

Visualizing your Data

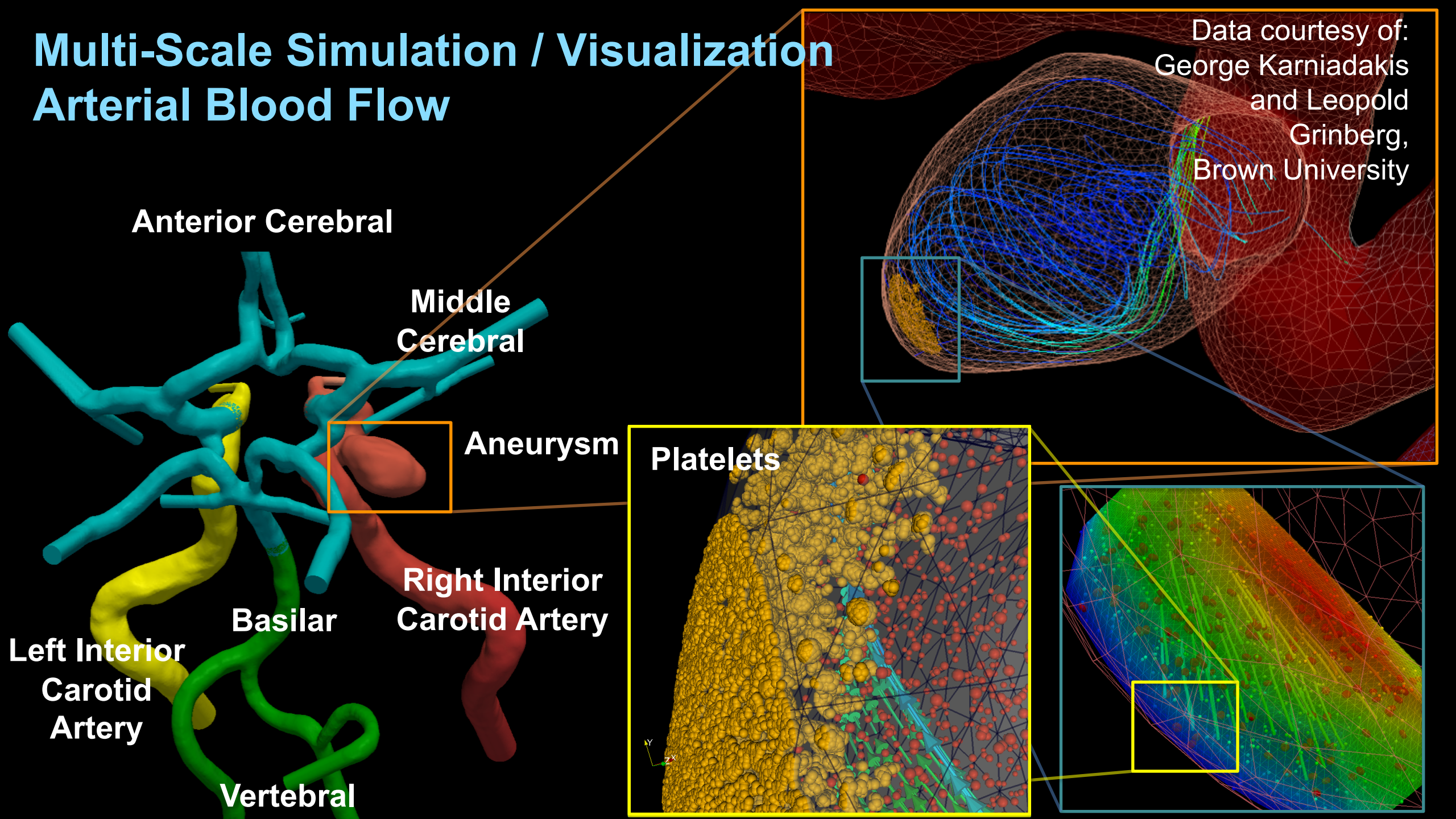
Joseph Insley
Lead, Visualization & Data Analytics
Argonne Leadership Computing Facility

Silvio Rizzi
Assistant Computer Scientist
Argonne Leadership Computing Facility

Here’s the plan…

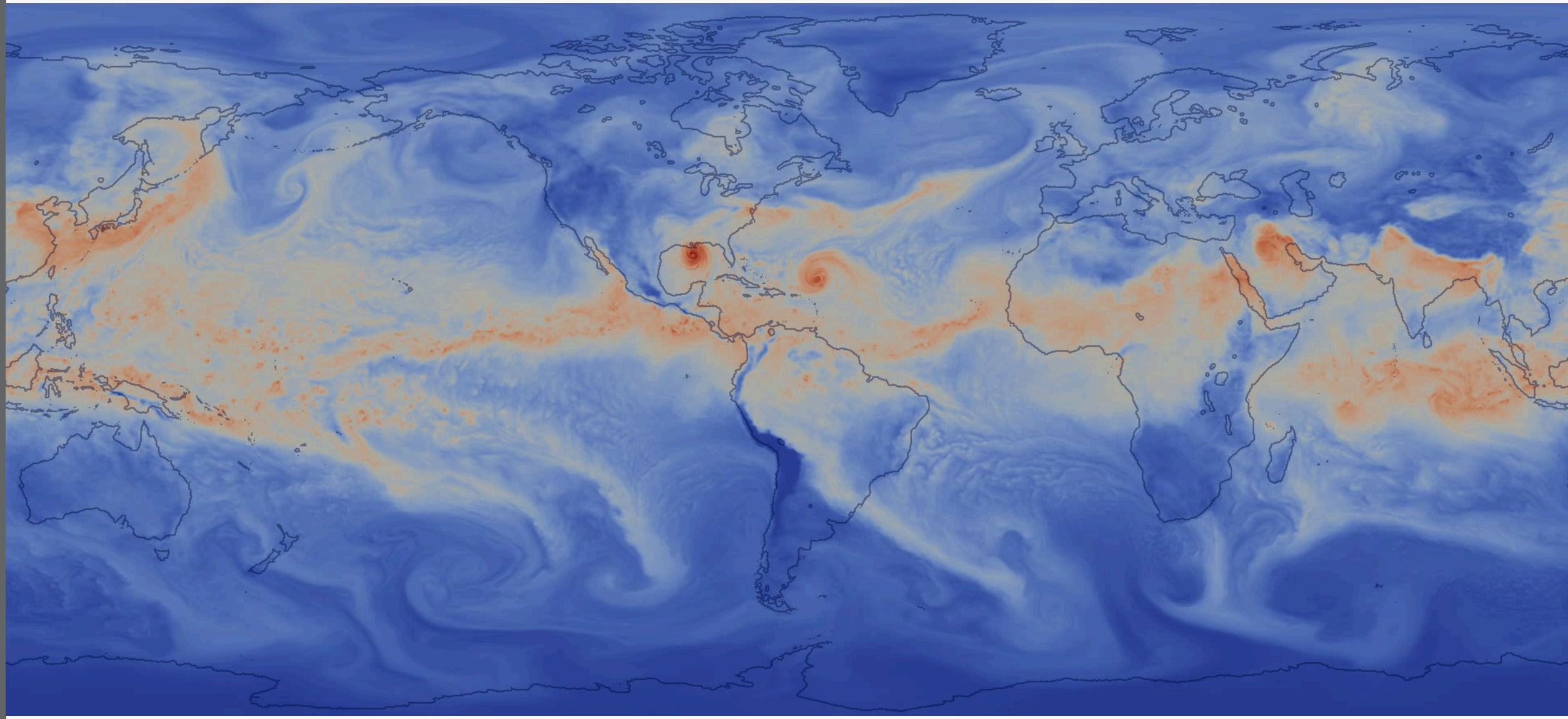
- Examples of visualizations
- Visualization resources
- Visualization tools and formats
- Data representations
- Visualization for debugging
- In Situ Visualization and Analysis

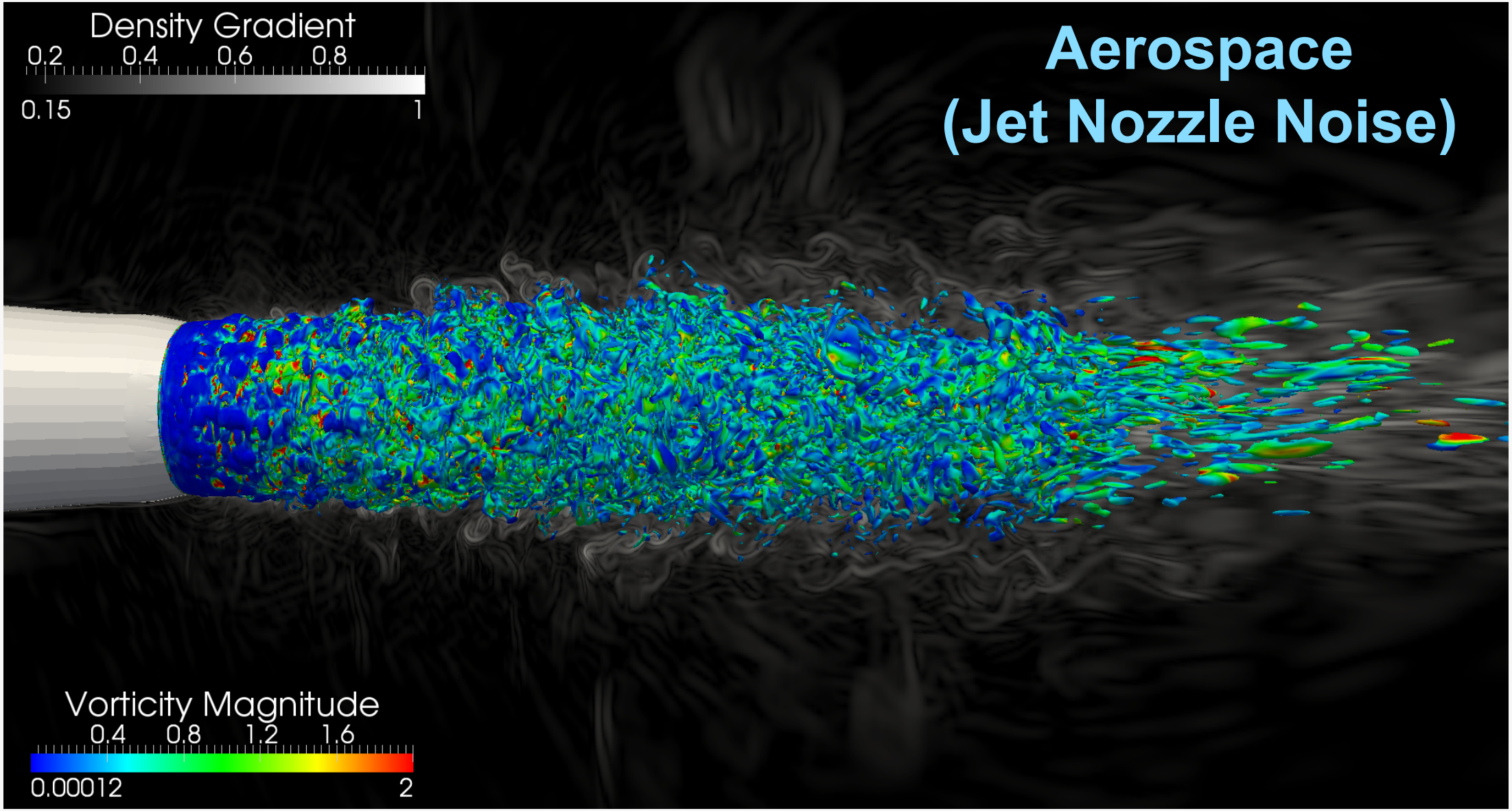
Multi-Scale Simulation / Visualization
Arterial Blood Flow



Climate

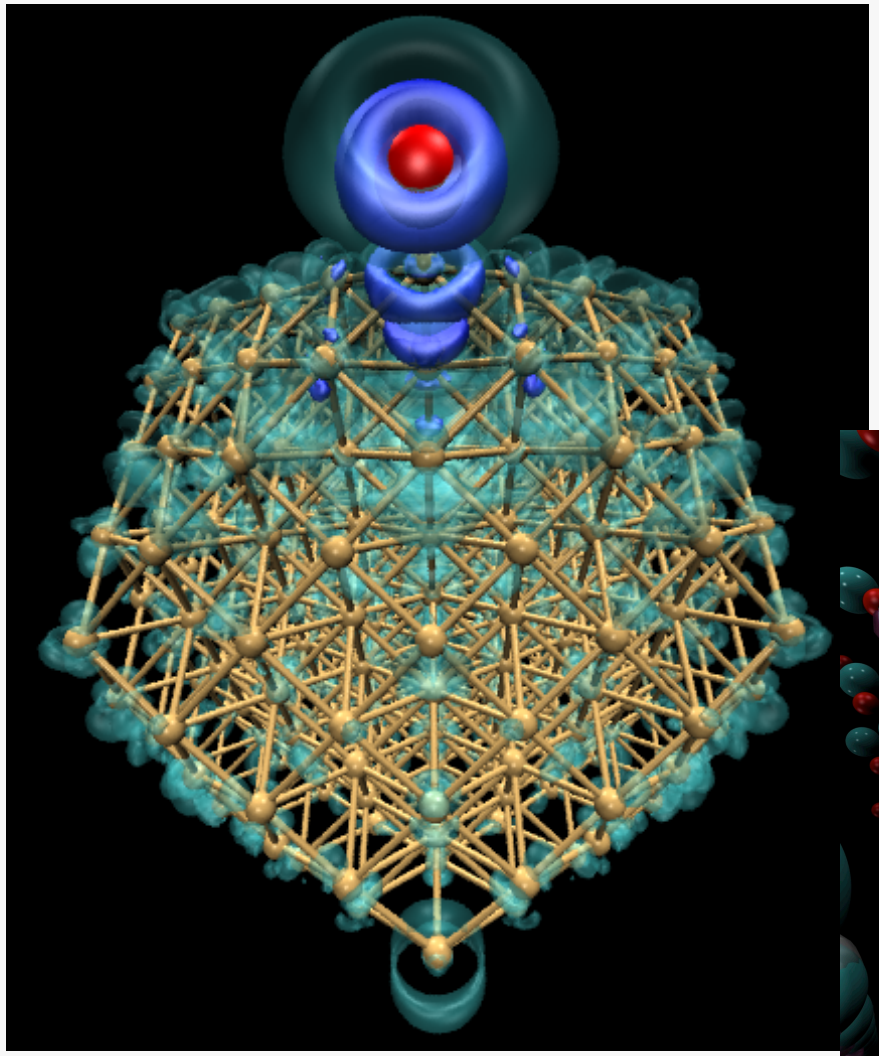
Data courtesy of: Mark Taylor, Sandia National Laboratory; Rob Jacob, Argonne National Laboratory; Warren Washington, National Center for Atmospheric Research



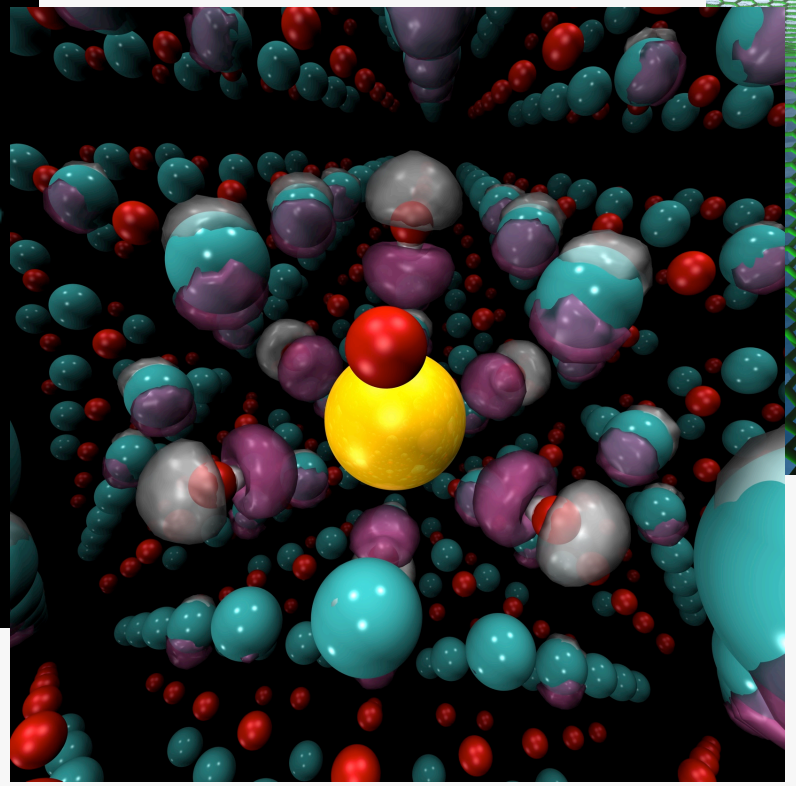


Data courtesy of: Anurag Gupta and Umesh Paliath, General Electric Global Research

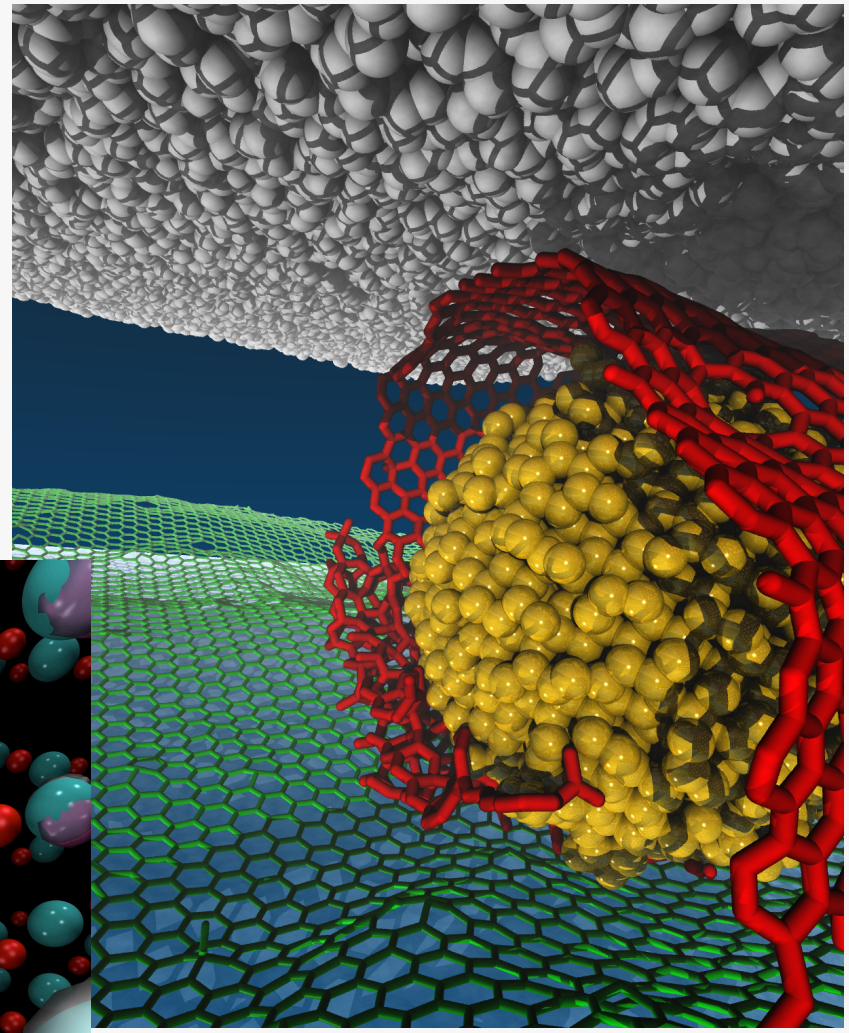
Materials Science / Molecular



Data courtesy of: Jeff Greeley, Nichols Romero, Argonne National Laboratory

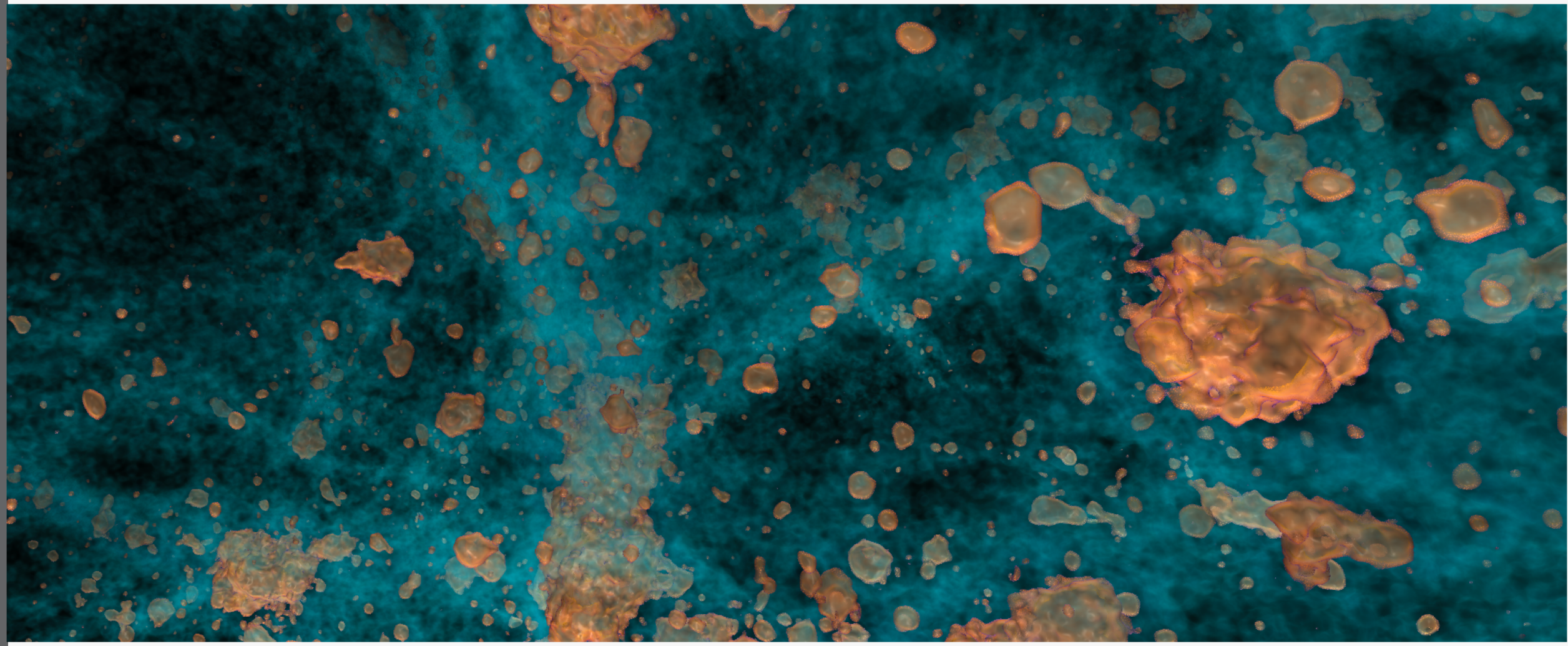


Data courtesy of: Subramanian Sankaranarayanan, Argonne National Laboratory



Data courtesy of: Paul Kent, Oak Ridge National Laboratory, Anouar Benali, Argonne National Laboratory

Cosmology



Data courtesy of: Salman Habib, Katrin Heitmann, and the HACC team, Argonne National Laboratory

Cooley: Analytics/Visualization cluster

Peak 223 TF

126 nodes; each node has

– Two Intel Xeon E5-2620 Haswell 2.4 GHz 6-core processors

– NVIDIA Telsa K80 graphics processing unit (24GB)

– 384 GB of RAM

Aggregate RAM of 47 TB

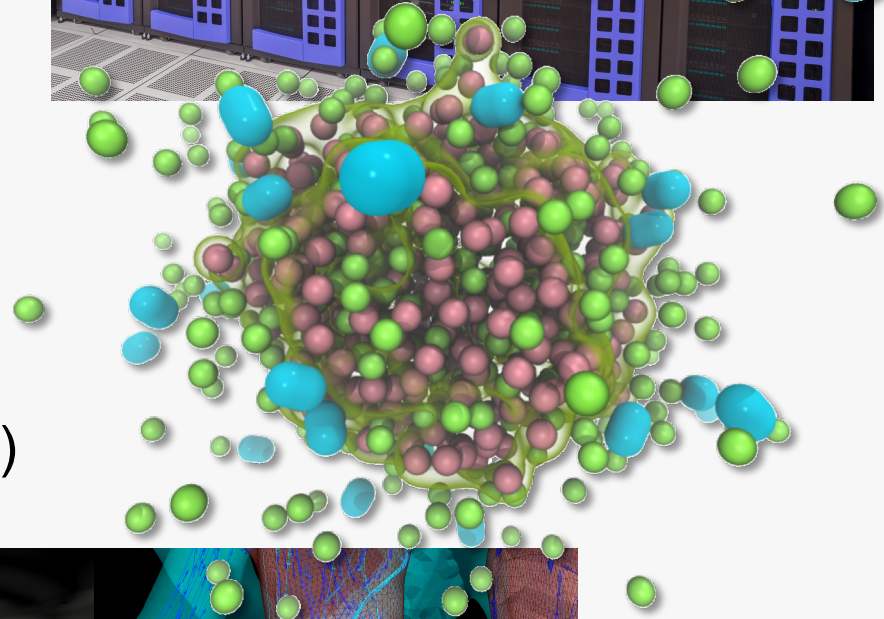
Aggregate GPU memory of ~3TB

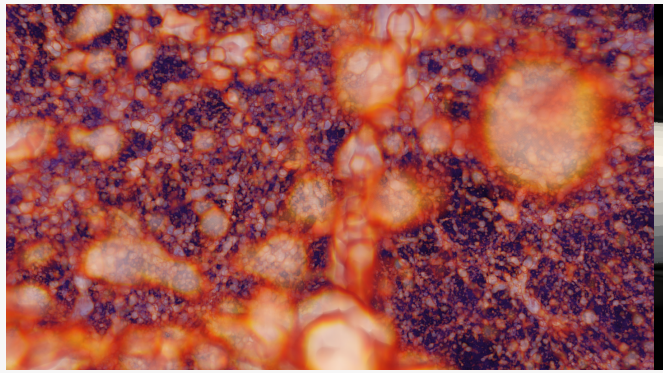
Cray CS System

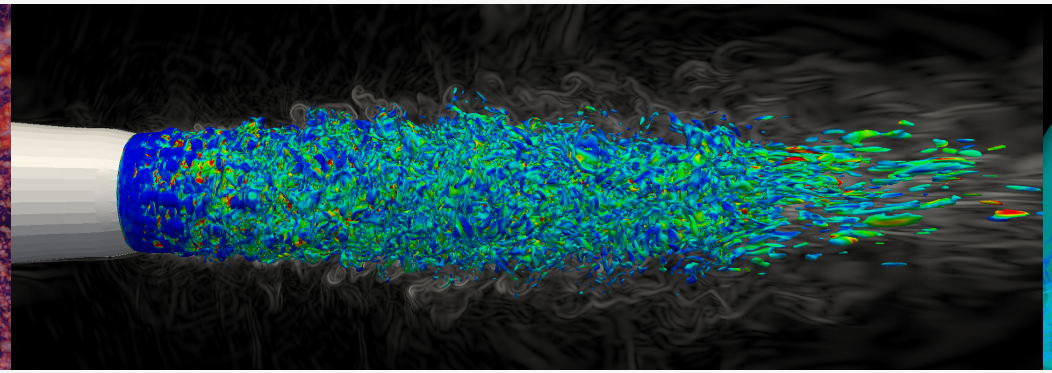
216 port FDR IB switch with uplinks to our QDR infrastructure

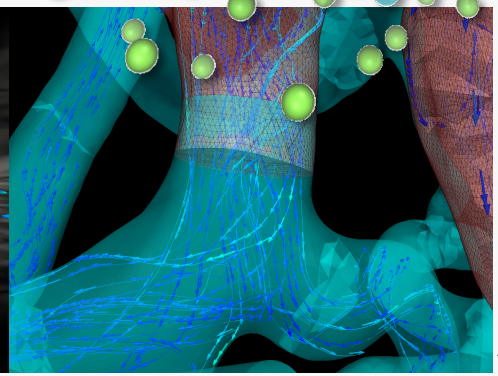
Mounts the same GPFS file systems as Mira, Cetus (not Theta)



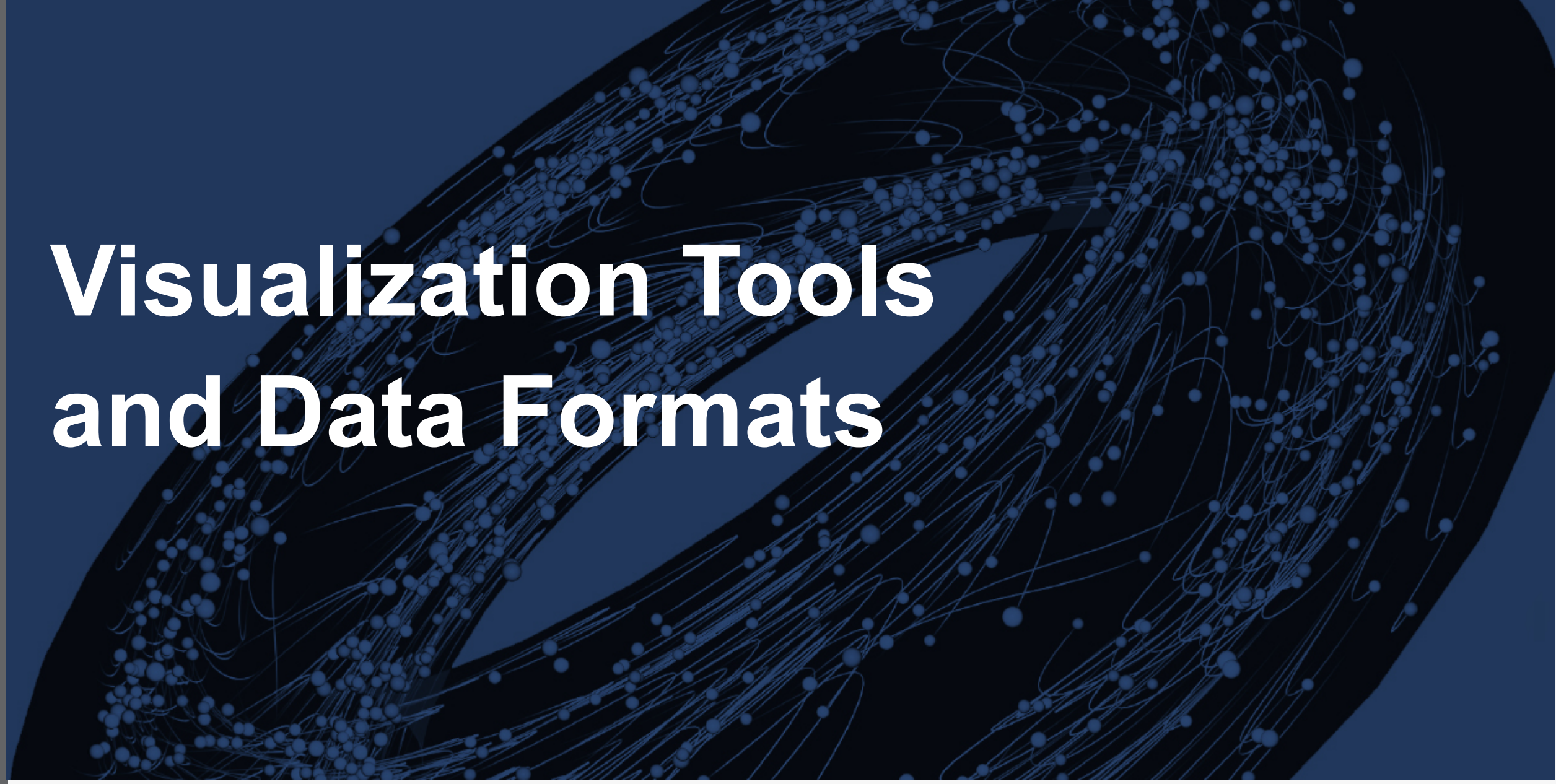








Argonne NATIONAL LABORATORY



Visualization Tools and Data Formats

All Sorts of Tools

Visualization Applications
- VisIt
- ParaView*
- EnSight

Domain Specific
- VMD, PyMol, Ovito

APIs
- VTK: visualization
- ITK: segmentation & registration

GPU performance
- vl3: shader-based volume and particle rendering

Analysis Environments
- Matlab
- Parallel R

Utilities
- GnuPlot
- ImageMagick

■ Available on Cooley

* Available on Theta

Argonne NATIONAL LABORATORY

ParaView & VisIt vs. vtk

ParaView & VisIt

- General purpose visualization applications
- GUI-based
- Client / Server model to support remote visualization
- Scriptable / Extendable
- Built on top of vtk (largely)
- *In situ* capabilities





vtk

- Programming environment / API
- Additional capabilities, finer control
- Smaller memory footprint
- Requires more expertise (build custom applications)



Data File Formats (ParaView & VisIt)

VTK

Parallel (partitioned) VTK

VTK MultiBlock (MultiGroup, Hierarchical, Hierarchical Box)

Legacy VTK

Parallel (partitioned) legacy VTK

EnSight files

EnSight Master Server

Exodus

BYU

XDMF

PLOT2D

PLOT3D

SpyPlot CTH

HDF5 raw image data

DEM

VRML

PLY

Polygonal Protein Data Bank

XMol Molecule

Stereo Lithography

Gaussian Cube

Raw (binary)

AVS

Meta Image

Facet

PNG

SAF

LS-Dyna

Nek5000

OVERFLOW

paraDIS

PATRAN

PFLOTRAN

Pixie

PuReMD

S3D

SAS

Tetrad

UNIC

VASP

ZeusMP

ANALYZE

BOV

GMV

Tecplot

Vis5D

Xmdv

XSF

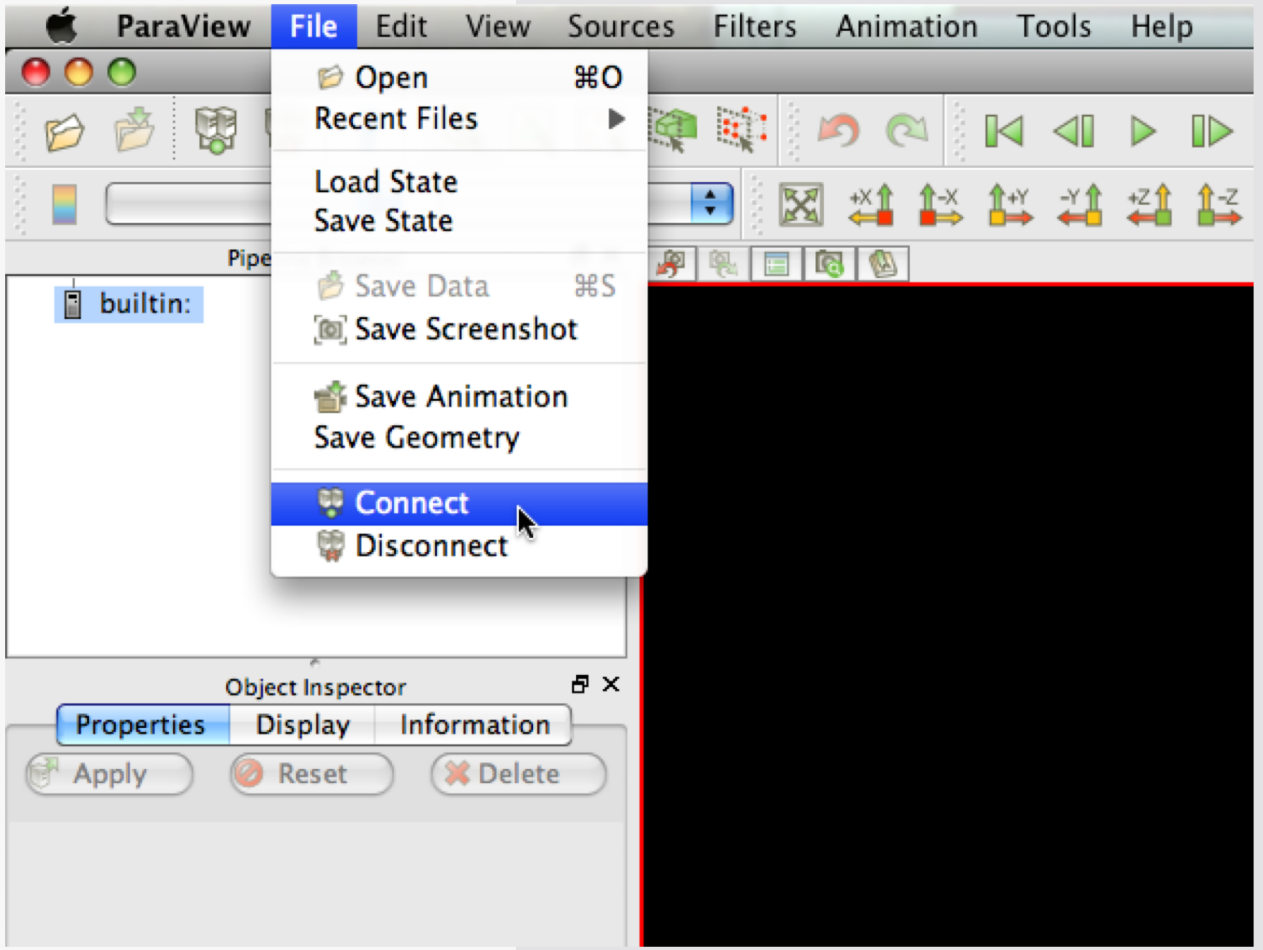
ParaView on Theta

Version 5.5.2 (Client and Server versions must match)

After launching client locally

- Connect
- Fetch servers (first time only)
- Fetch Theta configuration
- Connect
- Configure server settings
- Connecting: Enter Password
- Open File





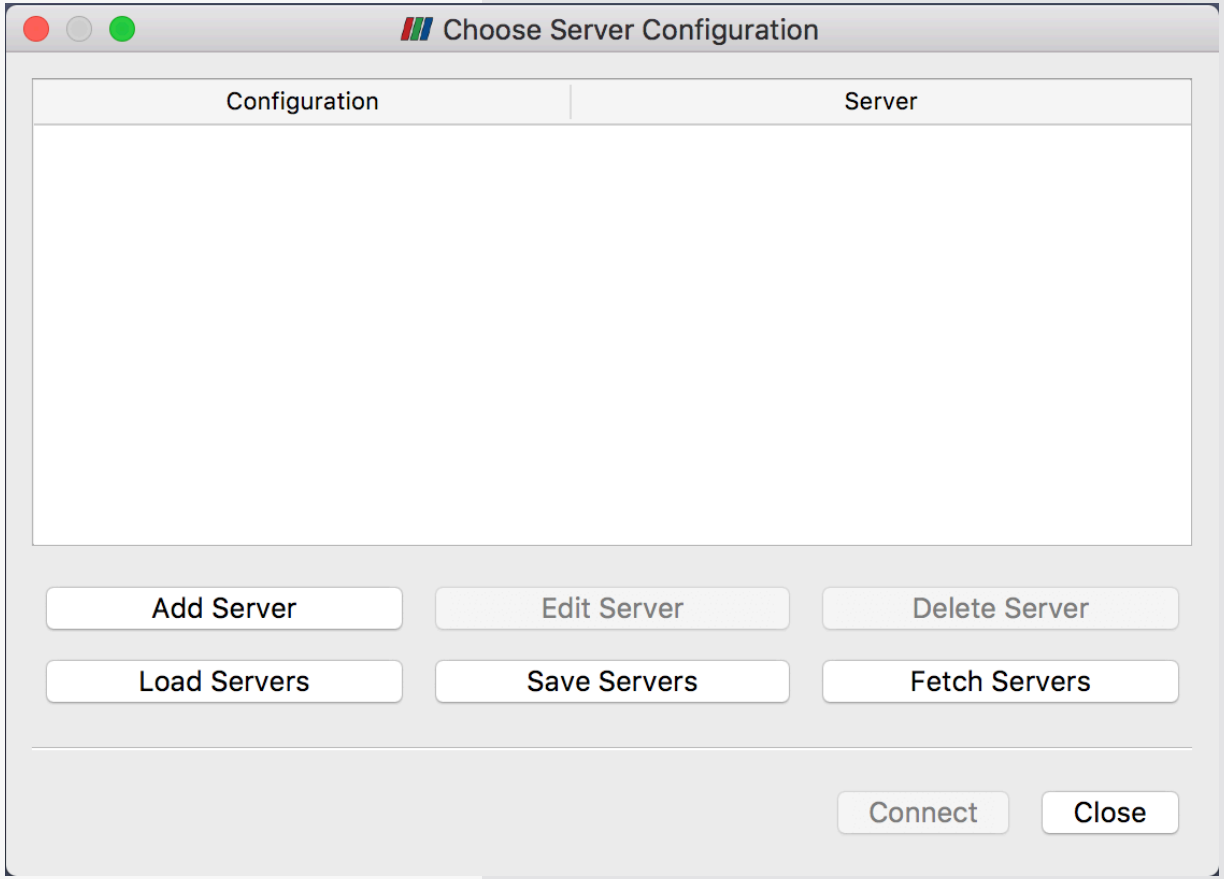
Argonne NATIONAL LABORATORY

ParaView on Theta

Version 5.5.2 (Client and Server versions must match)

After launching client locally

- Connect
- Fetch servers (first time only)
- Fetch Theta configuration
- Connect
- Configure server settings
- Connecting: Enter Password
- Open File

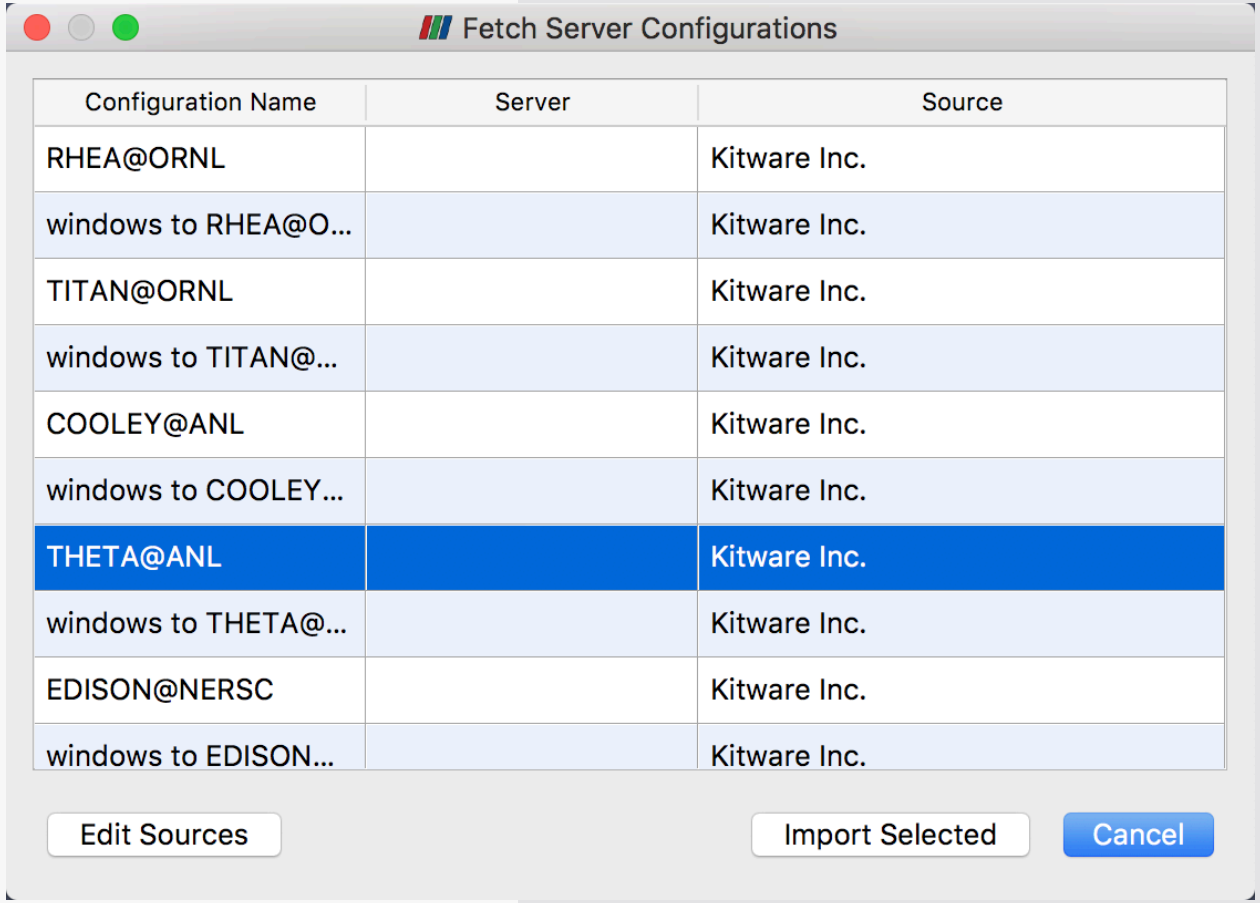


ParaView on Theta

Version 5.5.2 (Client and Server versions must match)

After launching client locally

- Connect
- Fetch servers (first time only)
- Fetch Theta configuration
- Connect
- Configure server settings
- Connecting: Enter Password
- Open File

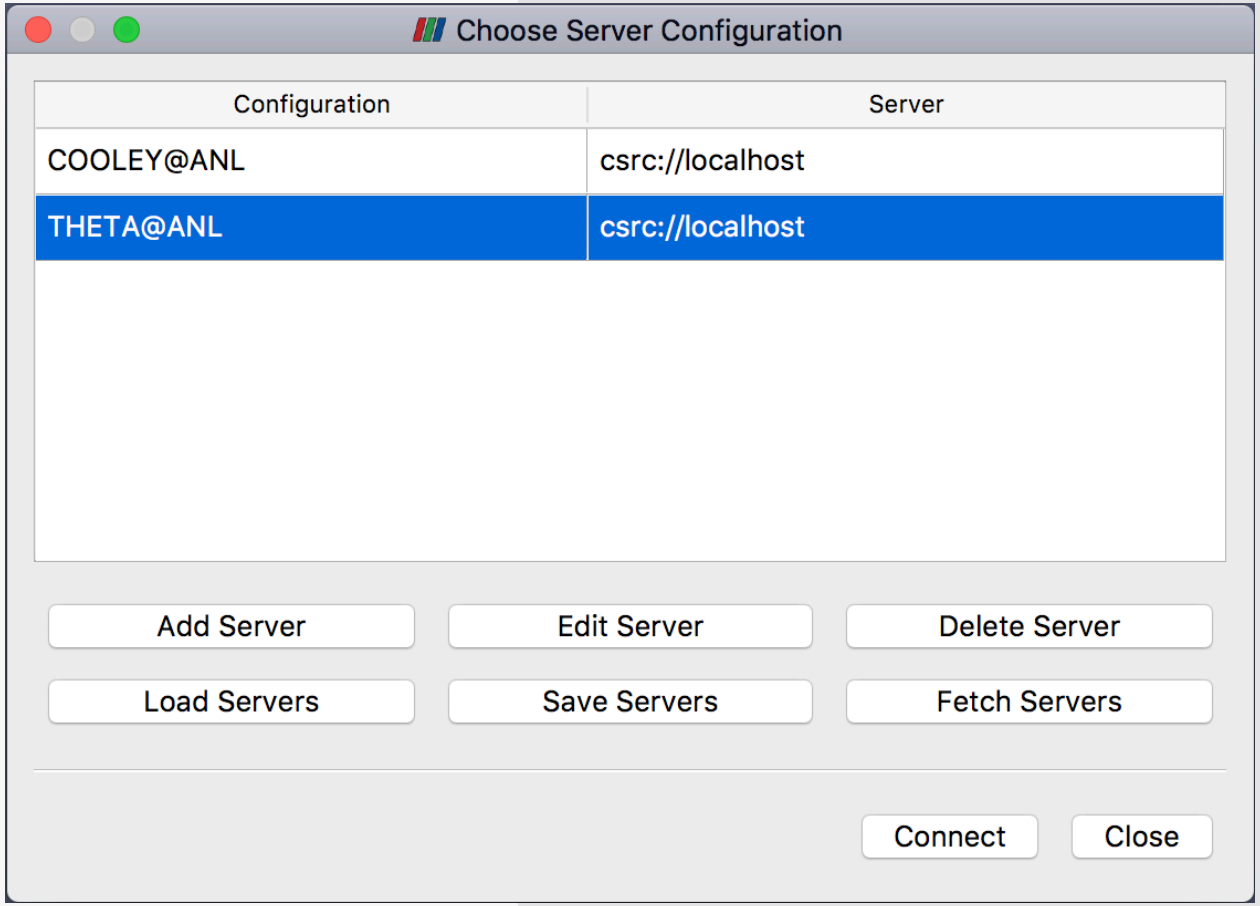


ParaView on Theta

Version 5.5.2 (Client and Server versions must match)

After launching client locally

- Connect
- Fetch servers (first time only)
- Fetch Theta configuration
- Connect
- Configure server settings
- Connecting: Enter Password
- Open File

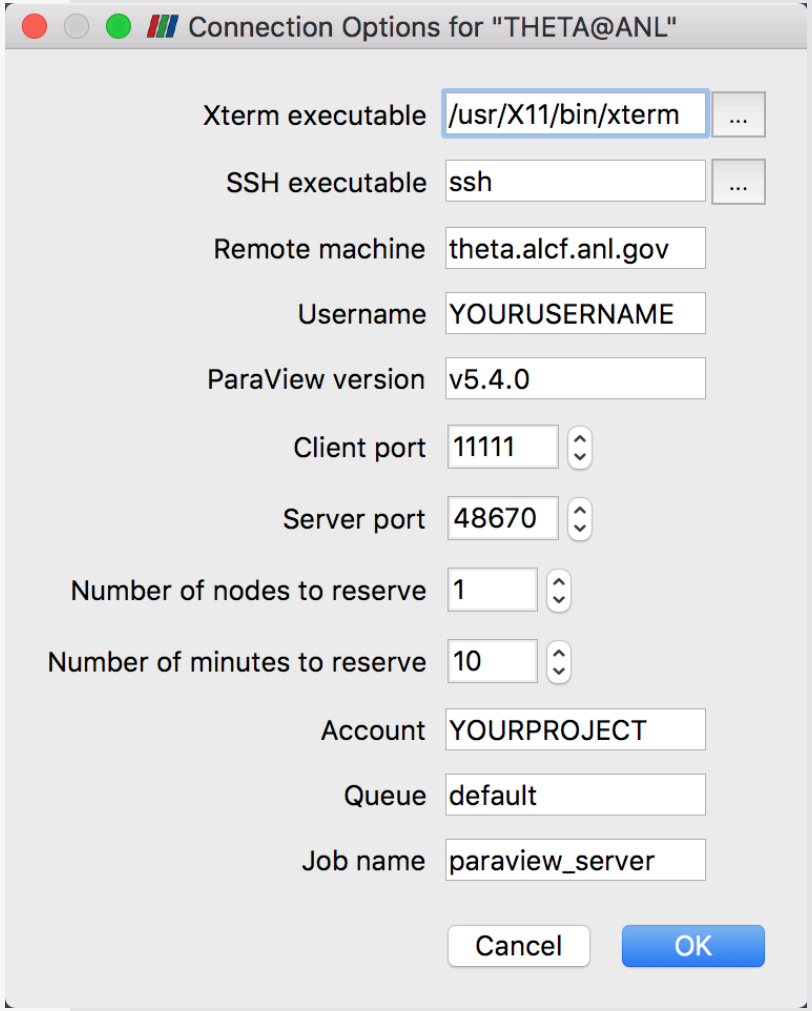


ParaView on Theta

Version 5.5.2 (Client and Server versions must match)

After launching client locally

- Connect
- Fetch servers (first time only)
- Fetch Theta configuration
- Connect
- Configure server settings
- Connecting: Enter Password
- Open File

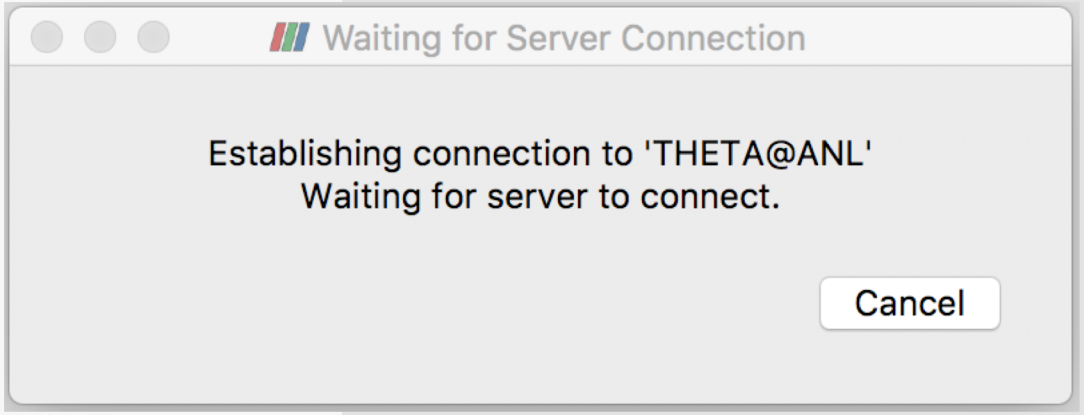


ParaView on Theta

Version 5.5.2 (Client and Server versions must match)

After launching client locally

- Connect
- Fetch servers (first time only)
- Fetch Theta configuration
- Connect
- Configure server settings
- Connecting: Enter Password
- Open File

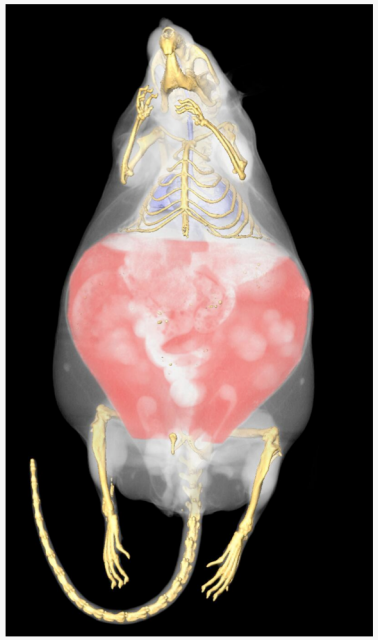


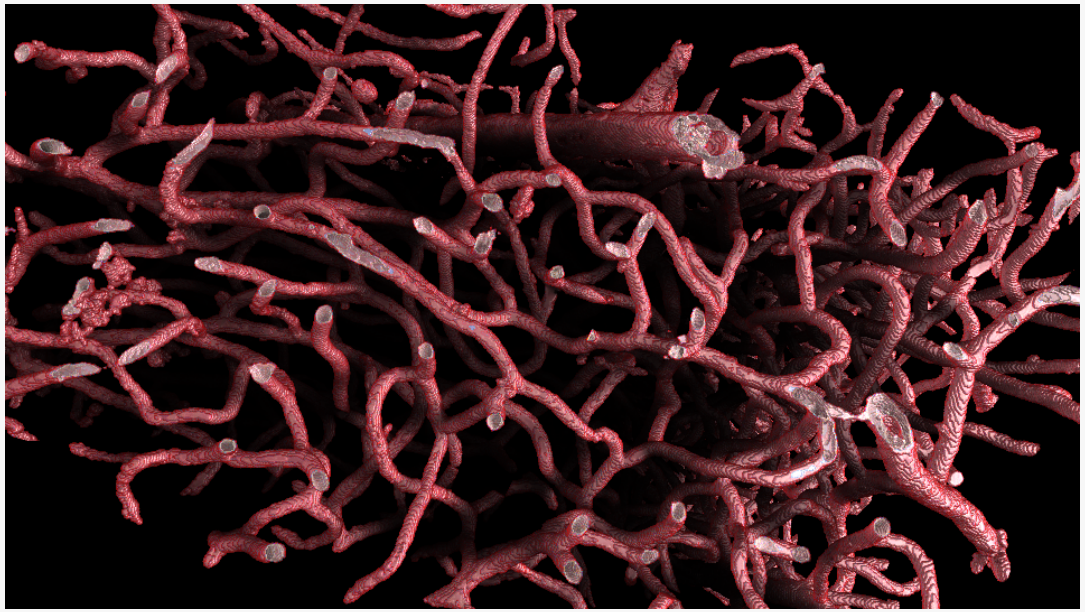


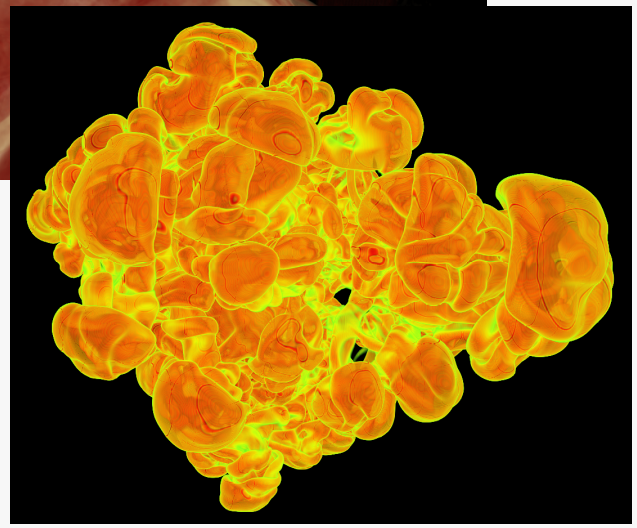
Data Representations

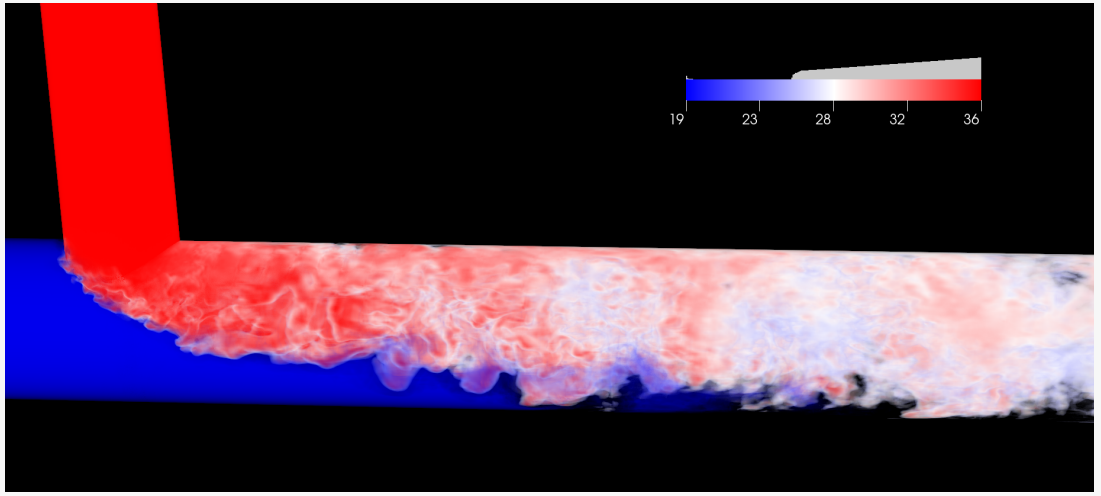
Data Representations: Volume Rendering











Data Representations: Glyphs

2D or 3D geometric object to represent point data

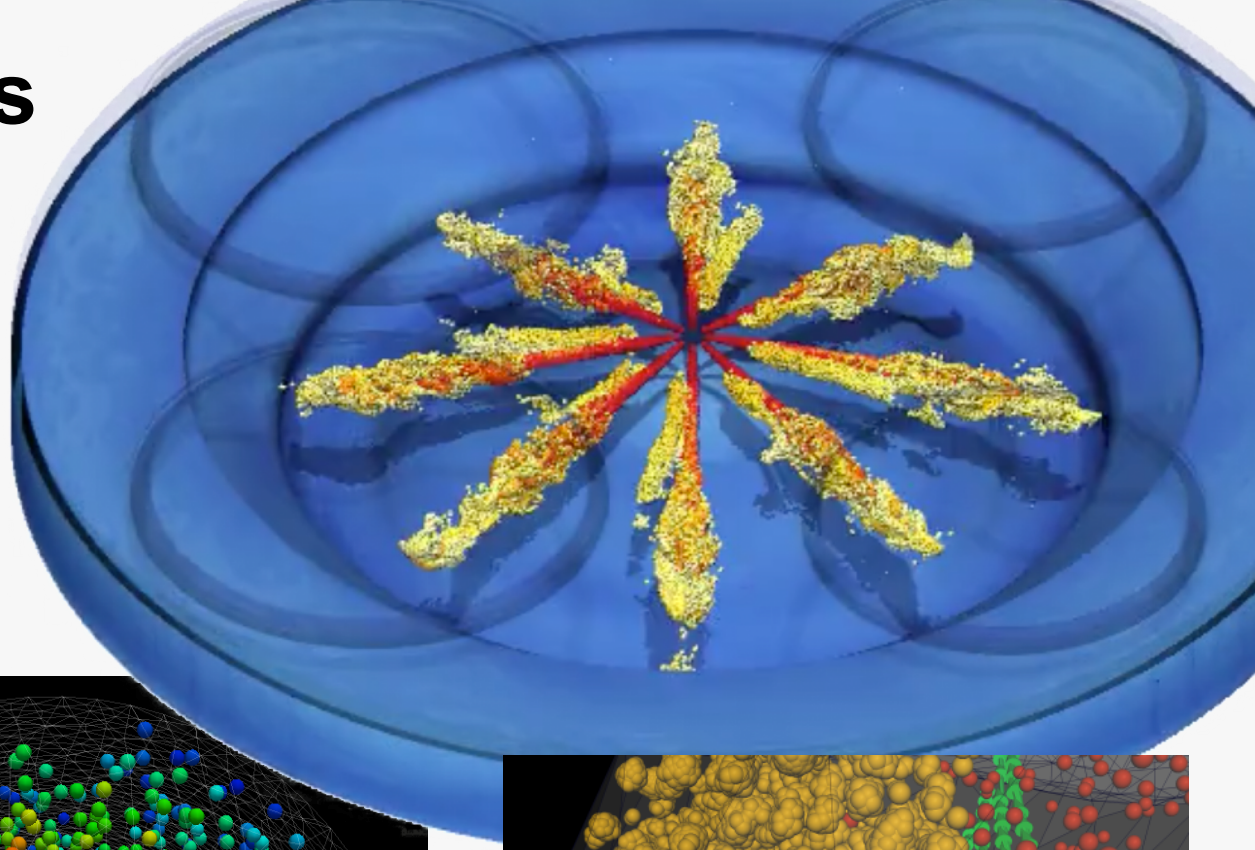
Location dictated by coordinate

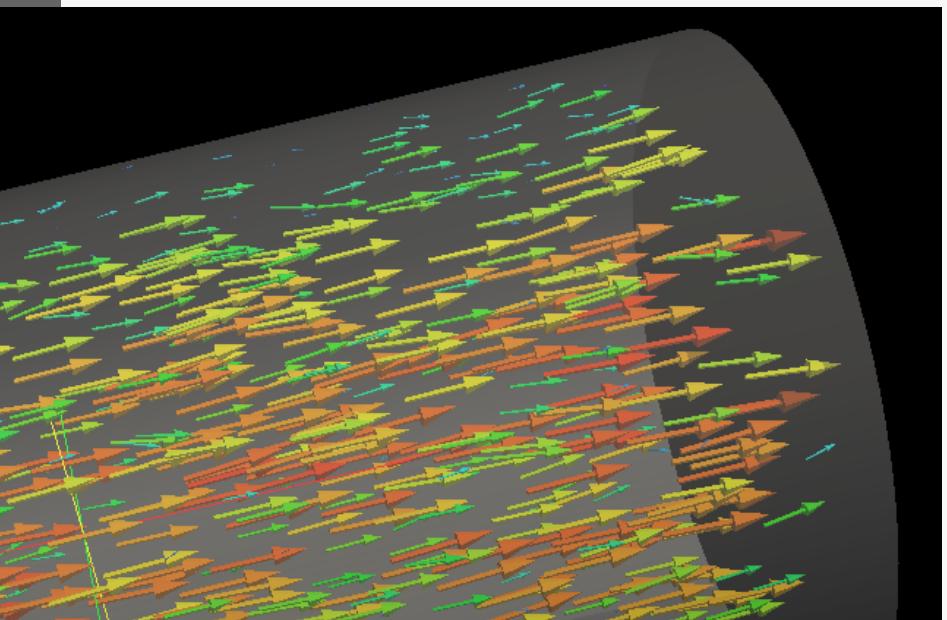
– 3D location on mesh

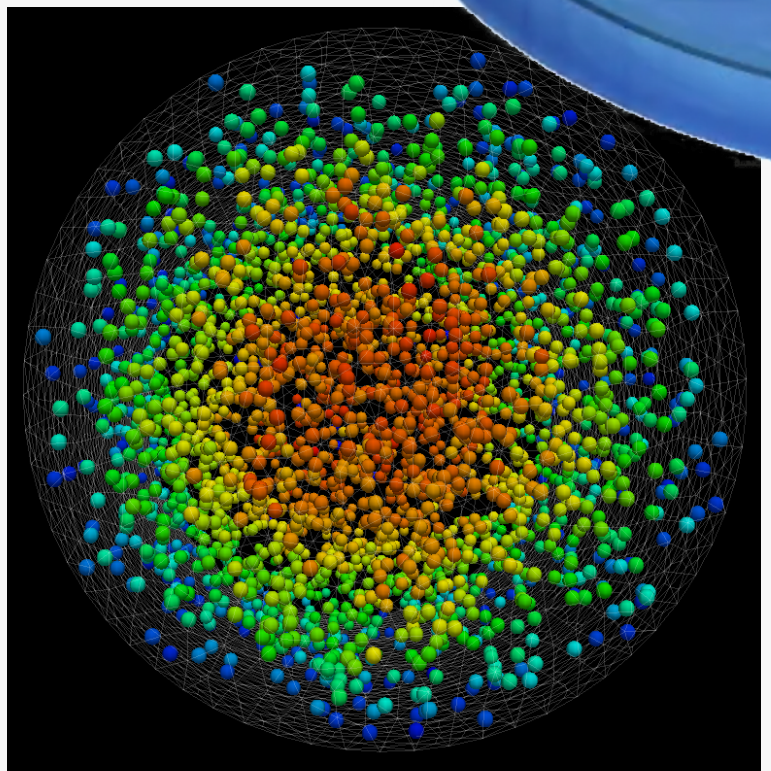
– 2D position in table/graph

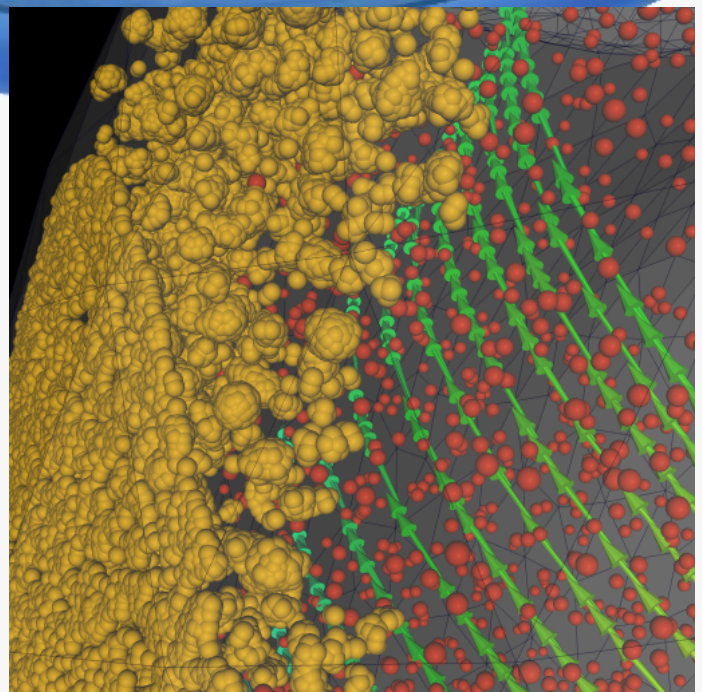
Attributes of graphical entity dictated by attributes of data

– color, size, orientation









Data Representations: Contours (Isosurfaces)

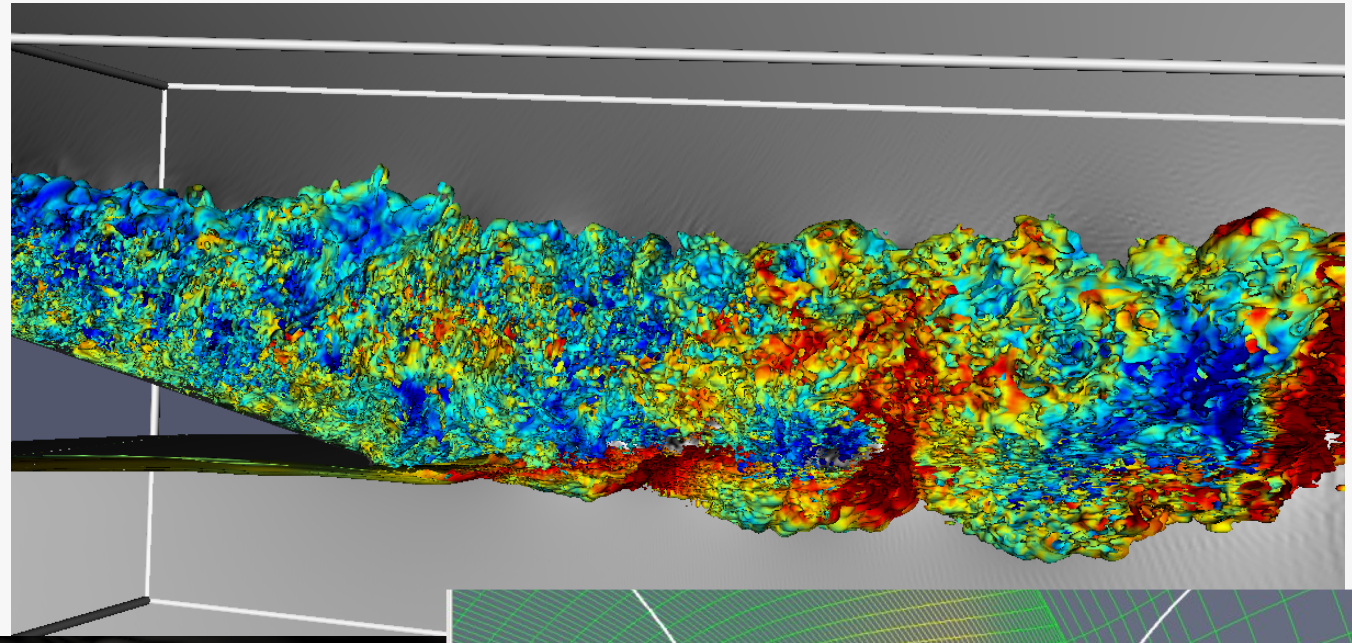
A Line (2D) or Surface (3D), representing a constant value

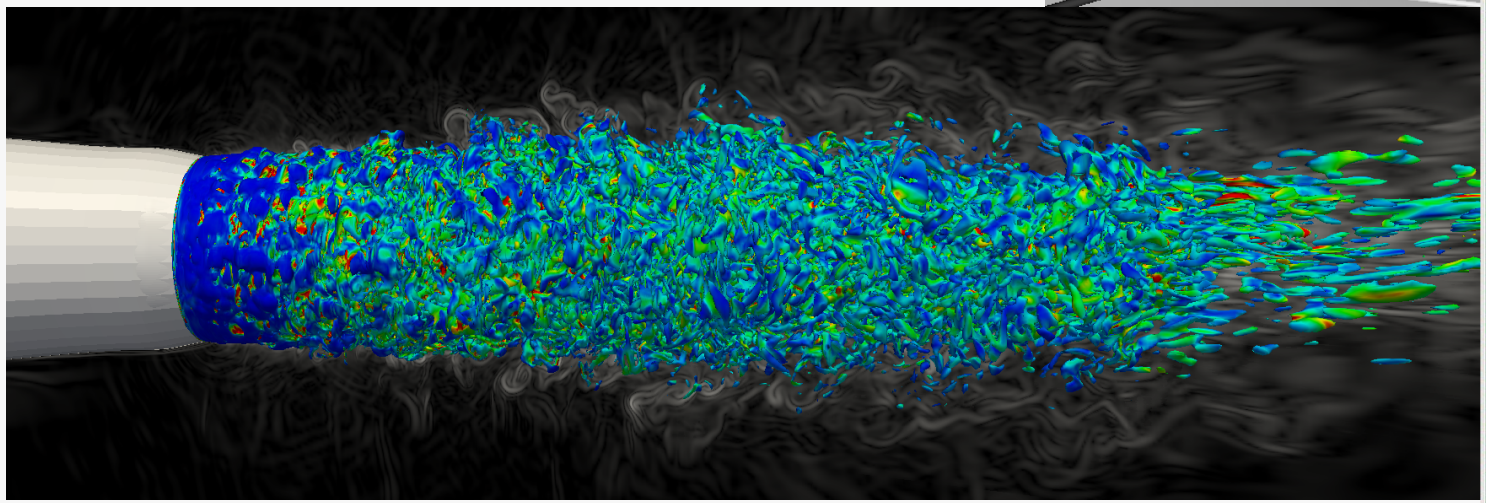
VisIt & ParaView:

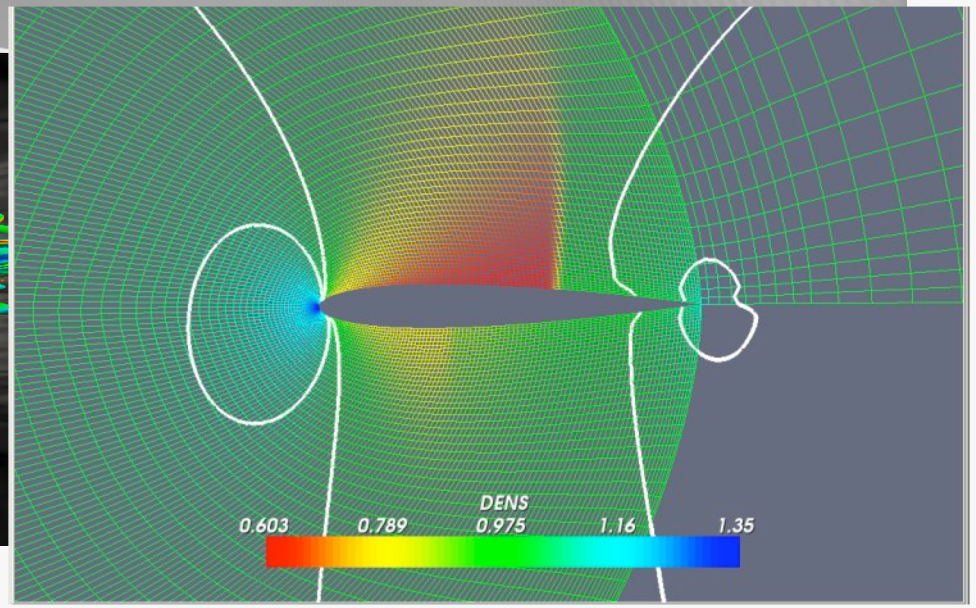
– good at this

vtk:

– same, but again requires more effort







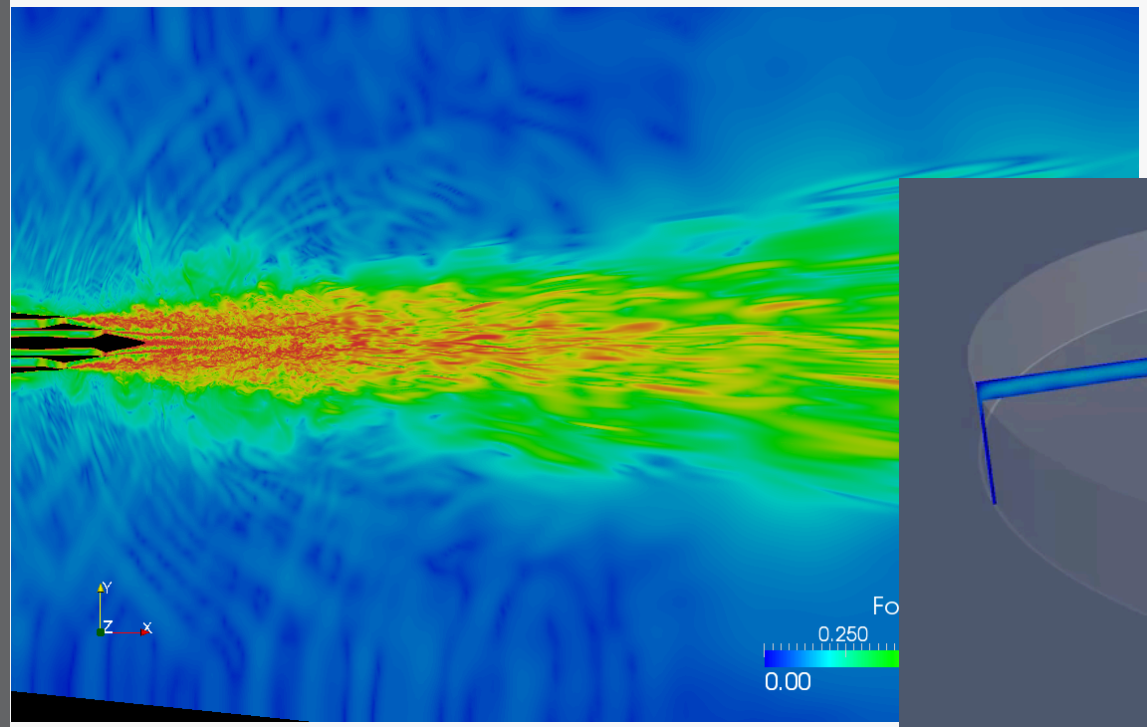
Data Representations: Cutting Planes

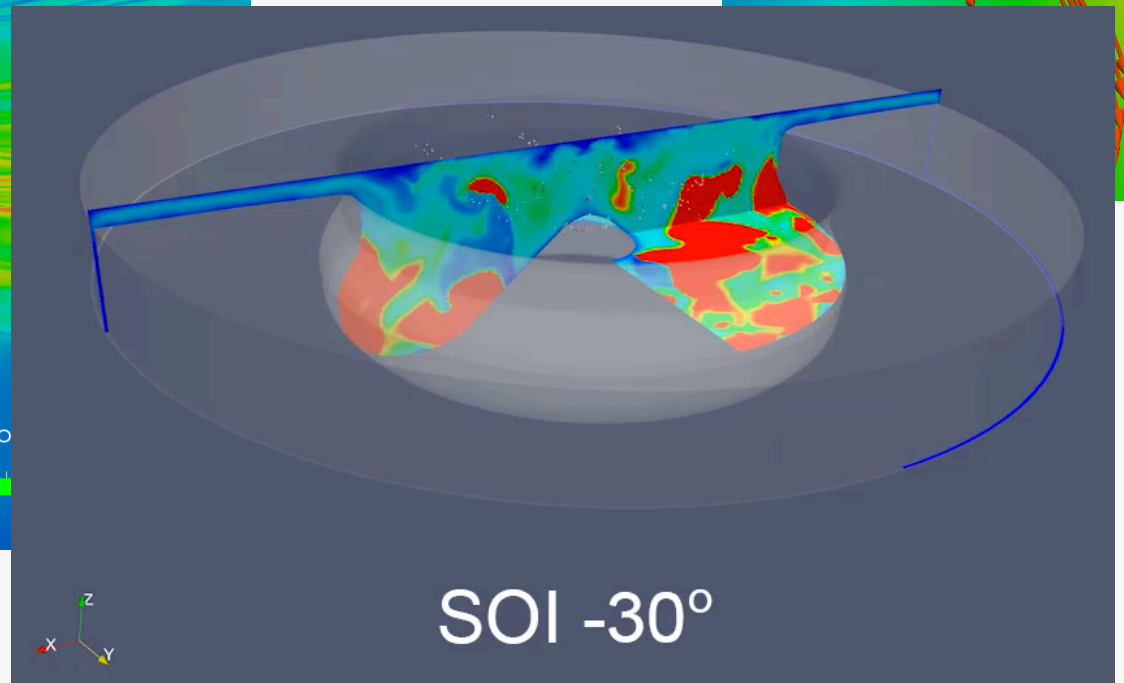
Slice a plane through the data

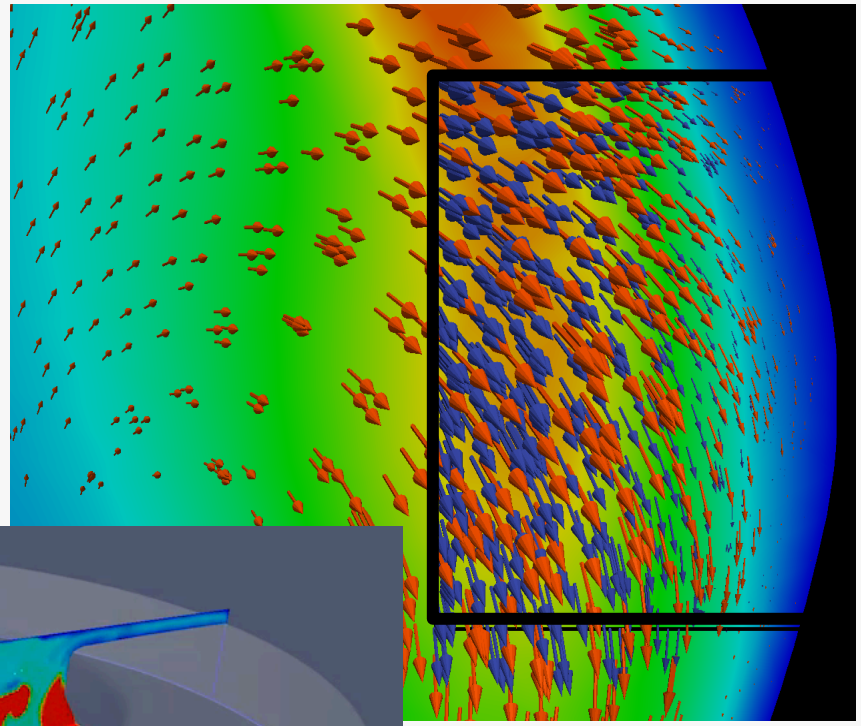
– Can apply additional visualization methods to resulting plane

VisIt & ParaView & vtk good at this

VMD has similar capabilities for some data formats





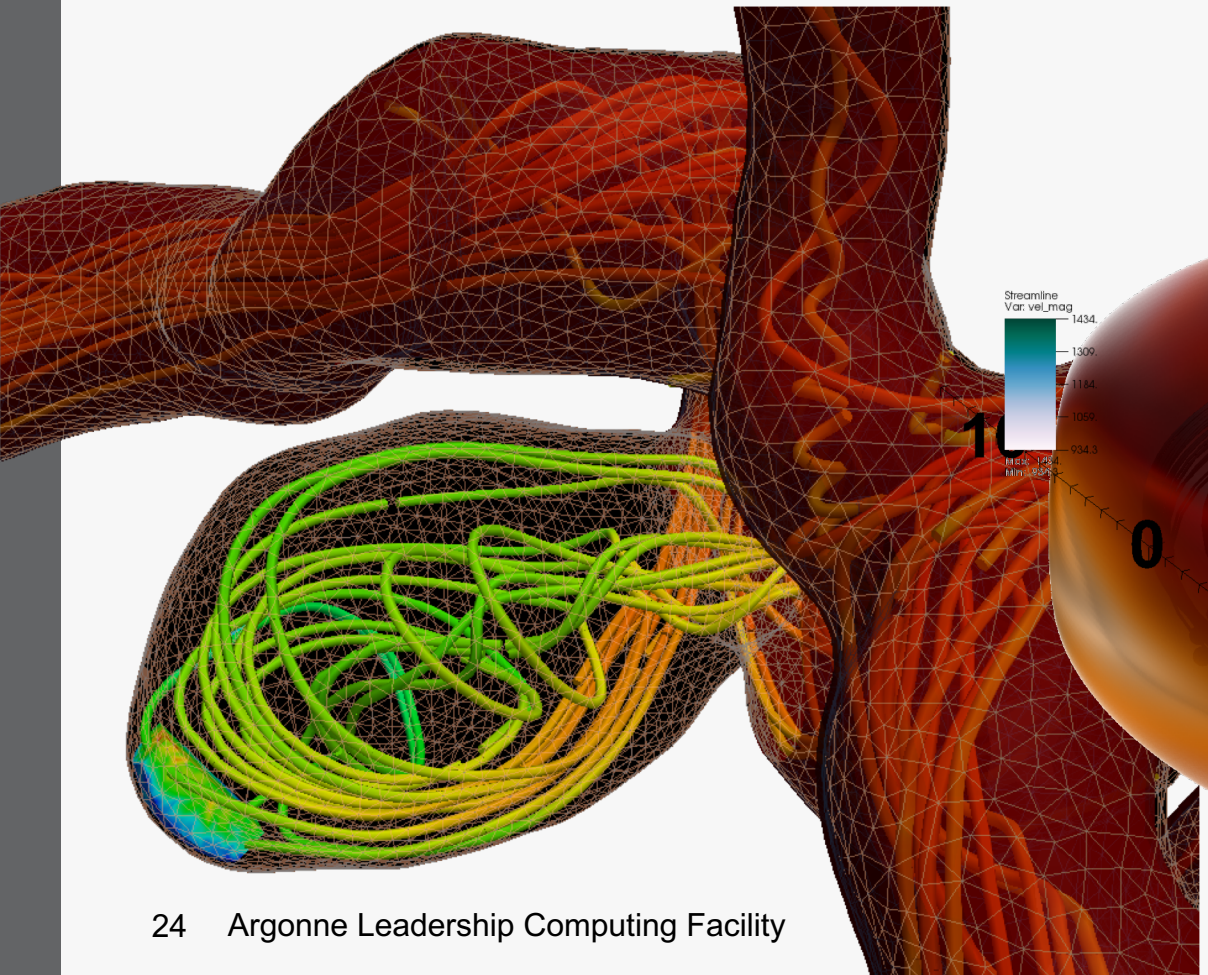


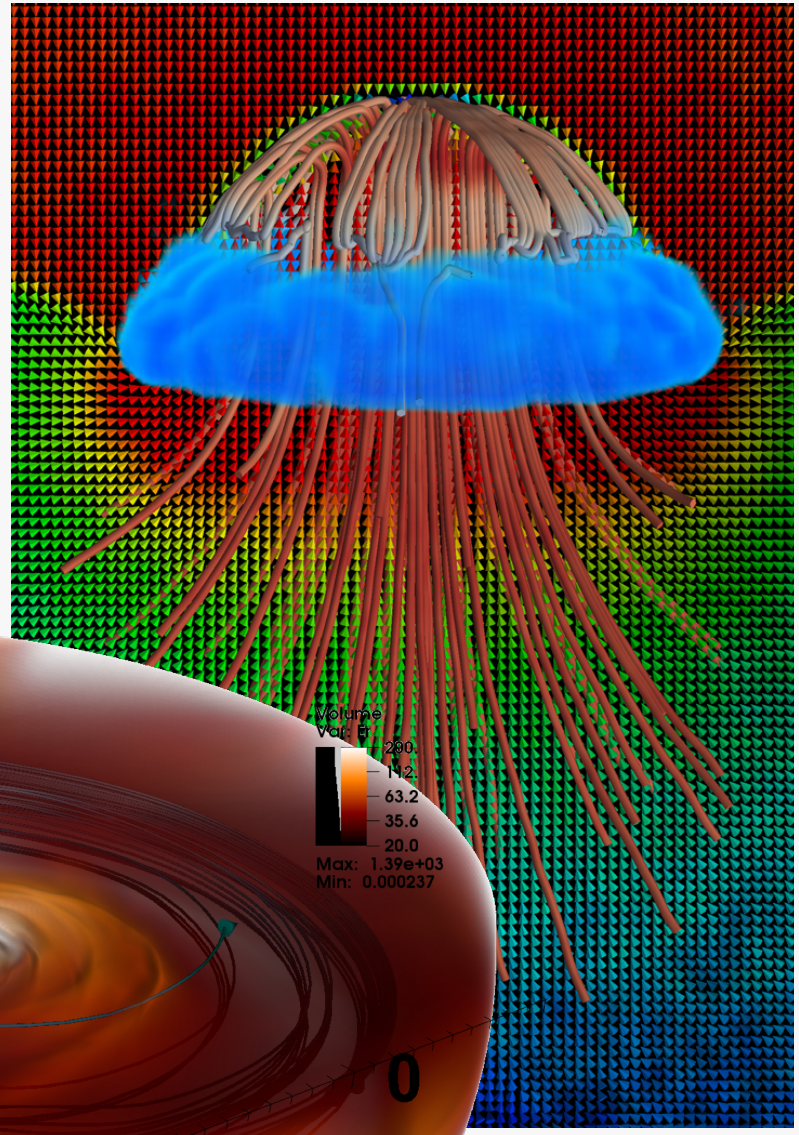
Data Representations: Streamlines

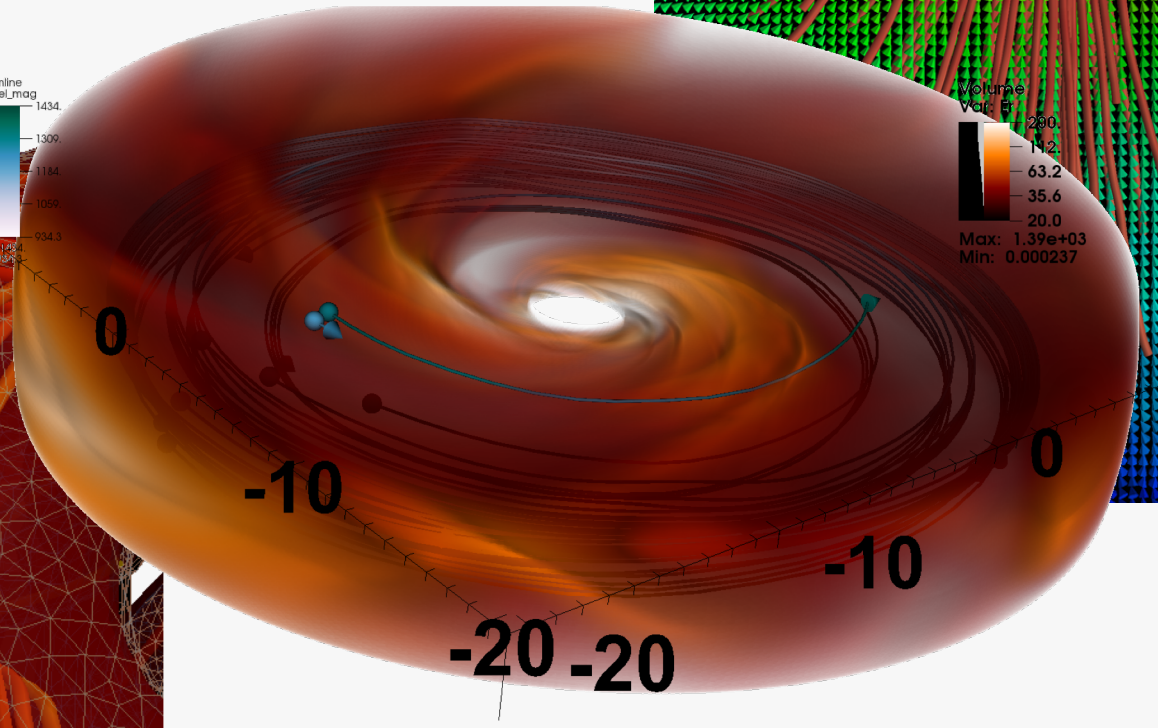
From vector field on a mesh (needs connectivity)

– Show the direction an element will travel in at any point in time.

VisIt & ParaView & vtk good at this







Molecular Dynamics Visualization

VMD:

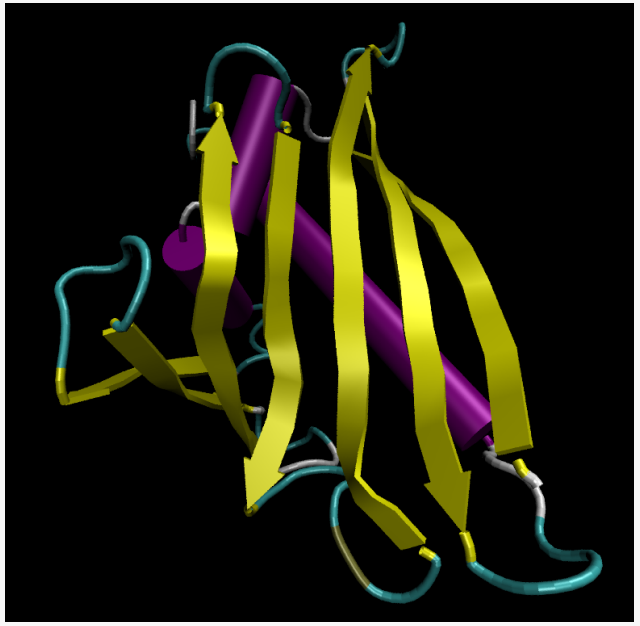
- Lots of domain-specific representations
- Many different file formats
- Animation
- Scriptable

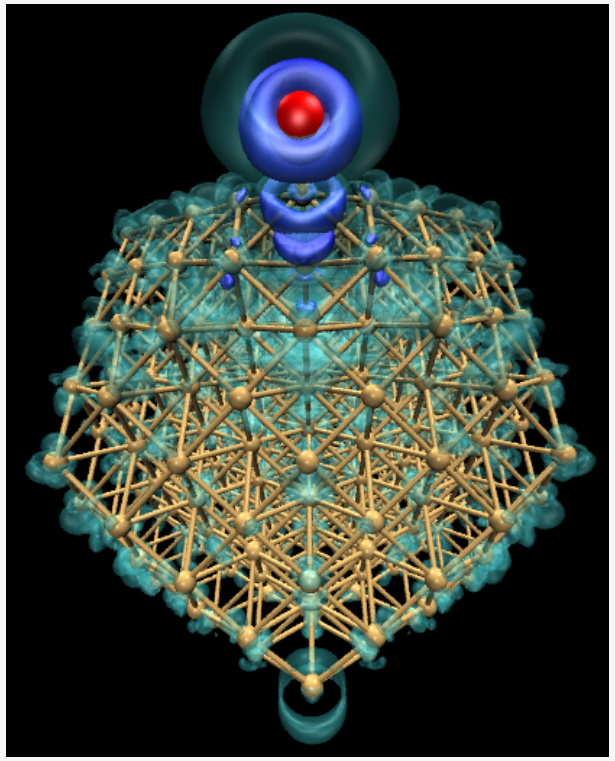
VisIt & ParaView:

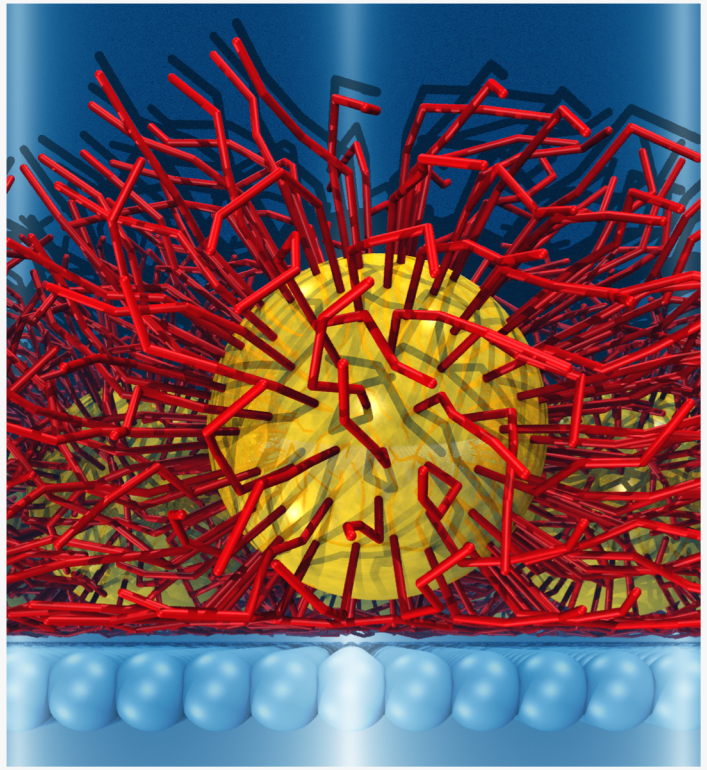
- Limited support for these types of representations, but improving

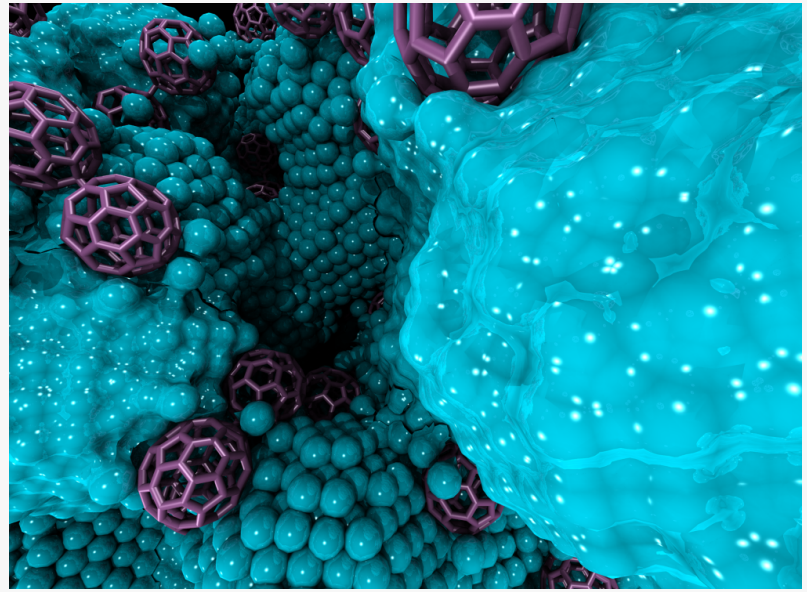
VTK:

- Anything's possible if you try hard enough



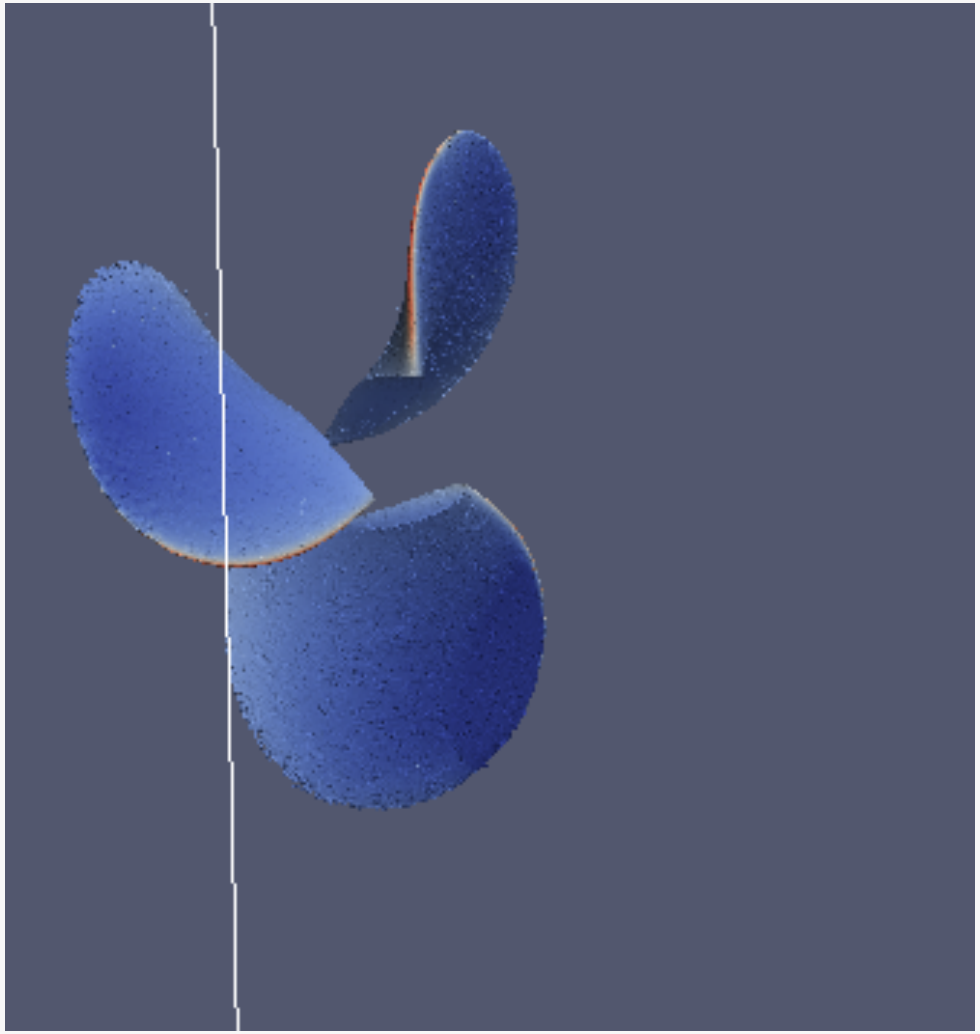






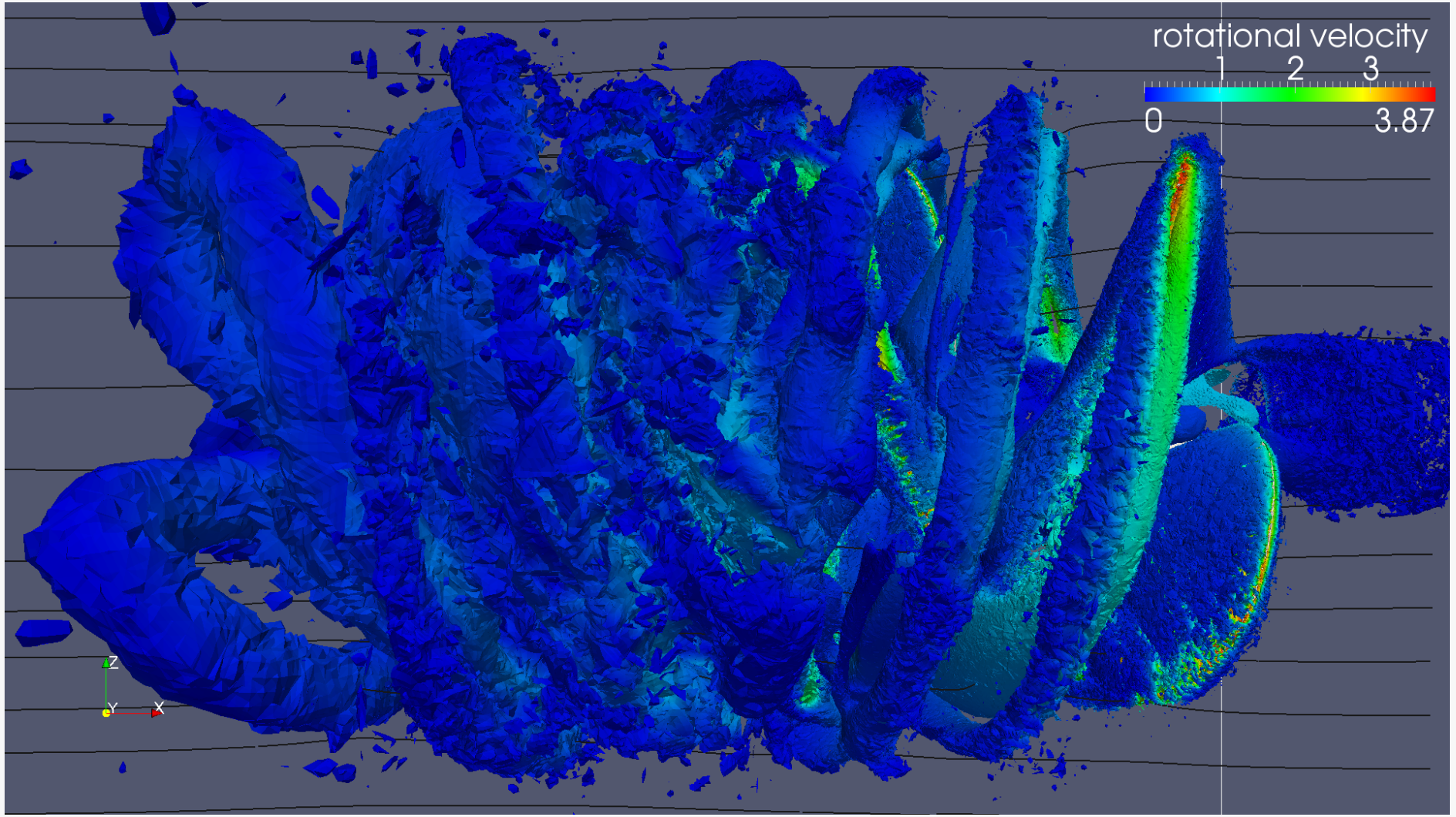
Visualization for Debugging

Visualization for Debugging

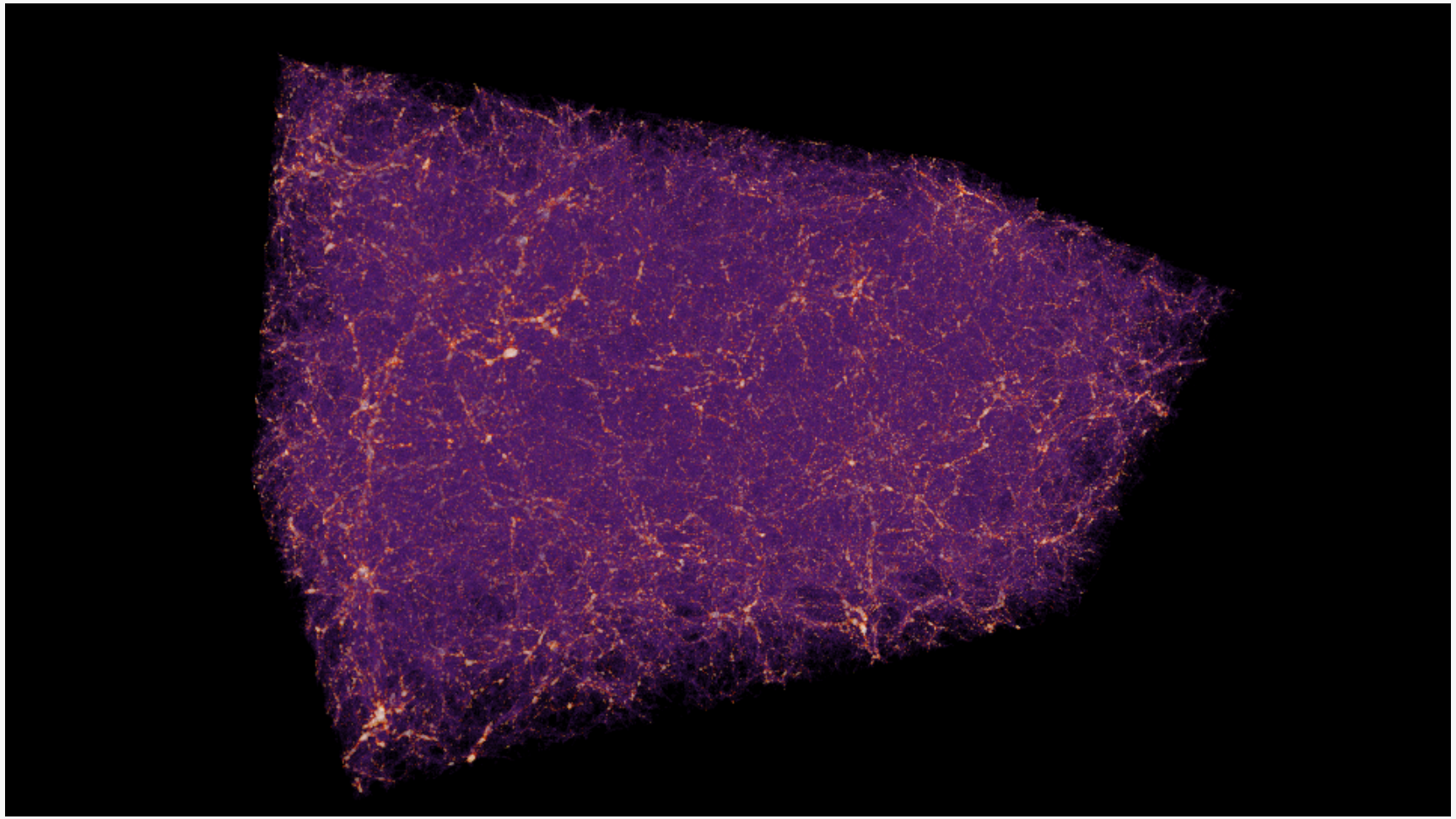


Argonne NATIONAL LABORATORY

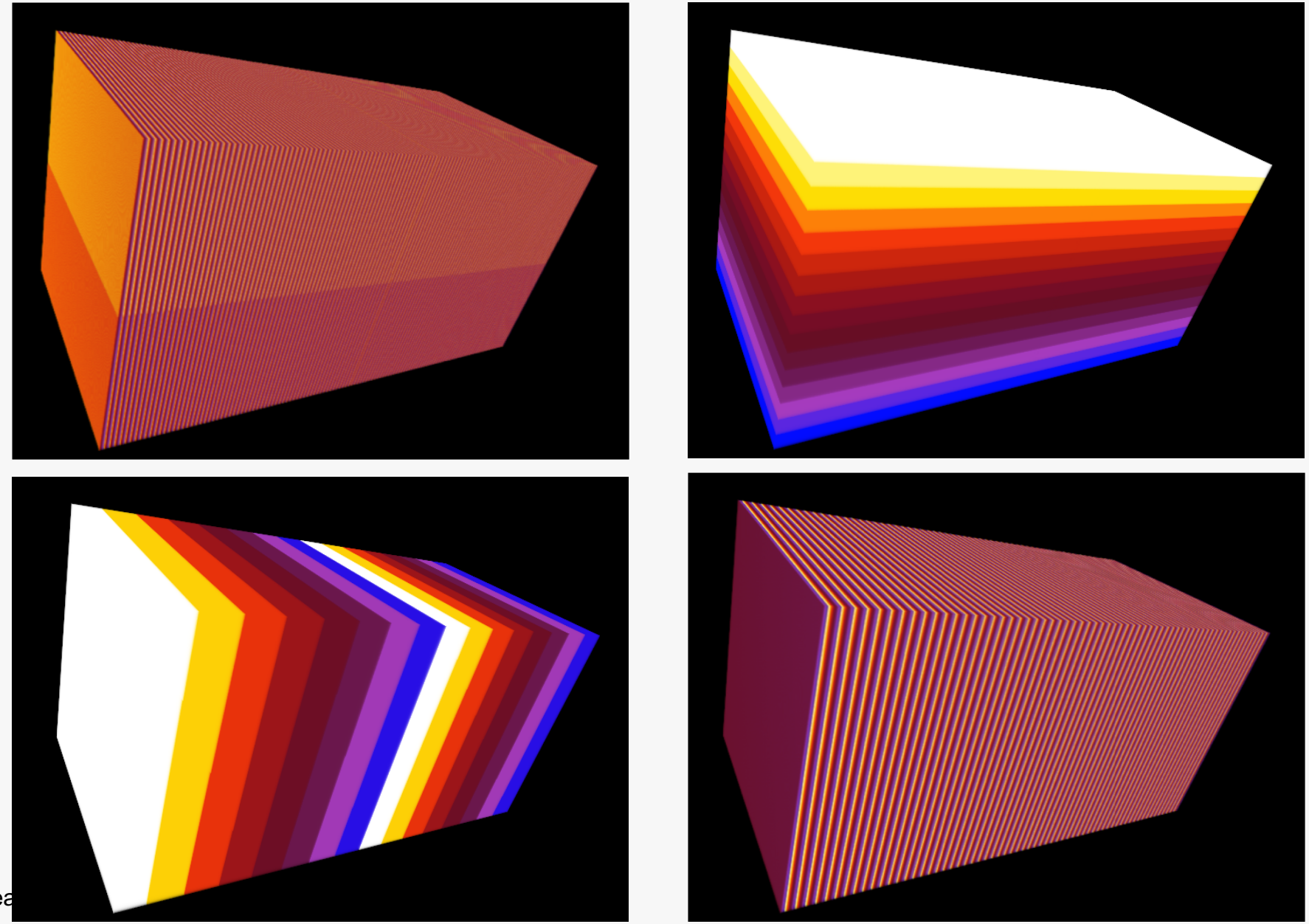
Visualization for Debugging



Visualization as Diagnostics: Color by Thread ID



Visualization as Diagnostics: Color by Thread ID



In Situ Visualization and Analysis

In situ visualization and analysis with SENSEI

Slides courtesy SENSEI in situ project:

www.sensei-insitu.org













The Need of *In Situ* Analysis and Visualization

Research challenges for enabling scientific knowledge discovery at extreme-scale concurrency

Widening gap between FLOPs and I/O capacity

– will make full-resolution, I/O-intensive post hoc analysis prohibitively expensive, if not impossible.

OUR APPROACH

Data model – to pass data between Simulation & Analysis

API – for instrumenting simulation and analysis codes

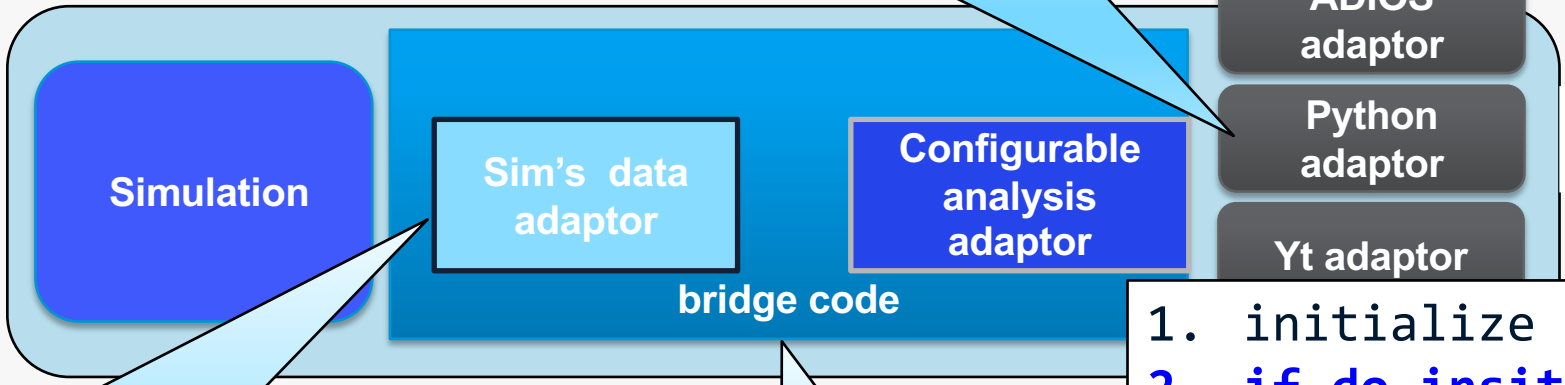


SENSEI In Situ Architecture

XML selects *one* of these at *runtime*

"write once, run everywhere"
A simulation can use any back-end through a single API & swap the back-ends at runtime

SENSEI's analysis adaptors provide the API for simulations to drive analysis and vis



Catalyst adaptor
Libsim adaptor
ADIOS adaptor
Python adaptor
Yt adaptor

ParaView Catalyst
VISIT
ADIOS
python
yt

SENSEI's data adaptor API and data model expose simulation data structures to the analysis back-end

Bridge/instrumentation code is added to call SENSEI Analysis. Typically: Initialize, Execute, Finalize

```
1.  initialize sim
2.  if do_insitu bridge::initialize
3.  do
4.    compute new state
5.    if do_io write plot file
6.    if do_insitu bridge::execute
7.  while !done
8.  if do_insitu bridge::finalize
9.  finalize sim
```

Using the configurable analysis adaptor

- Enable analysis in .xml file
- Run instrumented simulation

```
<sensei>
  <!-- Custom Analyses -->
  <analysis type="histogram" mesh="atoms" array="type" association="point"
    bins="10" enabled="0" />

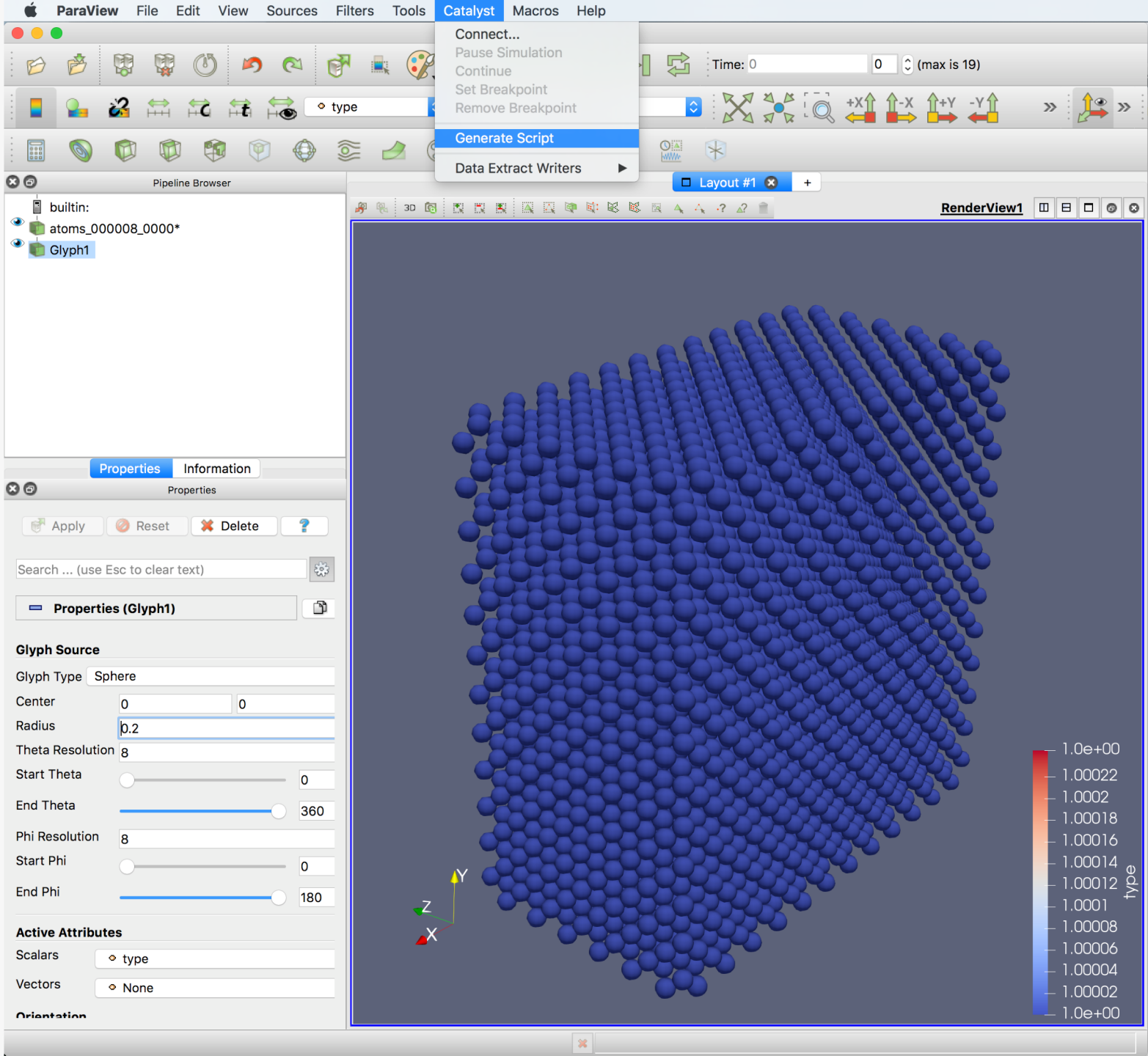
  <analysis type="histogram" mesh="atoms" array="id" association="point"
    bins="10" enabled="1" />

  <!-- Available with ENABLE_VTK_IO -->
  <analysis type="PosthocIO" mode="paraview" output_dir="./vtkio" enabled="0">
    <mesh name="atoms">
        <point_arrays> type, id</point_arrays>
    </mesh>
  </analysis>
</sensei>
```


Argonne NATIONAL LABORATORY

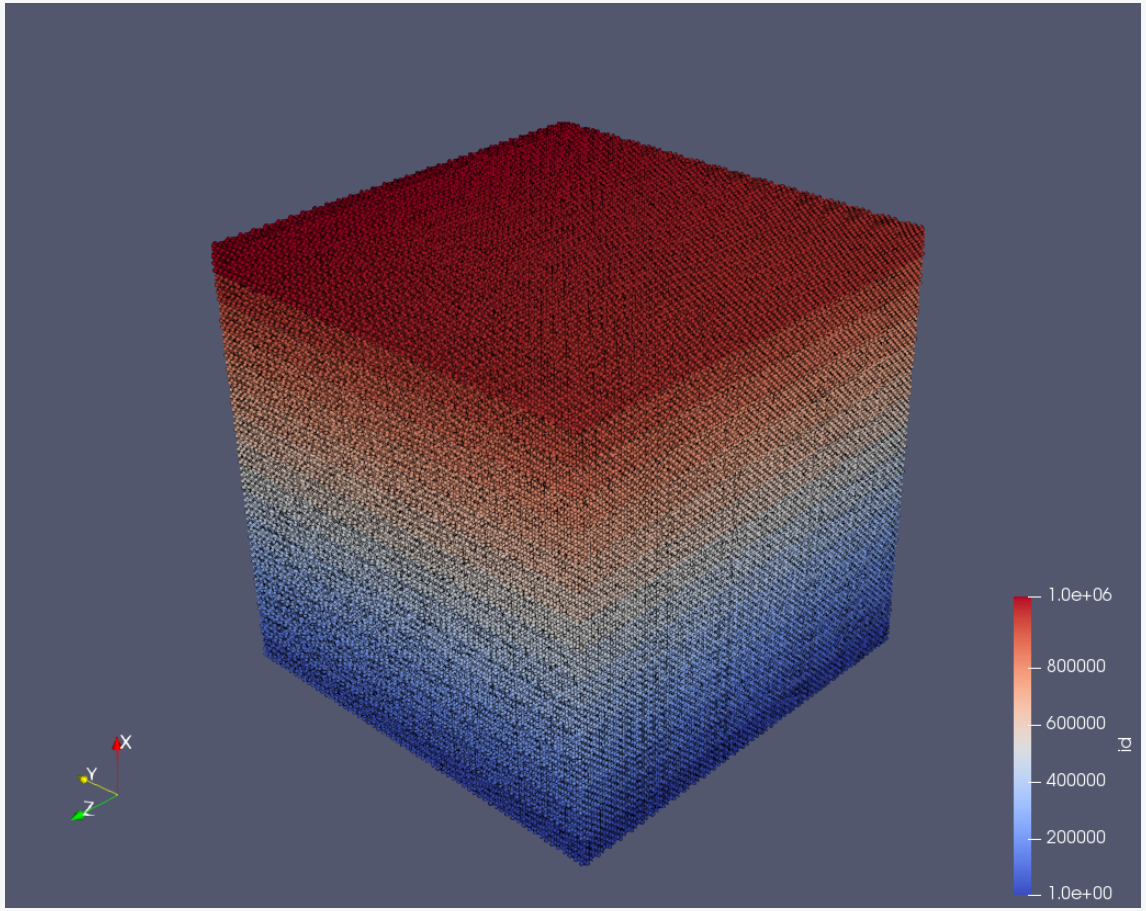
Catalyst example

- Load a representative dataset in ParaView
- Define your visualization pipeline
- Export Catalyst Python script



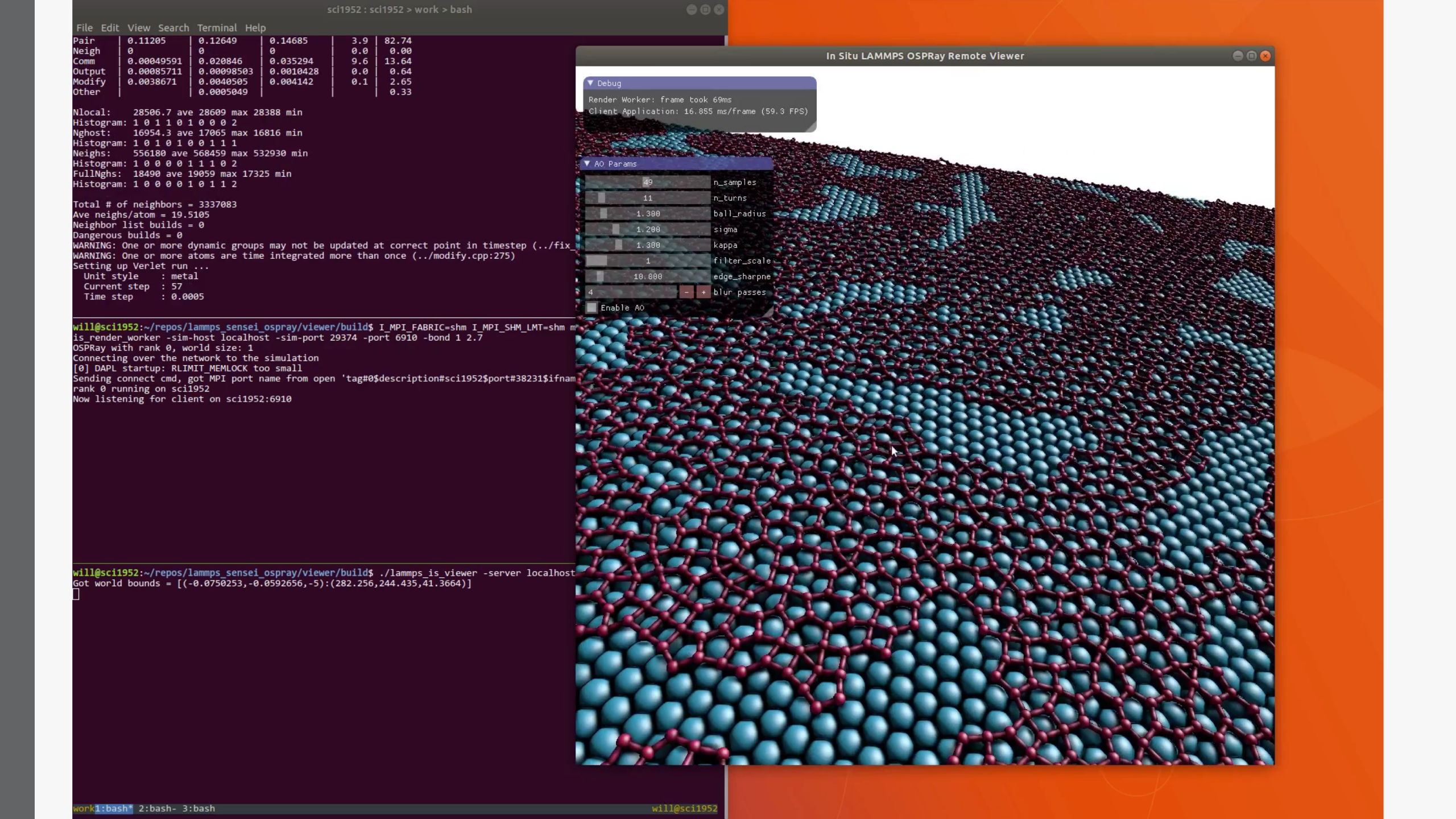
Catalyst example

- Configure XML file
- Run instrumented simulation
- Result: one .png image per simulation timestep




```
<sensei>
  <!-- Available with ENABLE_CATALYST -->
  <analysis type="catalyst" pipeline="pythonscript"
    filename="gaussianptsbyid.py" enabled="1" />
</sensei>
```


Argonne NATIONAL LABORATORY


sci1952 : sci1952 > work > bash
File Edit View Search Terminal Help
Pair | 0.11205 | 0.12649 | 0.14685 | 3.9 | 82.74
Neigh | 0 | 0 | 0 | 0.0 | 0.00
Comm | 0.00049591 | 0.020846 | 0.035294 | 9.6 | 13.64
Output | 0.00085711 | 0.00098503 | 0.0010428 | 0.0 | 0.64
Modify | 0.0038671 | 0.0040505 | 0.004142 | 0.1 | 2.65
Other | | 0.0005049 | | | 0.33
Nlocal: 28506.7 ave 28609 max 28388 min
Histogram: 1 0 1 1 0 1 0 0 0 2
Nghost: 16954.3 ave 17065 max 16816 min
Histogram: 1 0 1 0 1 0 0 1 1 1
Neighs: 556180 ave 568459 max 532930 min
Histogram: 1 0 0 0 0 1 1 1 0 2
FullNghs: 18490 ave 19059 max 17325 min
Histogram: 1 0 0 0 0 1 0 1 1 2
Total # of neighbors = 3337083
Ave neighs/atom = 19.5105
Neighbor list builds = 0
Dangerous builds = 0
WARNING: One or more dynamic groups may not be updated at correct point in timestep (../fix_
WARNING: One or more atoms are time integrated more than once (../modify.cpp:275)
Setting up Verlet run ...
Unit style : metal
Current step : 57
Time step : 0.0005
will@sci1952:~/repos/lammps_sensei_ospray/viewer/build$ I_MPI_FABRIC=shm I_MPI_SHM_LMT=shm m
is_render_worker -sim-host localhost -sim-port 29374 -port 6910 -bond 1 2.7
OSPRay with rank 0, world size: 1
Connecting over the network to the simulation
[0] DAPL startup: RLIMIT_MEMLOCK too small
Sending connect cmd, got MPI port name from open 'tag#0$description#sci1952$port#38231$ifnam
rank 0 running on sci1952
Now listening for client on sci1952:6910
will@sci1952:~/repos/lammps_sensei_ospray/viewer/build$ ./lammps_is_viewer -server localhost
Got world bounds = [(-0.0750253,-0.0592656,-5):(282.256,244.435,41.3664)]
work 1:bash* 2:bash- 3:bash
will@sci1952
In Situ LAMMPS OSPRay Remote Viewer
Debug
Render Worker: frame took 69ms
Client Application: 16.855 ms/frame (59.3 FPS)
AO Params
49 n_samples
11 n_turns
1.300 ball_radius
1.200 sigma
1.300 kappa
1 filter_scale
10.000 edge_sharpne
4 blur passes
Enable AO

In Situ Visualization and Analysis with Ascent

EXASCALE COMPUTING PROJECT

Slides courtesy:

Hank Childs, University of Oregon
Matthew Larsen, Lawrence Livermore National Laboratory
Cyrus Harrison, Lawrence Livermore National Laboratory
Kenneth Moreland, Sandia National Laboratories
David Rogers, Los Alamos National Laboratory



Ascent focuses on ease of use and efficient in-situ execution

Ascent Delivers

An easy to use API

- Designed to enable three use cases
  - Making Pictures
  - Transforming Data
  - Capturing Data
- Leverages Conduit (http://software.llnl.gov/conduit)
  - Underpins support for C, C++, Fortran, and Python
  - Simplifies handoff of mesh-based simulation data
  - Convention for specifying data called “Blueprint”

```
//
// Run Ascent
//

Ascent ascent;
ascent.open();
ascent.publish(data);
ascent.execute(actions);
ascent.close();
```

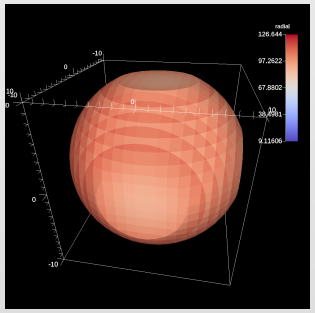
A flyweight design

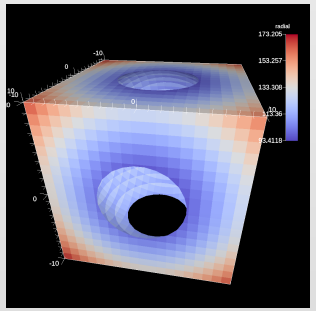
- Efficient distributed-memory + many-core execution
  - Leverages MPI, VTK-m (http://m.vtk.org/)
- Lower memory requirements then current tools
- Less dependencies than current tools (ex: no OpenGL)

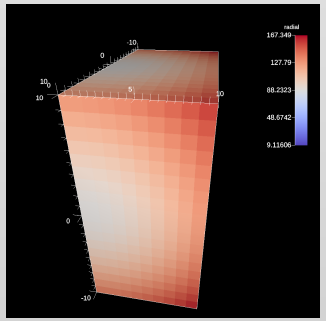


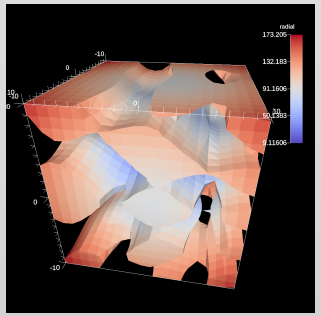


Ascent is ready for common visualization use cases

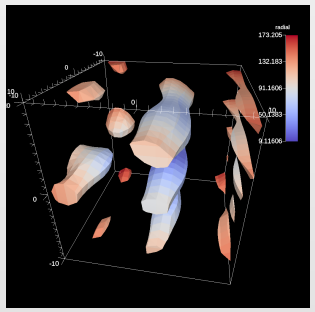




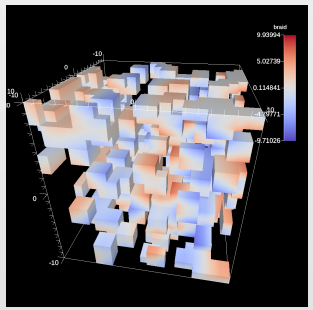




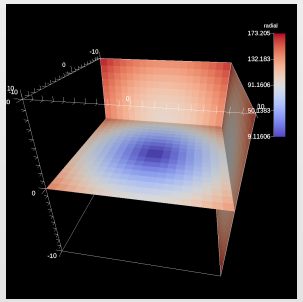
Clips



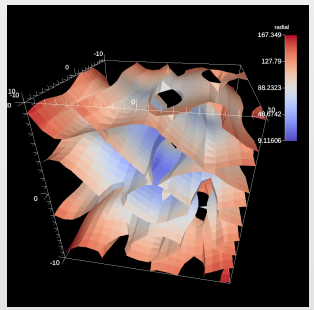
Iso-Volume



Threshold

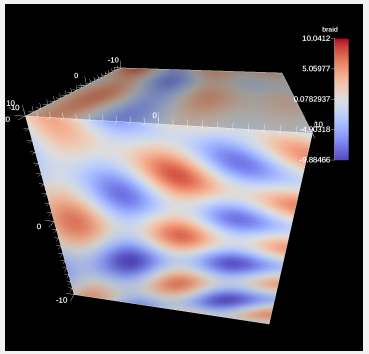


Slice

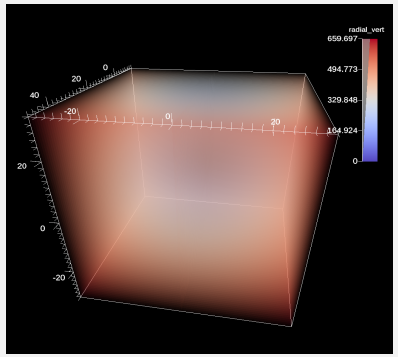


Contour

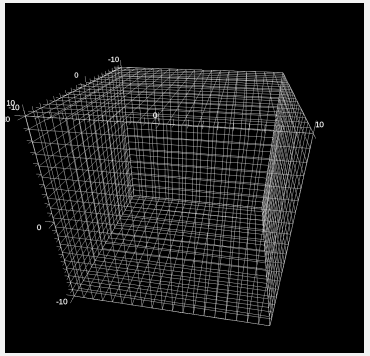
Rendering



Pseudocolor



Volume



Mesh

Design Goal: Support custom analysis as a first class citizen

Mainstream visualization only gets you so far

- Scientists often want something other than a contour

In-situ visualization frameworks need to be

- Flexible
- Easy to use
- Easy to connect with other “things”

Ascent supports multiple languages and output types

Language Bindings





Fortran

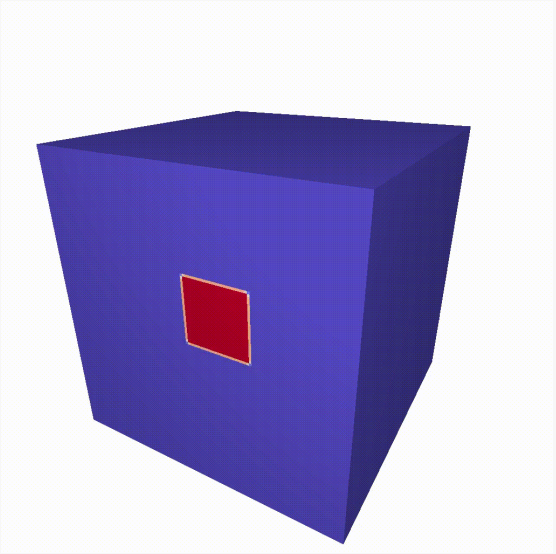
Output Types



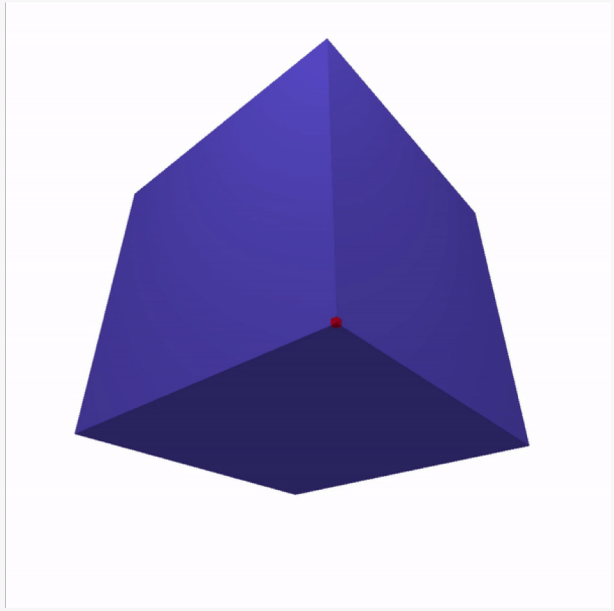




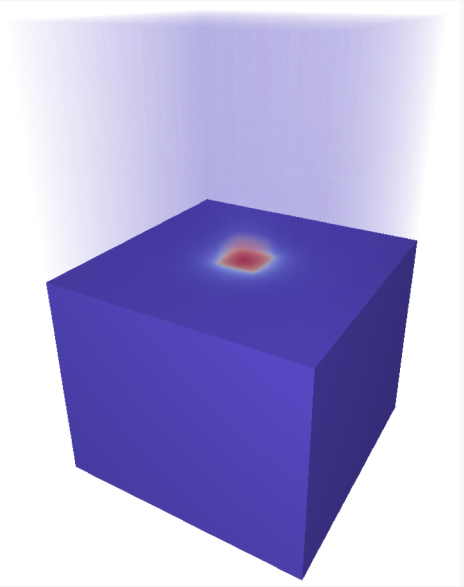
Ascent provides example integrations that also serve as built-in data sources



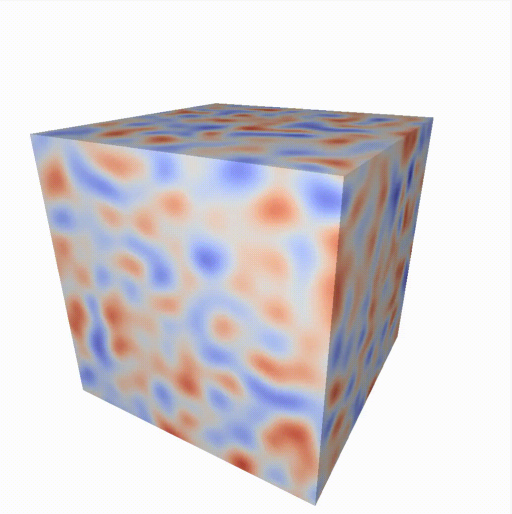
Cloverleaf3D



Lulesh



Kripke



Smooth Noise

Additional Resources

SENSEI additional resources



- SENSEI project page
  https://sensei-insitu.org/

- Gitlab repo
  https://gitlab.kitware.com/sensei/sensei

- SENSEI SC18 Tutorial - Slides and Virtual Machine
  https://sensei-insitu.org/tutorials/sc18.html

Ascent additional resources

Website + Docs:

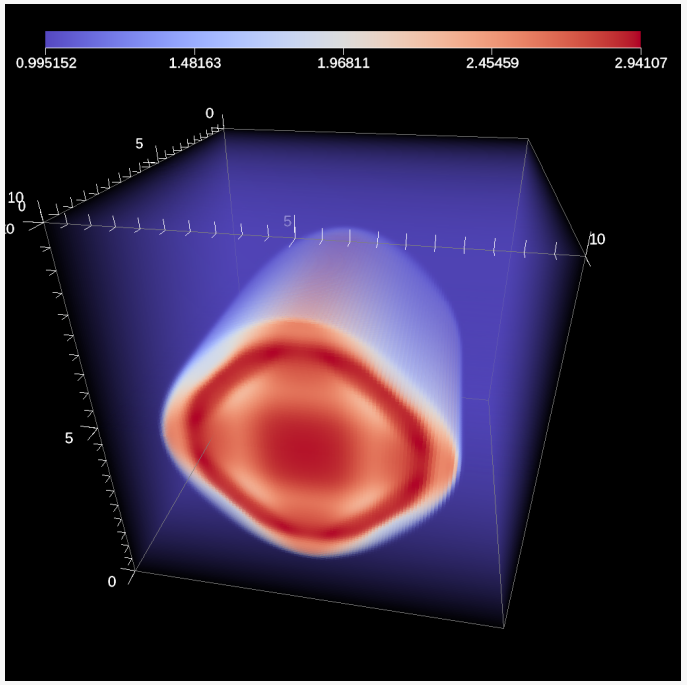
http://ascent-dav.org

GitHub Repo:

https://github.com/Alpine-DAV/ascent

Email Help:

help@ascent-dav.org



Example in-situ rendering created using Ascent

Resources for optional hands-on session

- ParaView installers
  Download version 5.5.2
  https://www.paraview.org/download/
- ALCF ParaView Red Blood Cell Tutorial
  https://www.alcf.anl.gov/user-guides/vis-paraview-red-blood-cell-tutorial
- ParaView on Cooley
  https://www.alcf.anl.gov/user-guides/paraview-cooley
- Data on Cooley
  `/lus/thetafs0/projects/Comp_Perf_Workshop/visualization/BLOODFLOW_TUTORIAL_DATA`

Argonne NATIONAL LABORATORY

QUESTIONS?

Joe Insley
insley@anl.gov

Silvio Rizzi
srizzi@anl.gov