Introduction to Visualizing Large Data with ParaView

Cory Quammen, Kitware, Inc.

kitware

About me

Cory Quammen

cory.quammen@kitware.com

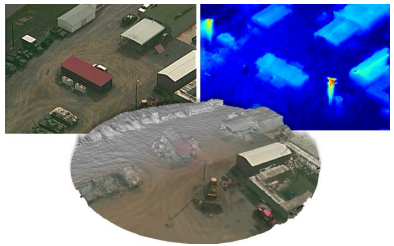
Technical Leader at Kitware

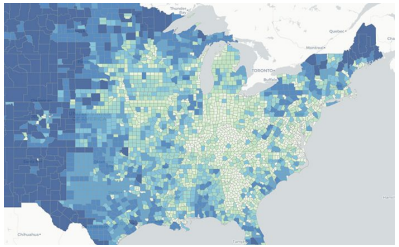
Using and developing ParaView since 2006

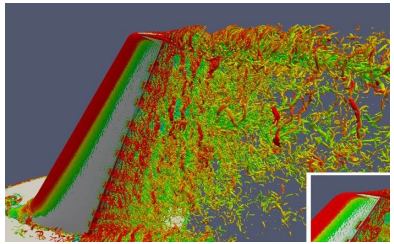
Software lead for ParaView

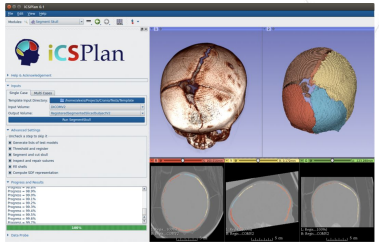
kitware

Kitware areas of expertise / Built on open source









Computer Vision

Data and Analytics

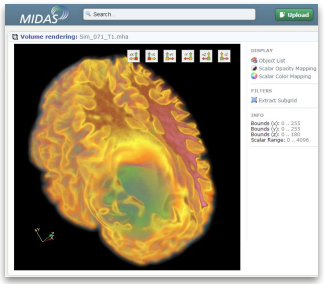
Scientific Computing

Medical Computing

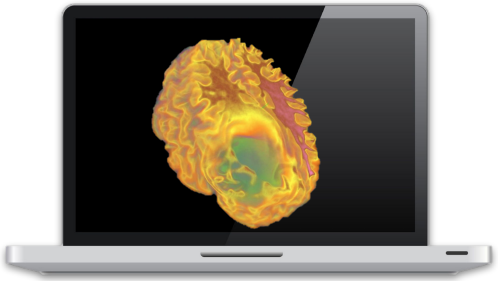
Software Solutions

kitware

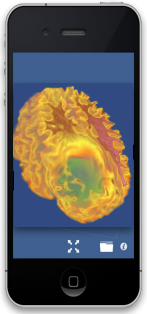
Kitware / Universal Platforms



Web



Desktop



Mobile

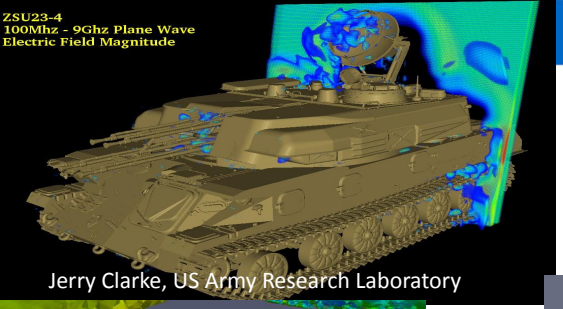
Cloud /HPC

kitware
Platforms

3D Slicer
ParaView
KWIVER
mstk
VTK
Pulse Physiology Engine
CMake
Resonant
tomviz
ITK

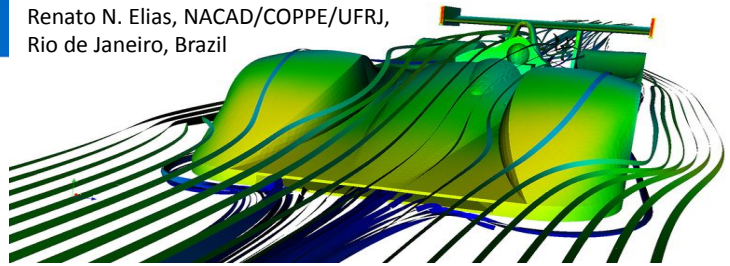
What is ParaView?

kitware

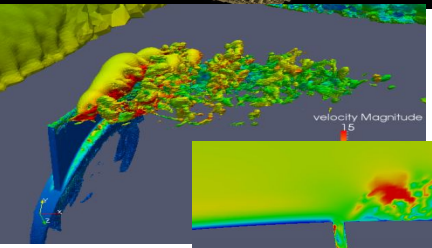


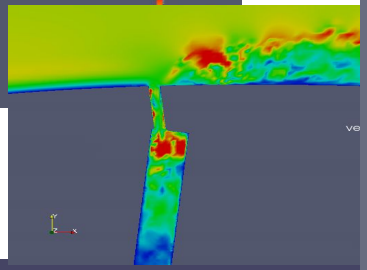
Jerry Clarke, US Army Research Laboratory

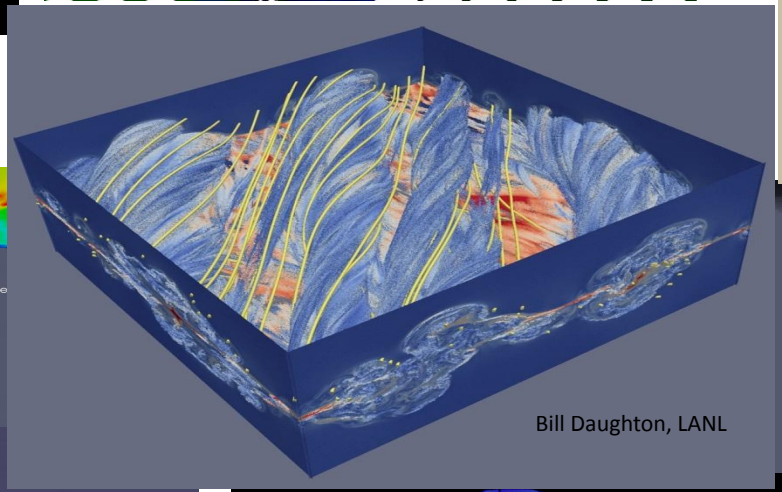
Renato N. Elias, NACAD/COPPE/UFRJ, Rio de Janeiro, Brazil



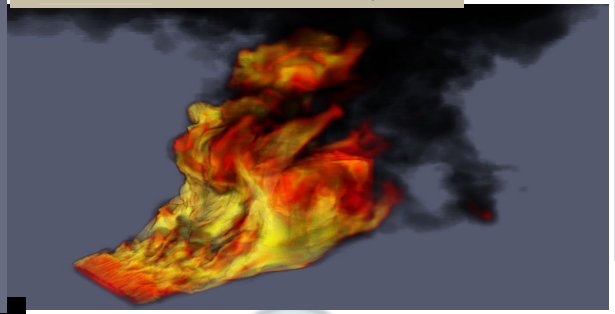
Swiss National Supercomputing Centre

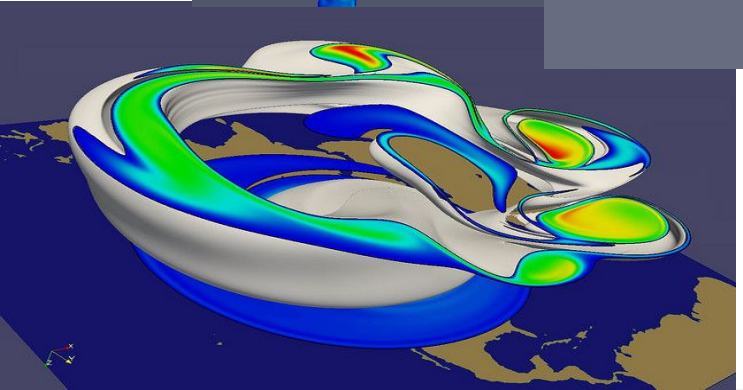


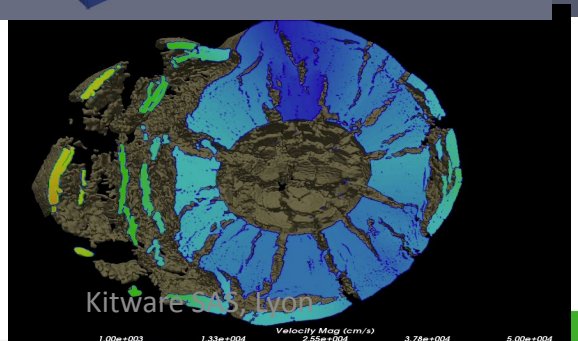




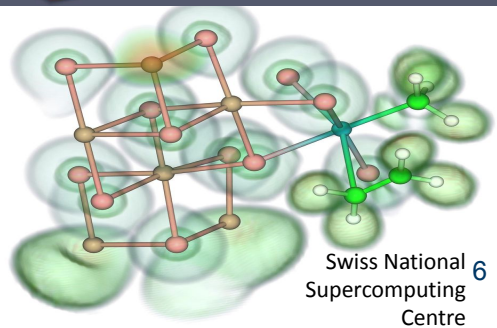
Bill Daughton, LANL

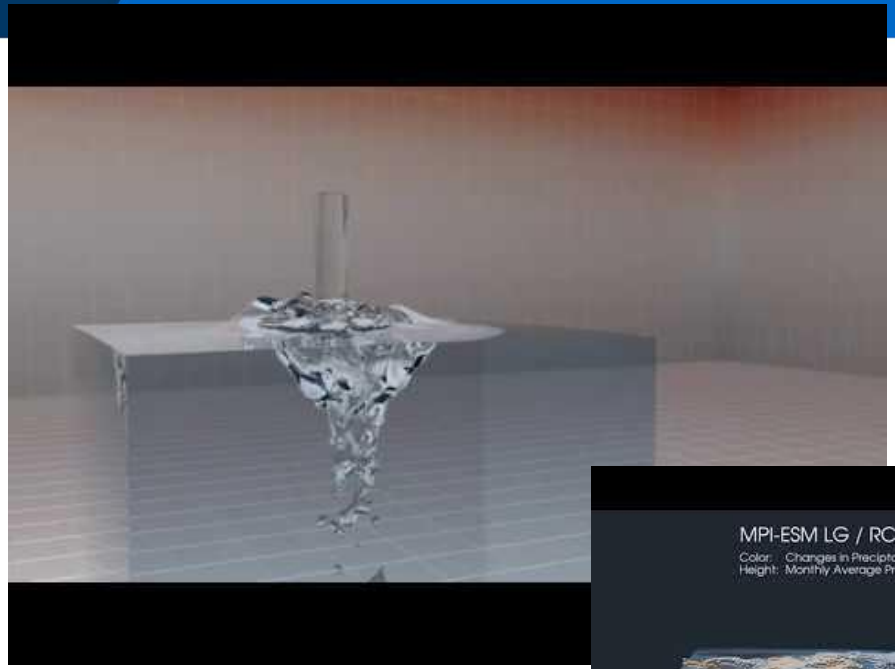


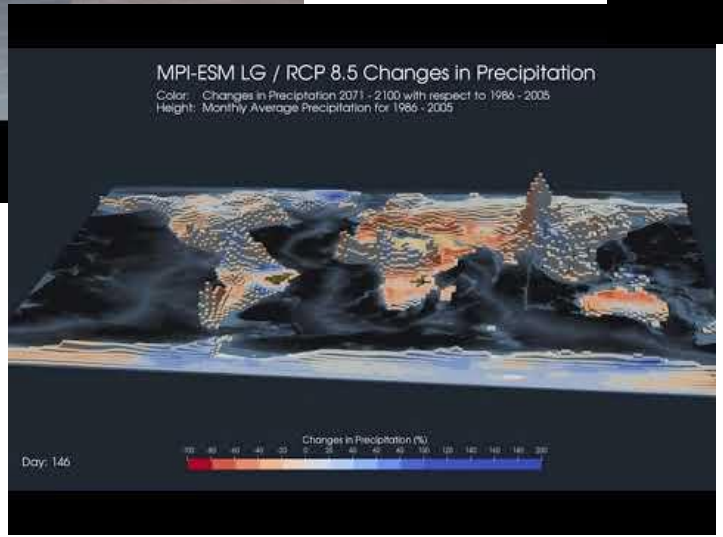




Kitware SAS, Lyon







https://vimeo.com/304370557

What is ParaView?

An **application** and architecture for display and analysis of massive scientific datasets.

- End User Visualization Tool
  - User does not have to be a programmer*
  - Wraps many of the important VTK filters into an Application**
  - Extensive searchable help
  - Undo/Redo feature
  - Selection tools
- Animation panel
- Configurable layout

* Can even be a rocket scientist

** Those not wrapped are easy to add

Current ParaView Usage

- Used by **academic**, **government**, and **commercial** institutions worldwide
- Downloaded ~100K times per year

Community	Members	Topics per month
VTK Discourse forum	1300	90
ParaView Discourse forum	2000	100

Current Funding

- US Dept. of Defense
- US Dept. of Energy
- EDF (France)
- Undisclosed commercial customers

- Other contributors
  - Swiss National Supercomputing Centre
  - DOE SLAC
  - Ohio State
  - Mississippi State
  - RPI
  - CEA

Sandia National Laboratories

kitware

Very active development

- Now on 6 month release schedule
- https://blog.kitware.com/
- 145,090 commits
  - 5000 commits from Jan 2019 - Jan 2020
- 260 contributors (~20/month)
- Code hosted on GitLab – a lot easier to include user's contribution
- Issue Tracker on Gitlab, for easier bugs/code links and tracking

Codebase (LOC)	1,646,346
Effort (est.)	466 Person Years

https://www.openhub.net/p/ParaView

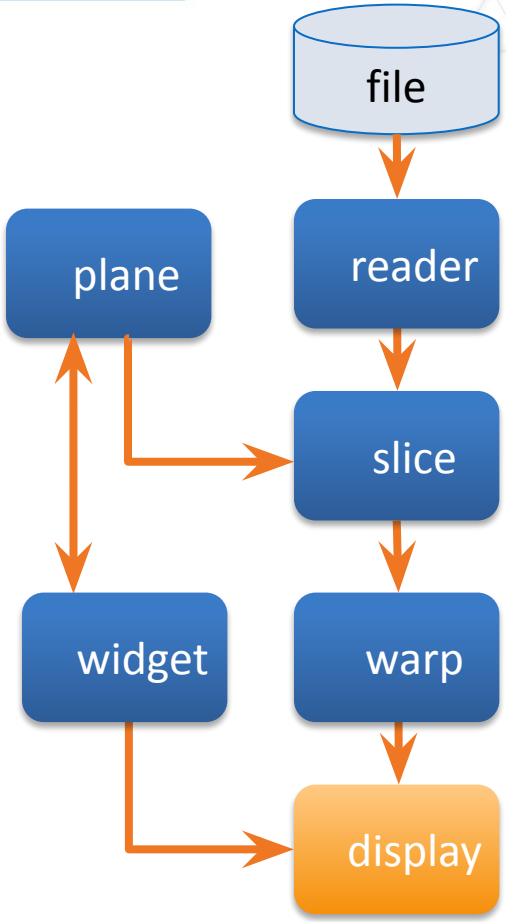
What is ParaView?

An application and **architecture** for display and analysis of massive scientific datasets.

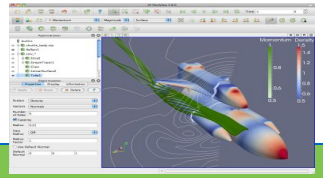
- Open source (BSD license) and Cross Platform
- Extensible
  - At compile time via External Code Module Interface
  - At runtime via Python scripting
  - At runtime via dynamically loaded C++ plugins
  - “Branding” infrastructure to create new applications
- Pick and choose what parts you need
- Extend and customize as you see fit

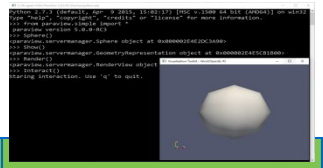
Visualization Pipeline

- Read in data: File->Open, hit Apply
  - Over **150 file formats** supported
  - Handles structured (uniform rectilinear, non-uniform rectilinear, and curvilinear grids), polygonal, unstructured, tabular, graph, multi-block, AMR and time varying data
- **Build a pipeline** to process data
  - Filter->Choose Filter
  - Tune filter parameters via Properties Tab and 3D Widgets, hit Apply
  - Tune rendering via Display Tab
- Repeat previous step as needed
- If you make a mistake, click Undo
- File->Save results in a variety of formats



ParaView Application Architecture





User Interface

Hardware

ParaView Client | pvpython | PVWeb | ParaView Catalyst | Custom App ...

Qt | python

ParaView Server Manager

VTK

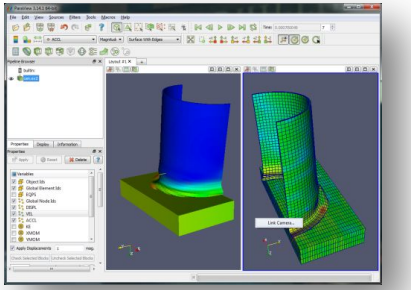
OpenGL | IceT Scalable Parallel Rendering | MPI | Etc.

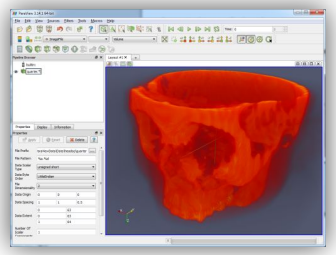
kitware

What is ParaView?

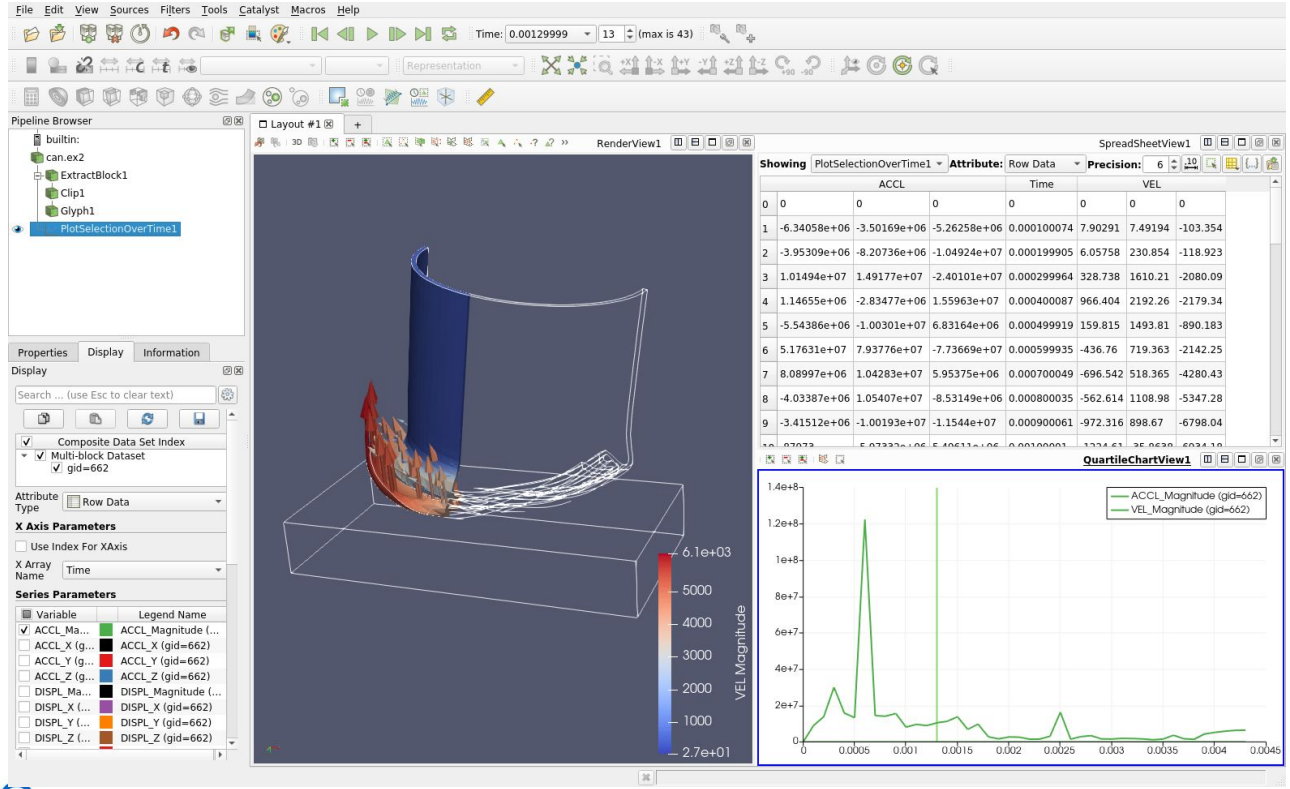
An application and architecture for **display** and analysis of massive scientific datasets.

- Displayed output of each filter can be independently controlled
  - Points, Surface/Edges, Volume Rendering representations
  - Interactive Lookup Table (LUT) Editing/Saving
- Simultaneous display in multiple viewing windows
- Augment view with a variety of annotations

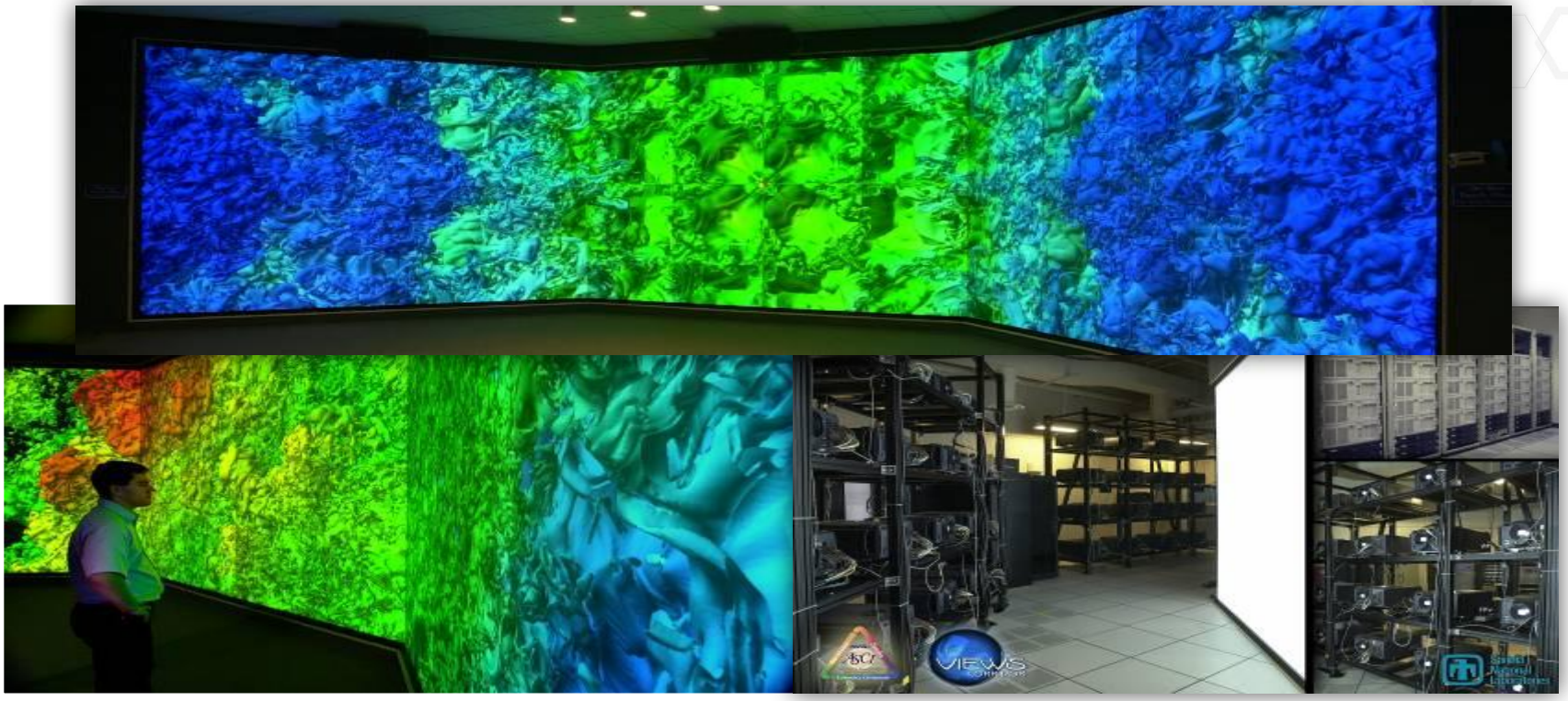




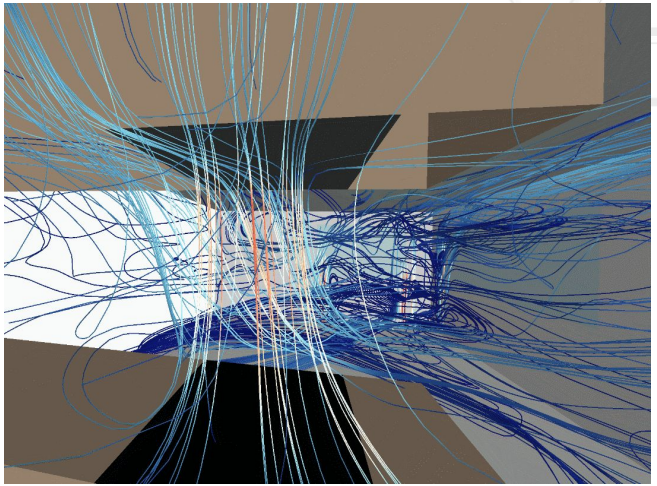
ParaView User Interface

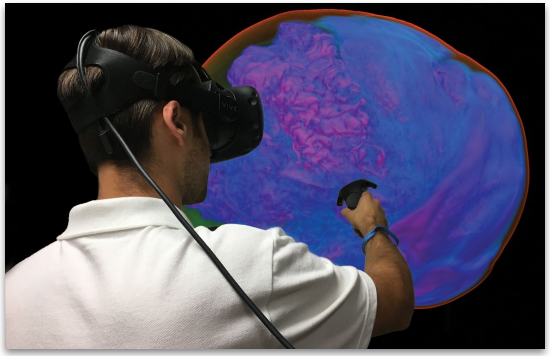


ParaView Support for Tiled Displays



Display in VR Environments

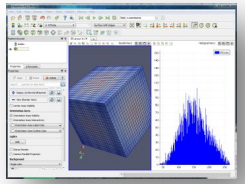


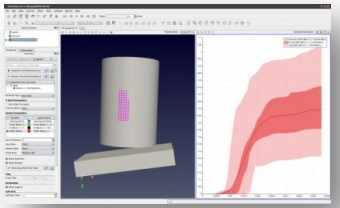


What is ParaView?

An application and architecture for display and **analysis** of massive scientific datasets.

- Chart, Plot, and Spreadsheet View types
  - Probe along a curve and get 2D plot of values along the line
  - Display coordinates and data values as text in a spreadsheet, or on screen via Selection labels
- Find Data queries
- Calculator and Programmable filters let you perform arbitrary data manipulation at runtime
- Statistics tools – Descriptive, consistency, K-means, PCA…
- Import and export to .csv and many other formats





What is ParaView?

An application and architecture for display and analysis of **massive** scientific datasets.

- Client/Server architecture lets it run on a variety of platforms
  - from notebooks
  - to the largest machines in the world
- Support for tile display and parallel rendering
- Level of detail techniques keep it interactive on huge data
- Can perform In-Situ analysis with Catalyst

Comparison to Other Image Analysis Software

ParaView Compared to ImageJ/Fiji

ImageJ/Fiji

- Java runtime memory limits
- Not designed to work on parallel systems*
- Embarrassingly parallel algorithms OK
- 3D visualization plugins available, not parallel
- Rich ecosystem of segmentation algorithms
- May need to install Java and do esoteric configuration on new systems (macOS)
- Plugin API is pretty simple, but need to compile code into bytecode

ParaView

- Built from the start to render huge datasets in parallel
- Parallel data readers exist
- 3D visualization focus
- Slice-based visualization possible
- Thresholding/find data is only basic segmentation method available
- Packages built with everything
- Some easy ways available to create plugins

ParaView Compared to 3D Slicer

3D Slicer

- Can render large datasets on capable workstation
- Not designed to operate remotely or in parallel
- Strong slice-based and 3D visualization capabilities
- Large set of segmentation algorithms available
- Packages built with everything
- Extensions are possible with Python or C++ programs
- Large set of extensions in 3D Slicer App Store

ParaView

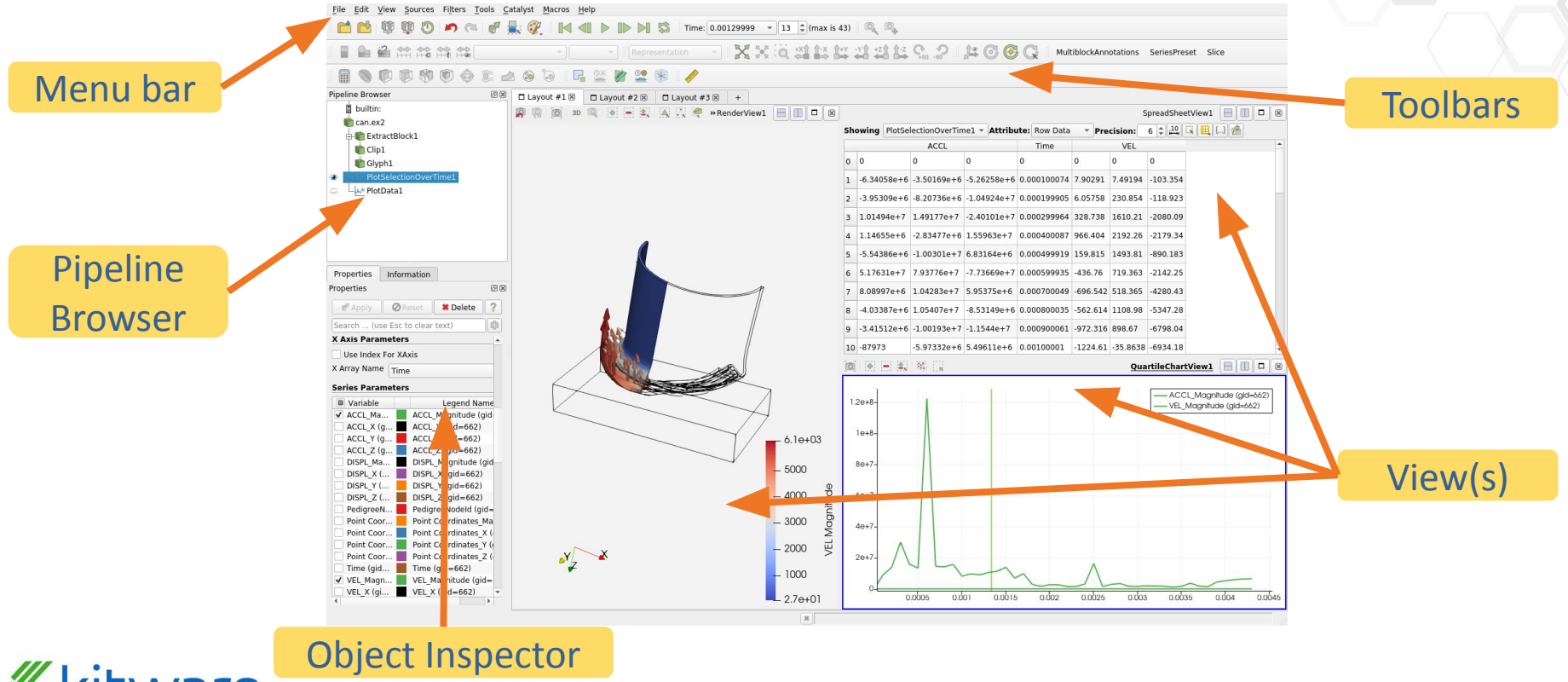
- Built from the start to render huge datasets in parallel
- Parallel data readers exist
- 3D visualization focus
- Slice-based visualization possible
- Thresholding is only basic segmentation method available
- Packages built with everything
- Some easy ways available to create plugins

ParaView User Interface

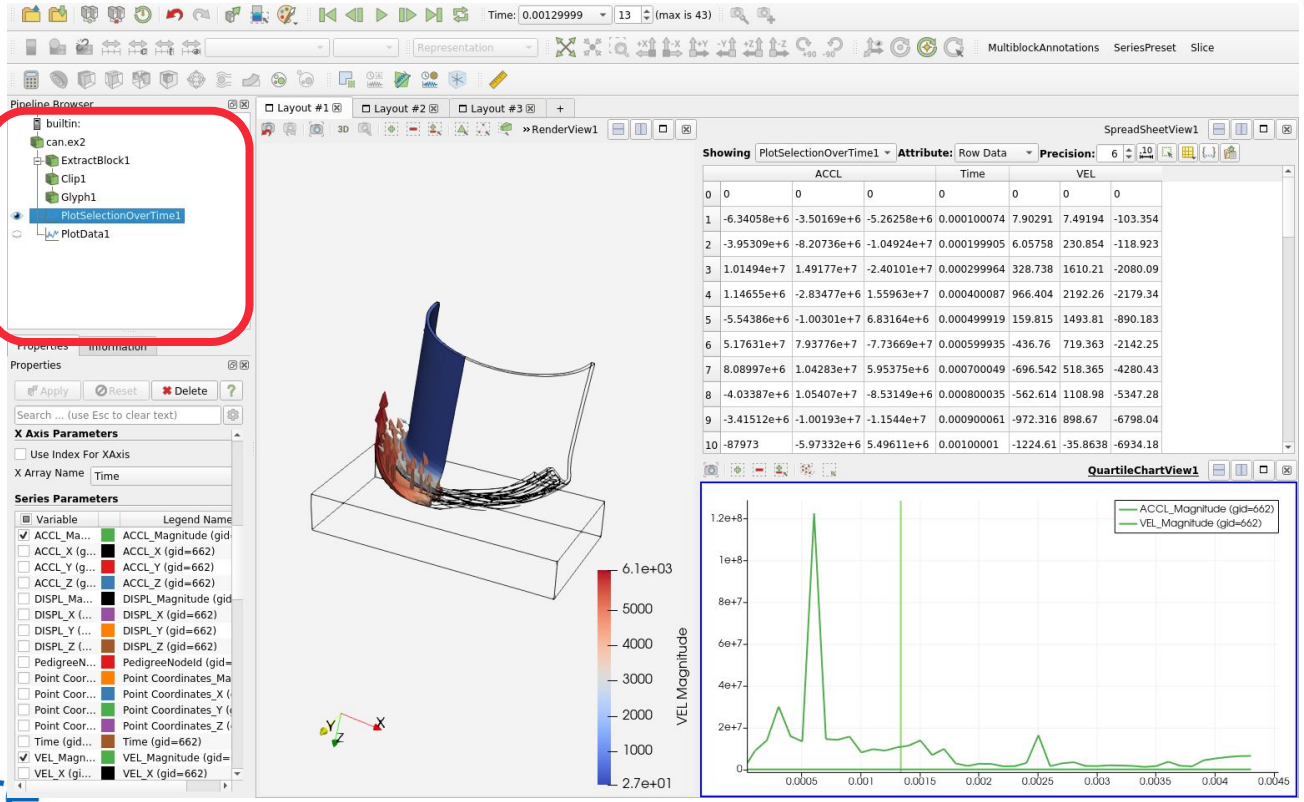
What is ParaView?
Comparison
ParaView User Interface
Data Filtering
Data Analysis
Distributed Processing
Distributed Rendering
System Requirements
Catalyst

kitware

ParaView - Graphical User Interface



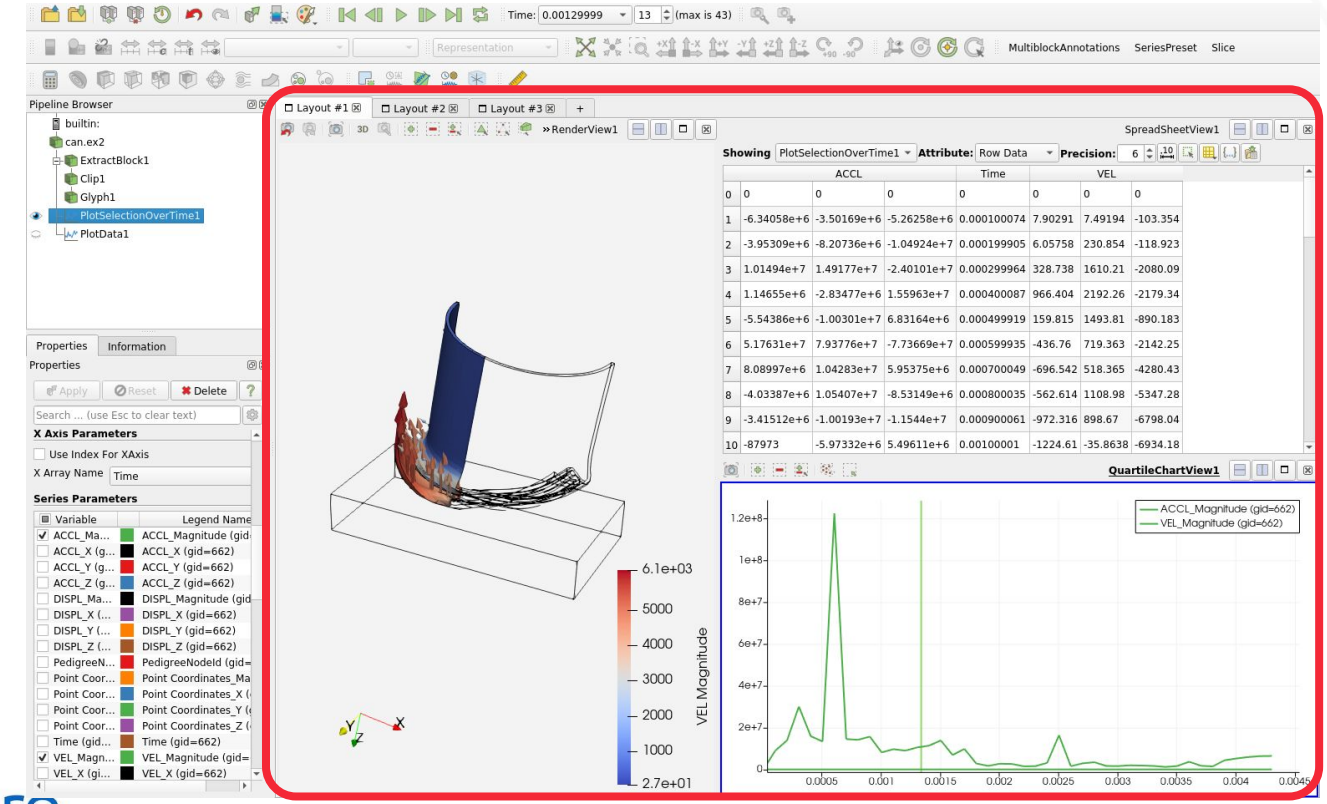
Pipeline Browser



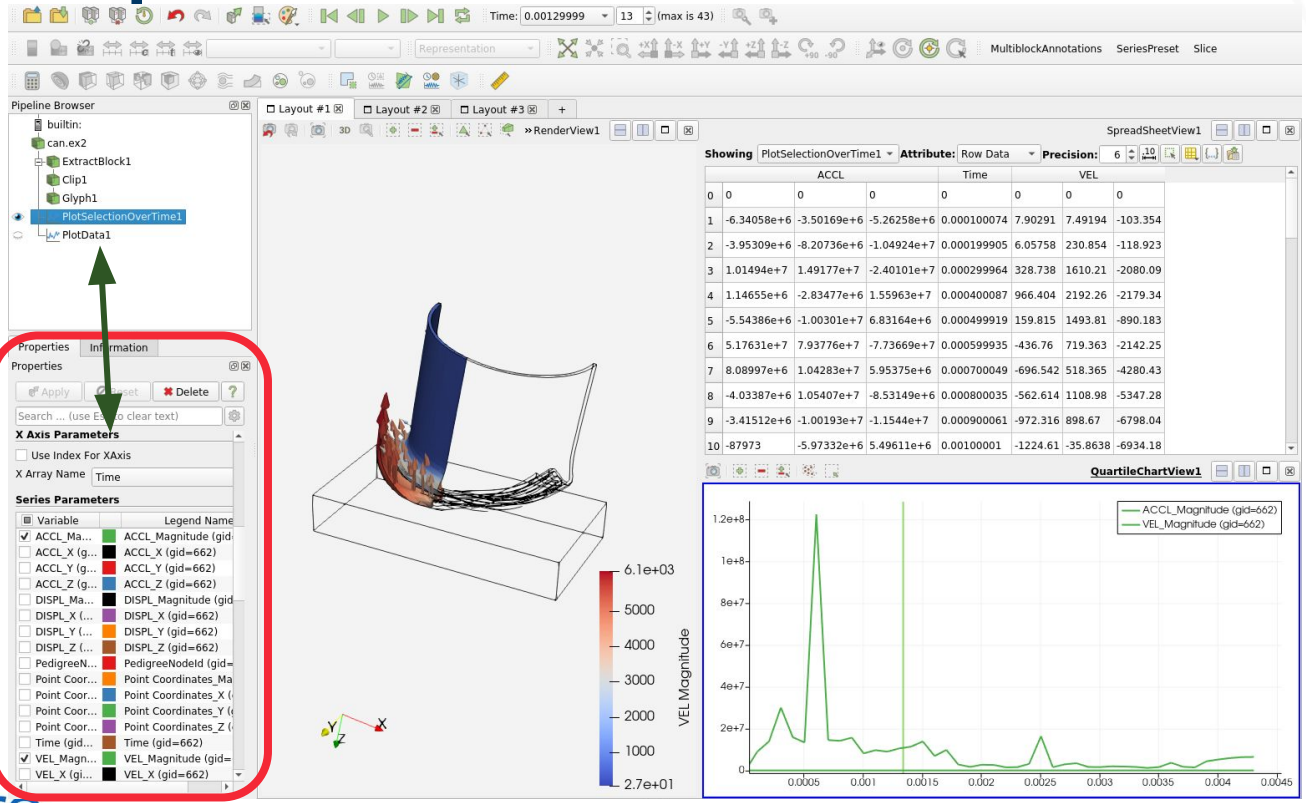
Active Source

- Source = pipeline element
- GUI can control only one thing at a time
- Click on pipeline browser to choose what
- Click on Apply to “Update” 

View Area



Object Inspector

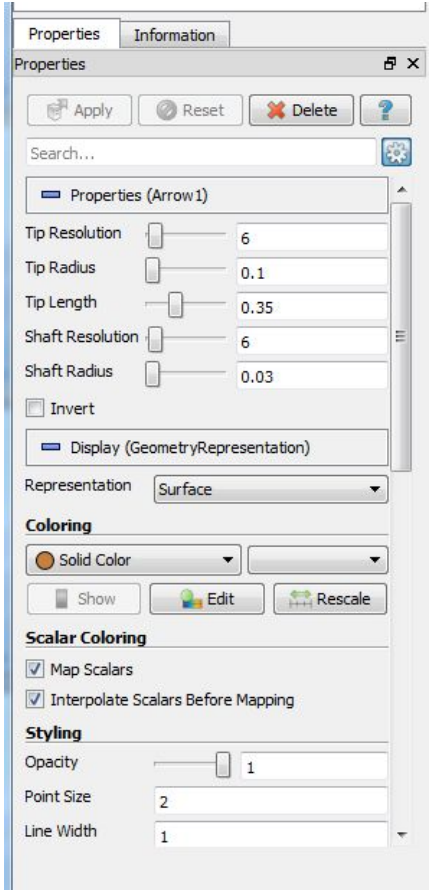


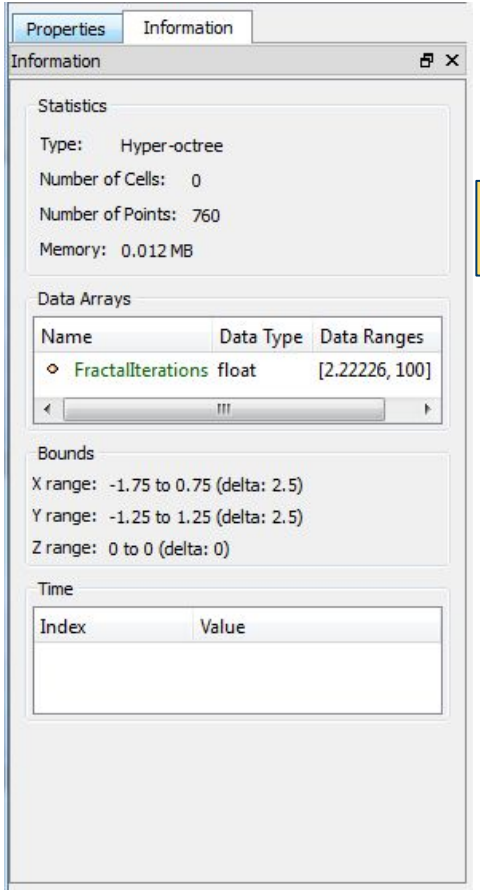
Concern only
Active source

Inspectors

Properties

Options for **Active Source**

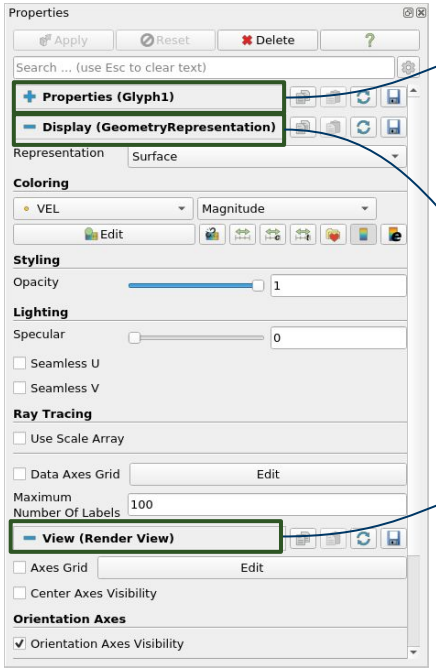




Information

Characteristics of Active Source **output**

Inspectors



Parameters options of **Active Source**

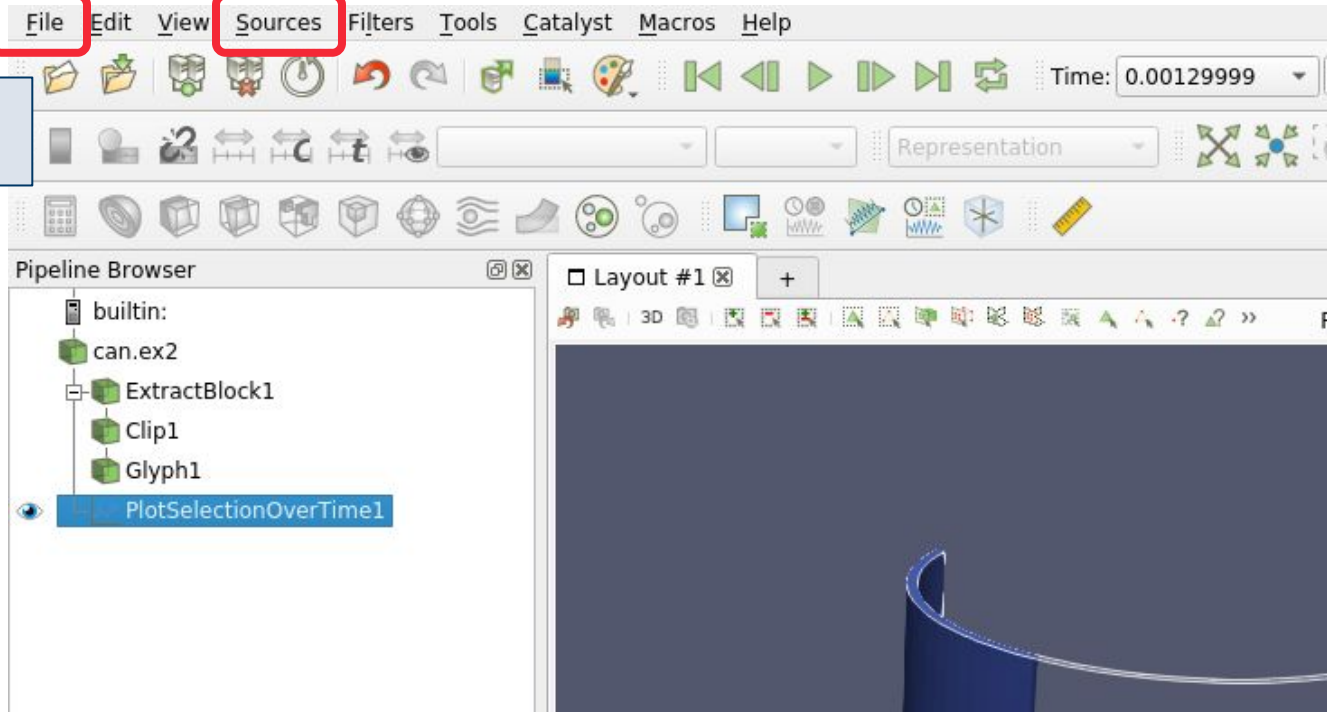
Display options of **Active Source in Active View**

Characteristics of Active **View**

Importing Data

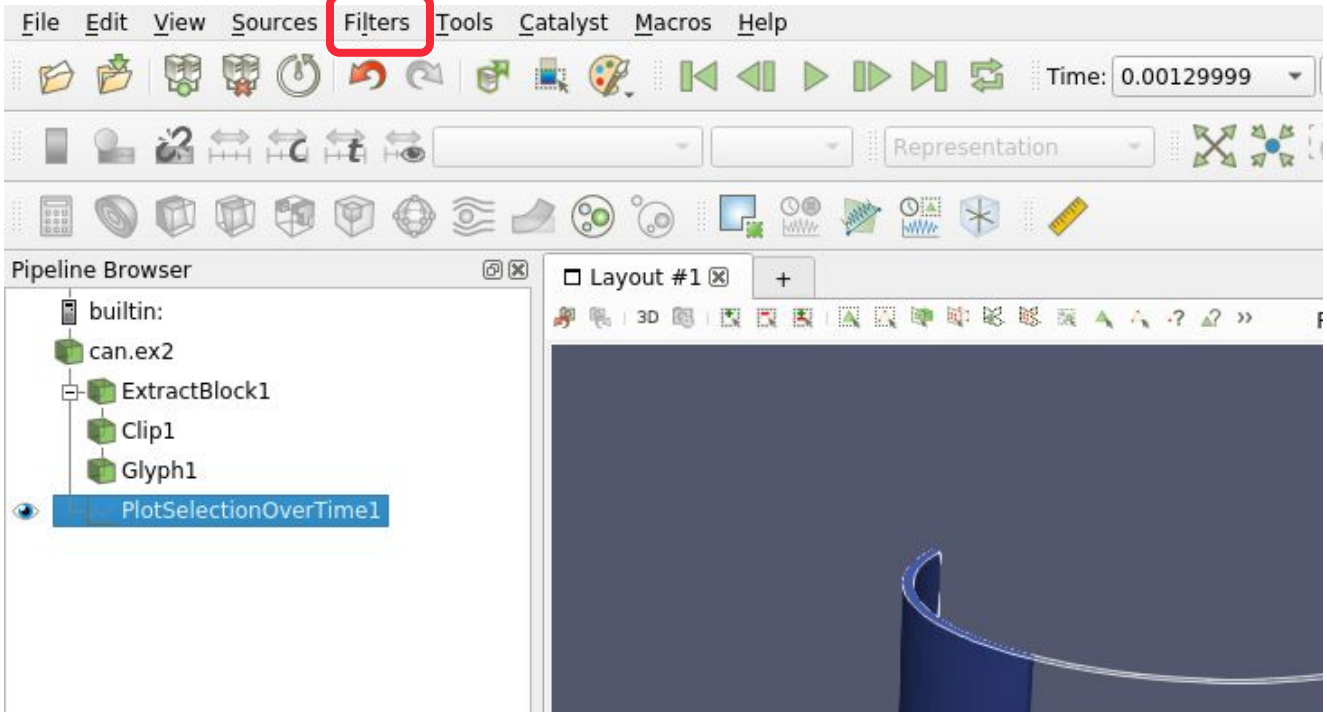
Procedural sources of data

Import data from files



Data Processing

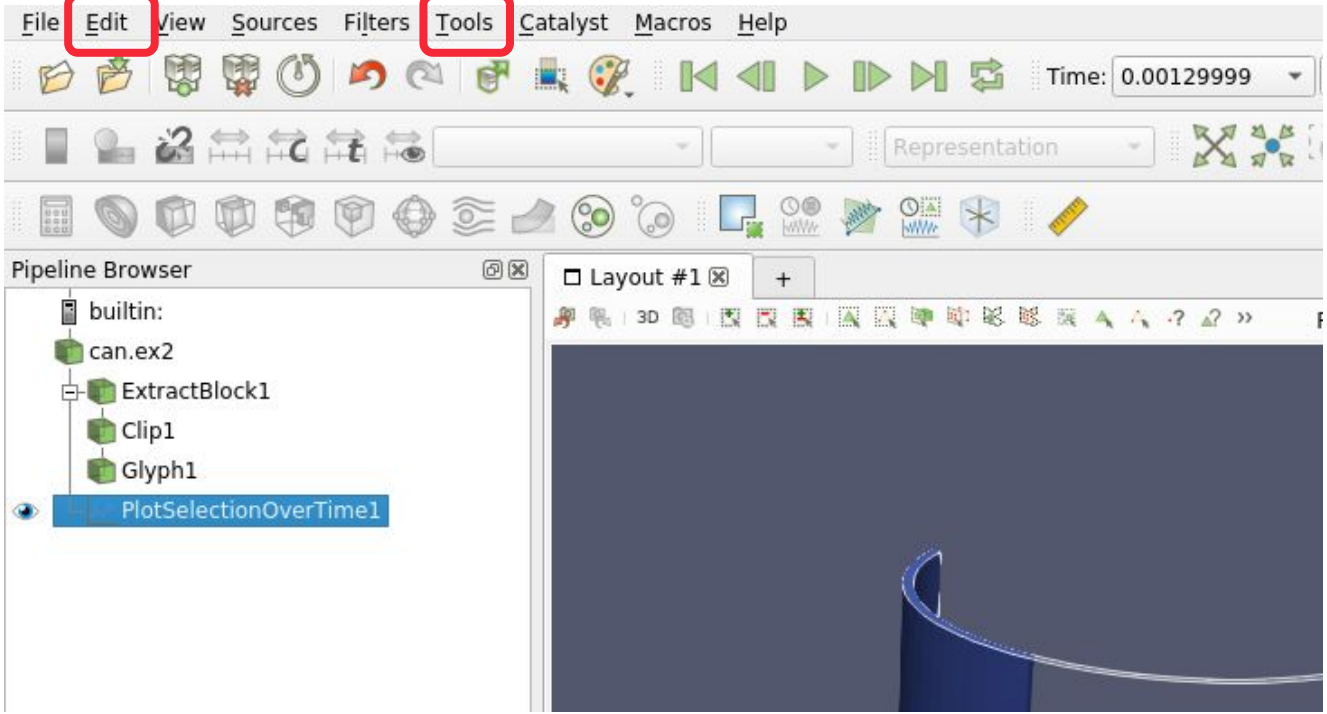
Subset of VTK filters applicable to the active source



Customize

Load plugins, define shortcuts ...

Edit settings



Undo/Redo

- ParaView keeps track of your changes
  - Filter creation, deletion
  - Parameter changes
  - View Splits
  - Camera Motion
- Undo/Redo lets you backtrack
- If you make a mistake:
  - before you Apply → hit Reset
  - after you apply → hit Undo


Undo

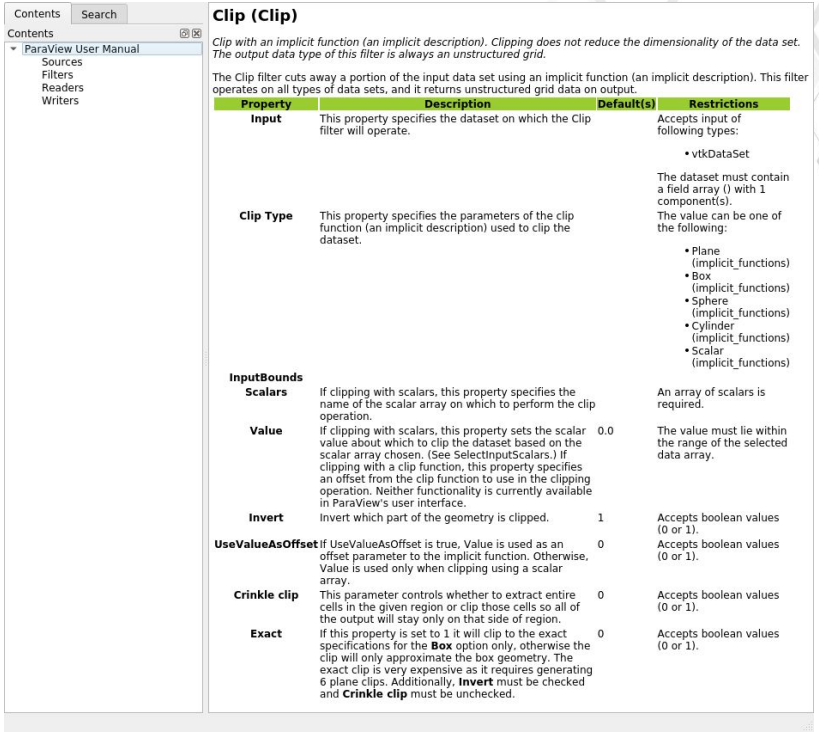

Redo


Camera Undo


Camera Redo

Getting Help

- *Help -> Getting Started*
- *Help -> Example Visualizations*
- F1 for ParaView user doc
- Generated pages for filters and sources
  - for every accessible method of every reader, source, filter and writer
- Duplicated online

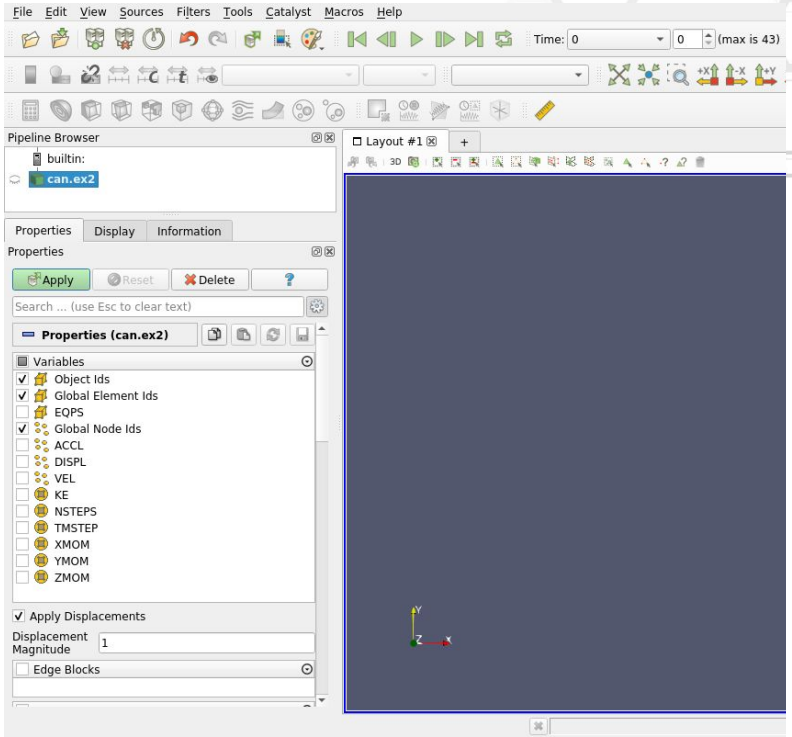


Important!

- **Active Source, Active View**
  - Specify exactly what your GUI actions modify
  - Specify what the next filter is applied to
- **Actions are committed**
  - Focus is large data, minor parameter changes ignored
  - **Nothing happens until Apply button hit**
- **Every part** of pipeline can be **displayed**
  - You have complete control over pipeline not just “final” result
- Only applicable filters are allowed
  - “Applicable” depends on data type, data arrays, etc

Loading a DataSet

- Menu *File/Open*
- Modify load options (**properties**)
  - e.g. arrays to load, timesteps ...
  - depends on file format
- Hit '**apply**' to read the data

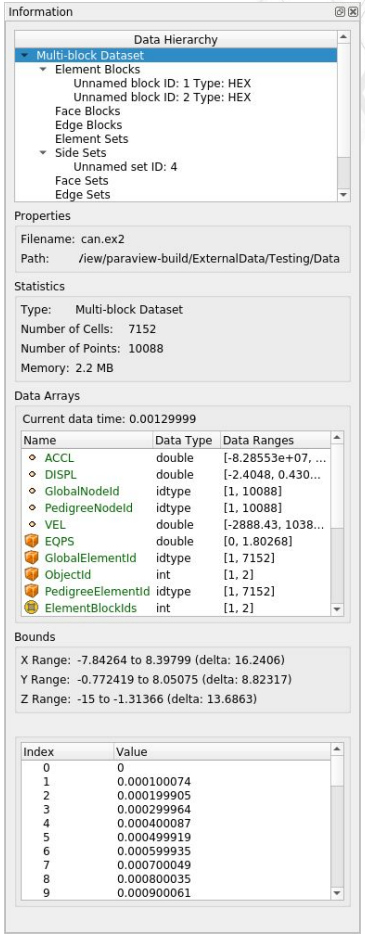


Supported File Formats

- 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)
- CGNS (*.cgns)
- 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)
- M3DCI (.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
- NetCDF (.ncdf, .nc)
- **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

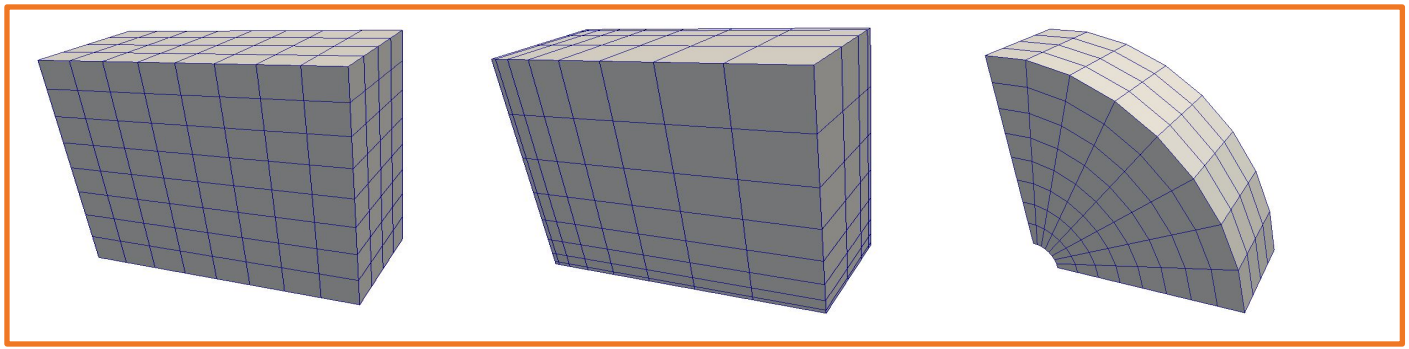
Information Tab

- Information about the Active **Source's output**
- Data object **structure**
- **Size** (#points, #cells, memory)
- Geometric **bounds**
- Structured bounds
- **Arrays**:
  - Association (point , cell , block)
  - Name
  - Data Type
  - Value Ranges
- **Temporal** Domain



Frequent Data Types

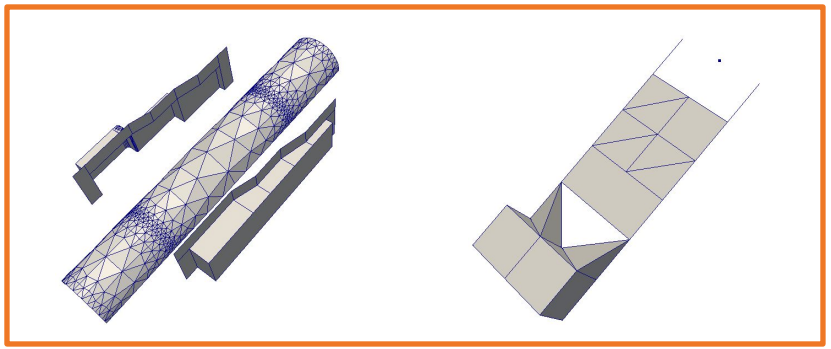
Structured



Uniform Rectilinear	Non-Uniform Rectilinear	Curvilinear
`vtkImageData`	`vtkRectilinearGrid`	`vtkStructuredGrid`

Frequent Data Types

Unstructured

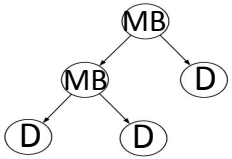


Polygonal
`vtkPolyData`

Unstructured Grid
`vtkUnstructuredGrid`

Frequent Data Types

Miscellaneous



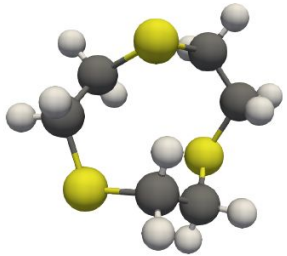
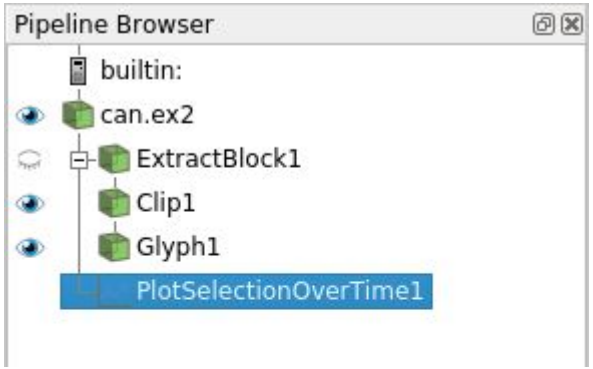


Table	Multiblock	Molecules
`vtkTable`	`vtkMultiblockDataSet`	`vtkMolecule`

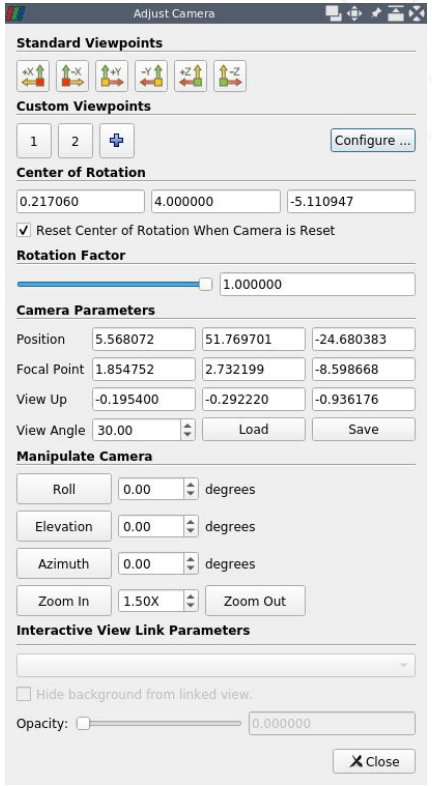
The Eye

- On **Pipeline Browser**
- **Shows** and Controls what is visible in Active View
- If eye is missing, it is not displayable in that type of View
- **Press Apply before the eye !**



Camera Manipulation Tools

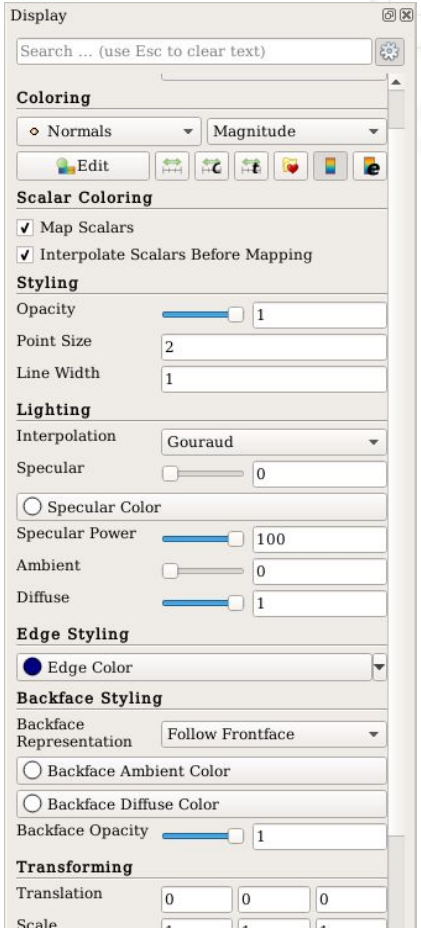
- **Reset** Camera
- Orthogonal Views
- **Center** of Rotation
- Camera **Undo/Redo**
  - View button
- **Adjust** Camera dialog
  - View button



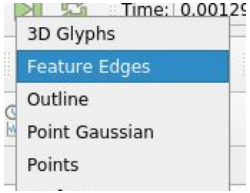
Display Properties

- Full Control of the **Appearance** of Active Source as shown in Active View
  - **Representation**
    - Bounding box
    - Wireframe
    - Surface
    - VolumeRender
  - **Color**
    - Transfer function
    - Solid, or Palette
  - **Texture**
  - **Transformation**
- Shortcuts to most important controls are on toolbar

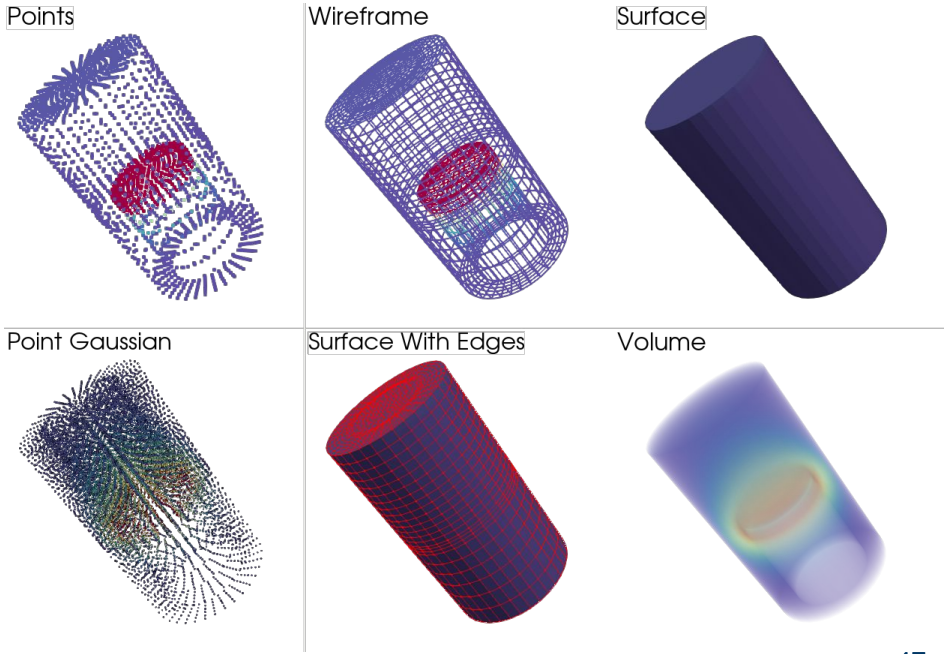
/!\ Press the cogwheel button ⚙ to show all properties



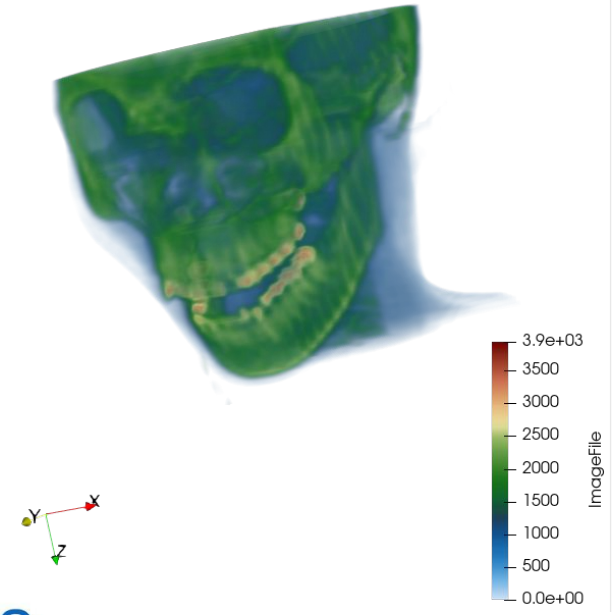
Display the data

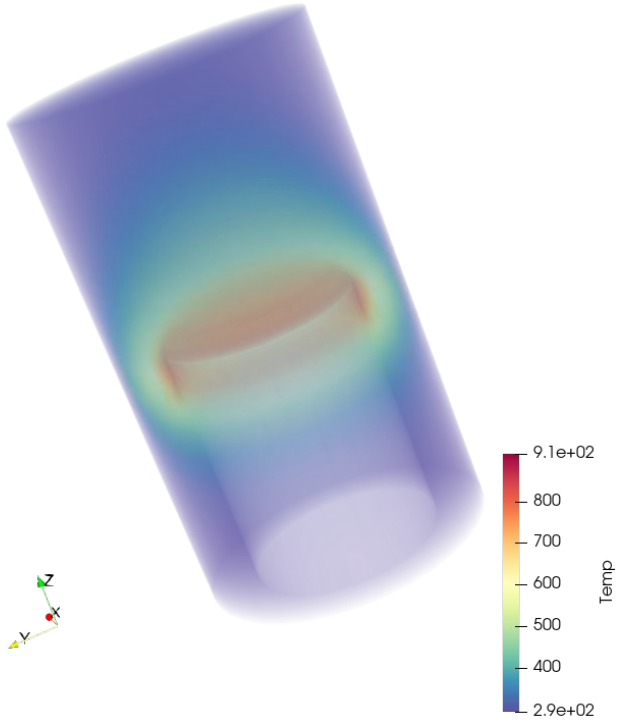


- Representations (aka Displays): visual characteristics of one particular data set
- (disk_out_ref.ex2) in one particular view

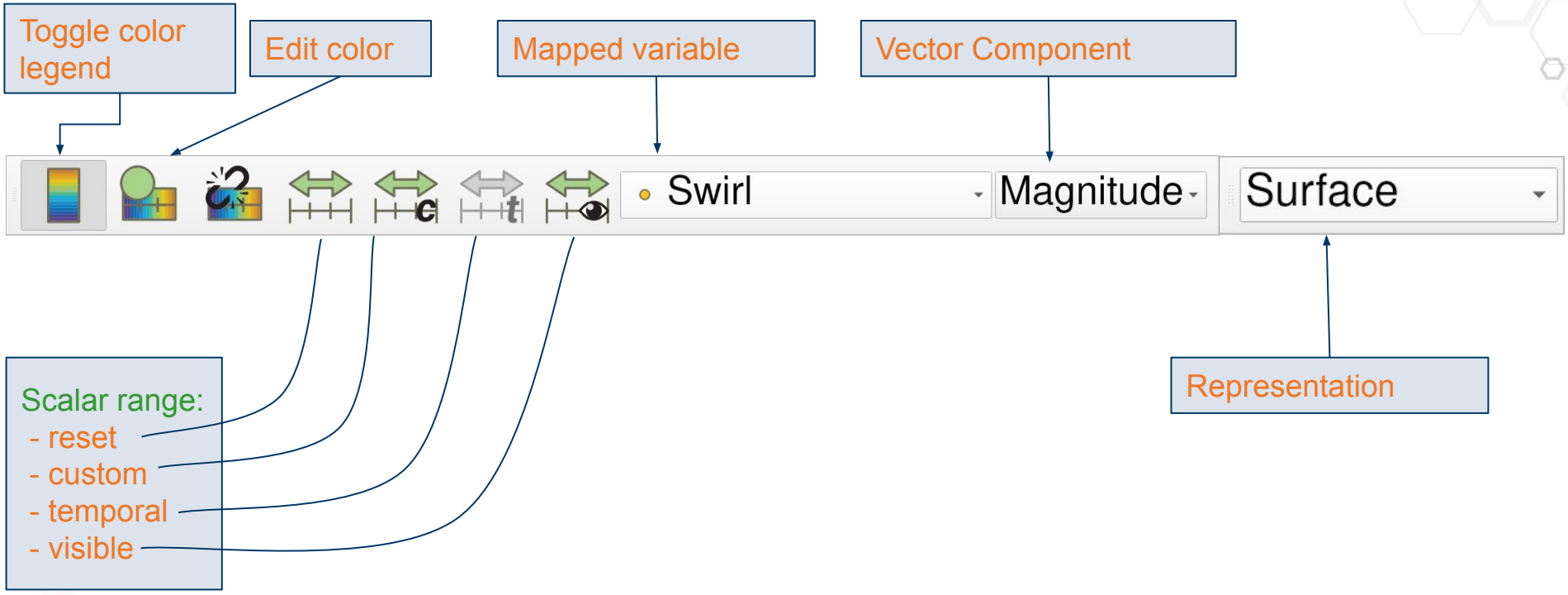


Volume Rendering





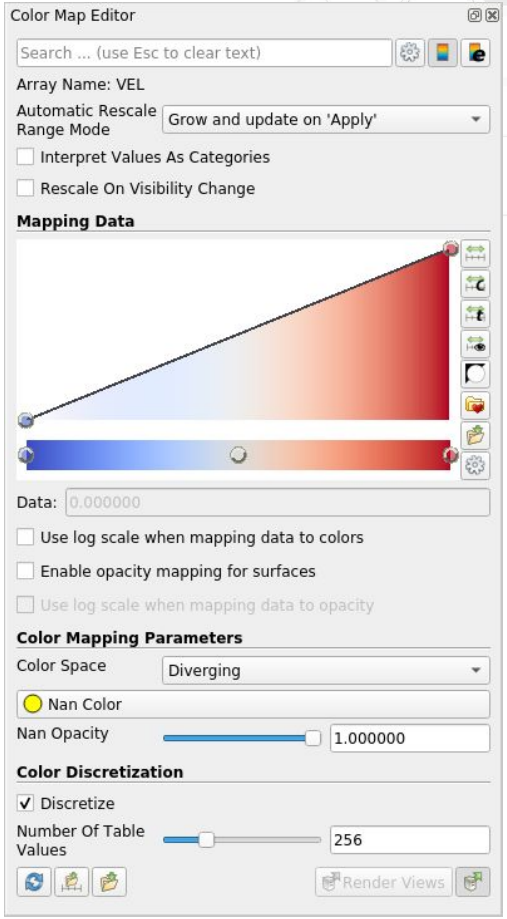
Representation Controls



Mapping Scalars to Colors

- If present, can **color with attribute**
- **Transfer function** (lookup table) maps values to colors (and opacities for volume rendering)
- Scalar **Range** of the lookup table determines how the scalars are mapped

- Open color scale **editor** dialog

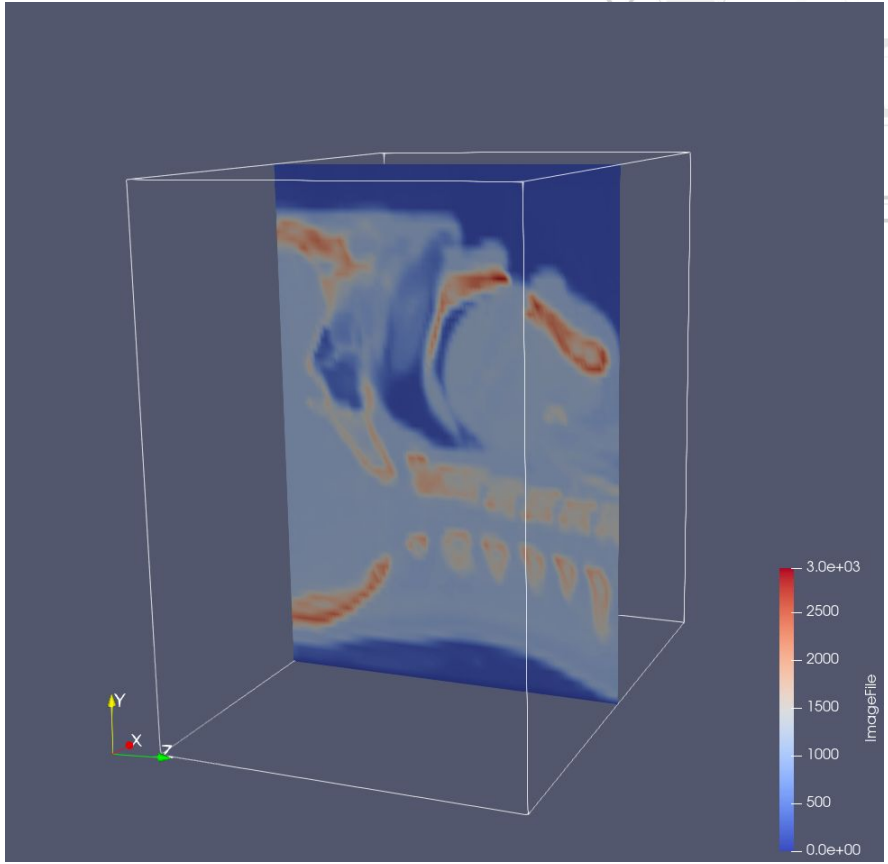


Data Filtering

kitware

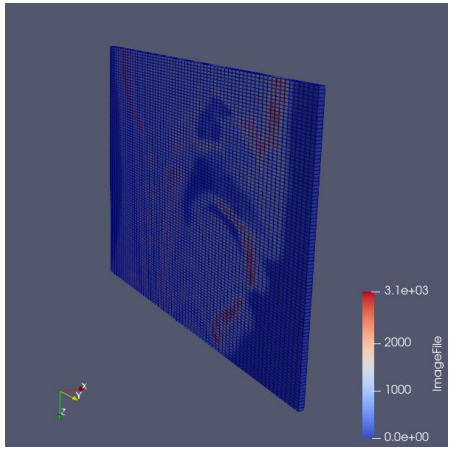
Slice

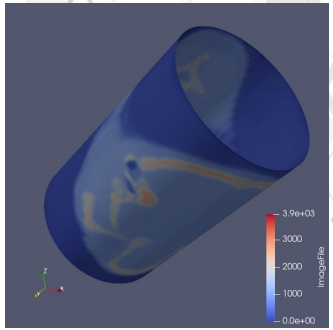
- Produces a slice of a dataset
- Provides a **widget** to aid in setting up the slice plane

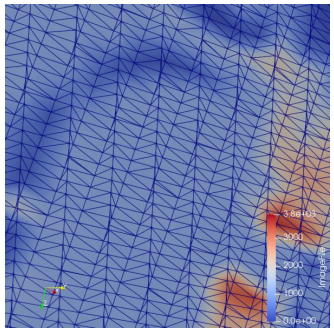


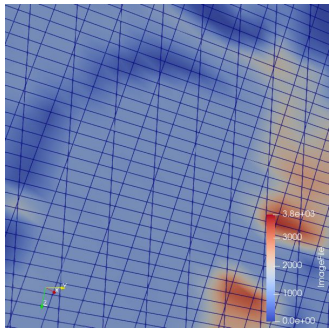
Slice

- Crinkle slice will only extract cells instead, does not reduce dimensions
- Triangulate will generate only triangles
- Type of slices: Box, Plane, Sphere, …



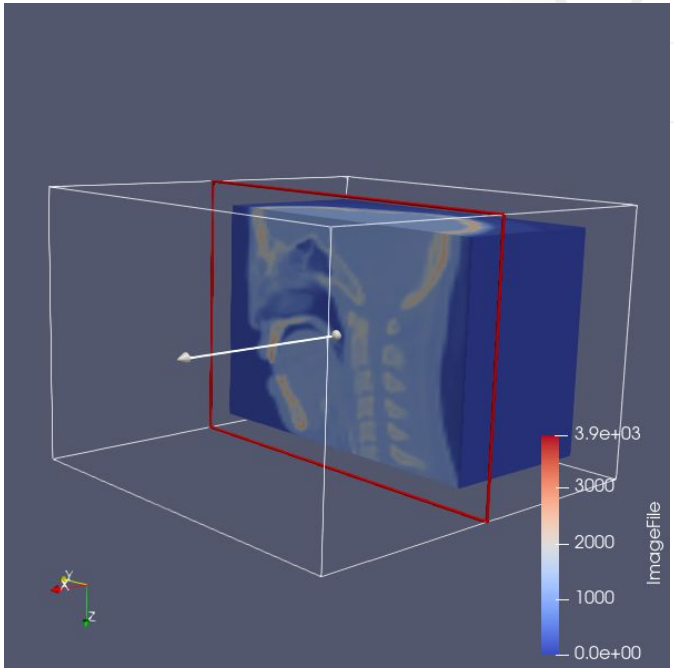






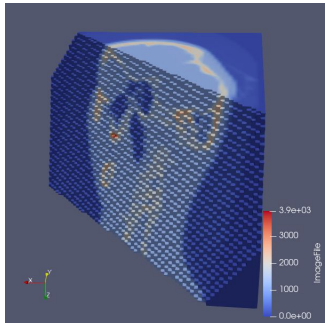
Clipping

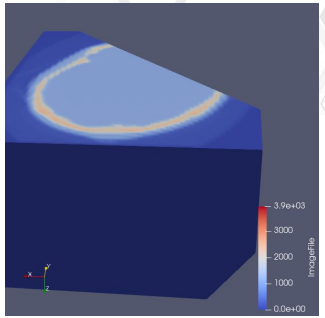
- Like slicing, but results in sub-volume
- Result is an **unstructured grid**

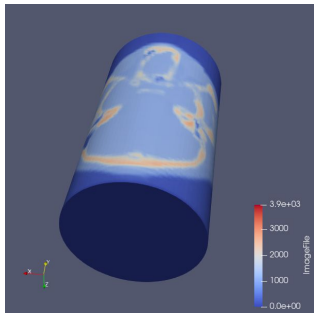


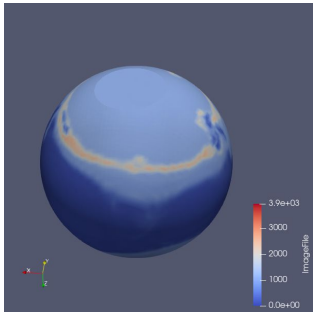
Clipping

- Crinkle clip will only extract cells instead
- Invert let you choose the side to clip
- Type of clip: Box, Plane, Sphere, **Scalar**, …



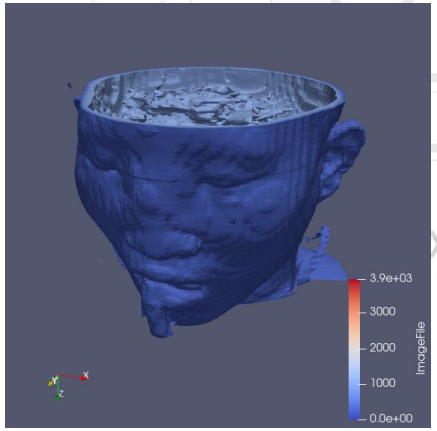


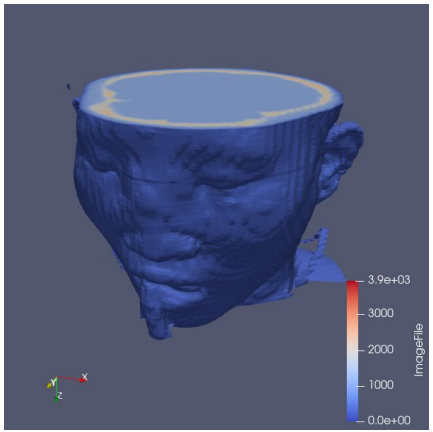




Clipping with scalars / Isovolume

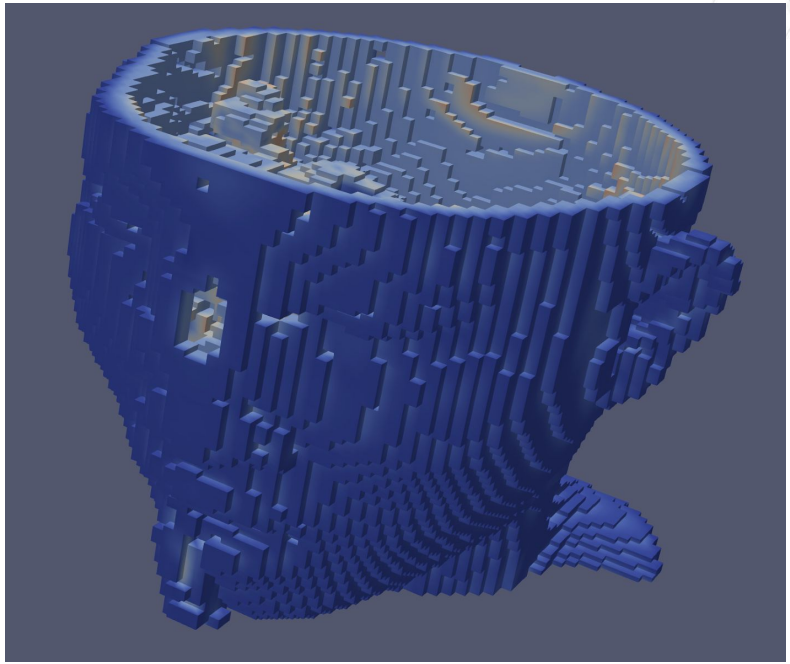
- Clipping with Scalars creates an isovolume with a single scalar
- Isovolume filter has a min and a max
- Result is an **unstructured grid**





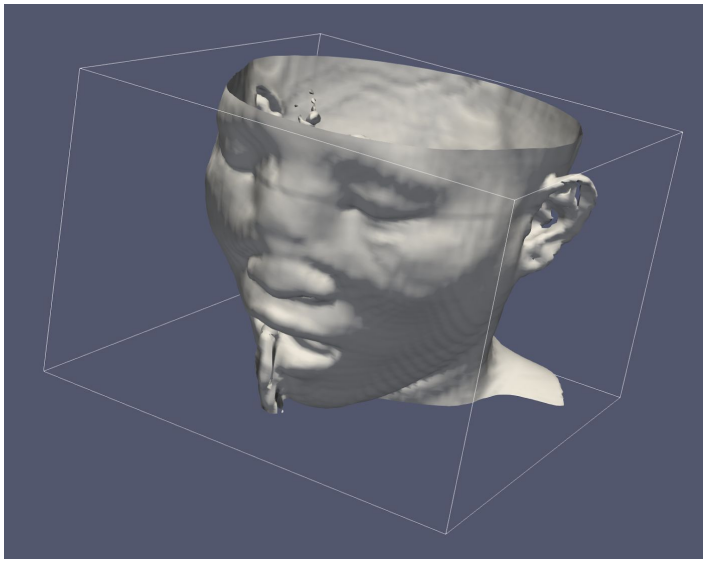
Thresholding

- Select cells by data values
- Like isovolume, but crinkle
- Result is **unstructured grid**
- Methods: between, min and max



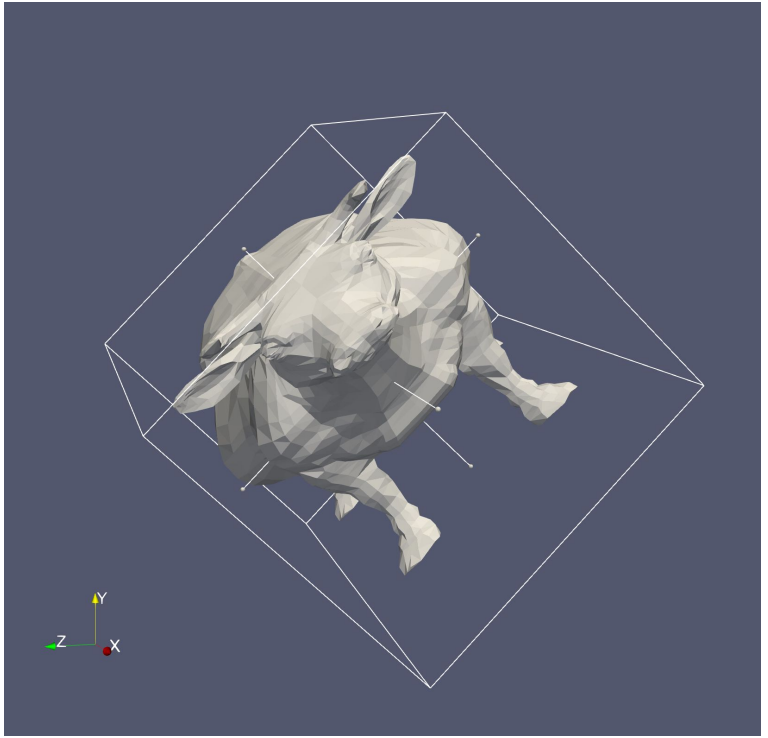
Contouring

- Connects points with same value in 2D or 3D
- Result a poly-line or a **polygonal** surface



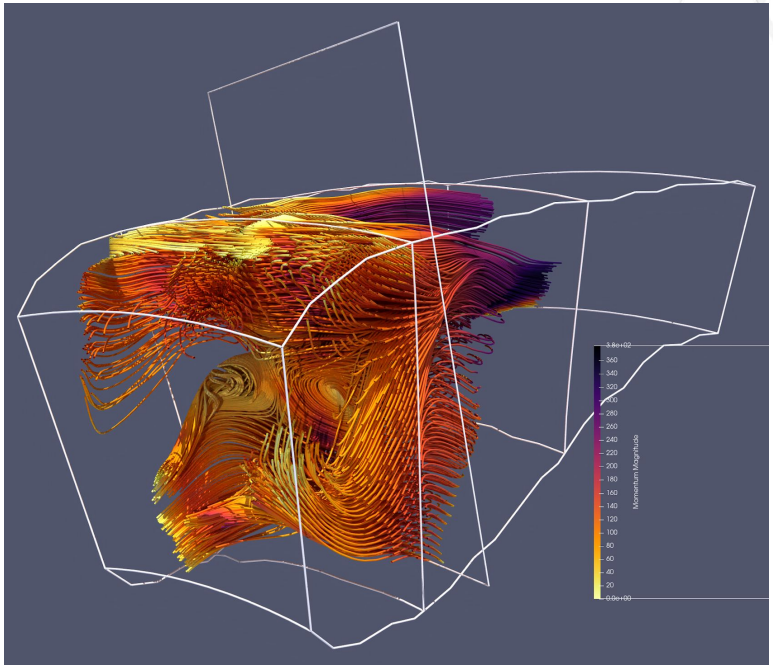
Transform

- Transforms the dataset
- Applies scale, translation, rotation to the input dataset
- Produces a new dataset with the transformed points



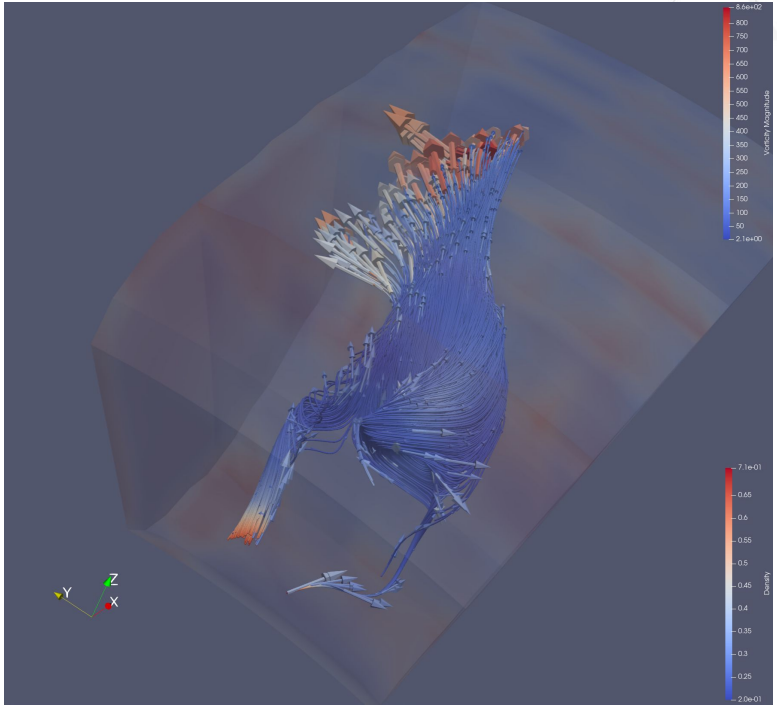
Stream Tracer

- Generate streamlines in a vector field from seed points using a vector field
- **Require point-centered vectors**
  - On any type of dataset
- **Require seeds**
  - Point cloud (X + Radius)
  - Line source (X1 -> X2)
  - Custom (use dedicated filter)
- A streamline terminates if
  - Crosses the exterior boundary
  - Reaches MaximumNumberOfSteps, TerminalSpeed or MaximumPropagation
- Output is **polylines** (polygonal data)



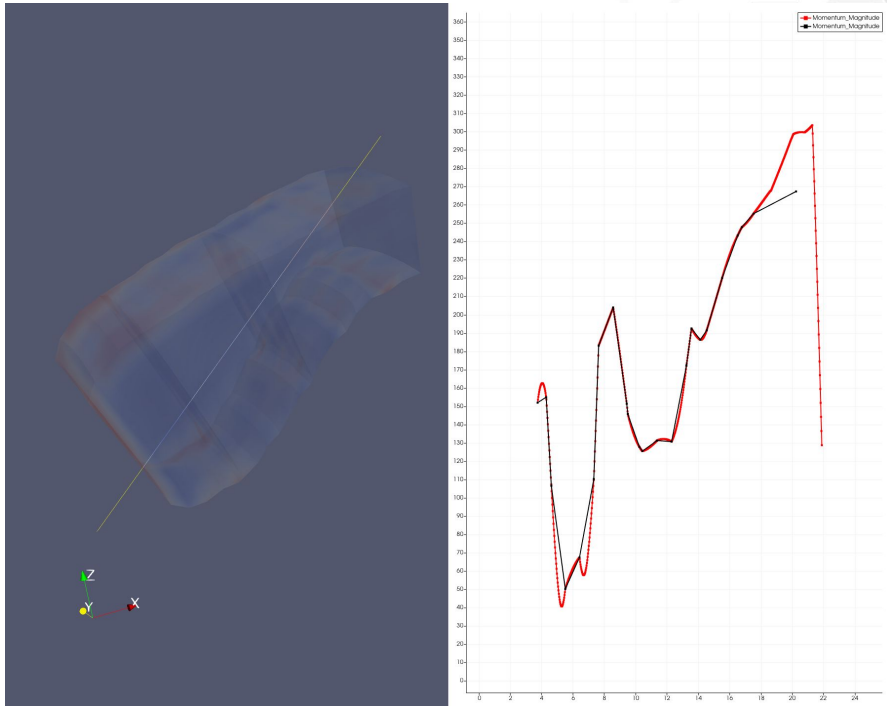
Glyph Filter

- Copy some **geometry to every point** of the model
  - Scale by scalar
  - Orient by vector
- Filter to reduce clutter
- Many glyph types



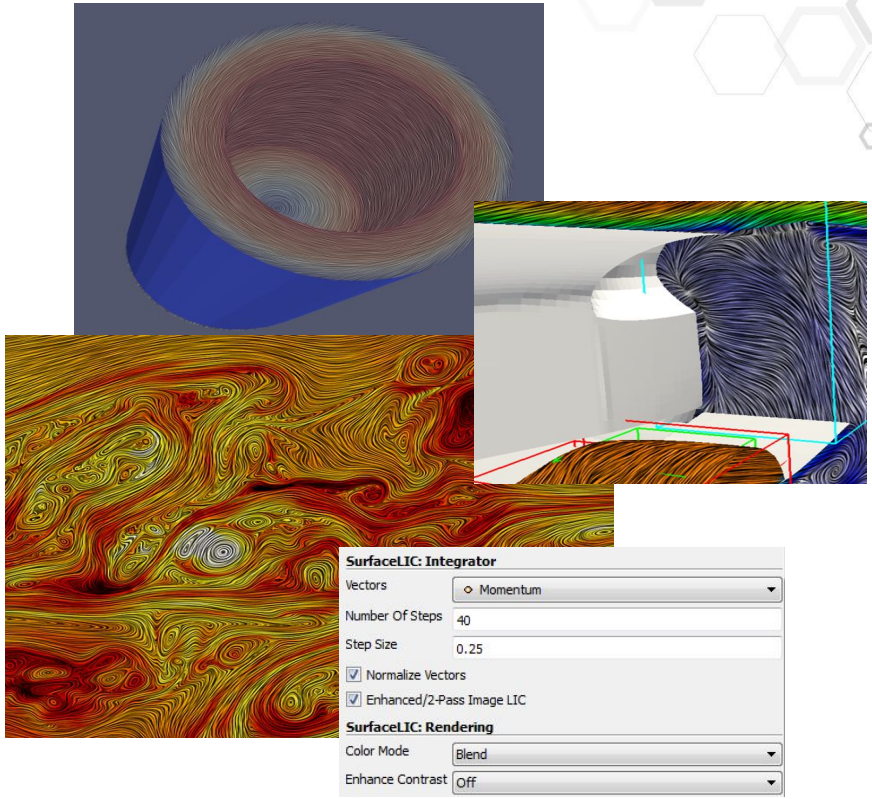
Plot Over Line

- Plot point and cell data over a specified line
- Line geometry can be set manually
- Use “Custom Source” to plot over any line geometry
- Different sampling pattern:
  - Plot using the input line point
  - Plot on the cell centers
  - Plot on the cell boundaries



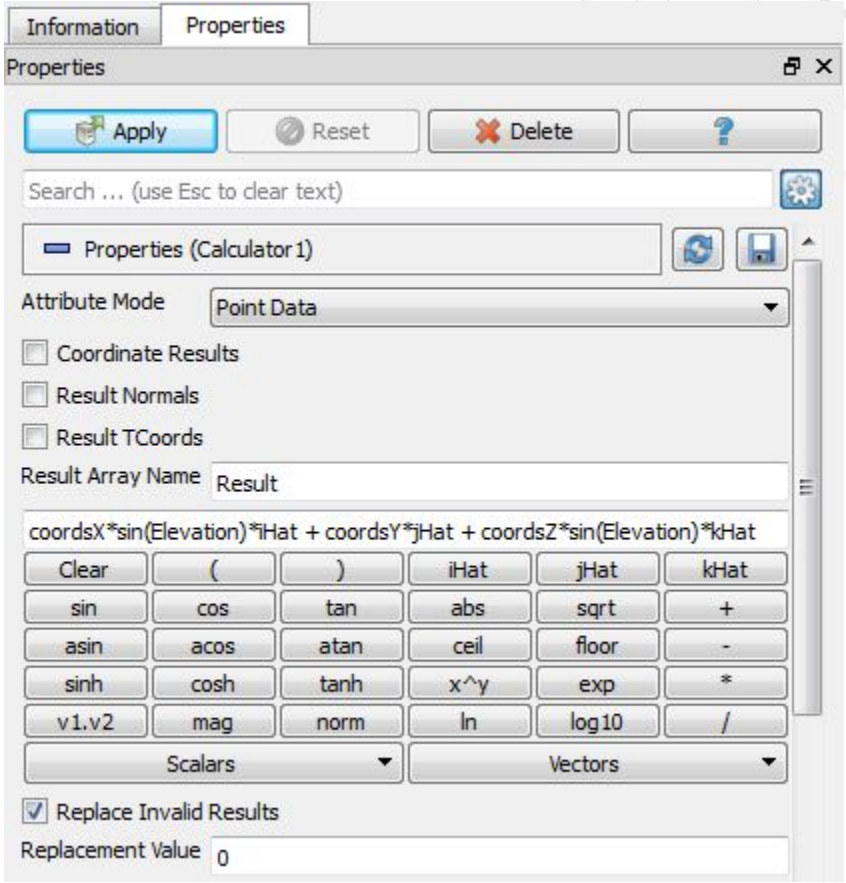
Surface LIC Representation

- The line integral convolution(LIC) **vector field visualization** technique convolves noise with a vector field producing streaking patterns that follow vector field tangents
- Displayed on top of the surface of any dataset



Calculator

- Write **expression** to derive new data from input
- Expression takes in:
  - **point centered** scalars/vectors
  - point coordinates
  - **OR cell centered** scalars/vectors
- **Runs over each point or cell** and evaluates expression
- Expression produces either:
  - A new point centered array
  - A new cell centered array
  - New point coordinates



Python Calculator – Accessing Input Arrays

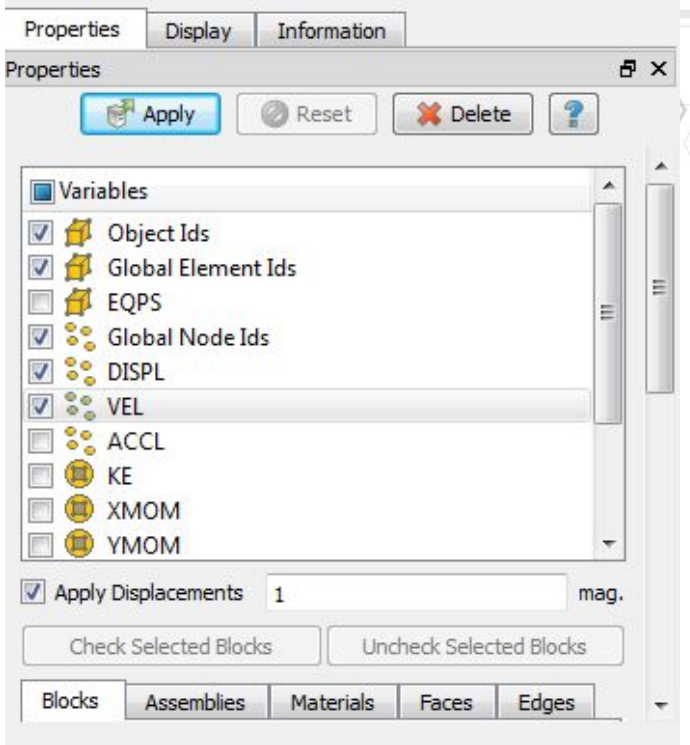
- Based on Python and **NumPy**-like syntax to access data arrays
- **`inputs[0]`** refers to the first input (ie. dataset) of the filter
- Accessing **point data**: `inputs[0].PointData['arrayname']`
- Accessing **cell data:** `inputs[0].CellData['arrayname']`
- Normals is equivalent to `inputs[0].PointData['Normals']`
- To access point **coordinates**: `inputs[0].Points[]`
  - `inputs[0].Points[:,0]` to extract the X coordinates

Filters and Data Type

- Not all filters can be used for all types of data
- /!\ **Filters can change data type**
  - Structured to unstructured (eg: Clip)
  - Quad/hex to triangle/tetra
  - Simple to multi-block dataset
  - Etc.

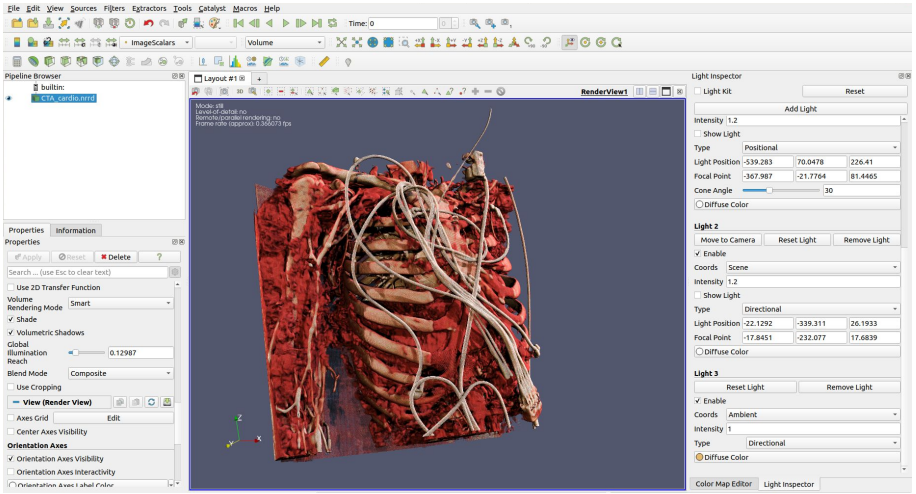
Filter Parameters

- **Properties** Tab of Object Inspector
- **Apply**
  - No effect until you press it
- **Reset**
  - Reset to previous settings
- **Delete**
  - Only for tree leaves
- **Help**



Filters for Medical Images

- **Filters**
  - Clip
  - Slice
  - Threshold
  - Resample To Image
- **Representations**
  - Volume
  - Multi Slice
  - Realistic Rendering



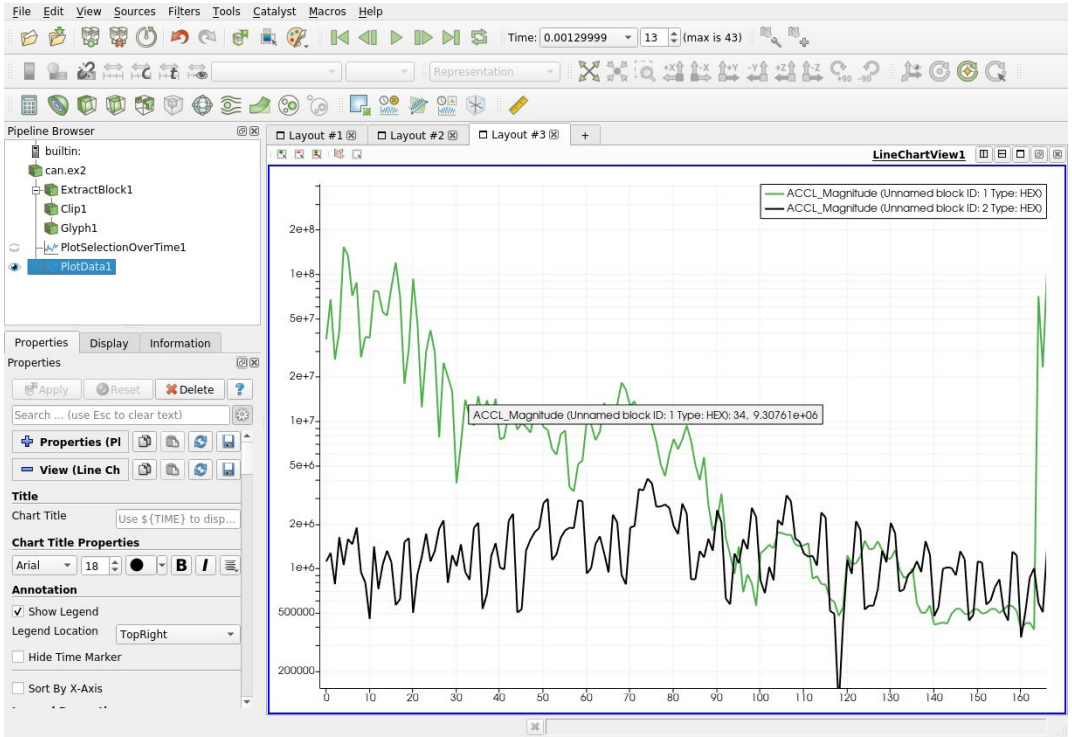
Data Analysis

kitware

Filters creating charts

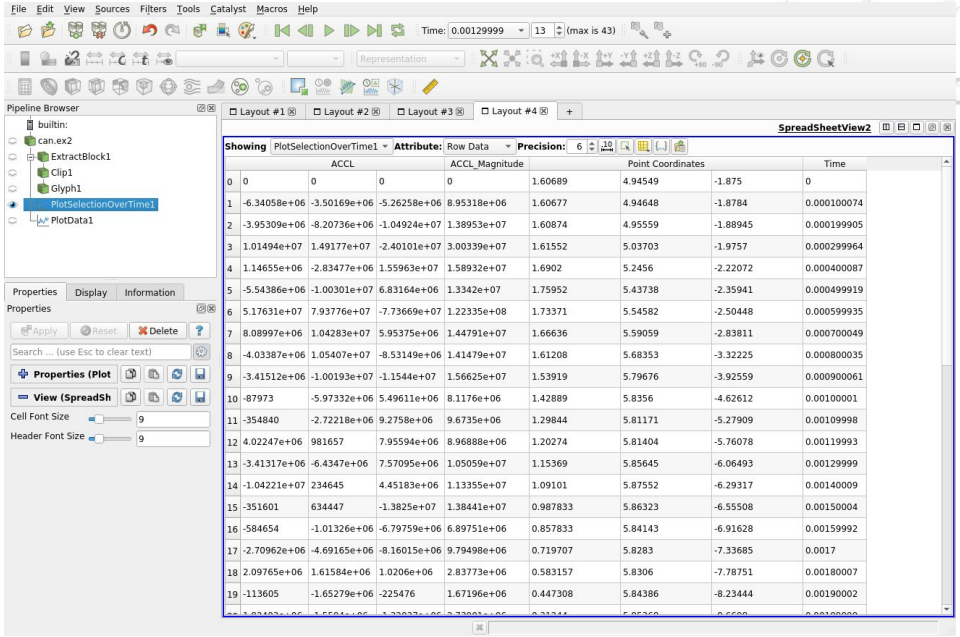
- **Histogram** filter
  - Bins attribute values to show distribution
  - Default view is a histogram view
- **Plot Data**
- **Plot** over line, Plot over intersection curve,
  - Probes attribute values at points along curves in space
- **Plot Selection** Over Time
  - Probe values of a particular:
    - cell
    - point
    - location
  - Over time
- View and Display properties control what quantities are plotted

Bar Chart and Line Chart Views



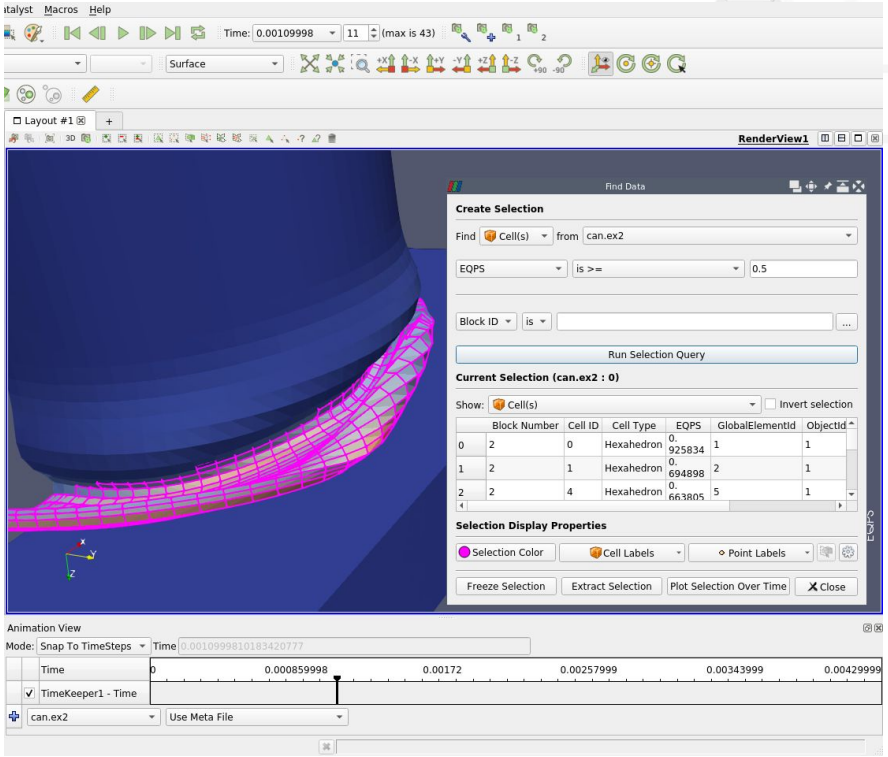
Spreadsheet View

- Shows data in **text** format
  - ! Read-only !
- Shows Point **Coordinates**
- Shows **Attribute** Values
  - Point
  - Cell
  - Field Data
- Streams small blocks of raw data at a time so does not overwhelm client



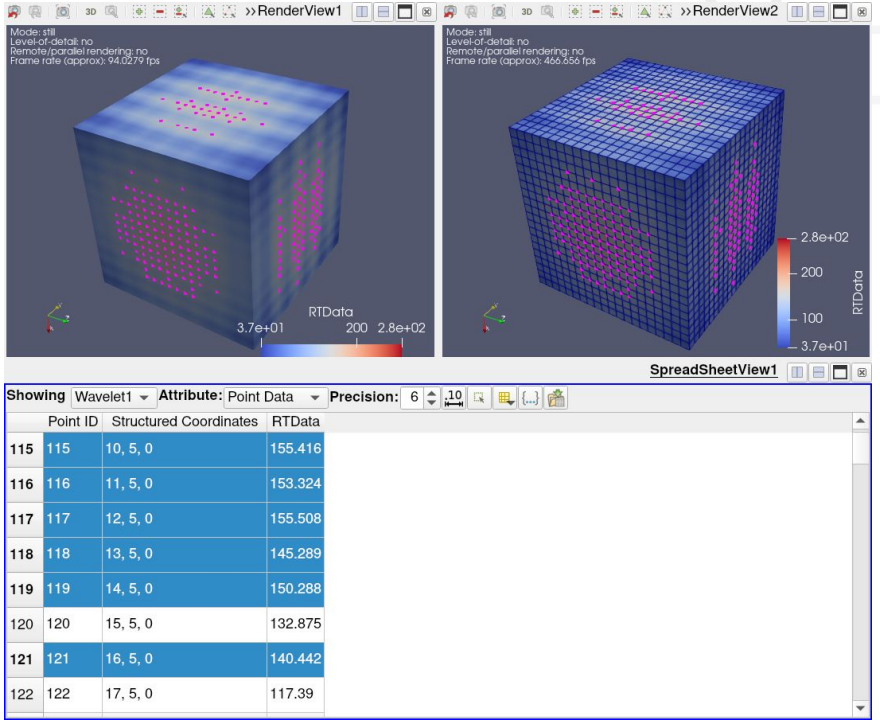
Selection

- A mechanism to **identify subset** of some dataset
- Why?
  - **Detailed** inspection of the subset
  - **Plot** subset over time
  - **Extracting** the subset
- Select with **queries**
  - *Edit->Find Data*
  - Enter characteristics you are searching for
  - Needs python and numpy



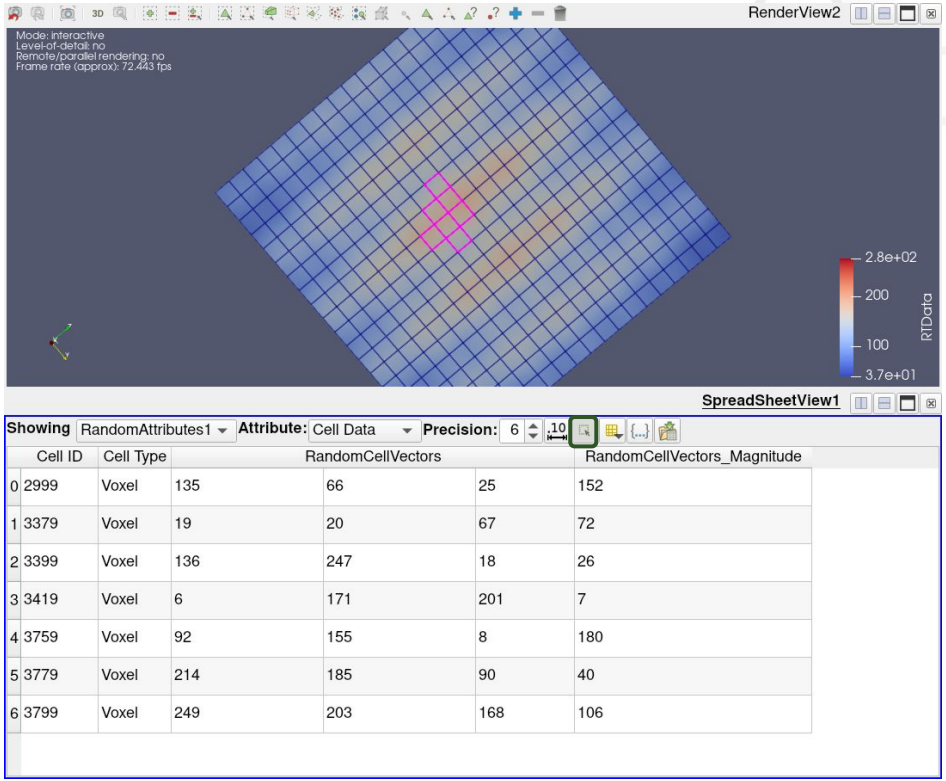
Selection Linking

- Active selection is shown in all views…
- Whenever possible (e.g., not in plot view)



Spreadsheet View

- Useful to view the **raw data** in selection
- Split an existing view and select “Spreadsheet view”
- Make sure the active filter is visible in the new view
- **Click to select** rows as one would in any spreadsheet application
- Check “**show only selected elements**”

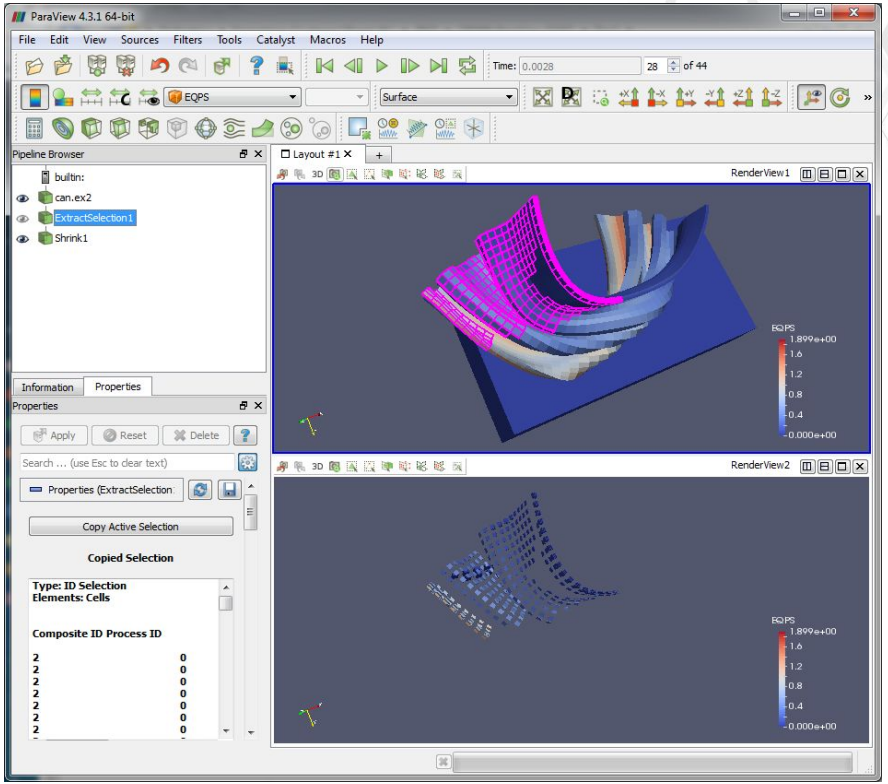


	Cell ID	Cell Type	RandomCellVectors			RandomCellVectors_Magnitude
0	2999	Voxel	135	66	25	152
1	3379	Voxel	19	20	67	72
2	3399	Voxel	136	247	18	26
3	3419	Voxel	6	171	201	7
4	3759	Voxel	92	155	8	180
5	3779	Voxel	214	185	90	40
6	3799	Voxel	249	203	168	106

Extract Selection

- Create a **dataset from selection**

- How to use:
  - Make a selection
  - Filters->‘Extract Selection’
  - Hit ‘Apply’
  - Update it with ‘Copy Active Selection’



Distributed Processing

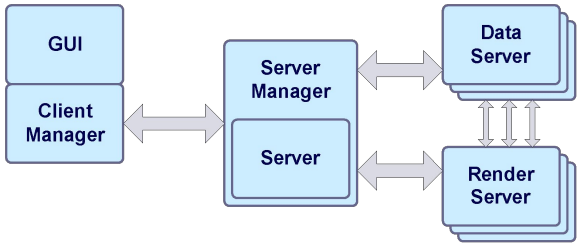
kitware

Distributed processing with ParaView

- What about large data visualization?
- **Distributed != Faster**
  - If data is **small enough** to process on **one machine**, running it on more machines probably won't make it faster
  - If data is **too large** the only way to run it is with **multiple machines**
  - With enough machines parallel interactive processing is feasible

Visualizing Massive Data

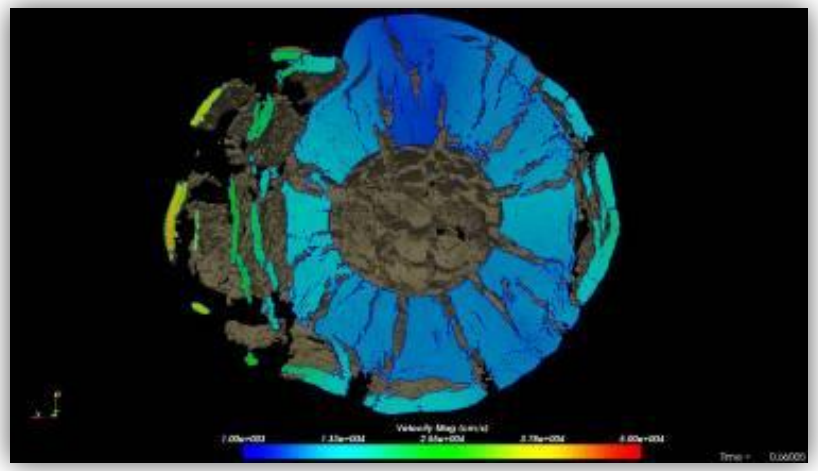
- Run data processing portion of ParaView (server) as a message passing parallel program (MPI) on a large cluster
- **Distribution = Data parallelism**
  - Server divides data, each of N processors gets 1/N'th (ideally)
  - Each processor runs identical processing pipeline
  - Result mesh is sent back to the client for local rendering OR image results are depth composited for local or remote display
- Run the front end ("GUI" or "Client") as normal but connect to remote server



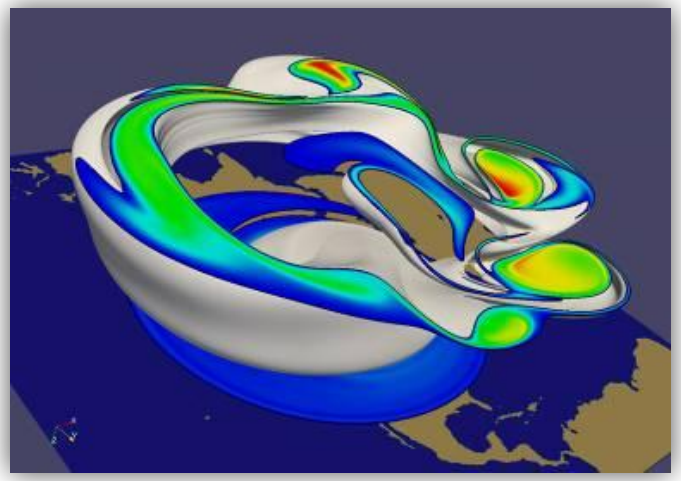
Extremely Large Data

Sandia National Laboratories

1 billion cell asteroid detonation simulation



½ billion cell weather simulation

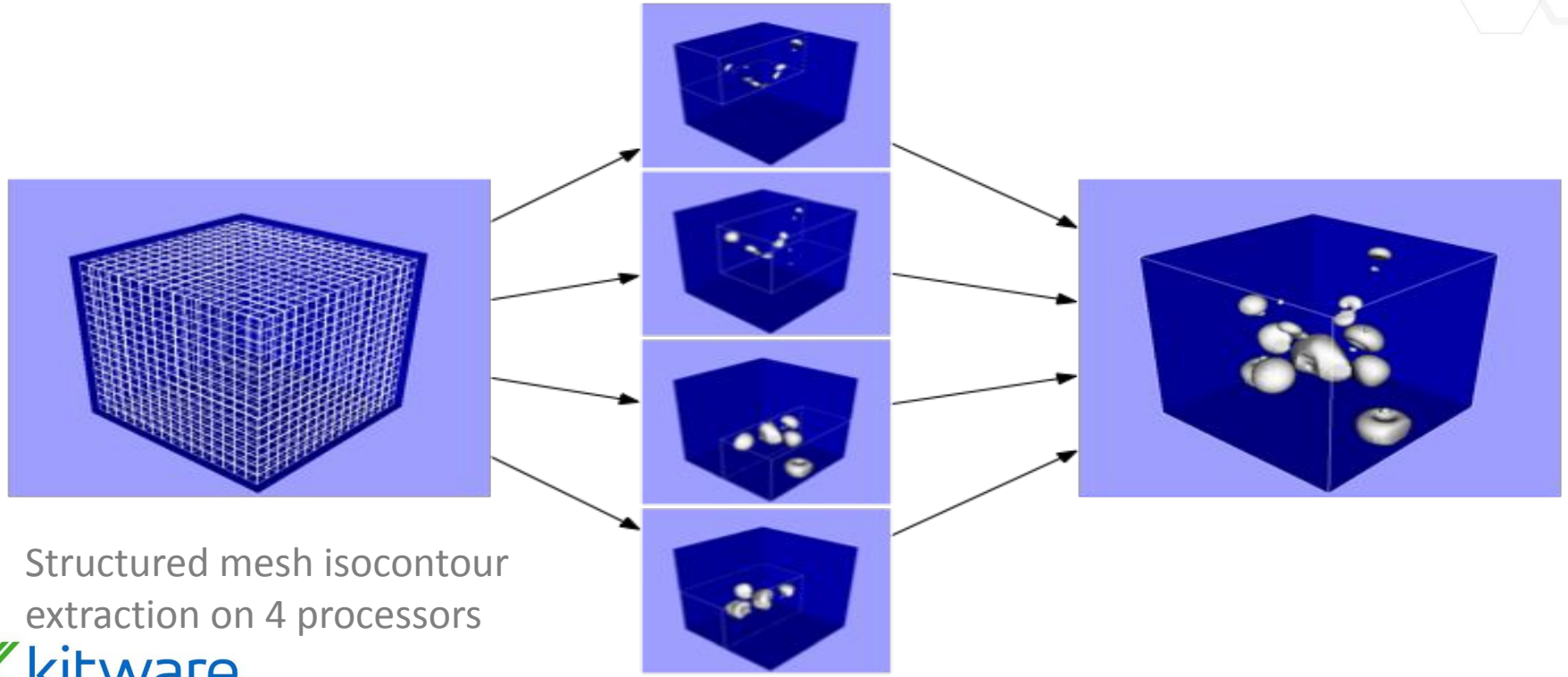


Source: Sandia National Lab

Data Distribution

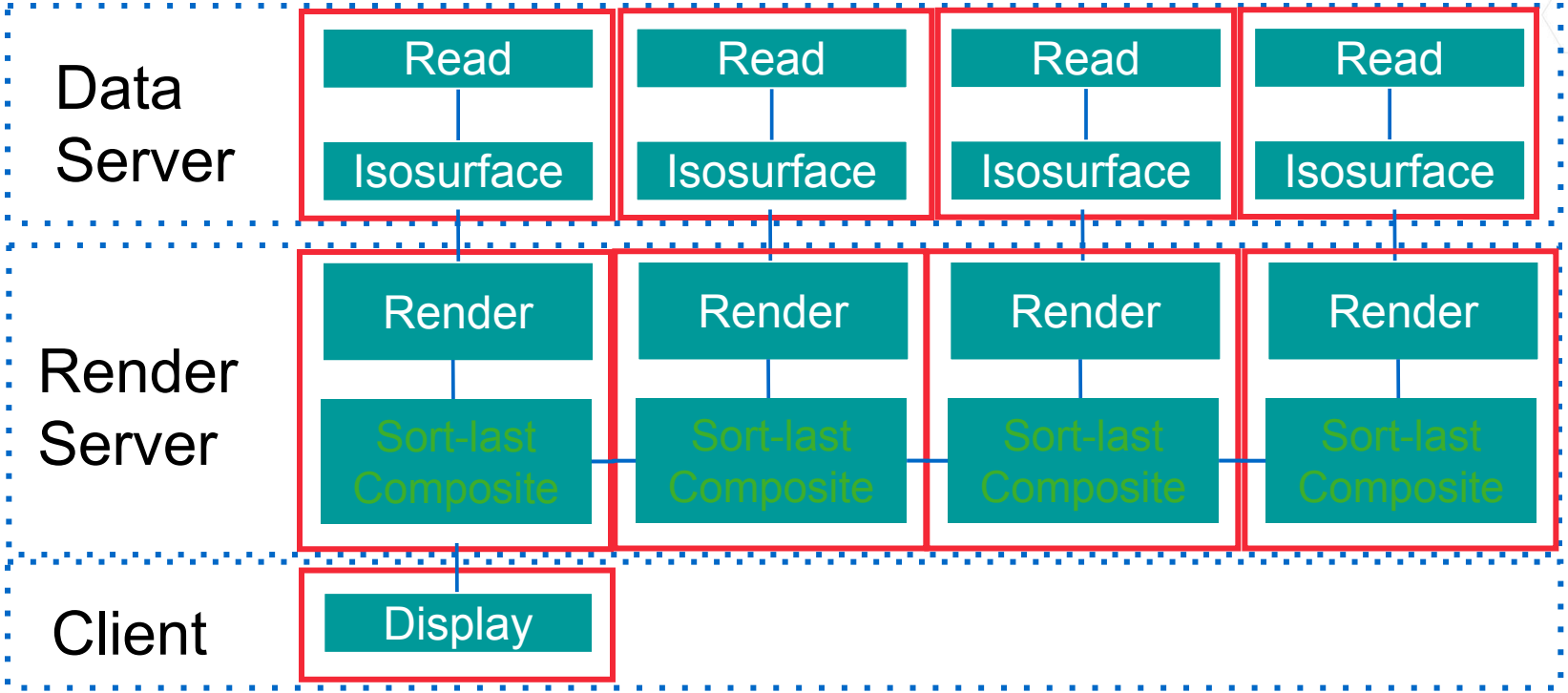
- Data is split across all processes
  - The full data model is never gathered on a single node
- Identical pipeline on all processes
- Sources/Readers are responsible for partitioning data
- Partitioning is automatic for structured data, based on data extents
- Filters can use MPI in execute methods
  - Most filters do not
- Repartitioning and load balancing filter is available, especially for unstructured datasets
  - D3 from Sandia (Legacy)
  - RedistributeDataSet
  - Distribute Point Cloud filter

Distributed Processing



Structured mesh isocontour extraction on 4 processors

Data Distributed Pipelines



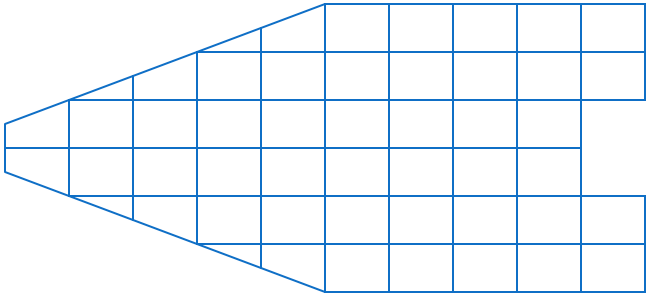
ParaView Components

- Client
  - Interface driving the visualization
- Data Server pvdataserver
  - Data processing – pipeline sources
- Render Server pvrenderserver
  - Data rendering – representations
- Server components can be run as MPI jobs, client cannot

Almost always combined into `pvserver`

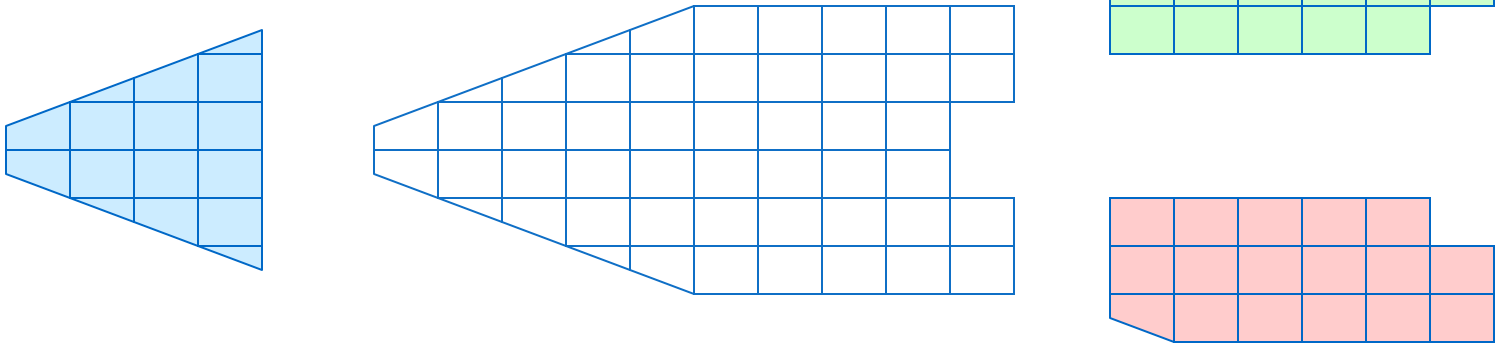
Data Distributed Pipelines

- Duplicate pipelines run independently on different partitions of data.



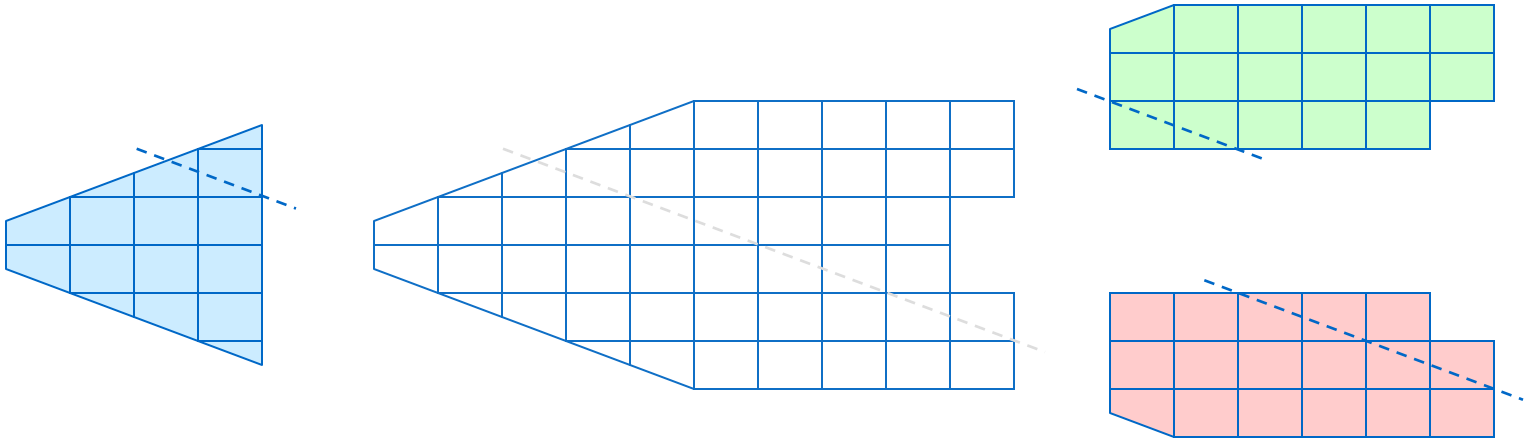
Data Distributed Pipelines

- Duplicate pipelines run independently on different partitions of data.



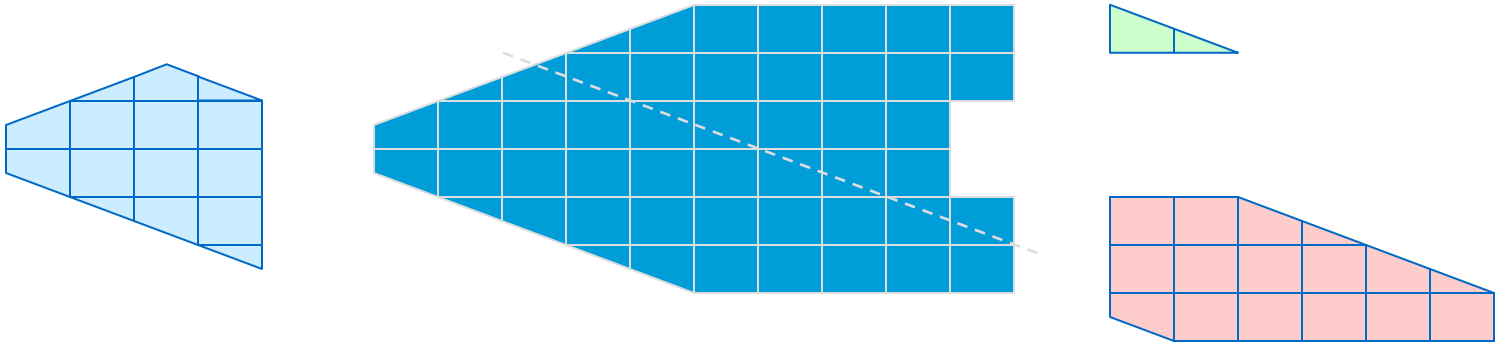
Data Distributed Pipelines

- Some operations will work regardless.
  - Example: Clipping.



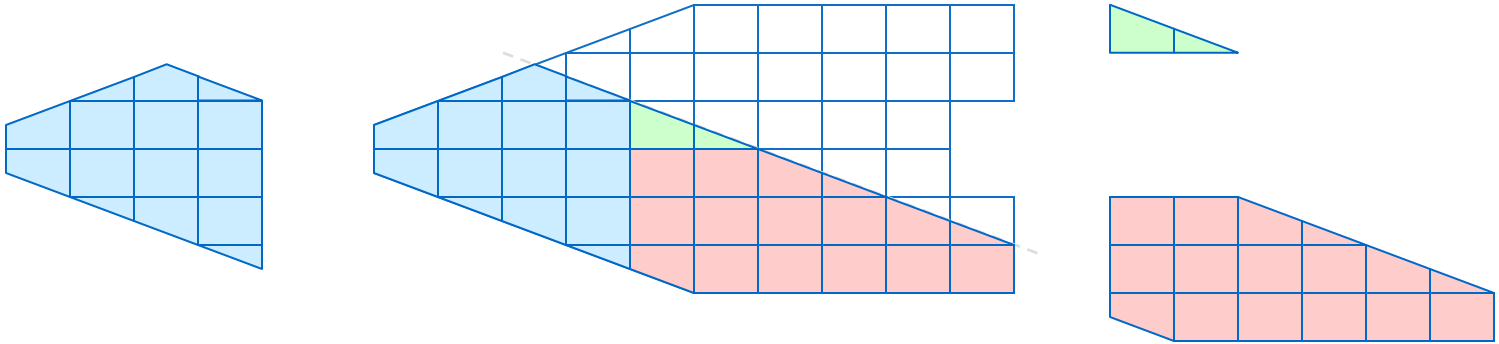
Data Distributed Pipelines

- Some operations will work regardless.
  - Example: Clipping.



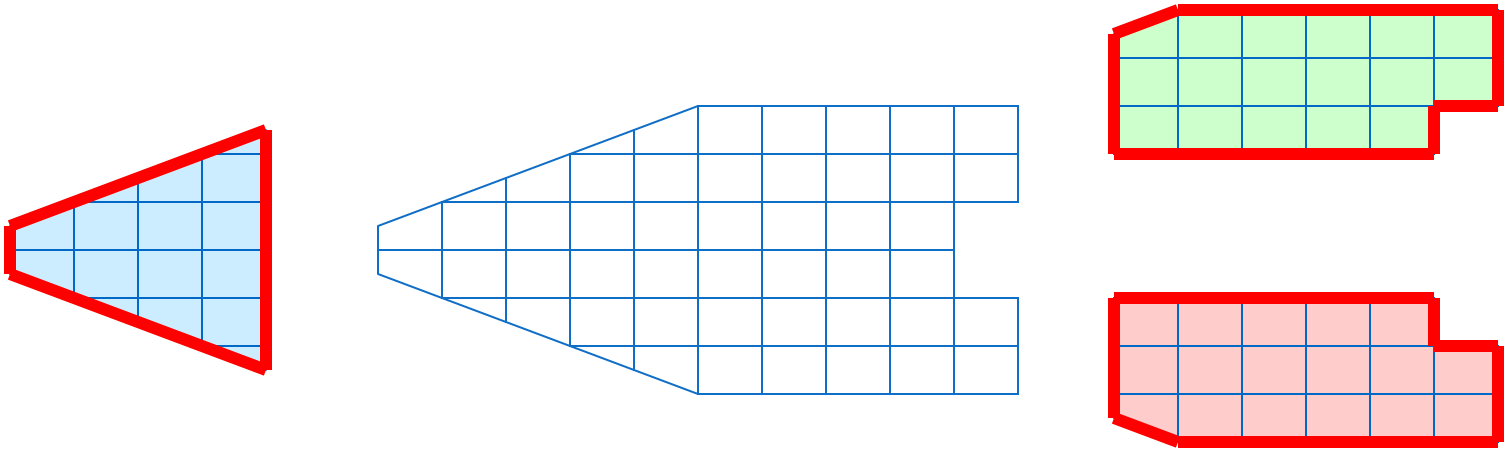
Data Distributed Pipelines

- Some operations will work regardless.
  - Example: Clipping.



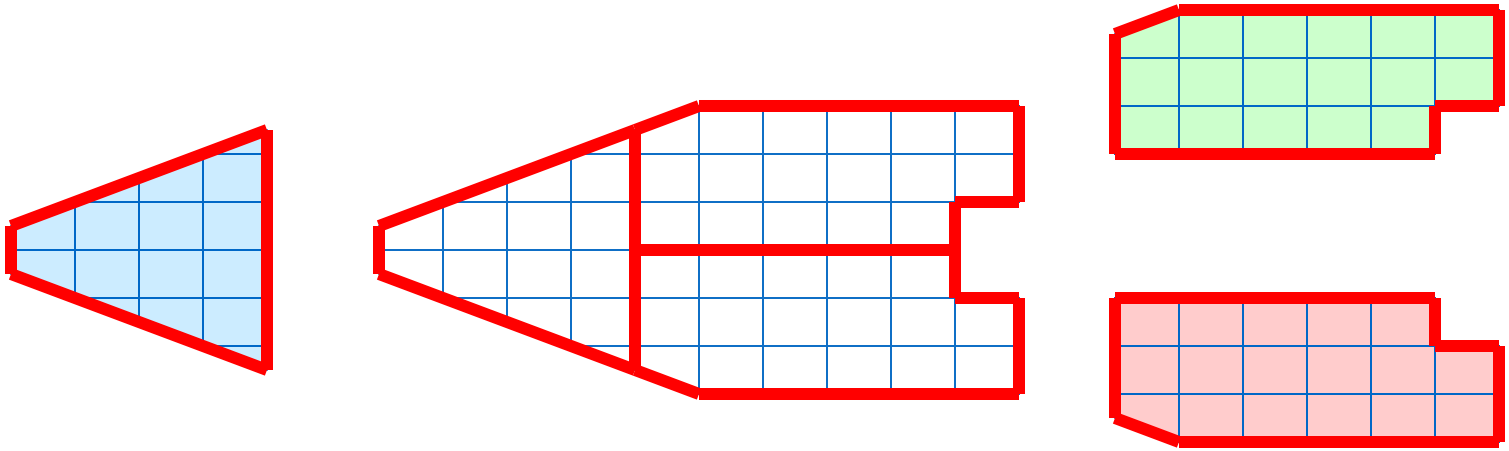
Data Parallel Pipelines

- Some operations will have problems.
  - Example: External Faces



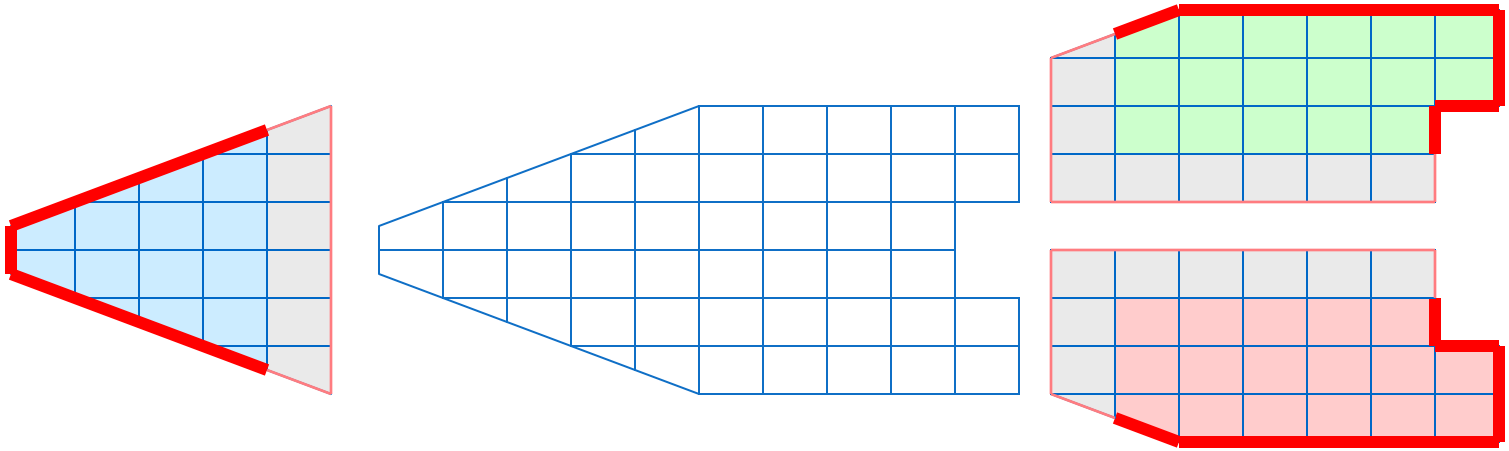
Data Distributed Pipelines

- Some operations will have problems.
  - Example: External Faces



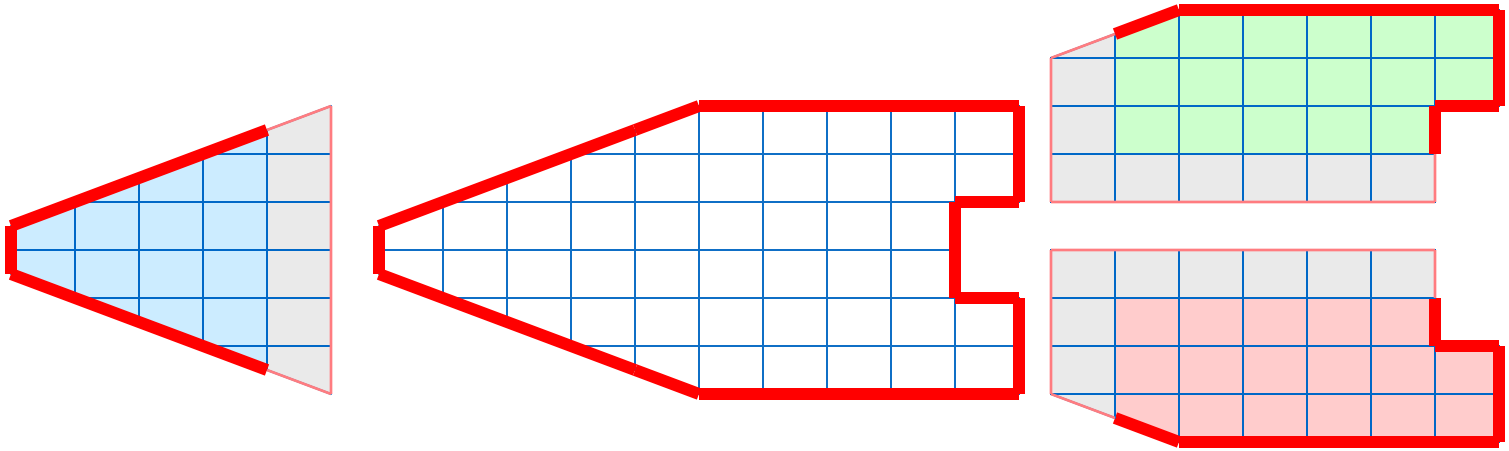
Data Distributed Pipelines

- Ghost cells can solve most of these problems



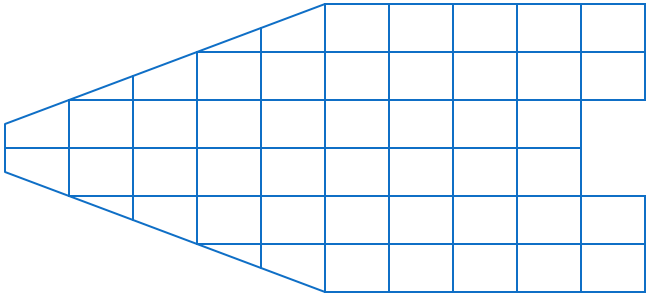
Data Distributed Pipelines

- Ghost cells can solve most of these problems



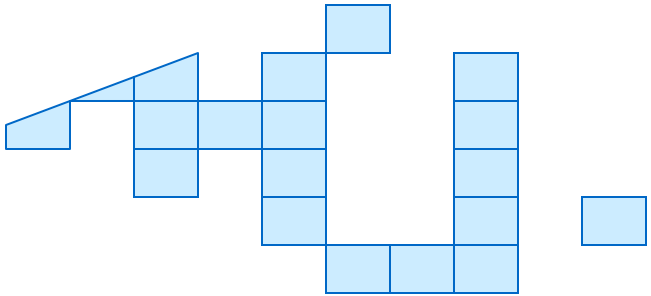
Data Partitioning

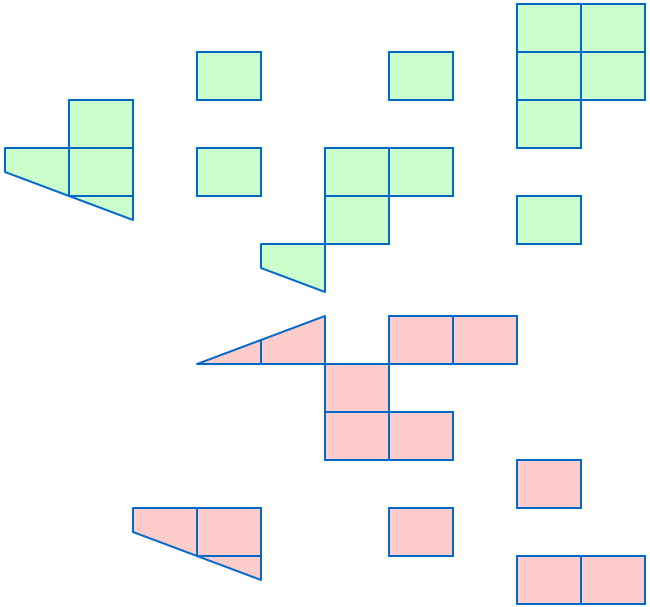
- Partitions should be load balanced and spatially coherent



Data Partitioning

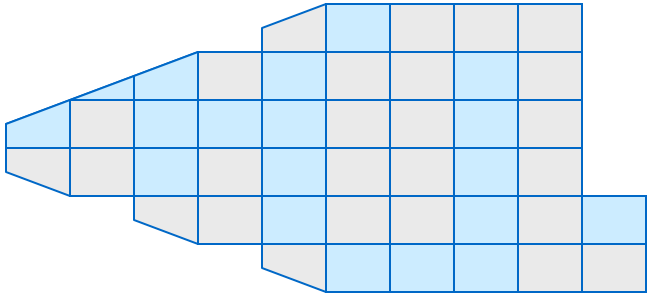
- Partitions should be load balanced and spatially coherent

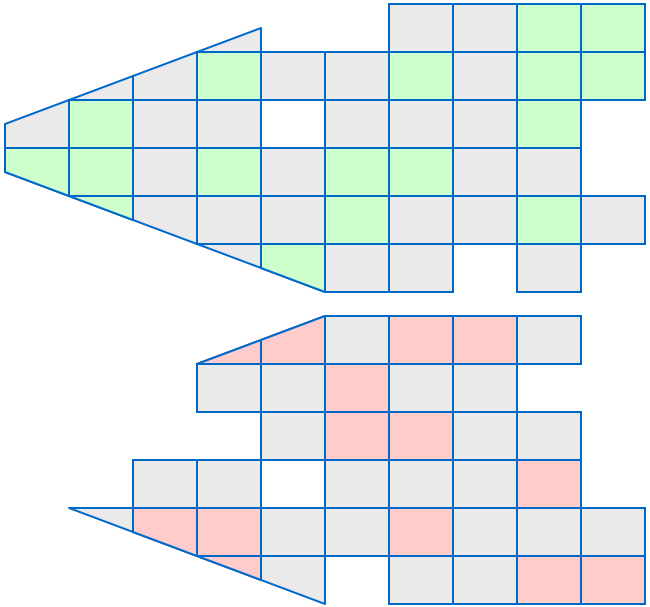




Data Partitioning

- Partitions should be load balanced and spatially coherent

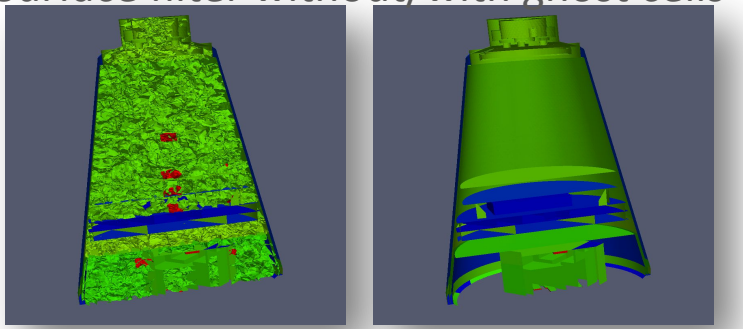




Load Balancing/Ghost Cells

- Automatic for Structured Meshes – global extents is split, readers ‘just’ have to support sub-extents requests
- Partitioning/ghost cells for unstructured is “manual”
- Use the **Redistribute Data Set** filter to redistribute data and compute ghost cells on unstructured grids
- Use the **Ghost Cells Generator** filter to compute ghost cells without redistributing the data

Surface filter without/with ghost cells



Distribute Data

Make sure that your data is distributed

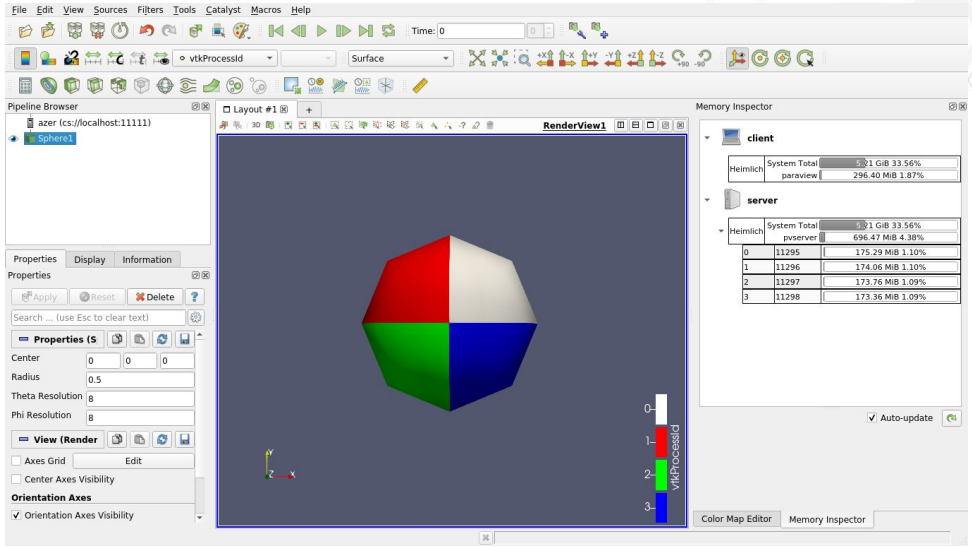
- Structured data will be automatically distributed
- Unstructured data will depend on the reader, see reference slide
- If the data is not distributed, use a distribution filter like **Redistribute Data Set**
- These filters are pass through in serial mode

Distributed Rendering

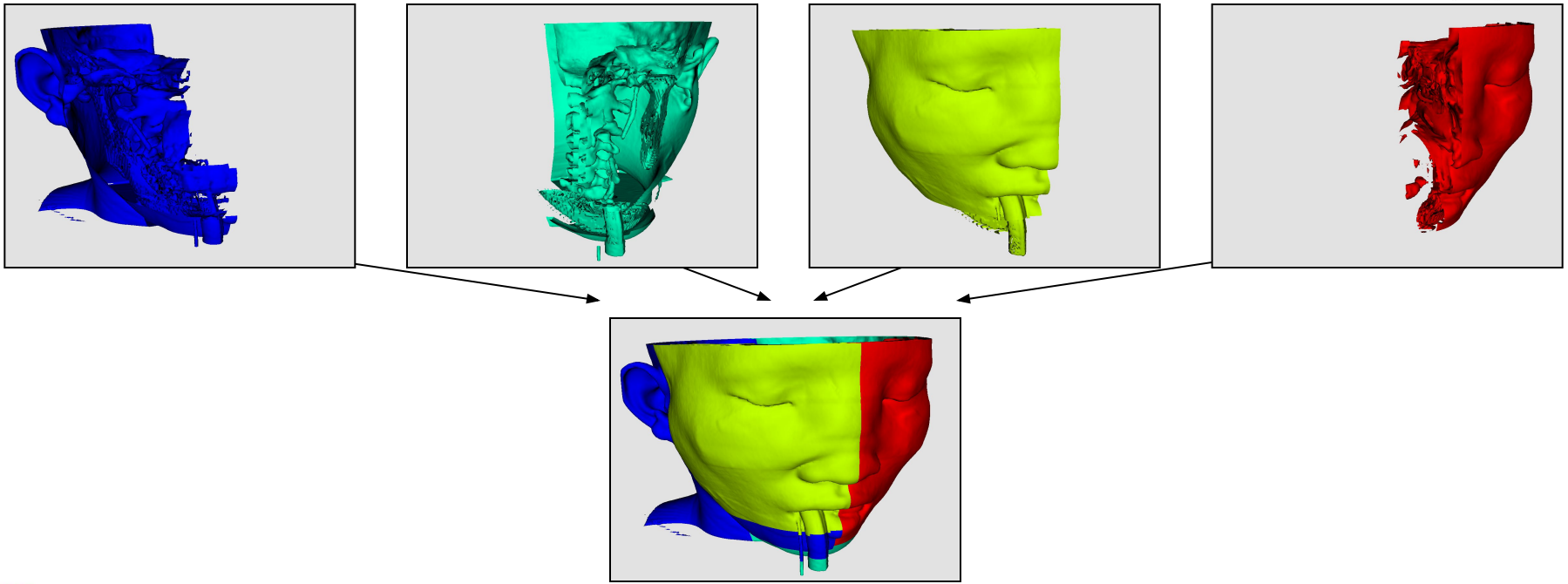
kitware

Compositing

- If data size < threshold
  - Client renders geometry locally
- If data size > threshold
  - N render server nodes render 1/N-th of all data over whole screen
  - Client receive images to display



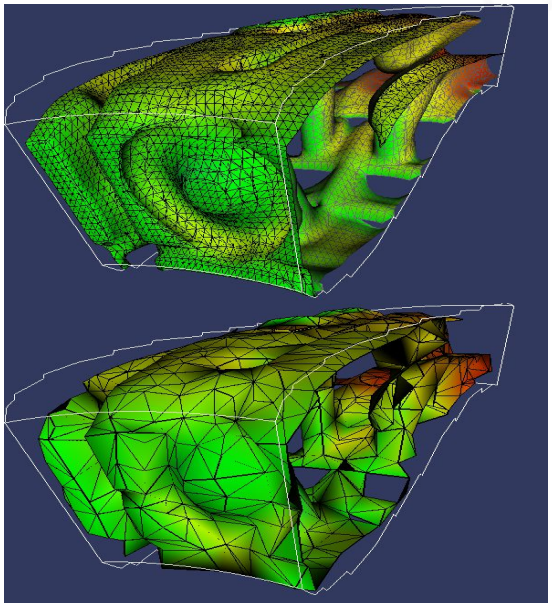
Parallel Rendering



Rendering Modes

- Still Render
  - Full detail render
  - Can be costly

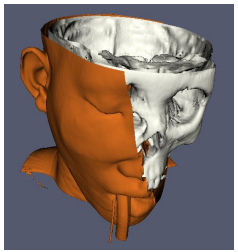
- Interactive Render
  - Used when interacting with 3D view
  - Sacrifices detail for speed
  - Provides fast rendering rate



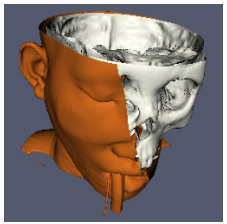
Still vs. interactive render

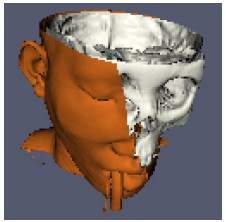
Subsampling – to maintain interactivity

- ParaView's parallel rendering overhead proportional to image size
- Can use smaller images for interactive rendering
- Image Reduction Factor



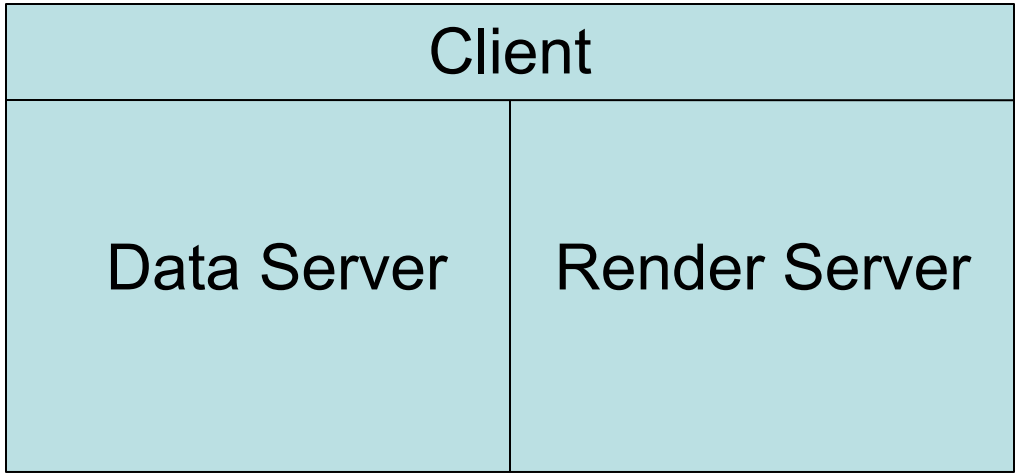
Original Data



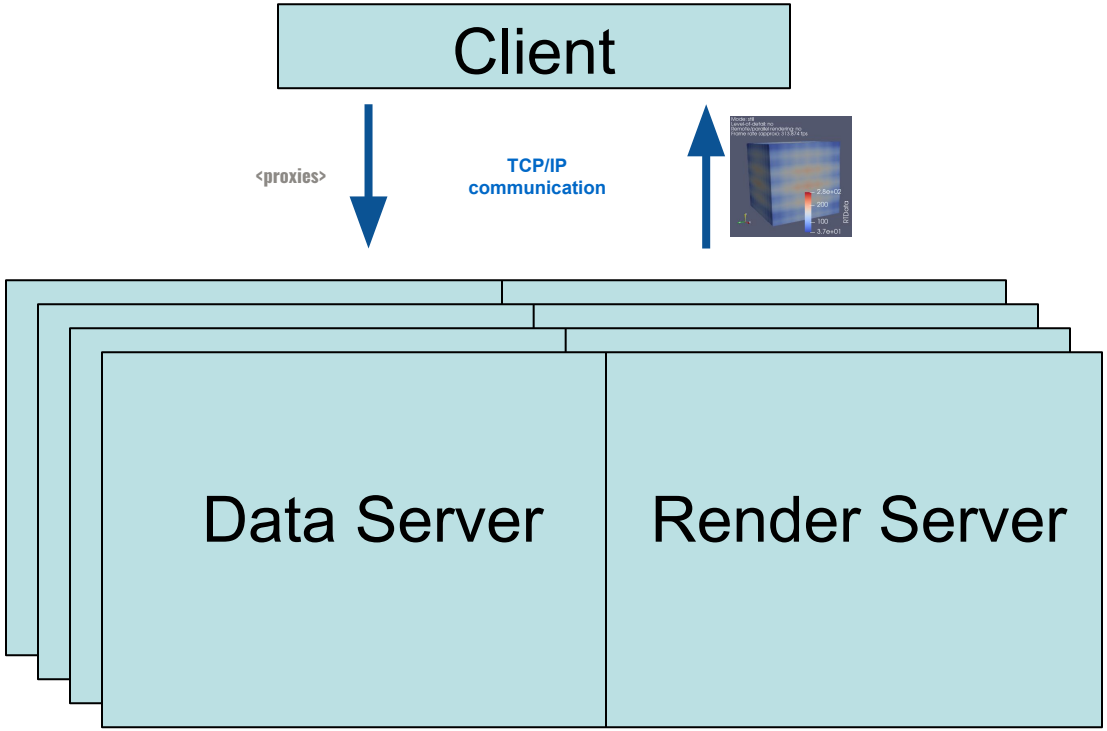


Subsample Rate: 2 pixels, 4 pixels, 8 pixels

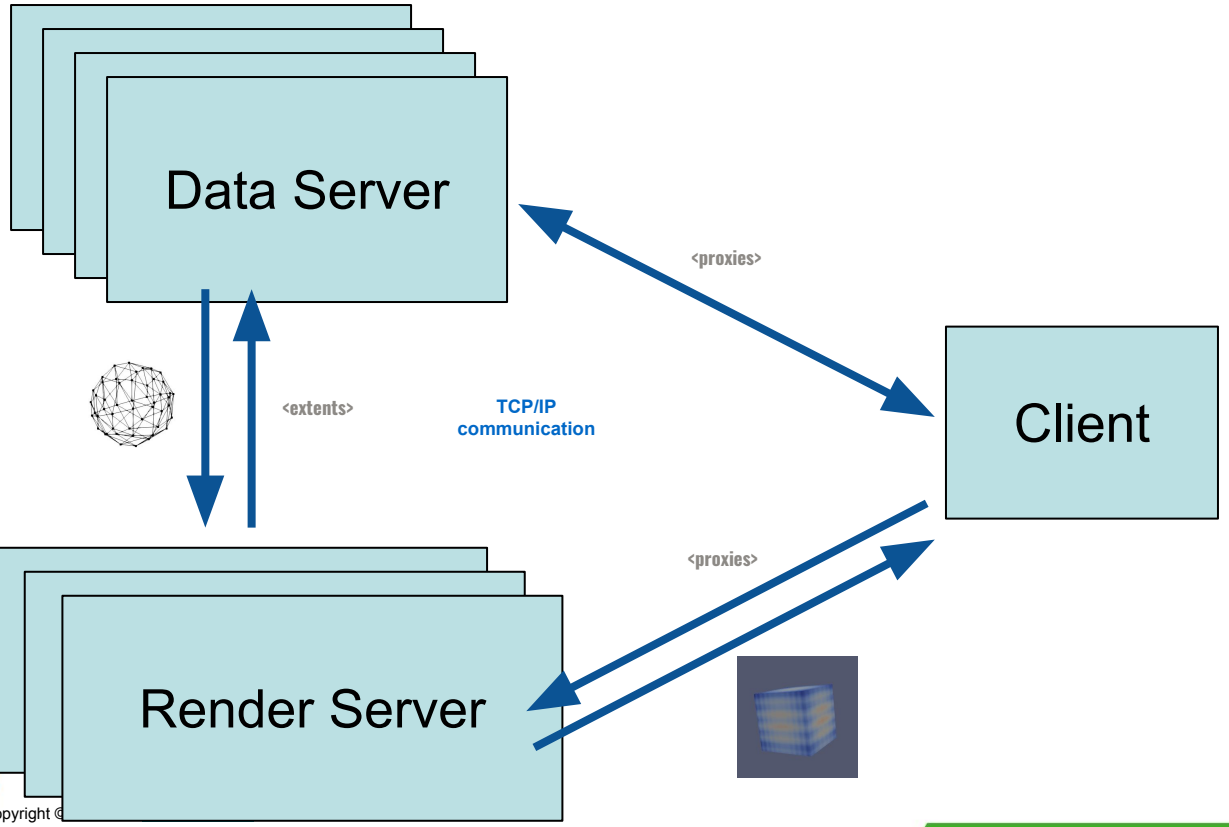
Standalone - Connect to Builtin Server



Client-Server



Client-Render Server-Data Server



Running a server

- Linux / OSX

```
> cd path/to/paraview/bin
> ./mpiexec -np 4 ./pvserver
```

- Windows

```
> cd path\to\paraview\bin
> .\mpiexec.exe -np 4 .\pvserver.exe
```

- Linux / OSX Compiled

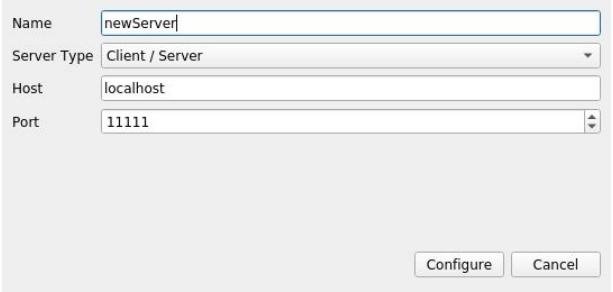
```
> cd path/to/paraview/bin
> mpirun -np 4 ./pvserver
```

- Windows Compiled

```
> cd path\to\paraview\bin
> mpiexec.exe -np 4 .\pvserver.exe
```

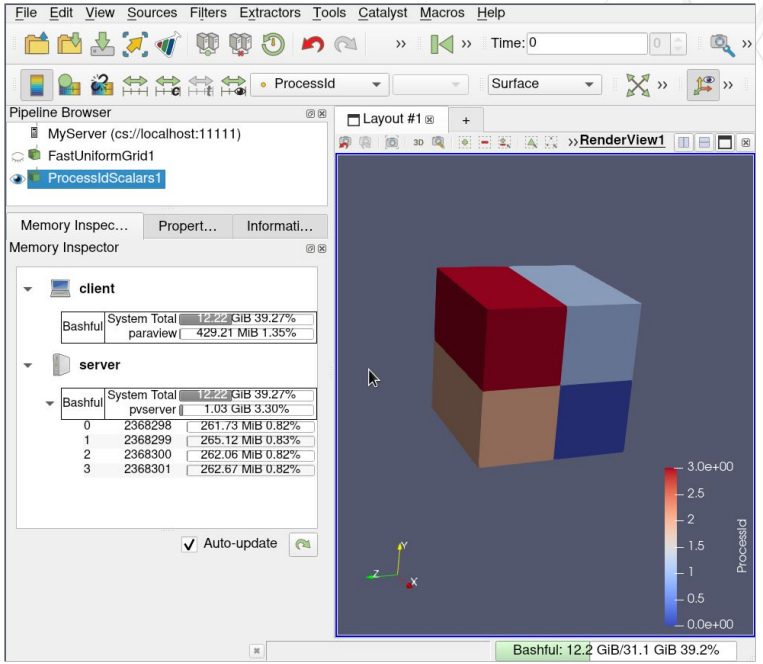
Configure server connection

- `File/Connect/Add Server`
- Name this connection to reuse it later
- Client/Server most common
- Host, Port = IP address of a machine to run pvserver on
- Startup. One of:
  - Command
    - a shell command to start pvserver on that machine
    - Ex. “ssh machine mpirun –np N pvserver”
  - Manual
    - If it is already running or you prefer to start it by hand



Connecting to a Server

- `File / Connect`
- Choose the connection you set up above
- When connected try “Process ID Scalars” filter. It shows which processor generated/owns what data

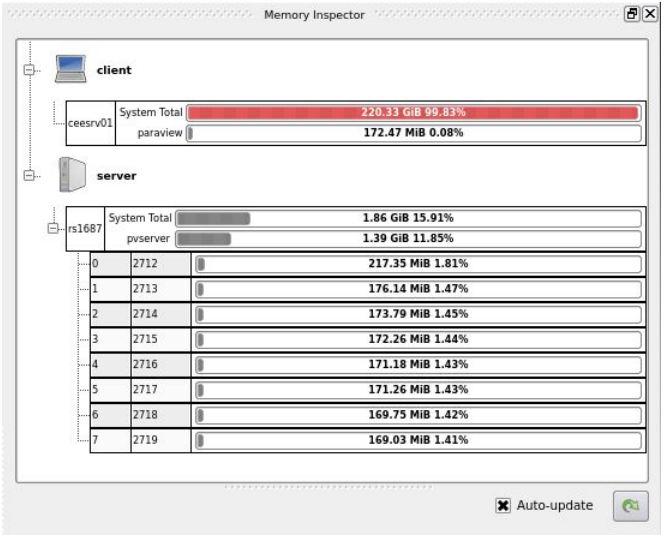


System Requirements

kitware

Memory Requirements

- Restricted to data that can fit in aggregate RAM
- Data parallelism replicates pipeline N times
- Each cluster node works on ~1/N-th
- Need at least as much as file size, plus enough for each filter's output
- Information Tab shows filter output size, but much of each filter's output is copied by reference of its input, so sum is <



Disk Requirements

- Each reader needs to see files
- Files shown in file browser are on server's file system
- Well written readers (ADIOS, Exodus, XDMF) read only local chunks
- "dumb" readers: only rank 0 reads the data
- Replication - works and minimizes contention but is a waste of disk space and preparation time
- NFS - better but has bottlenecks when all nodes read simultaneously
- Parallel file systems (e.g., PVFS, LUSTRE) typically have better performance, but are not perfect either

Display

- Multiple GPUs per node
  - Can be tricky to configure
  - Each GPU must have its own display
    - Eg. localhost:0.0 and localhost:0.1
  - For each process you will have to specify its display
  - Syntax to specify parameters for each MPI job is not part of specification and is implementation dependent
    - Example with OpenMPI (8 nodes with 2 GPUs)

```
> mpirun -bynode -np 8 /bin/env DISPLAY=localhost:0.0 ./pvserver : -np 8 /bin/env DISPLAY=localhost:0.1 ./pvserver
```

- Sharing GPUs amongst processes
  - Simply point multiple processes on the same host's display
  - Use off-screen-rendering flag to avoid windows/GRAM overlapping issues

```
> mpirun -np 32 ./pvserver -display :0.0 --use-offscreen-rendering
```

No Xorg server? No problem!

Without Xorg server on cluster, to still use the GPU, use EGL

- EGL – Embedded-System Graphics Library
  - Interface developed by Khronos, the same group that created OpenGL
  - Available in a dedicated release for linux ! Just use pvserver as usual !
  - For specific options, compilation is needed: https://kitware.github.io/paraview-docs/latest/cxx/Offscreen.html
  - To control it, use cli option `--displays --egl-device-index=` or env var VTK_DEFAULT_EGL_DEVICE_INDEX

OpenGL Vendor	NVIDIA Corporation
OpenGL Version	4.6.0 NVIDIA 470.57.02
OpenGL Renderer	NVIDIA GeForce GTX 1660/PCIe/SSE2
Headless support	EGL

No GPU? No problem!

Without GPUs on cluster, use OSMesa

- OSMesa – Off-Screen API of Mesa 3D Graphics Library
  - Available as a dedicated binary release for linux ! Just use pvserver as usual.
  - For windows or specific linux setup (llvmpipe is the most efficient), compilation is needed:
    http://www.paraview.org/Wiki/ParaView/ParaView_And_Mesa_3D
    https://kitware.github.io/paraview-docs/latest/cxx/Offscreen.html

OpenGL Vendor	VMware, Inc.
OpenGL Version	3.3 (Core Profile) Mesa 18.2.2
OpenGL Renderer	llvmpipe (LLVM 7.0, 128 bits)
Headless support	OSMesa

If all else fails

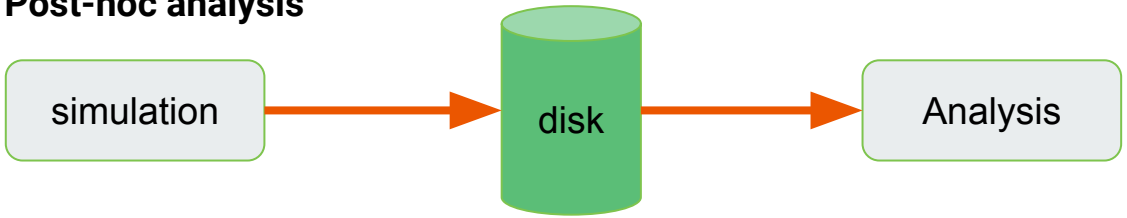
- `pvserver` can still be used
  - Will automatically disable the render server
  - Expect a warning on connection
  - Client will do the rendering
  - Communication overhead might increase

Catalyst

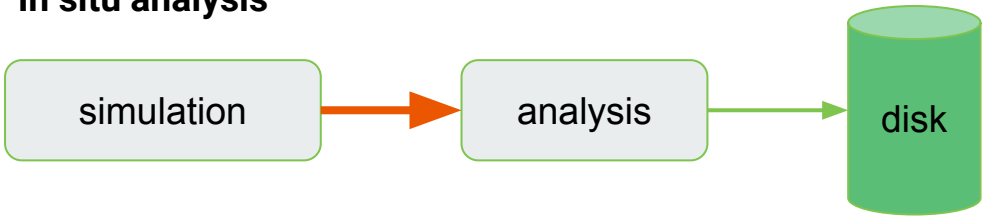
kitware

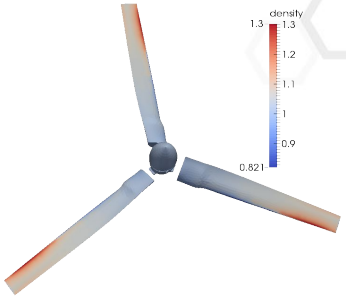
Catalyst and ParaView

Post-hoc analysis



In situ analysis





- Rotorcraft simulation, per timestep
  - Full data set – 448 MB
  - Surface of blades – 2.8 MB
  - Image – 71 KB

ParaView and Catalyst

Steps to use Catalyst

- Augment simulation with a few calls into Catalyst library
- Describe data structures using Conduit specification
- Link against stub Catalyst library that is easy to build
- At run time, set environment variable to point to ParaView's Catalyst implementation

Steps to write analysis algorithms

- Write ParaView or VTK code (more challenging)
- Use ParaView to generate Catalyst scripts that ParaView will execute during a run (easier)

Going Further ...

- ParaView.org
  - General information about ParaView and links to additional resources
- ParaView User Doc (Guide) – Official user's manual and reference guide
  - Accessible in the binary version of ParaView
  - Freely available as a website: https://docs.paraview.org
  - Printed version on Amazon
- Forum
  - Plenty of user and developer resources
  - https://discourse.paraview.org/