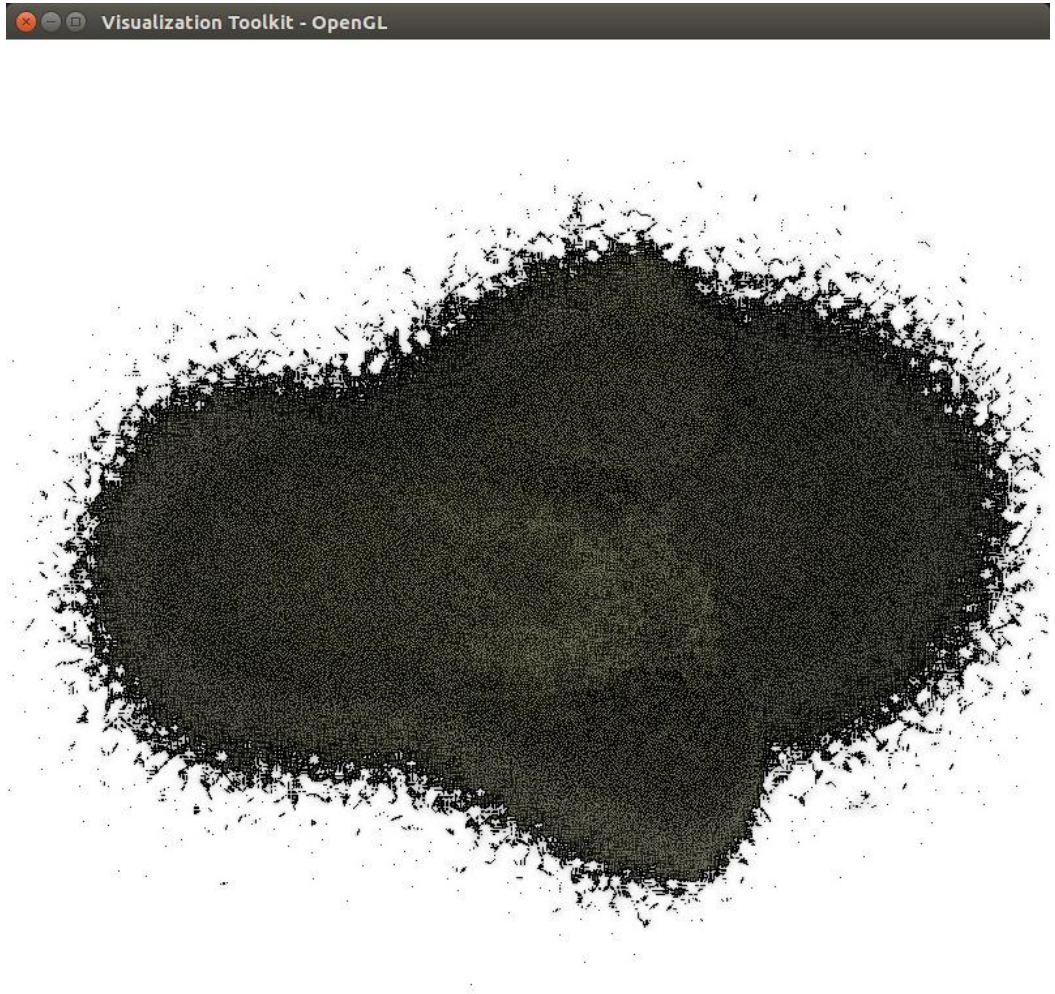
Performance:

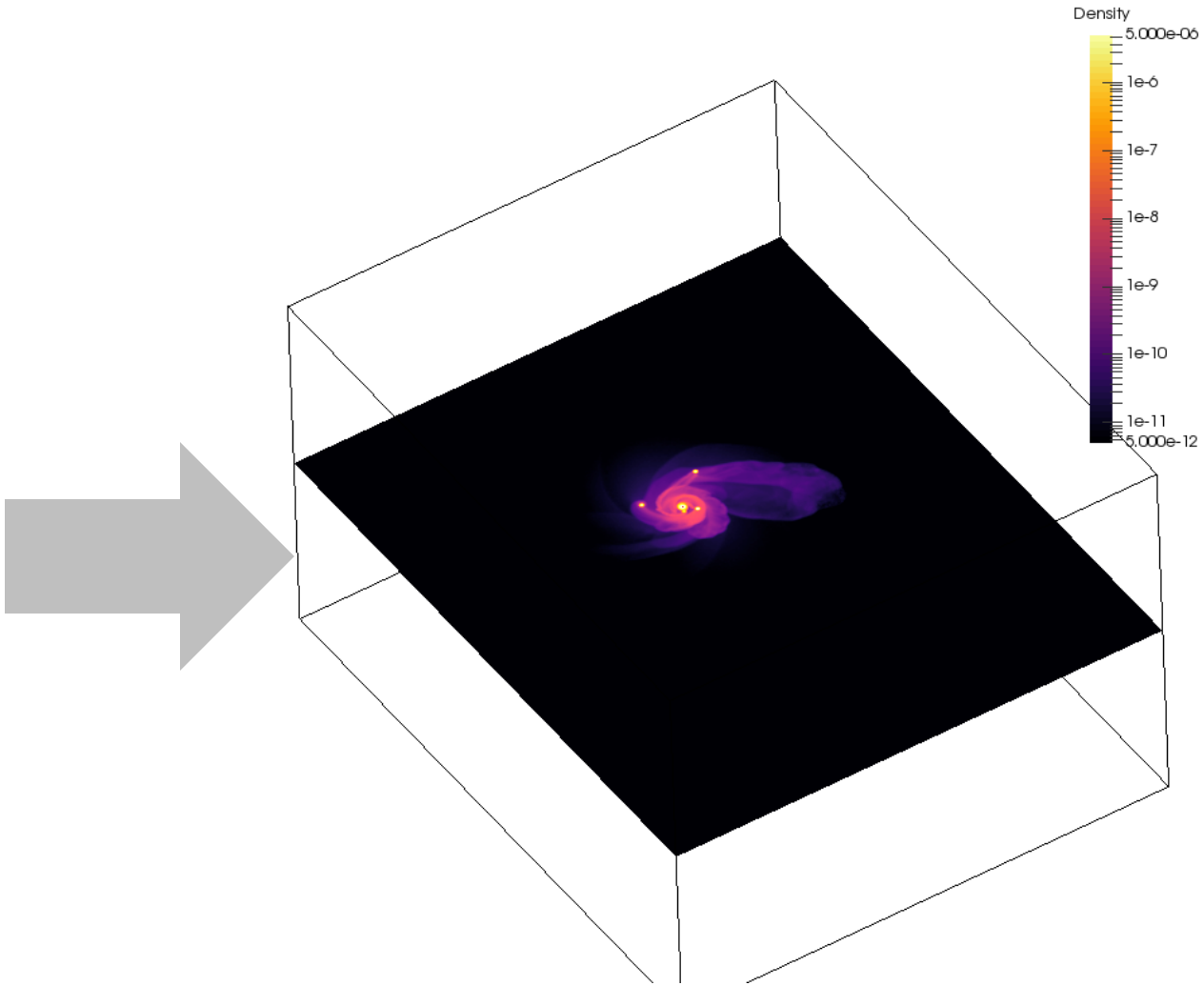
Using 1 daint-multi-core node (72 vcores) available

Exec time: 20 seconds

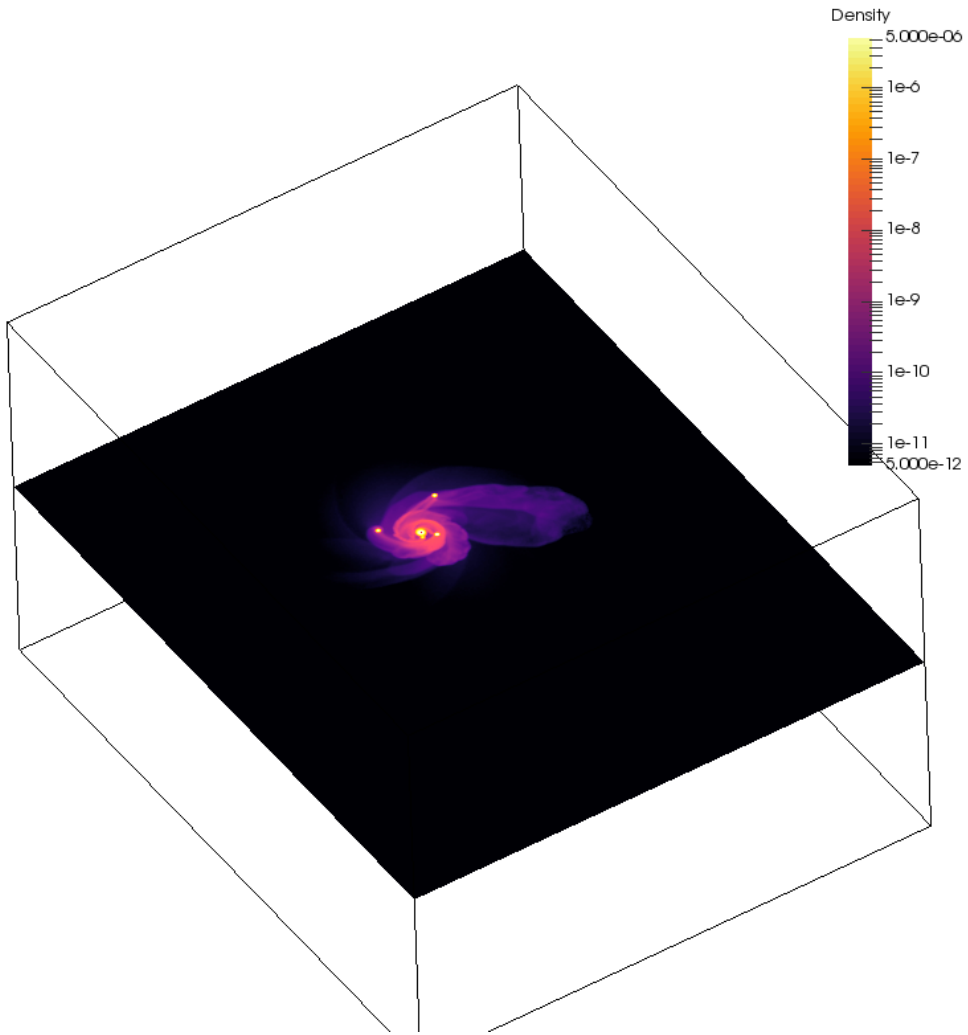
We have some new compute nodes at CSCS with 256-core nodes…testing is underway.

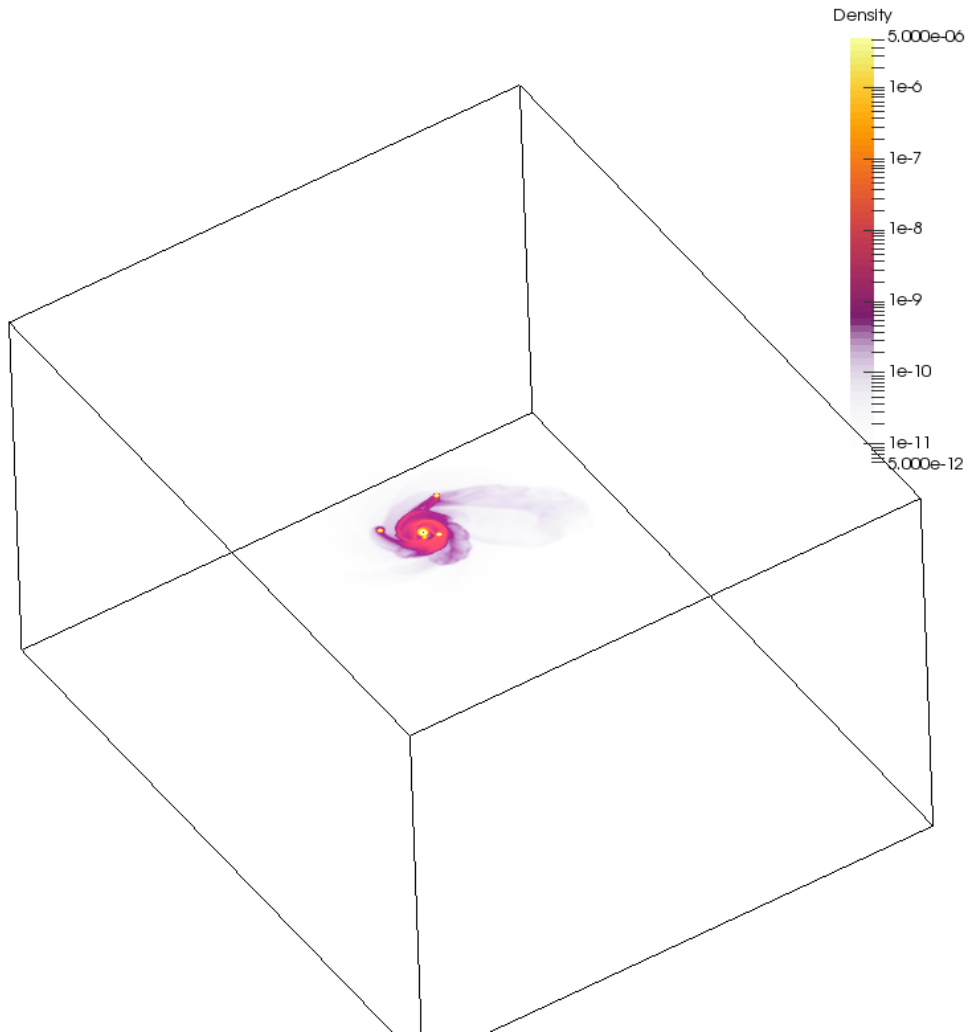
Particle clouds to slices





Set ParaView to use varying opacity for surfaces





Hierarchical Binning

- vtkHierarchicalBinningFilter creates a spatial, hierarchical ordering of input points.

This hierarchy is suitable for level-of-detail rendering, or multiresolution processing. Each level of the hierarchy is based on uniform binning of space, where deeper levels (and its bins) are repeatedly subdivided by a given branching factor.

Points are associated with different bins using a pseudo random process. The effect of executing this filter is simply to reorder the input points.

- vtkExtractHierarchicalBins

- See demonstration ParaView/Planets-VTK.ipynb

Query-based filtering examples

https://gitlab.kitware.com/paraview/paraview/blob/master/Wrapping/Python/paraview/selection.py

```
qs1="rho > 1e-05"
sel1 = QuerySelect(QueryString=qs1, Source=mergeBlocks1)
extractSelection1 = ExtractSelection(Input=mergeBlocks1)

rep1 = Show(extractSelection1)
rep1.Representation = 'Points'
ColorBy(rep1, ('POINTS','rho'))
```

Query-based filtering examples

```
# points is the array of coordinates


Qs2 = "np.logical_and(rho > 1e-05, points[:,0] < -12.15)"
sel2 = QuerySelect(QueryString=Qs2, Source=mergeBlocks1)
extractSelection2 = ExtractSelection(registrationName='ExtractSelection2', Input=mergeBlocks1)


rep2 = Show(extractSelection2)
rep2.Representation = 'Points'
ColorBy(rep2, ('POINTS','rho'))
```

Query-based filtering examples

```
# points is the array of coordinates


#Center = [-11.1431, 3.97869, -0.173805]; Radius = 10
Qs3 = "mag(points - [-11.1431, 3.97869, -0.173805]) < 10"
sel3 = QuerySelect(QueryString=Qs3, Source=mergeBlocks1)
extractSelection3 = ExtractSelection(registrationName='ExtractSelection3', Input=mergeBlocks1)



rep3 = Show(extractSelection3)
rep3.Representation = 'Points'
ColorBy(rep3, ('POINTS','rho'))
```

Query-based filtering

https://gitlab.kitware.com/paraview/paraview/blob/master/Wrapping/Python/paraview/selection.py

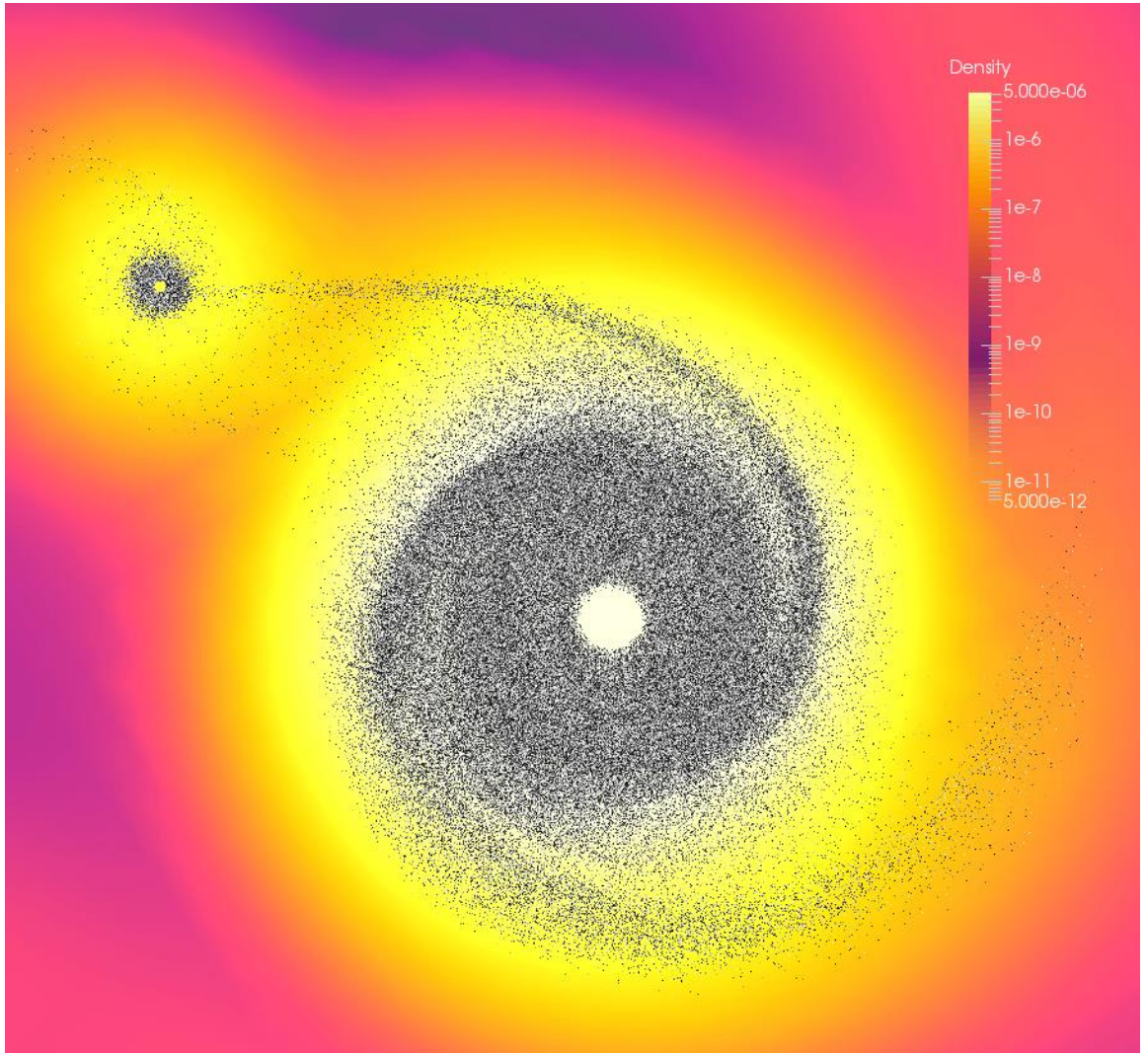
Other selections:

SelectThresholds(Thresholds=[-338000, 0], ArrayName= ='Potential', Source=GetActiveSource())

SelectIDs(IDs=[i for i in range(100000, 1100000)], , Source=GetActiveSource())

ParaView Python Programmable Filter

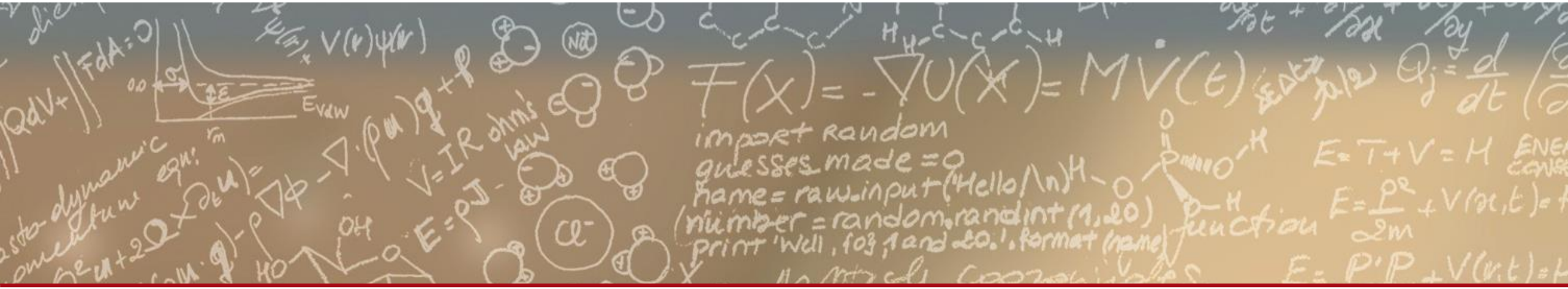
- Used to calculate non trivial derived fields
- Used to do selection
- Runs, in parallel, on the data server side of ParaView
- Can apply as many Boolean combinations of masks using AND. OR,..



Other types of gridless data

See blog article

LAS and PDAL Readers available in ParaView. Please send me email if interested.



Thanks for the attention