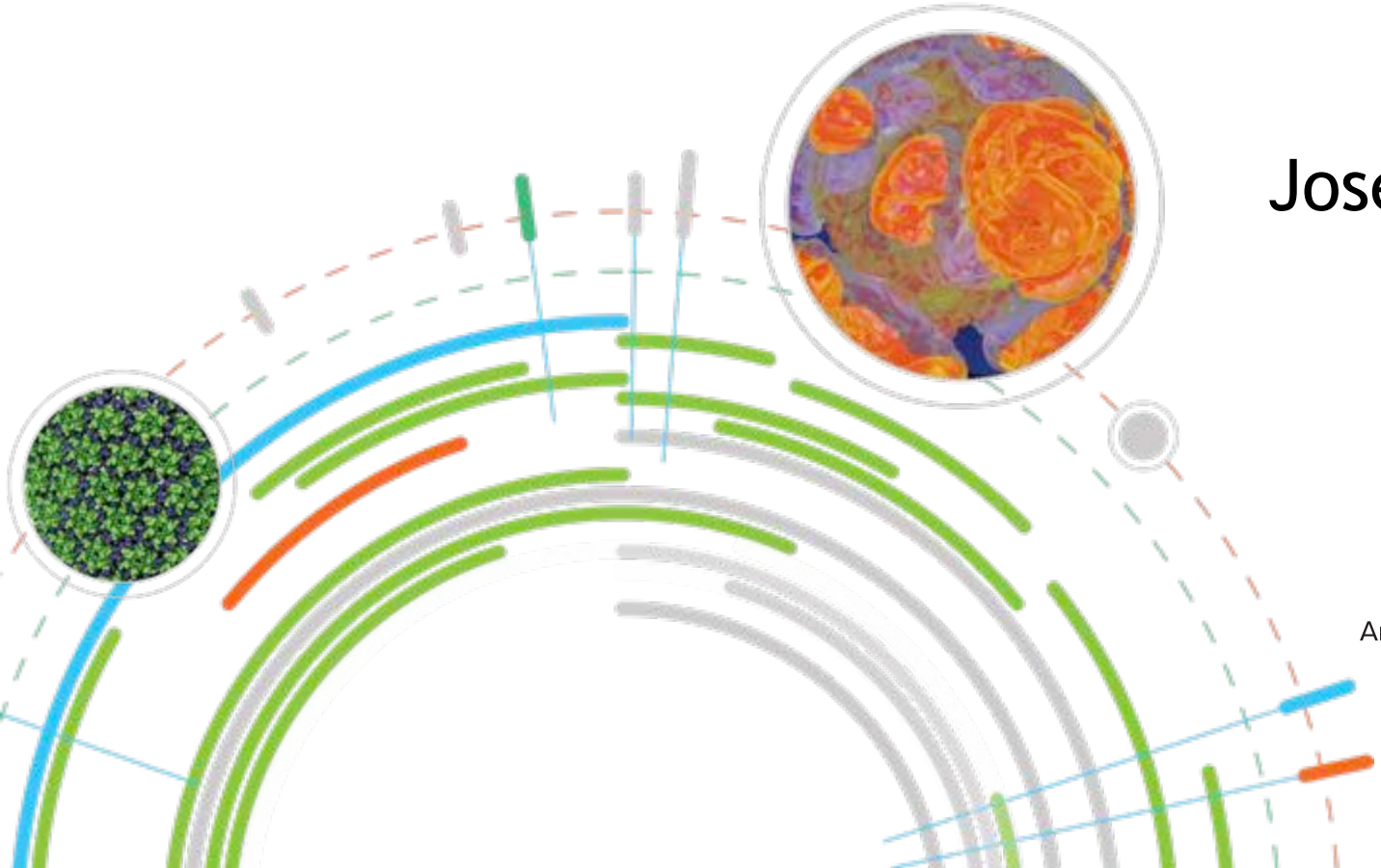
Your Data Analysis & Visualization Needs



Joseph A. Insley

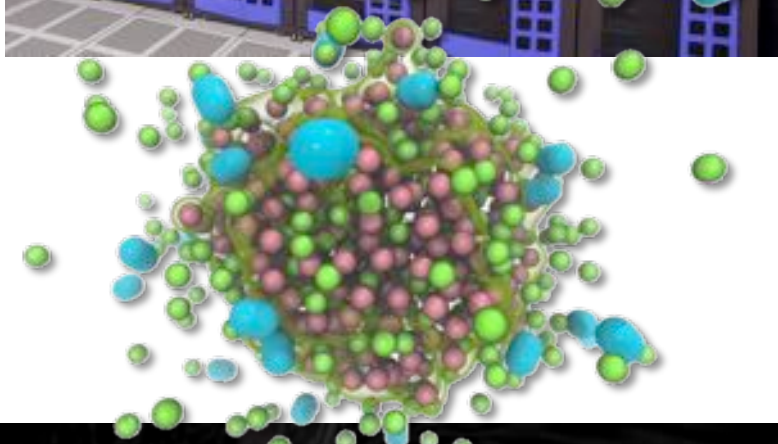
Scaling Your Science on Mira
May 25, 2016

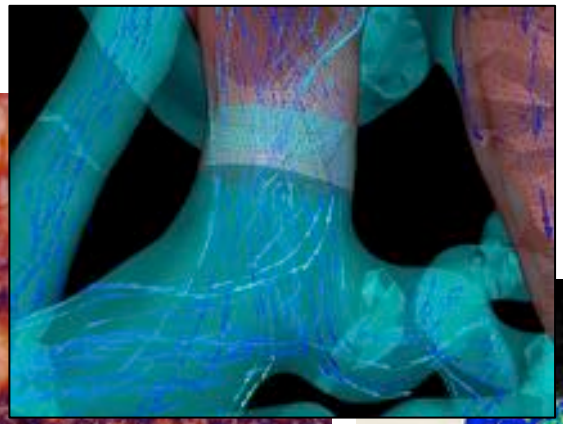
Argonne **Leadership Computing** Facility

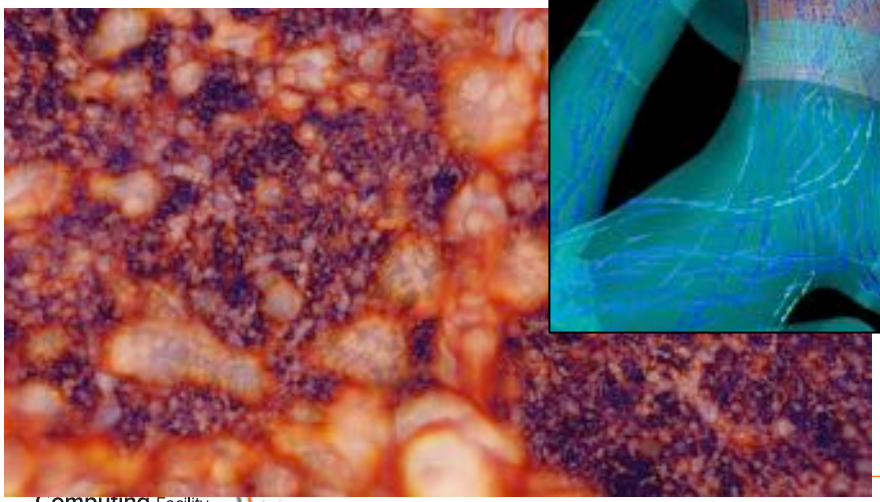
Argonne NATIONAL LABORATORY

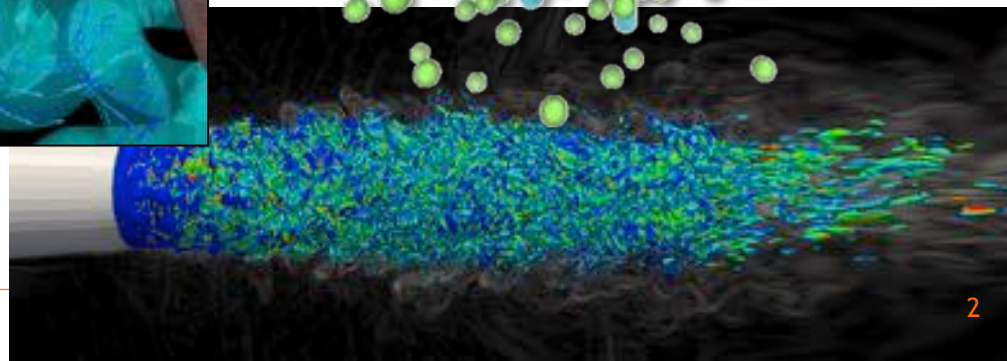
High Performance Visualization Resource: Cooley

- 126 nodes; each node has
  - 2 -Intel Xeon Haswell 2.4 GHz 6-core processors
  - NVIDIA Telsa K80 graphics processing unit (24GB)
  - **384 GB of RAM**
- Peak 223 TF (425 on Top500, Nov 2015)
- Aggregate RAM of 47 TB
- Aggregate GPU memory of~3TB
- Cross-Mounted file systems with Mira



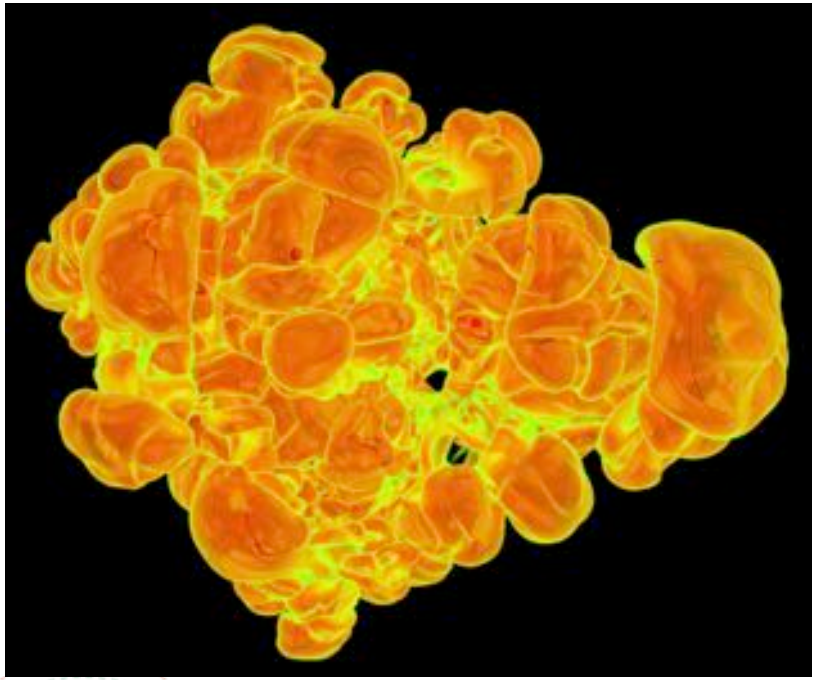


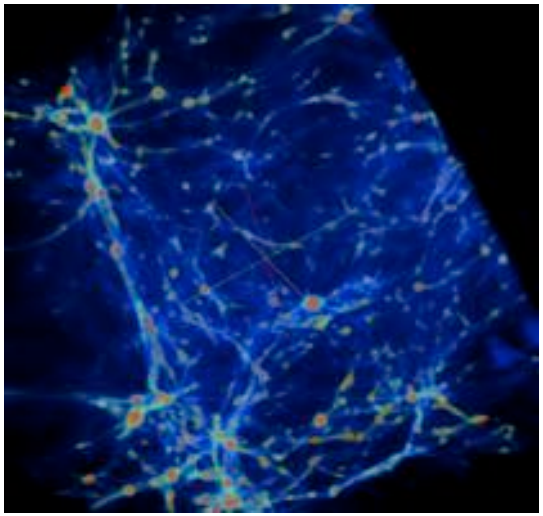


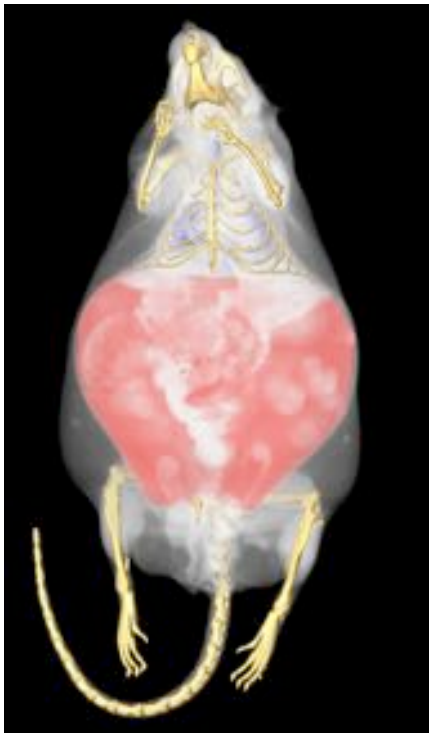


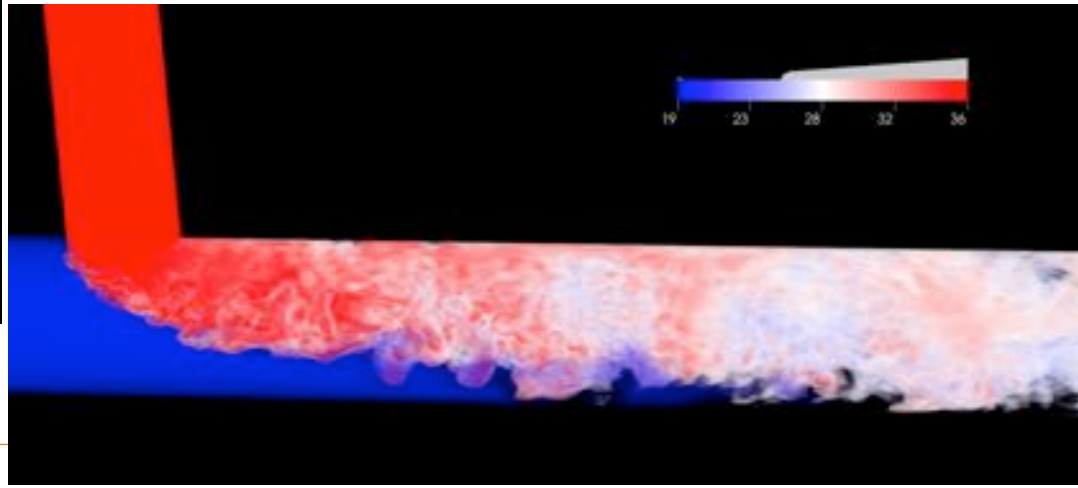
Data Representations: Volume Rendering

- Turn 2- and 3-dimensionsal datasets into 2D images
- Approximation: Volume ray casting



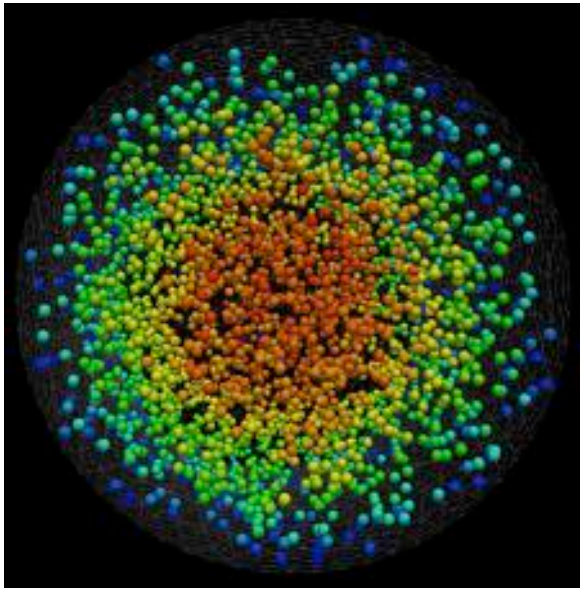


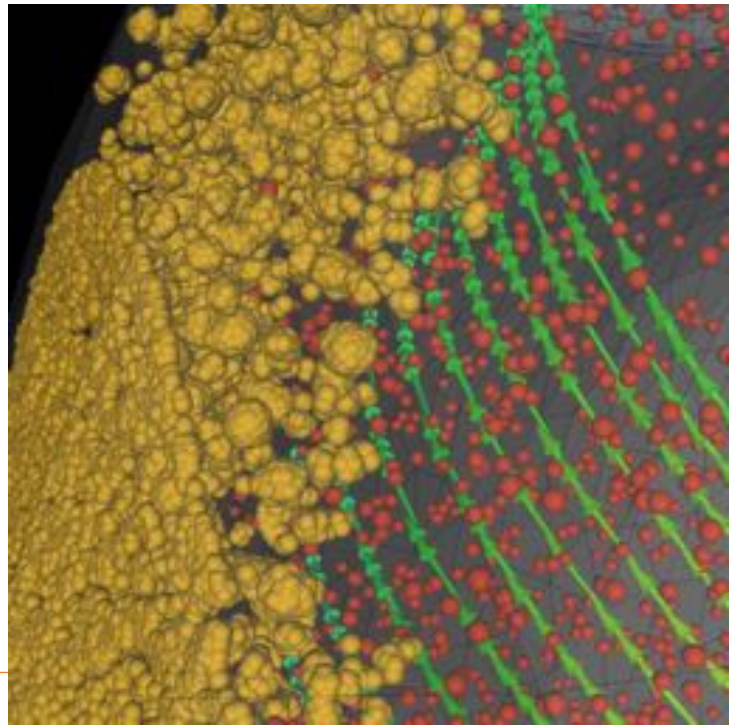


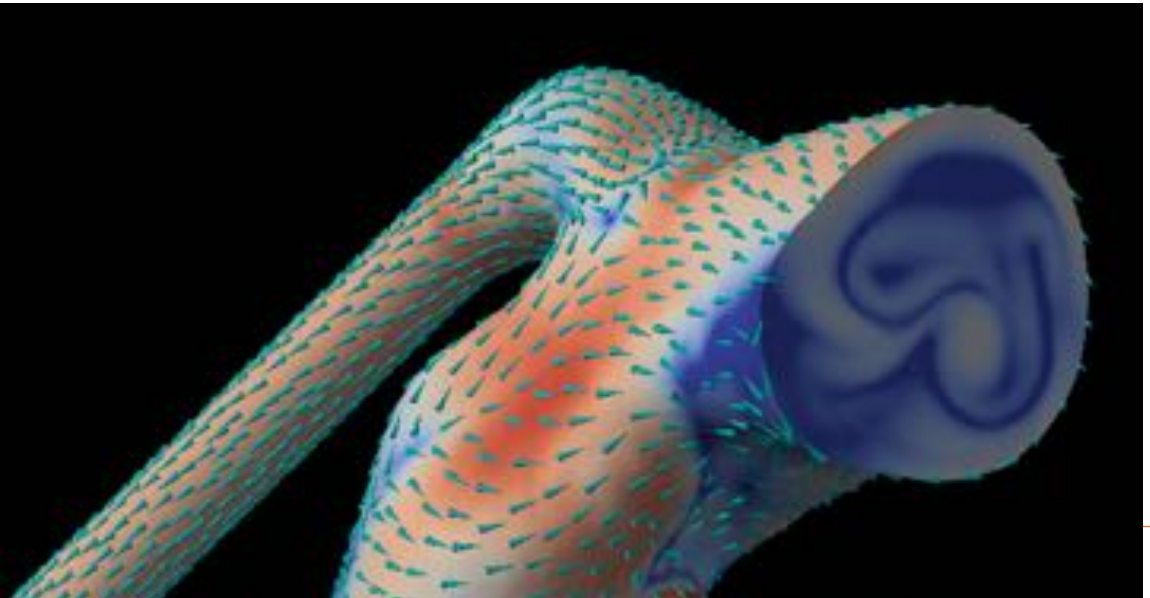


Data Representations: Glyphs

- 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 a data
  - color, size, orientation

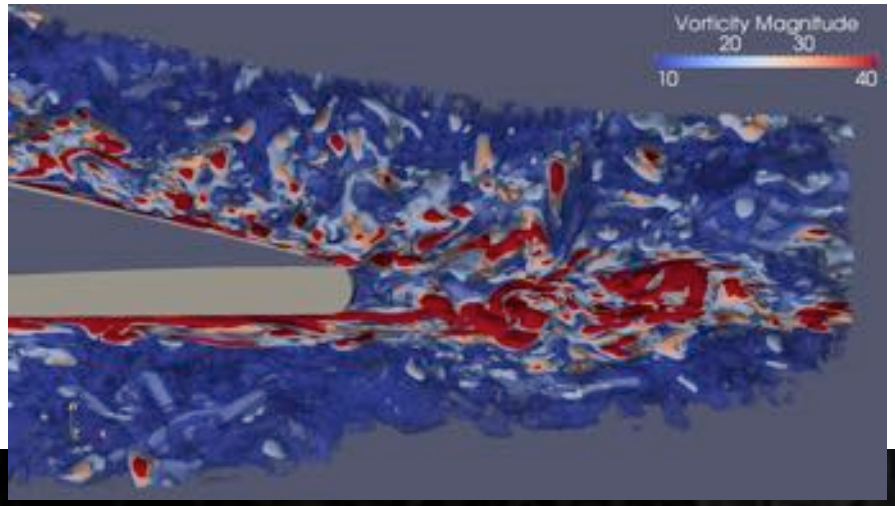


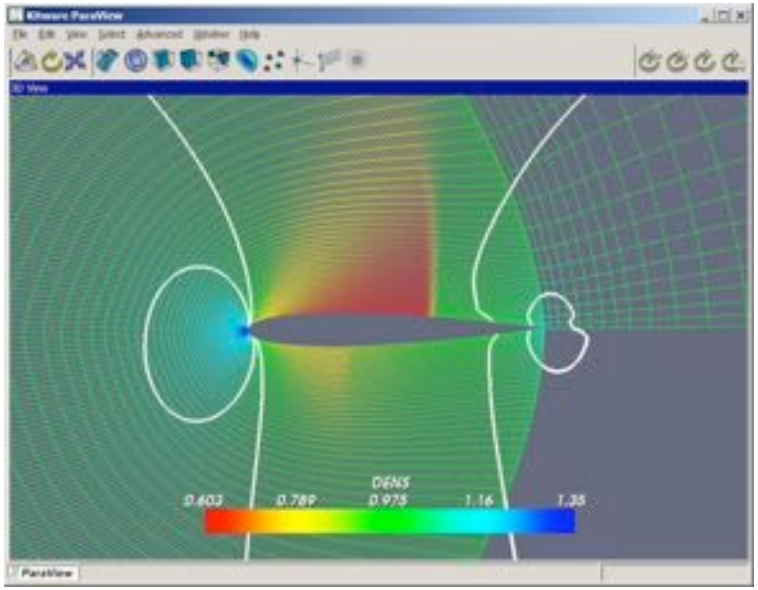


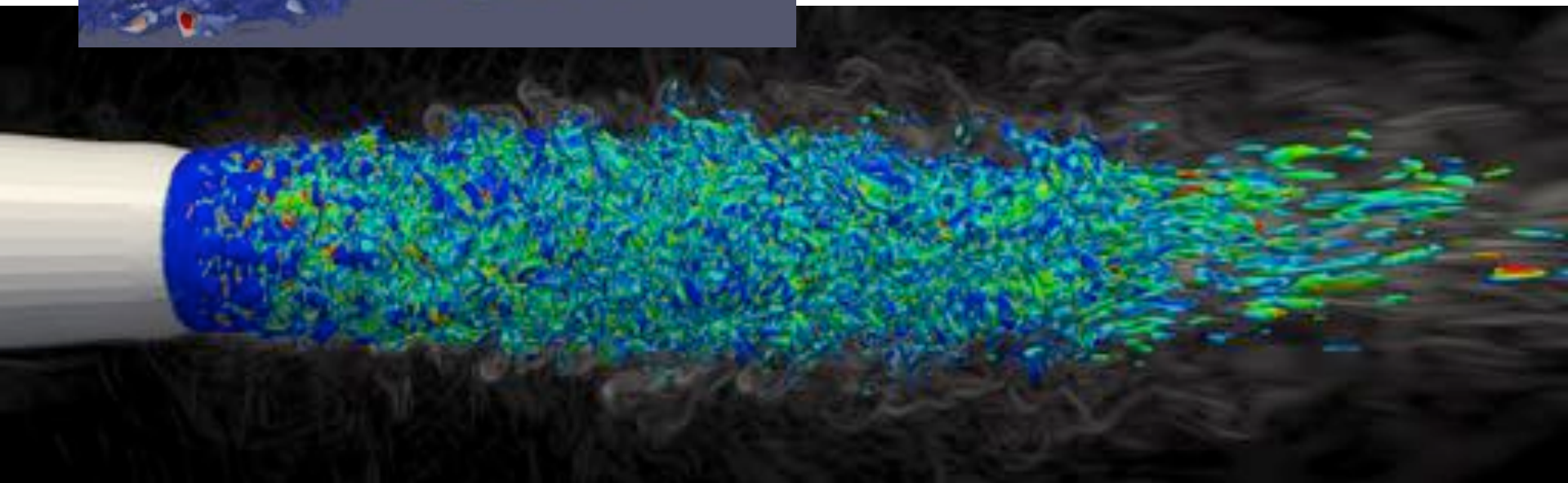


Data Representations: Iso-contours

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

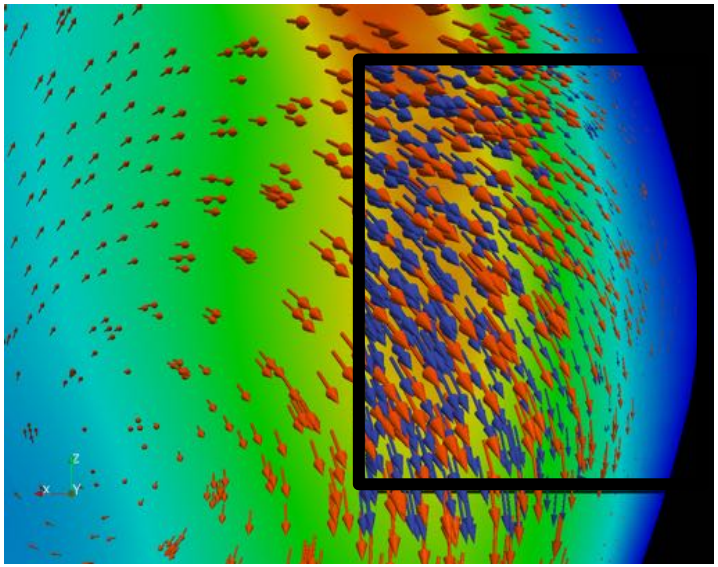


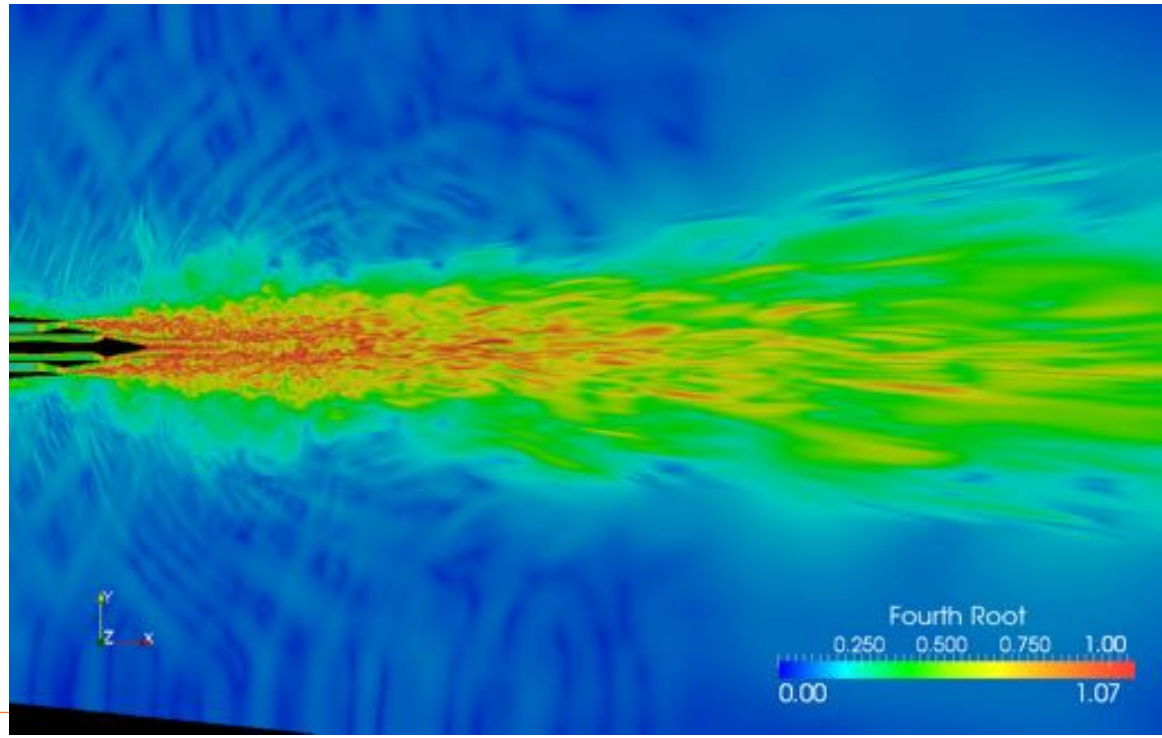




Data Representations: Cutting Planes

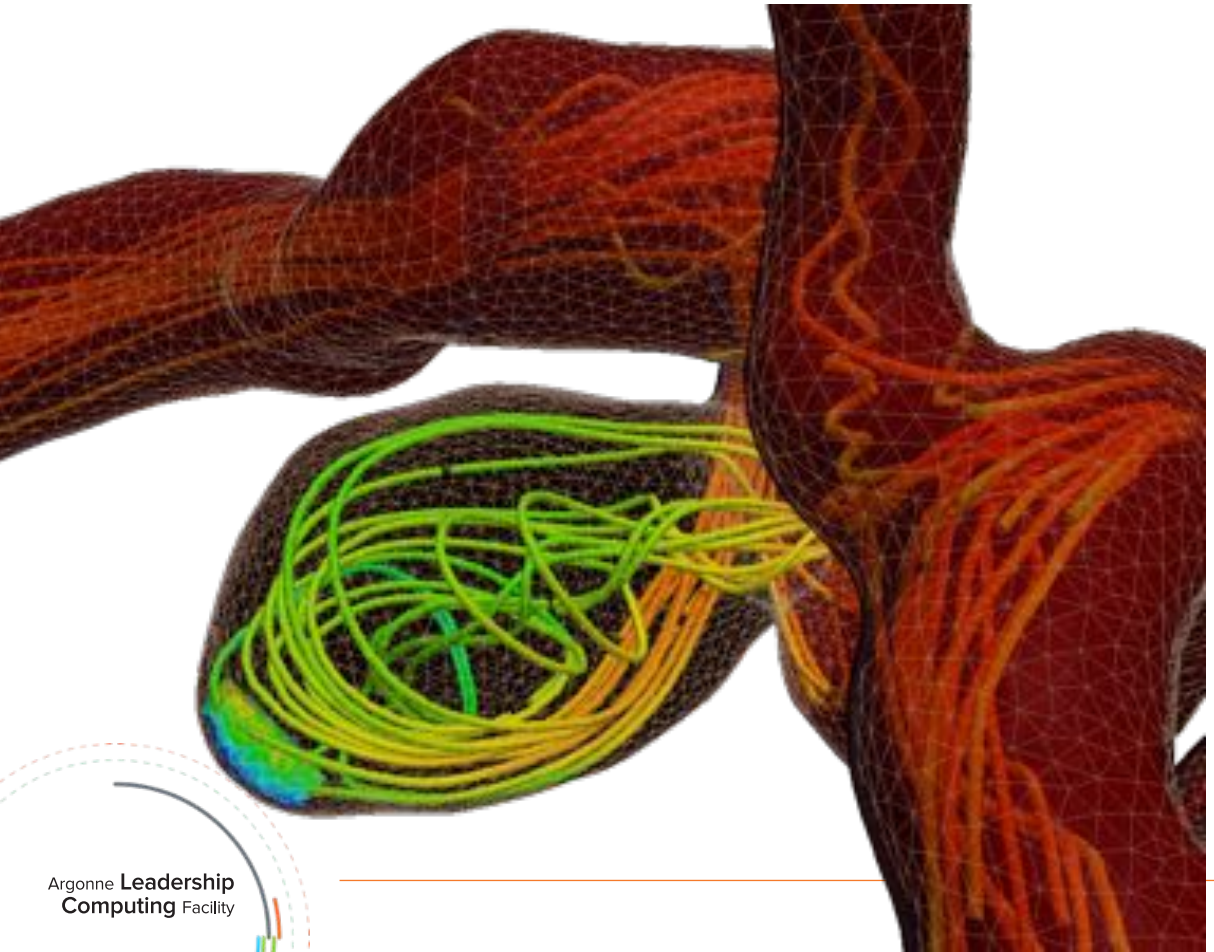
- Slice a plane through the data
  - Can apply additional visualization methods to resulting plane

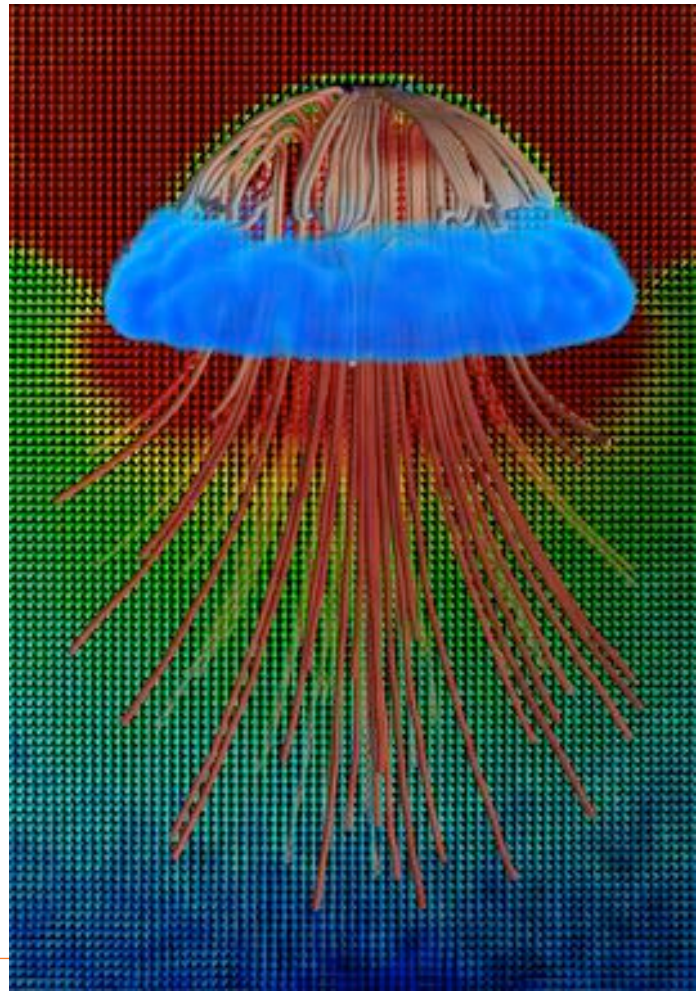




Data Representations: Streamlines

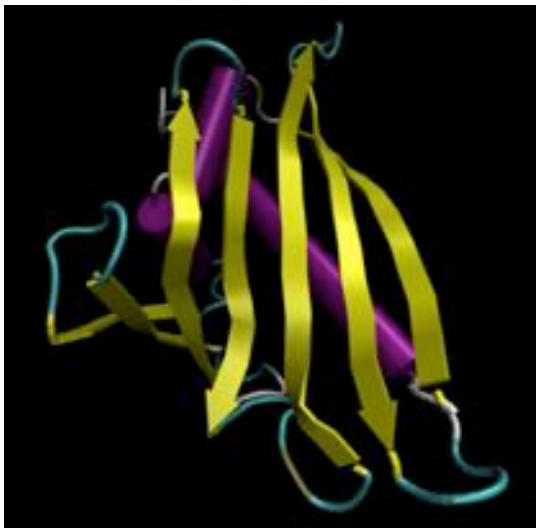
- From vector field on a mesh (needs connectivity)
  - Show the direction an element will travel in at any point in time.

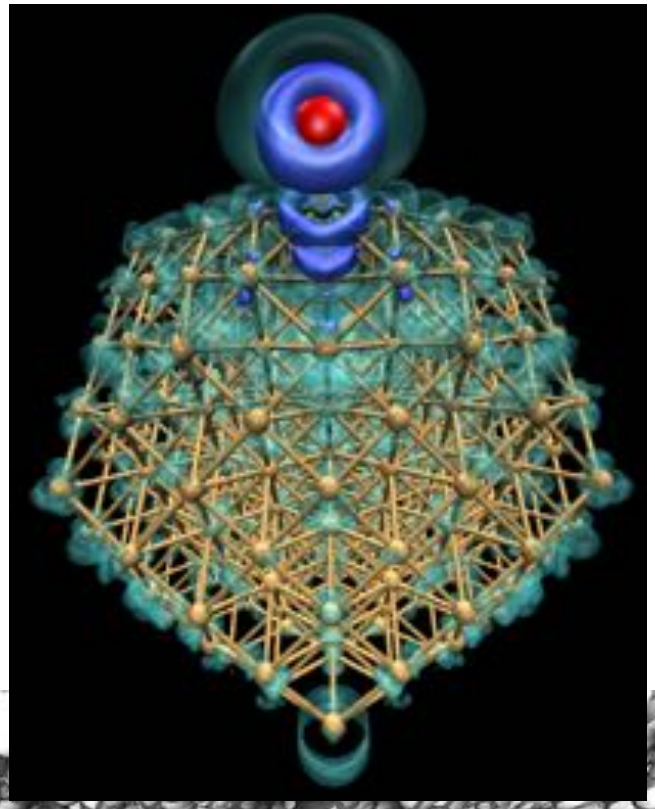


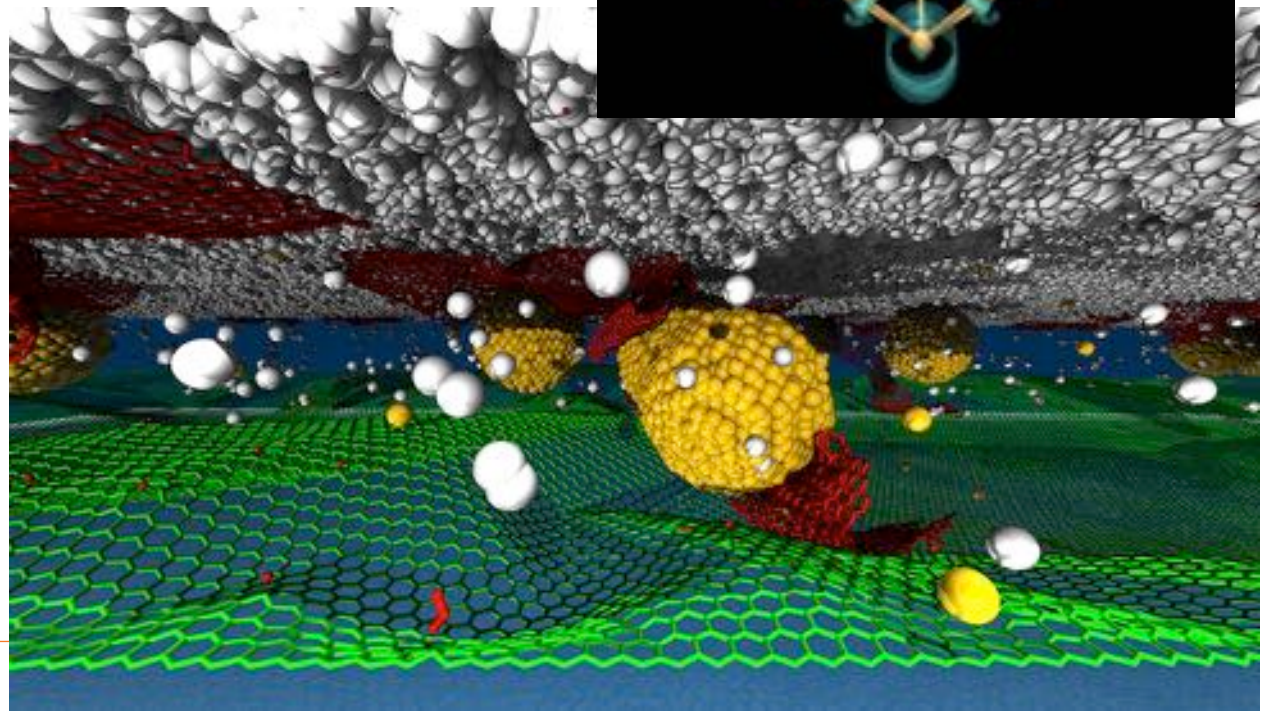


Molecular Dynamics Visualization

- Domain-specific representations
  - Backbone
  - Display attributes by atom type
  - Ball and stick
  - Ribbon representation
  - etc.







All Sorts of Tools (*not all available at ALCF)

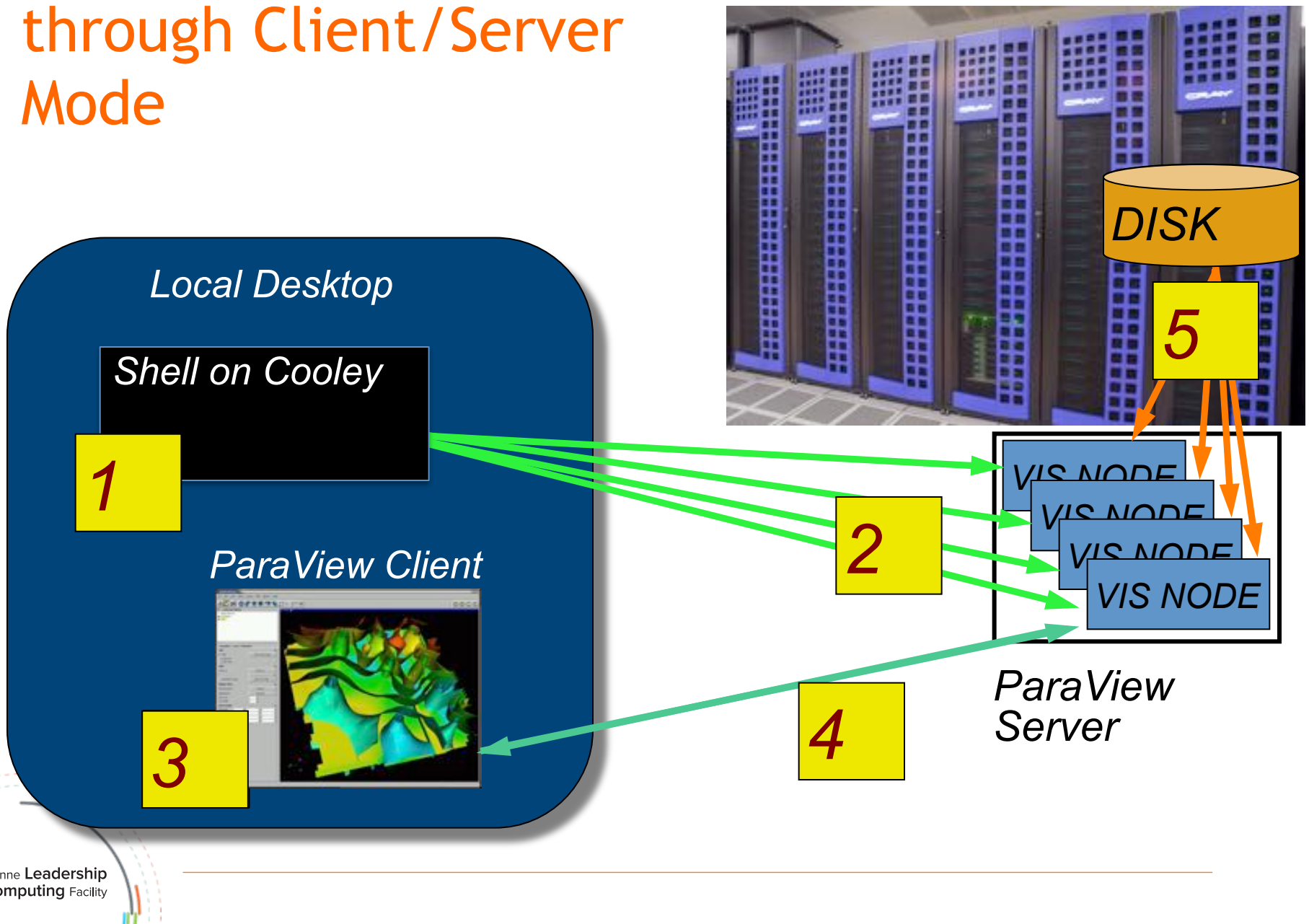
- Visualization Applications
  - VisIt
  - ParaView
  - EnSight
- Domain Specific
  - VMD, Ovito, PyMol, RasMol
- APIs
  - VTK: visualization
  - ITK: segmentat & registration
- GPU performance
  - vl3: shader-based vol rendering
- Analysis Environments
  - Matlab
  - Parallel R
- Utilities
  - GnuPlot
  - ImageMagick
- Visualization Workflow
  - VisTrails

Visualization Modes

- Interactive
  - Data exploration
  - Generating movies
- Batch
  - Known quantities/configurations
  - Generating movies

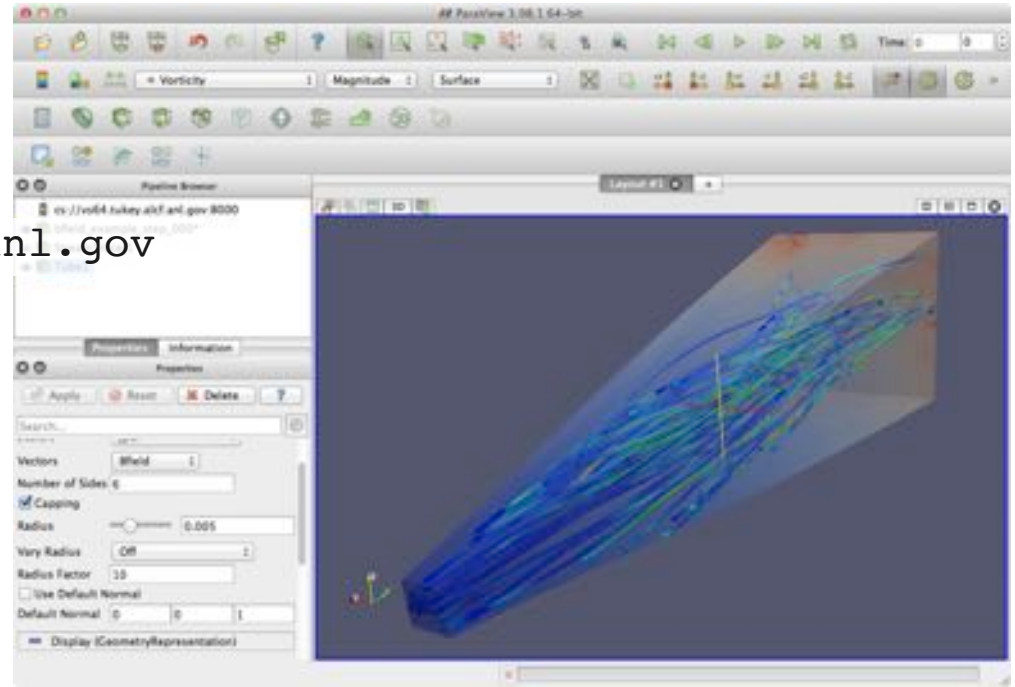
- Stand-alone
  - smaller data – move locally and visualize there
- Client/server mode
  - Use interactive sessions to produce state for visualizing larger data / time series in batch mode.

Remote Visualization through Client/Server Mode

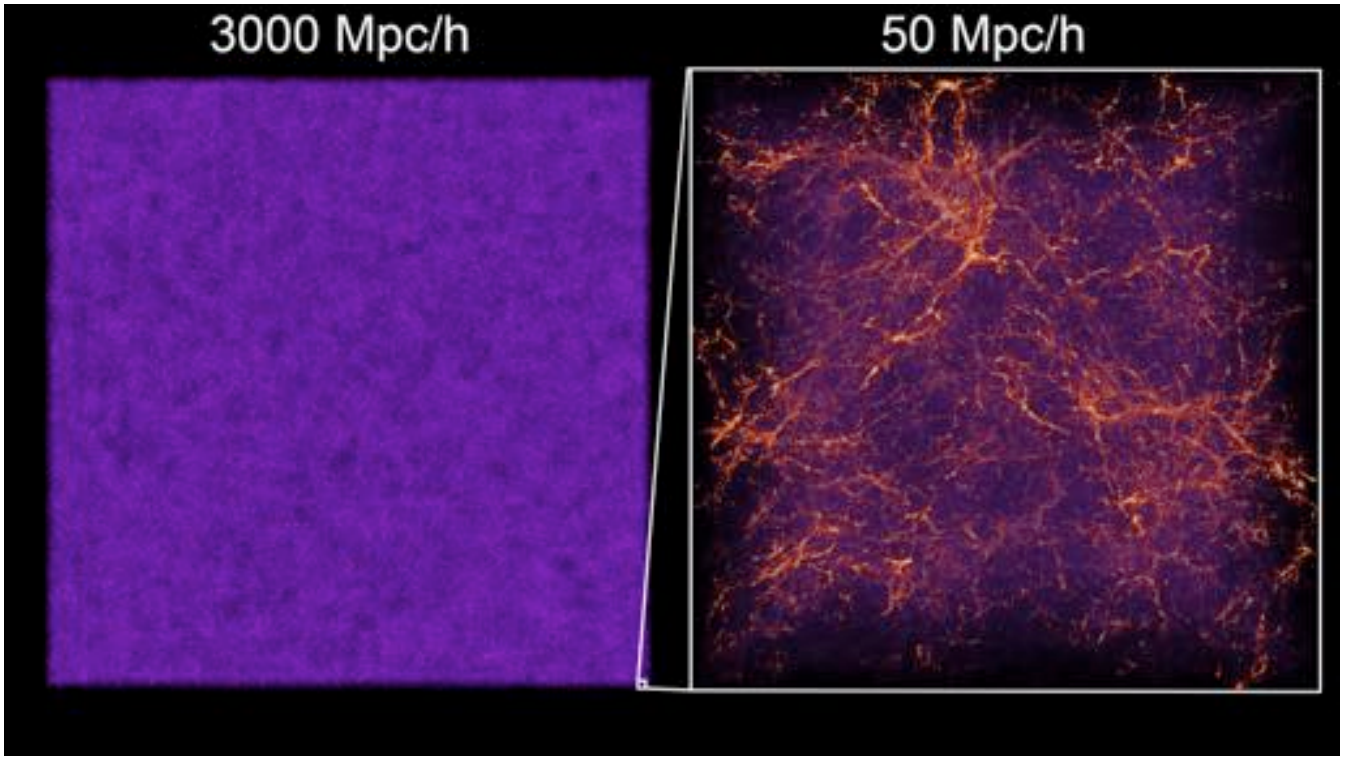


ParaView Example

- Tutorial Page:
  - www.alcf.anl.gov/user-guides/paraview-cooley
- Launch pvserver
  - `qsub -I -n 4 -t 60 -A project_id -q pubnet`
  - `mpiexec -f $COBALT_NODEFILE -np 4 pvserver --server-port=8000`
- ParaView Client
  - Configure server
    - `cc###.cooley.pub.alcf.anl.gov`
  - Connect
  - Do some vis



HACC: Cosmology Simulation



- Data: 10Kx10Kx800 (~316GB)
- Visualization considerations:
  - Number of visualization nodes, processes, GPUs
  - Trade-offs
    - Are you memory, compute, or rendering bound?

Data Logistics

- Storing data
  - /projects/<project_id>
- Backup data
  - HPSS Tape Archive
  - www.alcf.anl.gov/user-guides/using-hpss
- Moving data
  - scp
    - handy for small manageable data
  - Globus (www.globus.org)
    - useful for bigger/ more complex data
    - GridFTP servers
    - Local downloads (Globus Connect Personal)
    - Fire and forget

Data File Formats (ParaView & VisIt, partial list)

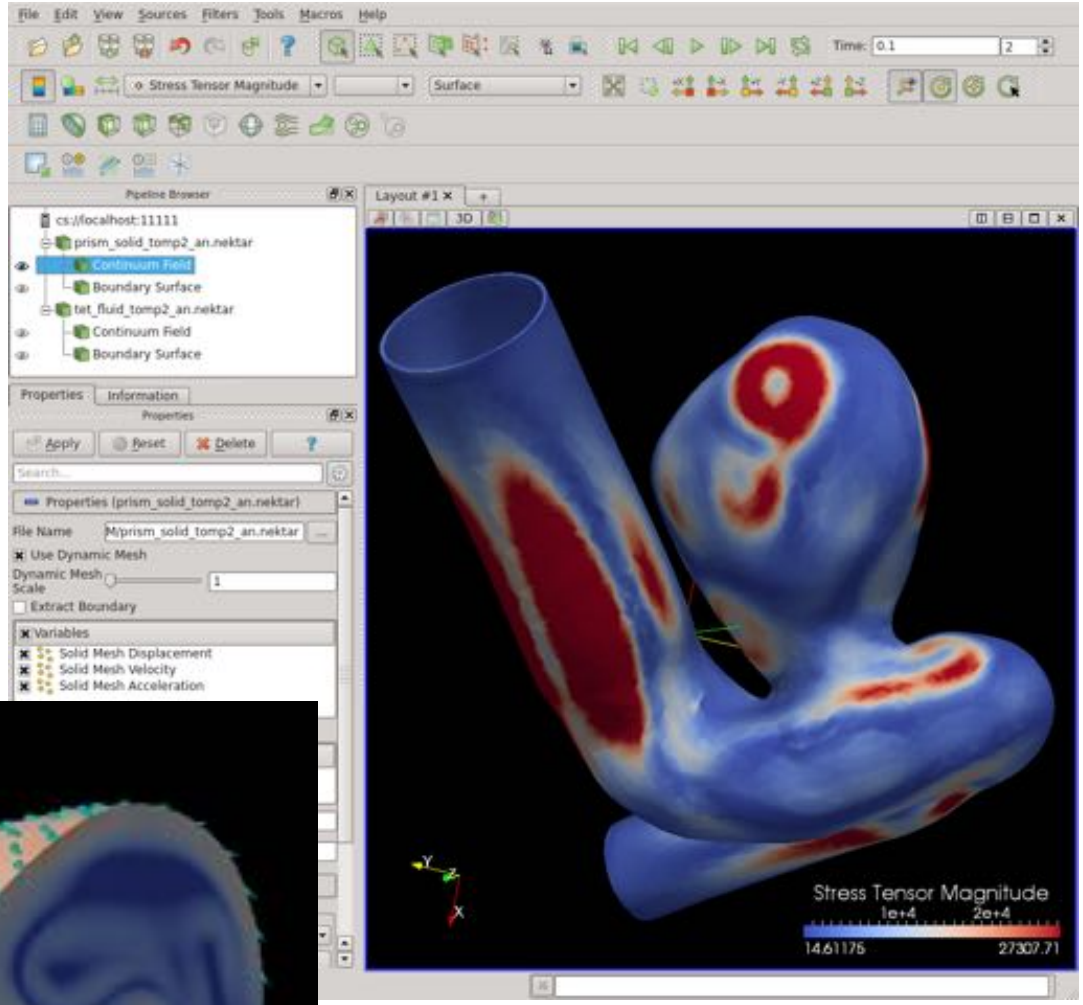
- ParaView Data (.pvd)
- VTK (.vtp, .vtu, .vti, .vts, .vtr)
- VTK Legacy (.vtk)
- VTK Multi Block (.vtm,.vtmb,.vtmg,.vthd,.vthb)
- Partitioned VTK (.pvtu, .pvti, .pvts, .pvtr)
- ADAPT (.nc, .cdf, .elev, .ncd)
- ANALYZE (.img, .hdr)
- ANSYS (.inp)
- AVS UCD (.inp)
- BOV (.bov)
- BYU (.g)
- CAM NetCDF (.nc, .ncdf)
- CCSM MTSD (.nc, .cdf, .elev, .ncd)
- CCSM STSD (.nc, .cdf, .elev, .ncd)
- CEAucd (.ucd, .inp)
- CMAT (.cmat)
- CML (.cml)
- CTRL (.ctrl)
- Chombo (.hdf5, .h5)
- Claw (.claw)
- Comma Separated Values (.csv)
- Cosmology Files (.cosmo, .gadget2)
- Curve2D (.curve, .ultra, .ult, .u)
- DDCMD (.ddcmd)
- Digital Elevation Map (.dem)
- Dyna3D(.dyn)
- EnSight (.case, .sos)
- Enzo boundary and hierarchy
- ExodusII (.g, .e, .exe, .ex2, .ex2v.., etc)
- ExtrudedVol (.exvol)
- FVCOM (MTMD, MTSD, Particle, STSD)
- Facet Polygonal Data
- Flash multiblock files
- Fluent Case Files (.cas)
- GGCM (.3df, .mer)
- GTC (.h5)
- GULP (.trg)
- Gadget (.gadget)
- Gaussian Cube File (.cube)
- JPEG Image (.jpg, .jpeg)
- LAMPPS Dump (.dump)
- LAMPPS Structure Files
- LODI (.nc, .cdf, .elev, .ncd)
- LODI Particle (.nc, .cdf, .elev, .ncd)
- LS-DYNA (.k, .lsdyna, .d3plot, d3plot)
- M3DCl (.h5)
- MFIX Unstructred Grid (.RES)
- MM5 (.mm5)
- MPAS NetCDF (.nc, .ncdf)
- Meta Image (.mhd, .mha)
- Miranda (.mir, .raw)
- Multilevel 3d Plasma (.m3d, .h5)
- NASTRAN (.nas, .f06)
- Nek5000 Files
- Nrrd Raw Image (.nrrd, .nhdr)
- OpenFOAM Files (.foam)
- PATRAN (.neu)
- PFLOTRAN (.h5)
- PLOT2D (.p2d)
- PLOT3D (.xyz, .q, .x, .vp3d)
- PLY Polygonal File Format
- PNG Image Files
- POP Ocean Files
- ParaDIS Files
- Phasta Files (.pht)
- Pixie Files (.h5)
- ProSTAR (.cel, .vrt)
- Protein Data Bank (.pdb, .ent, .pdb)
- Raw Image Files
- Raw NRRD image files (.nrrd)
- SAMRAI (.samrai)
- SAR (.SAR, .sar)
- SAS (.sasgeom, .sas, .sasdata)
- SESAME Tables
- SLAC netCDF mesh and mode data
- SLAC netCDF particle data
- Silo (.silo, .pdb)
- Spheral (.spheral, .sv)
- SpyPlot CTH
- SpyPlot (.case)
- SpyPlot History (.hscth)
- Stereo Lithography (.stl)
- TFT Files
- TIFF Image Files
- TSurf Files
- Tecplot ASCII (.tec, .tp)
- Tecplot Binary (.plt)
- Tetrad (.hdf5, .h5)
- UNIC (.h5)
- VASP CHGCA (.CHG)
- VASP OUT (.OUT)
- VASP POSTCAR (.POS)
- VPIC (.vpc)
- VRML (.wrl)
- Velodyne (.vld, .rst)
- VizSchema (.h5, .vsh5)
- Wavefront Polygonal Data (.obj)
- WindBlade (.wind)
- XDMF and hdf5 (.xmf, .xdmf)
- XMol Molecule

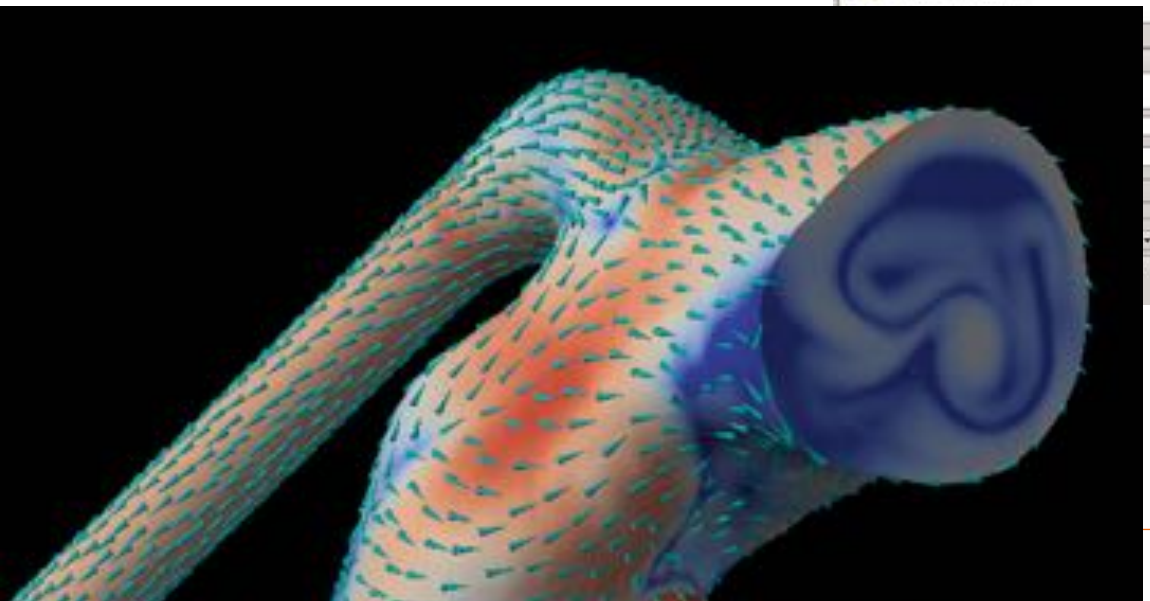
Data Wrangling

- XDMF
  - XML wrapper around HDF5/binary data
  - Can define
    - data sets, subsets, hyperslabs
- vtk
  - Could add to your simulation code
  - Can write small utilities to convert data
    - Use your own read routines
    - Write vtk data structures
  - C++ and Python bindings
- Silo
  - Could add to your simulation code
  - Can write small utilities to convert data
    - Use your own read routines
    - Write Silo data structures
  - C and Fortran bindings

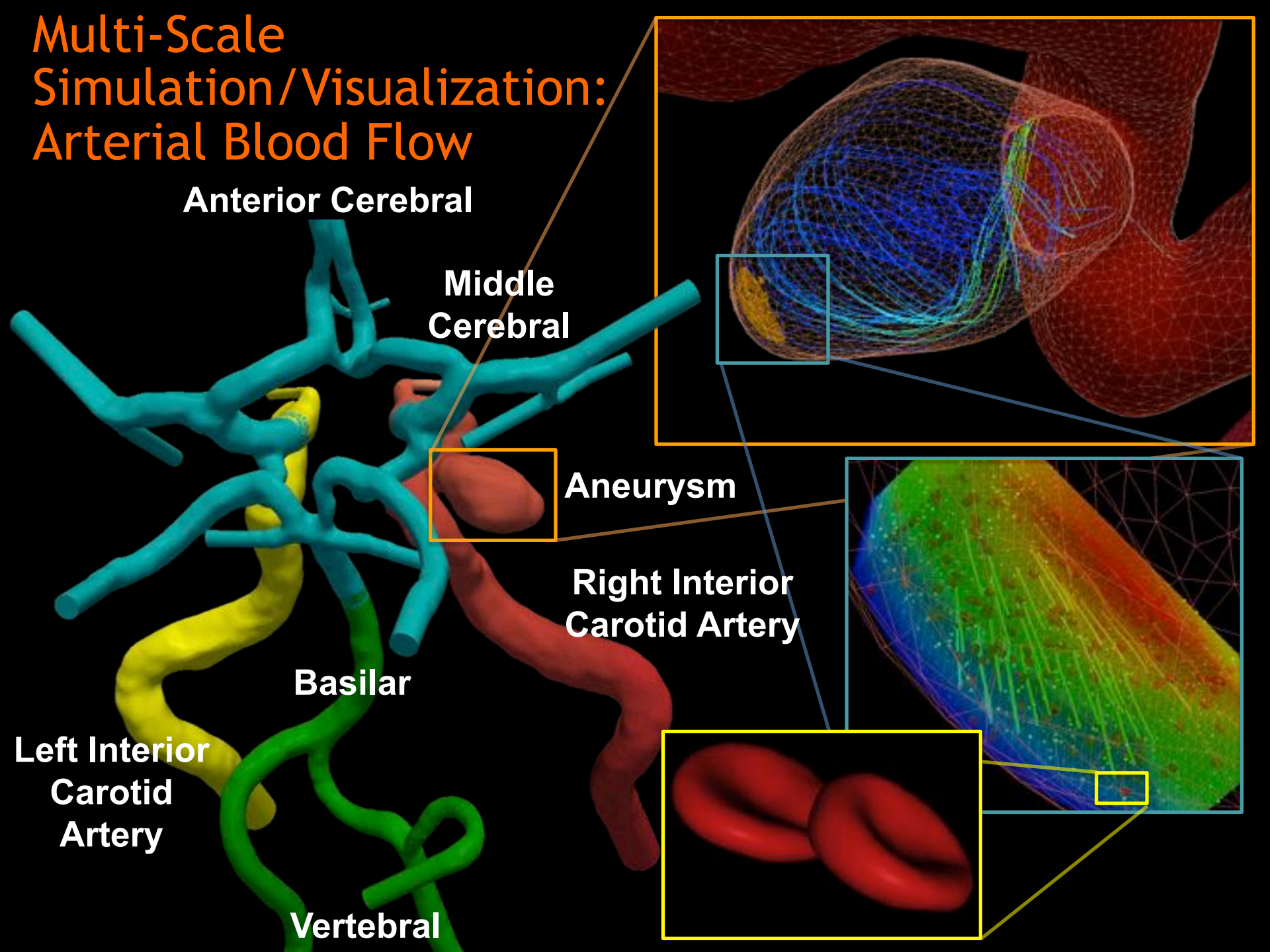
Data Organization

- Format
  - Existing tools support many flavors
  - Use one of these formats
  - Write a custom reader for existing tool
    - NekTar example
  - Write your own custom vis tool





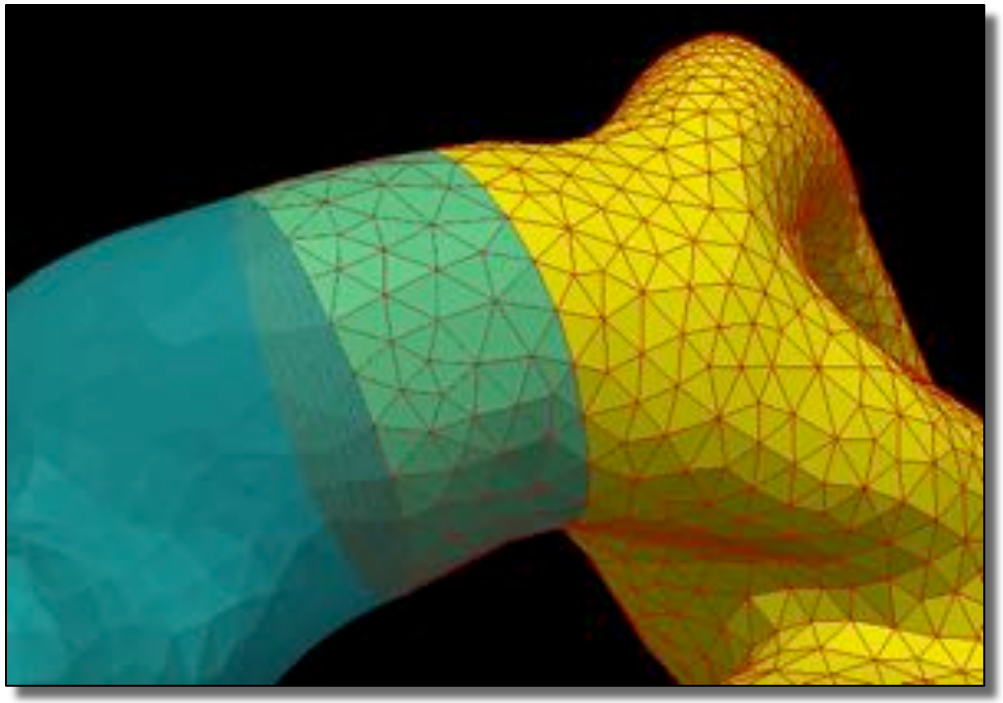
Multi-Scale Simulation/Visualization: Arterial Blood Flow



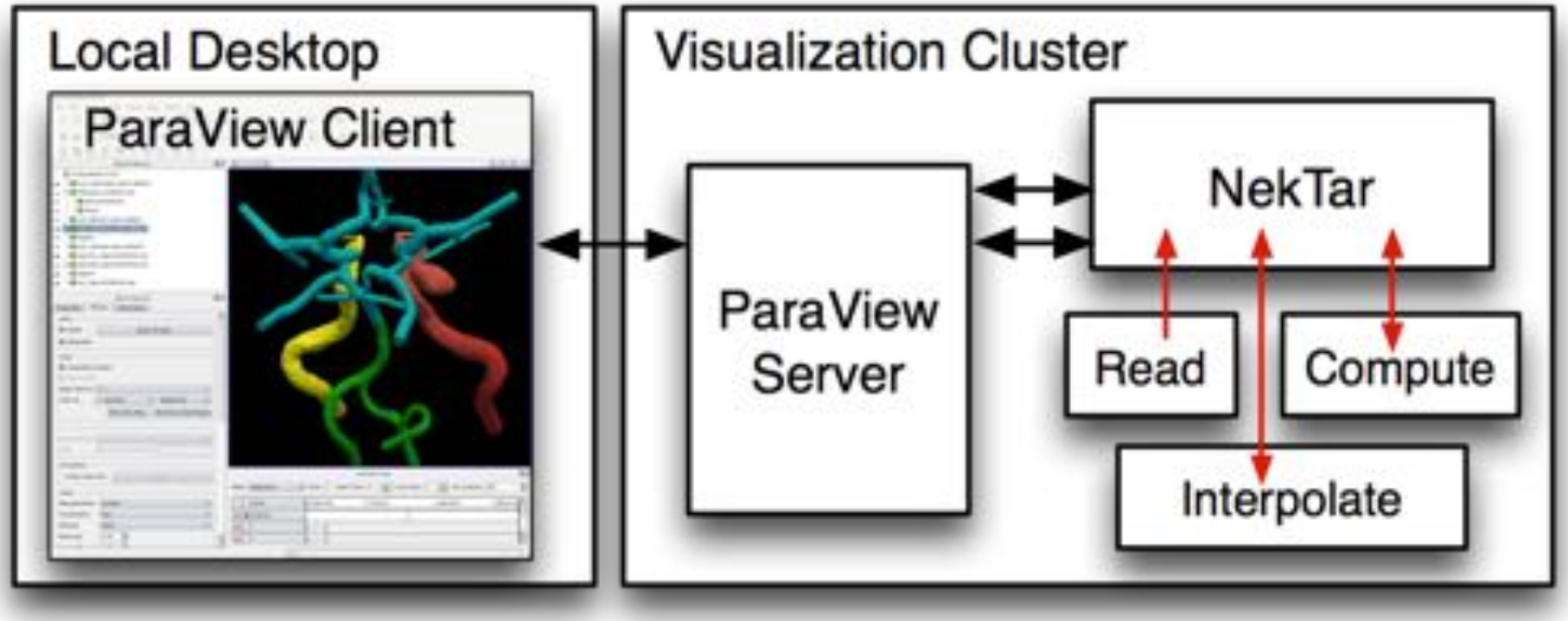
Macroscale Simulation (NekTar)

- NekTar: Spectral/hp element method (SEM)
  - Non-overlapping elements
  - Multi-level approach
    - Domain decomposed into overlapping patches

- NekTar Data
  - Saved in Modal space
  - Mesh (geometry)
  - Solution data



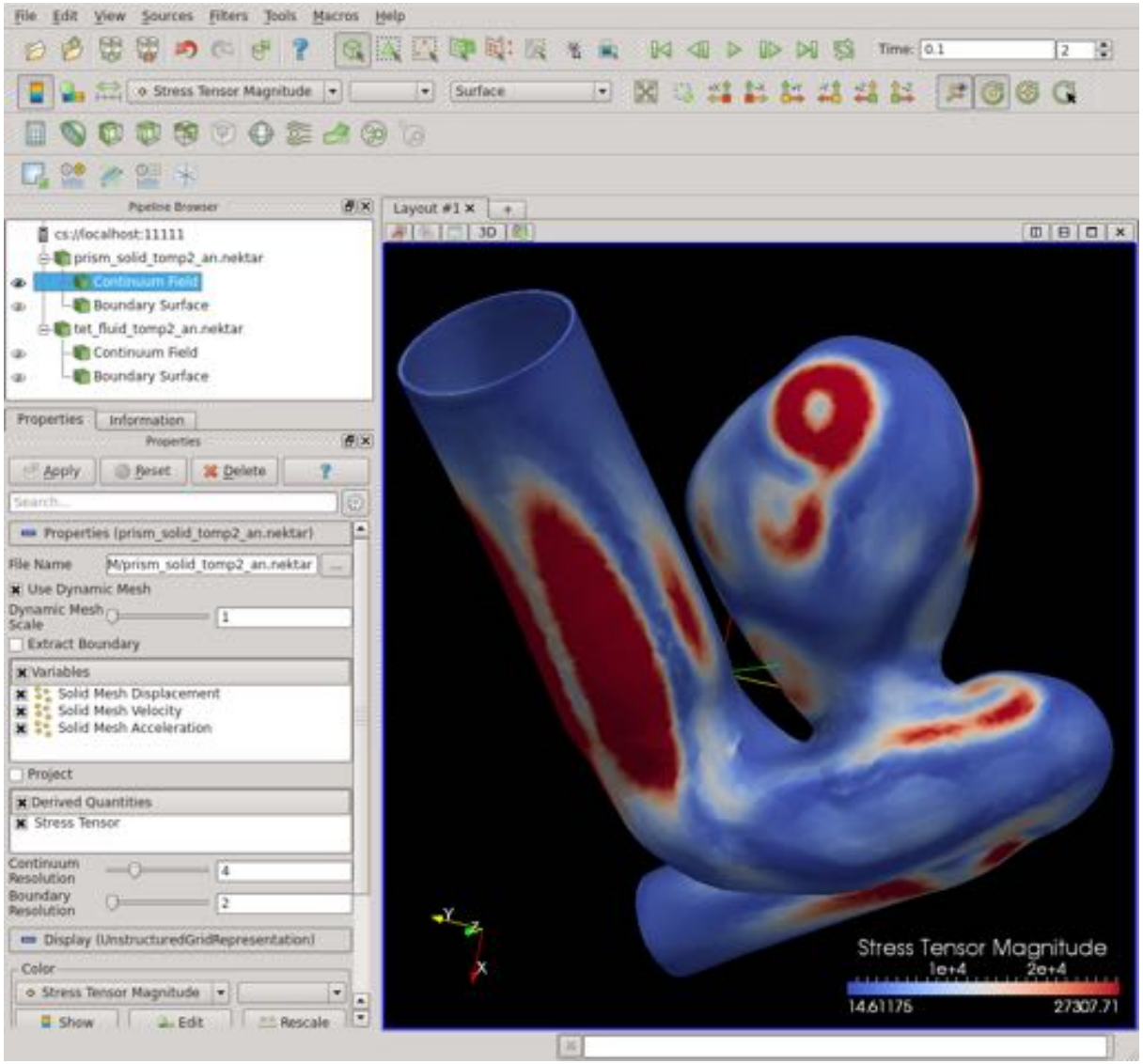
NekTar-ParaView Coupling



- NekTar for parallel I/O and computation
- ParaView for parallel visualization and rendering
- Implement methods:
  - RequestInformation()
  - RequestData()

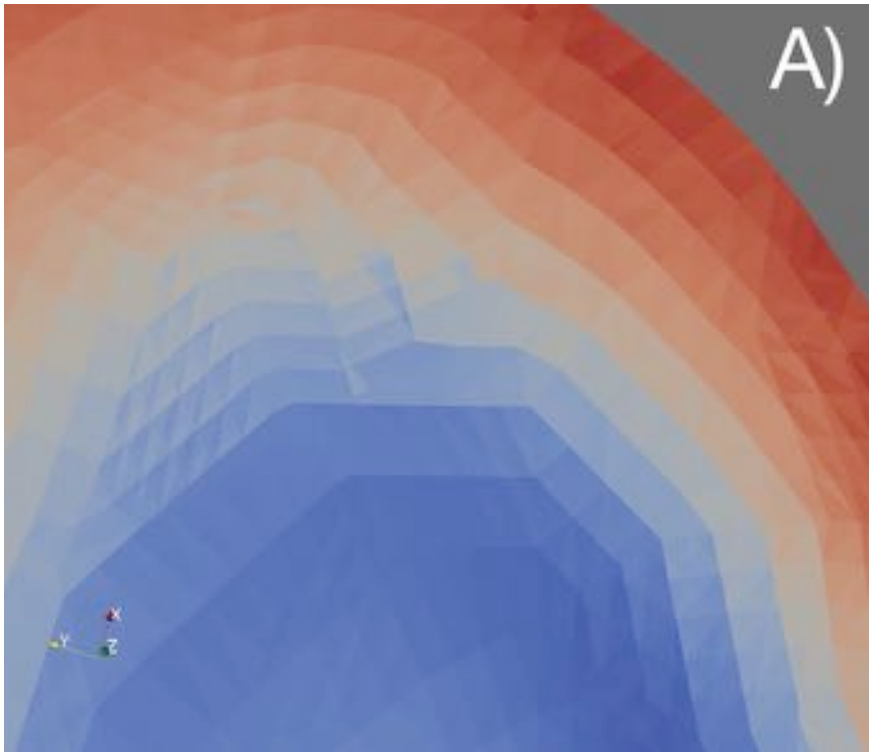
Plug-in Controls

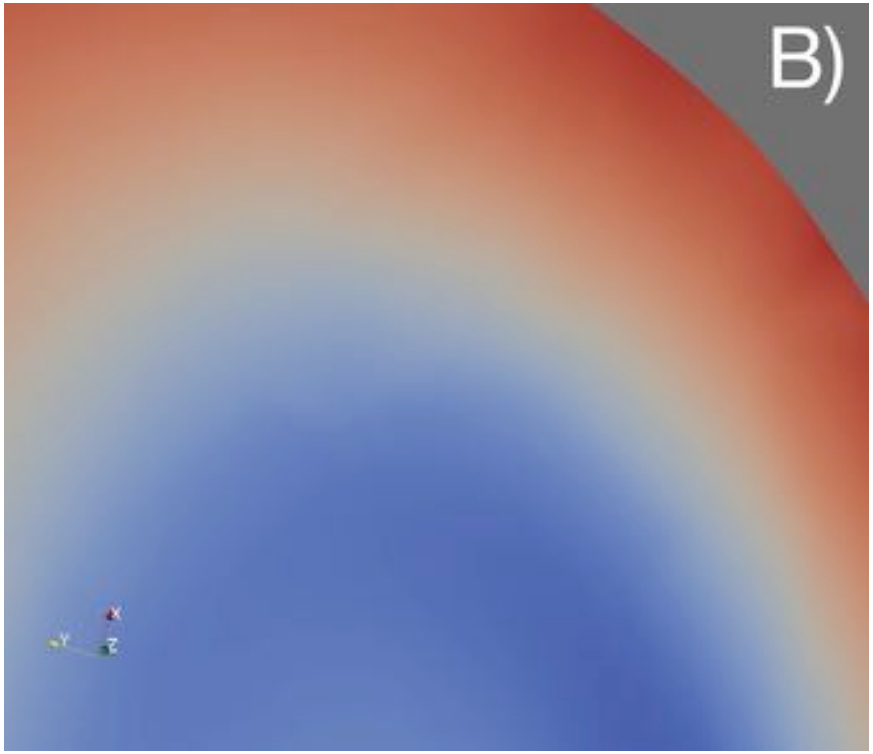
- Select variables
- Interactively set data resolution
  - No need to re-read data from disk
- Time varying data
  - Only new data read from disk, not geometry



Derived Quantities: Vorticity

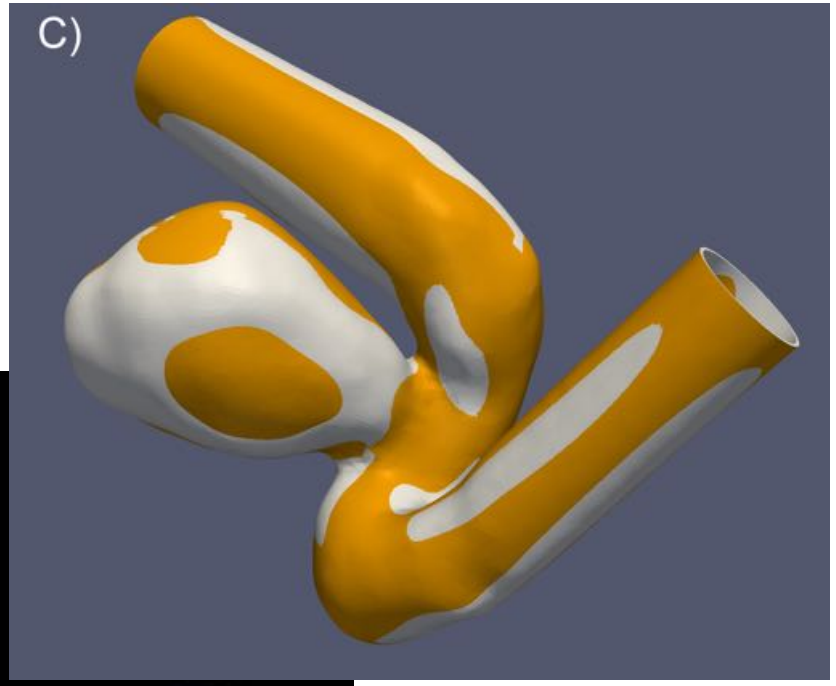
- Data computed with high-order spectral accuracy
  - Grid consistent with simulation resolution

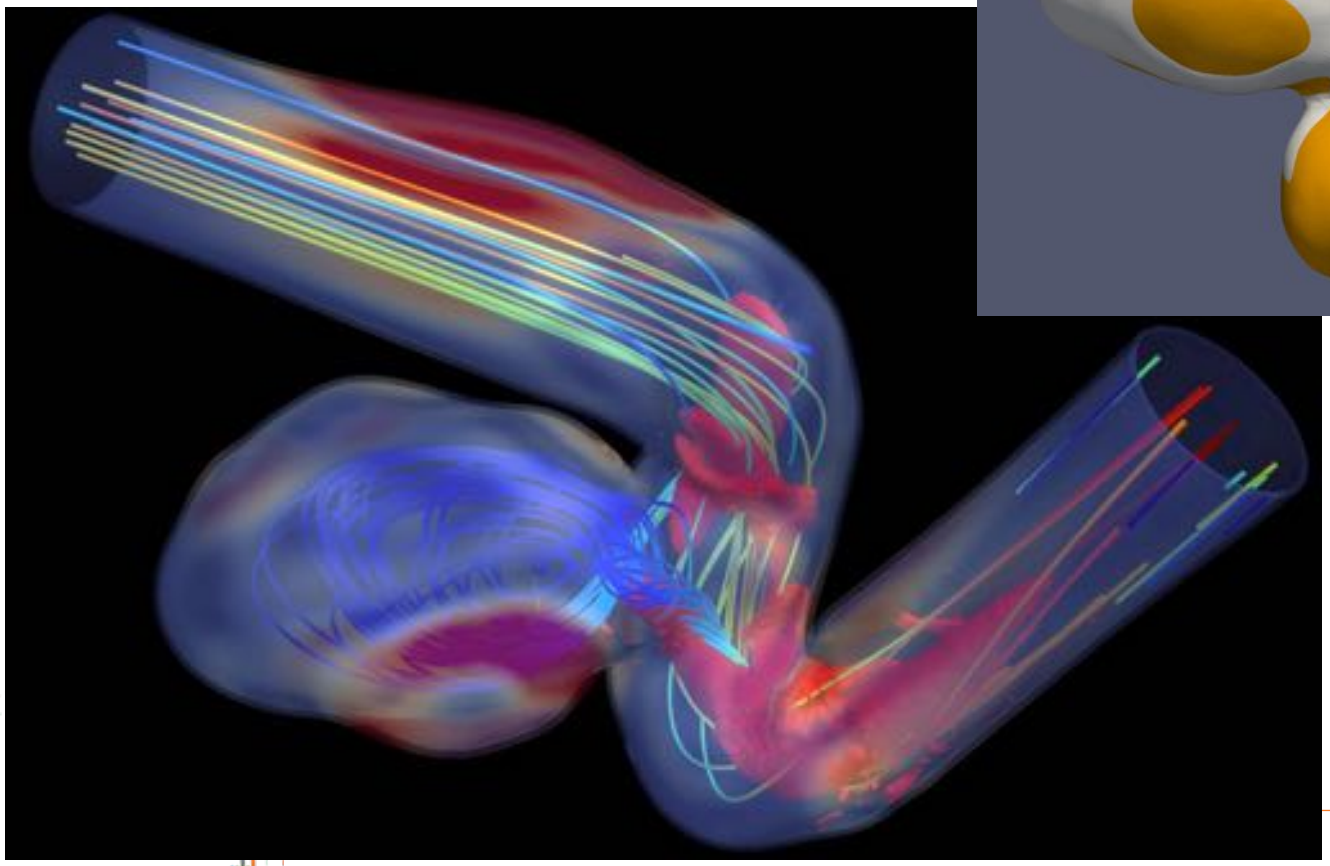




Fluid-Structure Interaction

- Dynamic mesh
- Stress Tensor





Data Organization

- Serial vs. Parallel/Partitioned
  - Single big file vs. many small files: middle ground generally best
    - vtk data types
    - XDMF for ParaView and VisIt

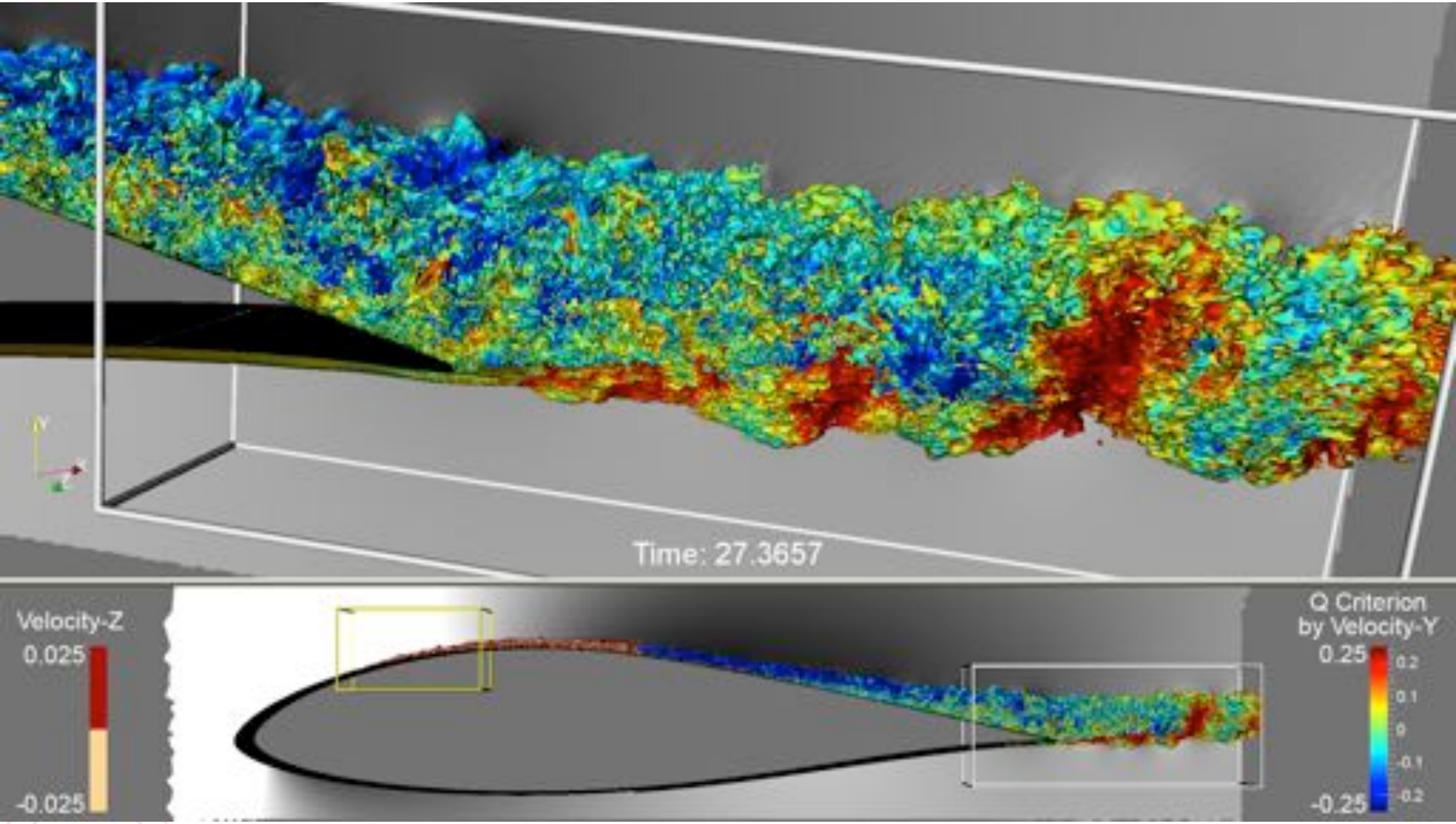
- Serial vs. Parallel/Partitioned
  - Performance trade-offs
    - vtk/paraview: Single serial file: all data read on head node, partitioned and distributed
    - vtk/paraview: Parallel files: made up of smaller serial files, partitioned across processes, read in parallel

Data Organization

Performance example:

- Single serial .vtu file (unstructured grid)
  - Data size: ~3.8GB
  - Read time on 64 processes: > 15 minutes
    - most of this was spent partitioning and distributing
- Partitioned .pvtu file (unstructured grid)
  - Data size: ~8.7GB (64 partitions)
  - Read time on 64 processes: < 1 second

Data Organization



ParaView States and Scripting

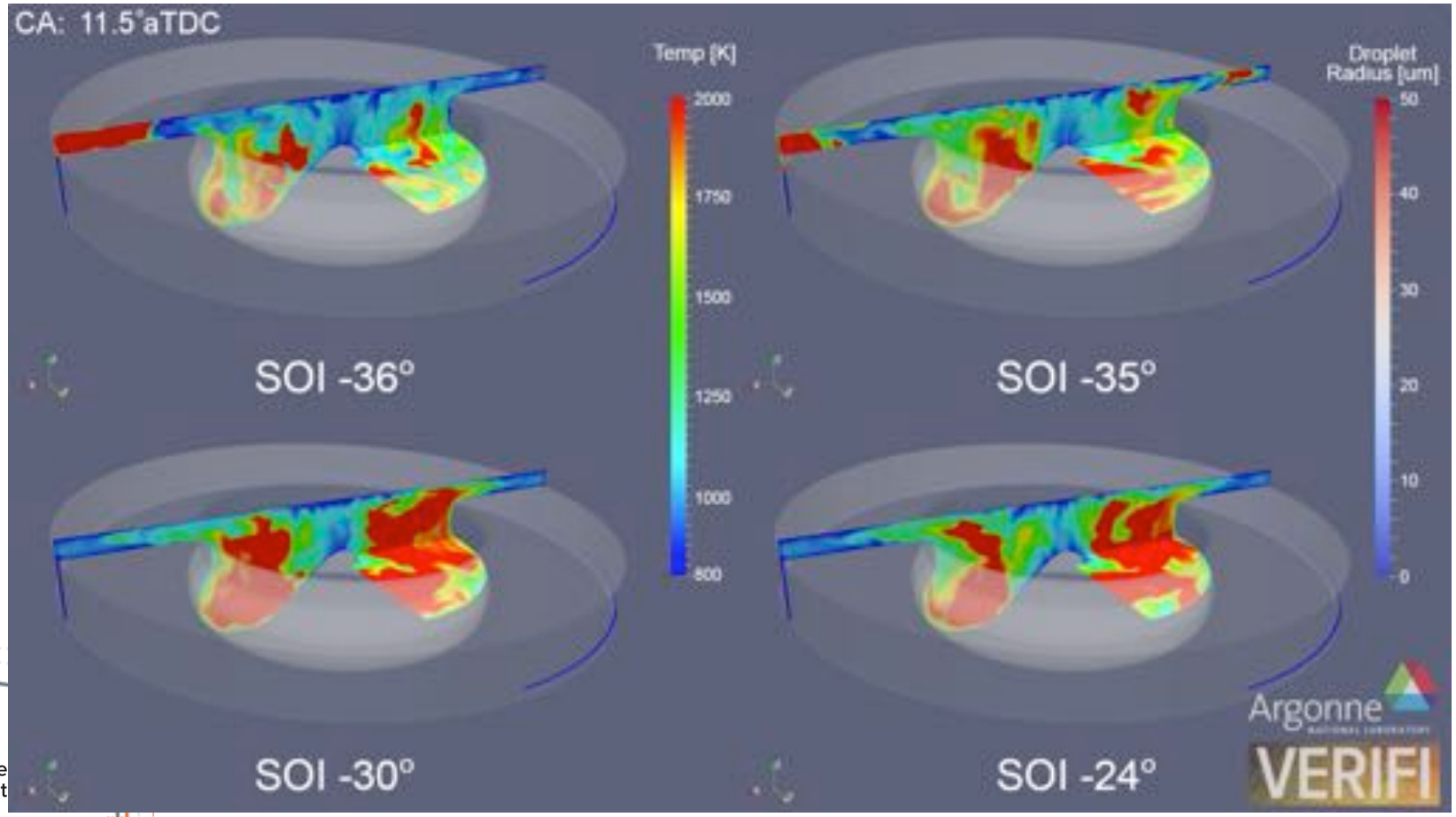
- Choose: File →Save State…
  - .pvsm (for restoring state in interactive mode)
  - saved on the client side

- Choose: File →Save State…
  - .py (for use with pvbatch)
  - saved on the client side

- Edit .py script
  - short example, loop over time steps, saving images

VERIFI: Multi-case Temperature

- 4 cases (SOI -36, -35, -30, -24)
- Each case:
  - 330 time steps
  - VTK files: ~1.1TB



ParaView States and Scripting

```
IMAGE_DIR="/projects/my_project/FRAMES/”
IN_DATA_BASE=”/projects/my_project/DATA/soi-24-"
STEP_START=1
STEP_COUNT=330
STEP_INC=1

start_frame=int(sys.argv[1])
num_frames=int(sys.argv[2])

DATA_FILES = []

for i in range(STEP_START, STEP_START+(STEP_COUNT*STEP_INC),
STEP_INC):
   TEMP_FILE=”%s%06d%s" % (IN_DATA_BASE, i, “.vtu”)
   DATA_FILES.append(TEMP_FILE)
```

ParaView States and Scripting

```
…
try: paraview.simple
except: from paraview.simple import *
paraview.simple._DisableFirstRenderCameraReset()
RenderView1 = CreateRenderView()
RenderView1.ViewSize = [1920, 1080]
…
soiData = XMLUnstructuredGridReader( guiName="soi-Data*",
CellArrayStatus=['temp', 'equiv_ratio', 'rank'], FileName=DATA_FILES)
```

ParaView States and Scripting

```
...
#Render()

time_vals = soiData.TimestepValues

for i in range(start_frame, start_frame+num_frames):

    RenderView1.ViewTime = time_vals[i]
    RenderView1.StillRender()

    IMAGE_FILE="%s/frame_%04d.png" % (IMAGE_DIR, i)
    print "saving: " + IMAGE_FILE
    WriteImage(IMAGE_FILE)
```

ParaView Batch Mode

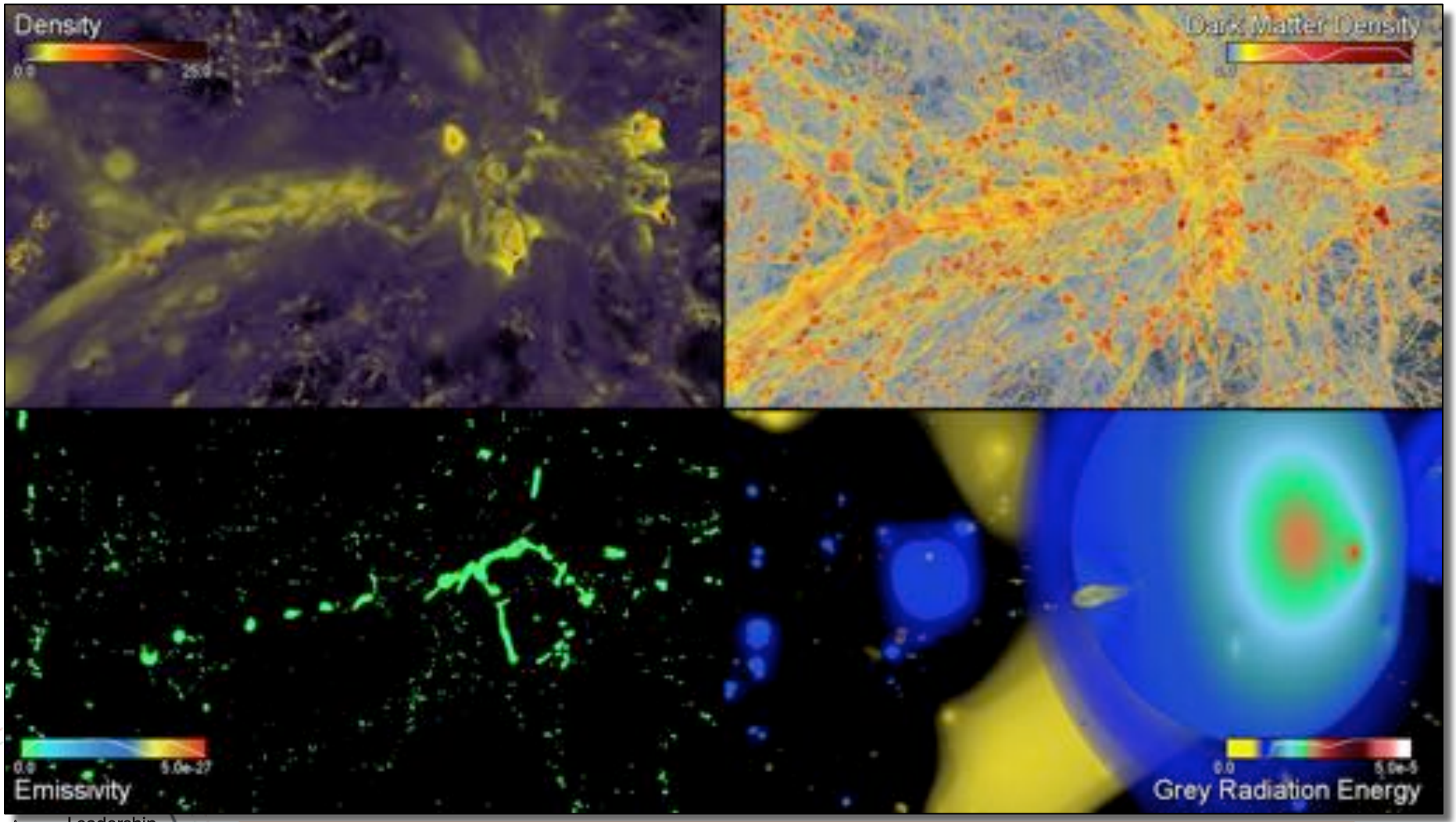
- Copy .py script
  - need to copy the .py script to Cooley
- pvbatch
  - `mpiexec -f $COBALT_NODEFILE -np 4 pvbatch <script.py> <start_frame> <num_frames>`
  - could run in qsub -I mode, or totally batch

Movie Making

- VisIt/ParaView: save animation
  - can save animation file
  - can save individual images
    - I tend to do this, allows for more flexibility
- Adobe Premiere, Final Cut, etc.
  - commercial tools enable lots of options, but do cost money (ALCF does not provide these tools)
- ImageMagick tools for annotating and combining images in different ways

Annotation, compositing, scaling…

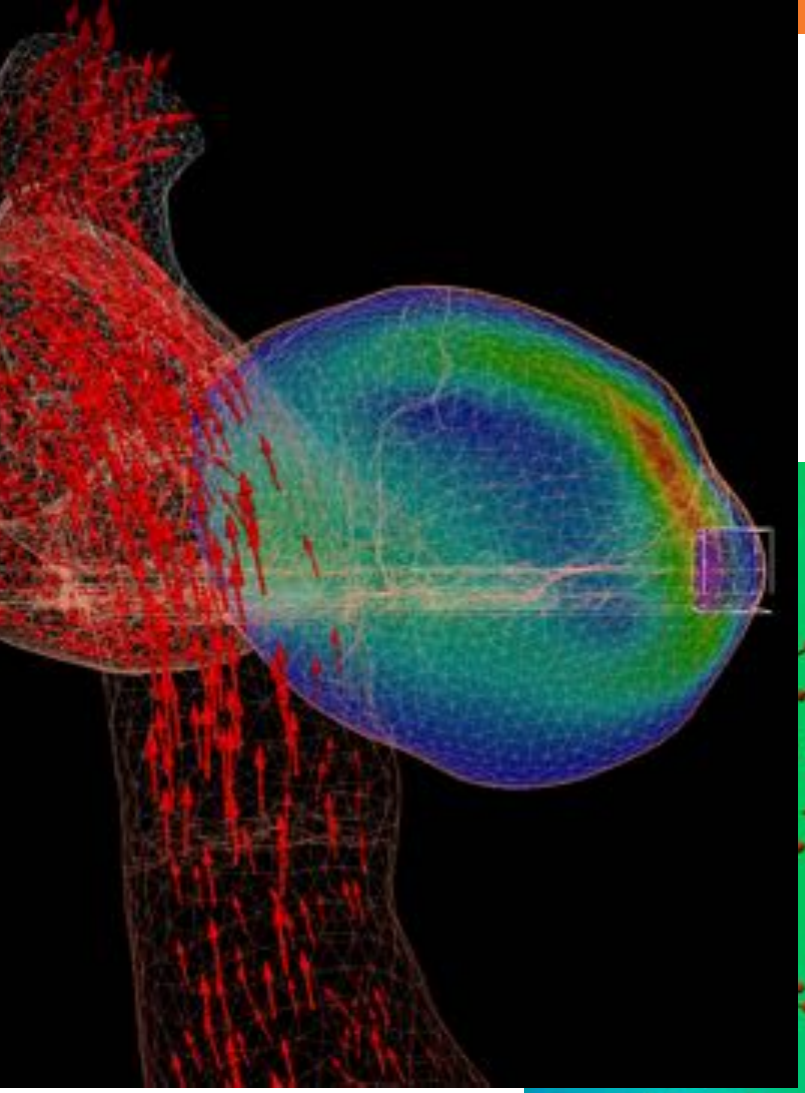
- ImageMagick
  - convert, composite, montage, etc.

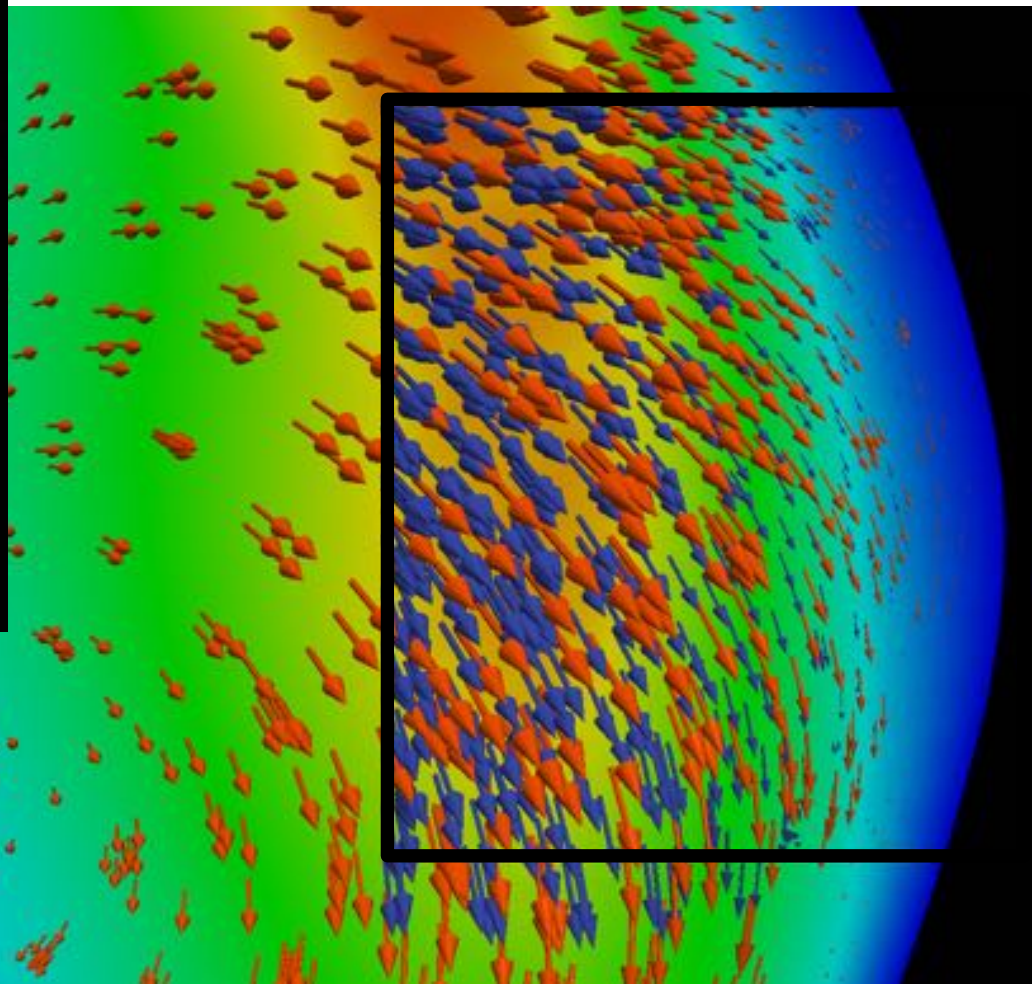


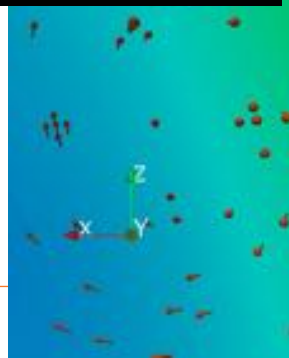
Movie Creation

- ffmpeg: Movie encoding
  - softenv key: +ffmpeg-1.0.1
  - ffmpeg –sameq –i frame.%04d.png movie.mp4
- Combine multiple segments of frames
  - Create a directory of symbolic links to all frames in order

Visualization for Verification

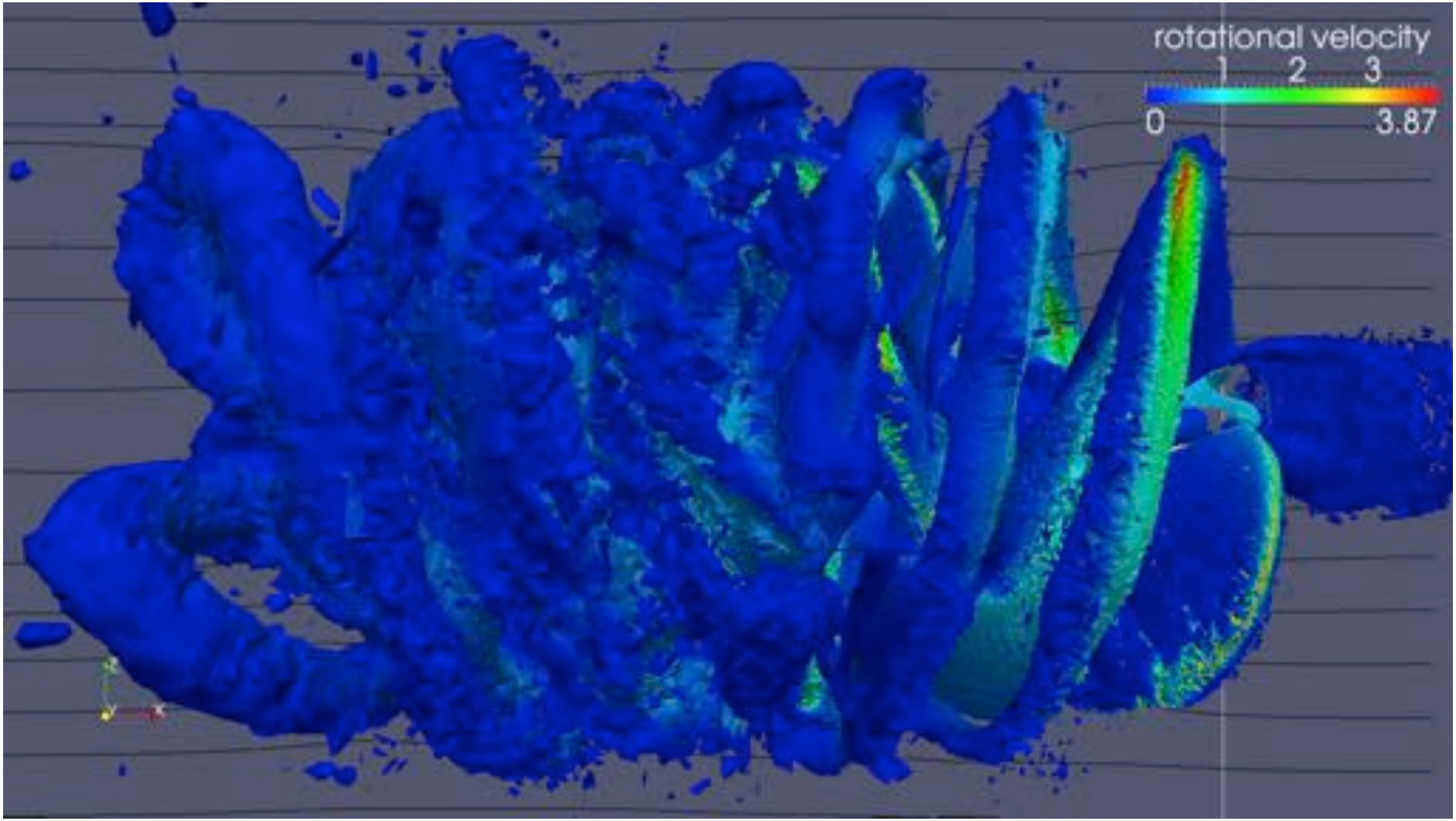




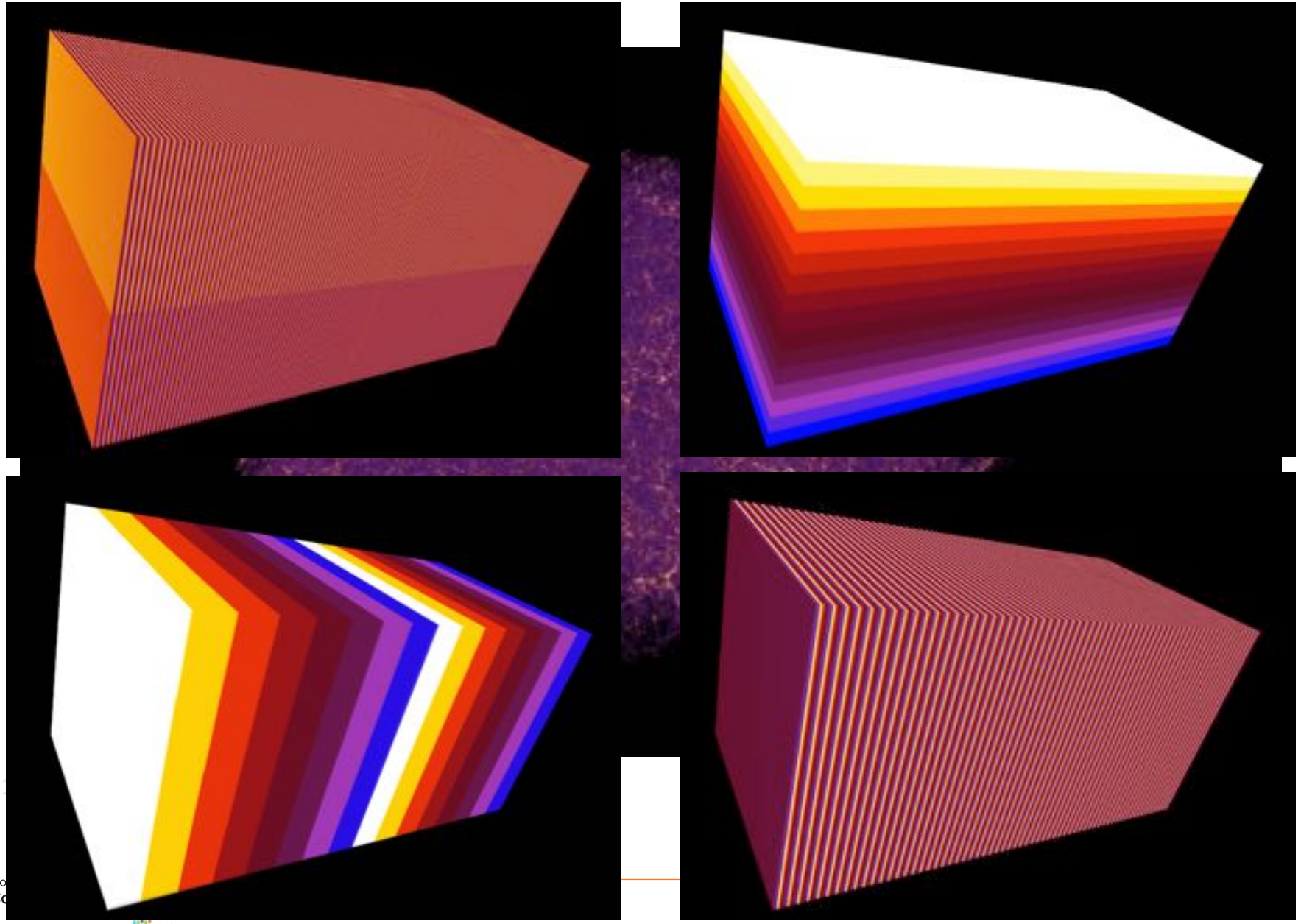


Visualization for Debugging

Visualization for Debugging



Visualization as Diagnostics: Color by Thread ID



More info…

- Cooley user guide:
  - www.alcf.anl.gov/user-guides/cooley
- Online ParaView tutorial:
  - www.alcf.anl.gov/user-guides/vis-paraview-red-blood-cell-tutorial