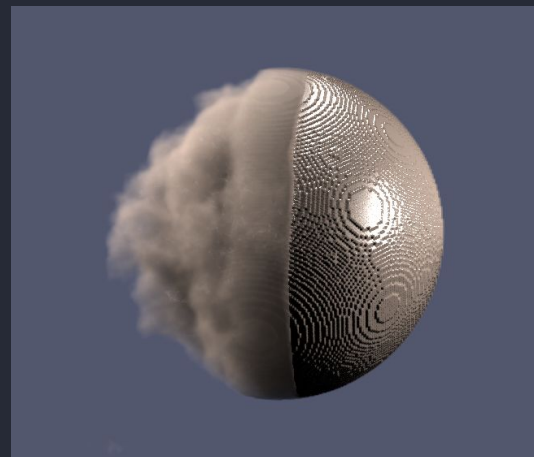


Large Scale Visualization with ParaView

Cory Quammen and Patrick Avery

Outline

- ◆ **Introduction to ParaView**
- ◆ **ParaView Demo**
- ◆ **Visualizing Datasets with Distributed Memory Parallelism**
- ◆ **Python scripting in ParaView**
- ◆ **In situ Computing with ParaView**
- ◆ **ParaView + AI**



Volumetric Rendering in VTK and ParaView:
Introducing the Scattering Model on GPU

Kitware / Leader in AI & scientific open source solutions

Software and AI R&D Services

Customers in government, industry, and academia
300+ active projects worldwide



Sustained Growth

100% employee-owned
Over \$50 million revenue



200 employees Worldwide

6 offices across USA/Europe
NY, NC, NM, VA, MN and France



Deep Expertise

Strong academic reputation
65% staff hold a graduate degree



25+ years of expertise

Kitware USA, 1998
Kitware Europe, 2010

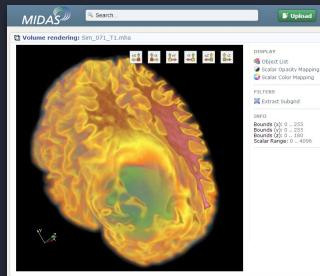


Founded on Open Source

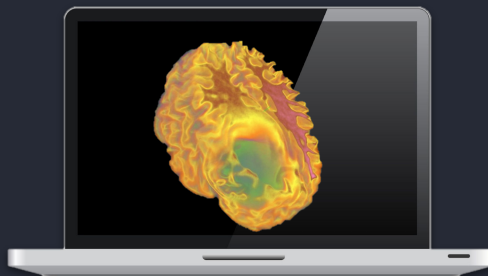
Vibrant, enduring platform/communities
Strong commitment to Open Science



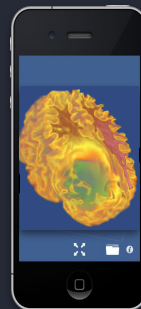
Applications / Universal Platforms



Web



Desktop



Mobile



Cloud /HPC

kitware
Platforms



3D Slicer



ParaView



KWIVER



TeleSculptor



tomviz



LidarView



Slicer SALT
Shape Analysis Toolbox



Resonant



VolView



Pulse
Physiology Engine



CMake



trame



CMB



VTK



ITK

Kitware – HPC and Visualization

VTK



ParaView



CMB



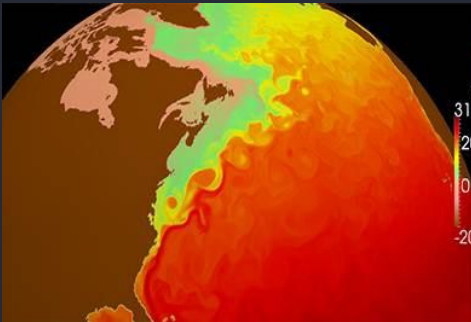
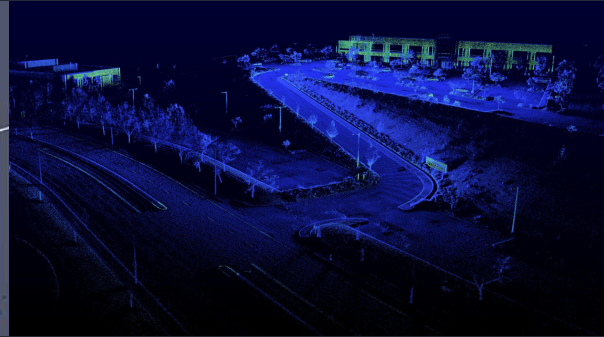
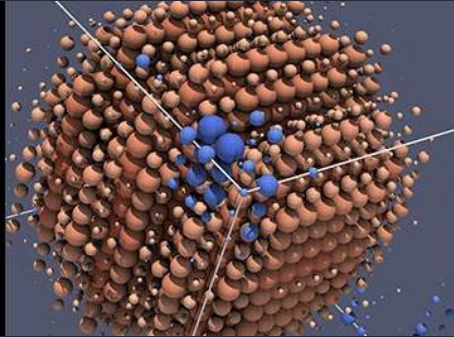
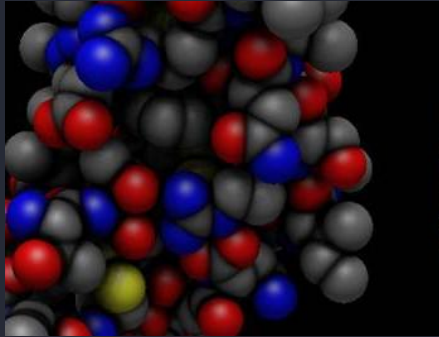
tomviz



LidarView



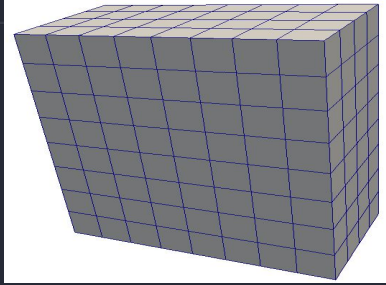
Trame



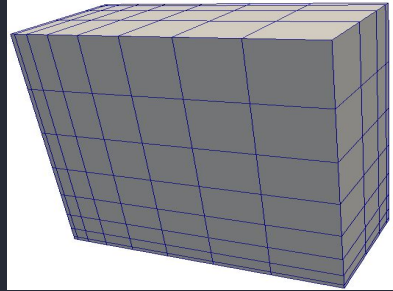
What is ParaView?

- An open-source (BSD 3 Clause License), **scalable**, multi-platform visualization application based on VTK
- Processing paradigms:
 - distributed computing (MPI)
 - shared memory multiprocessing (SMP) (vtkSMPTools)
 - GPU processing (viskores).
- Has an open, flexible, and intuitive user interface
- Has an **extensible, modular architecture**

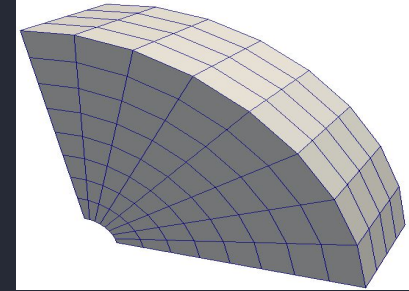
ParaView (VTK) Data Types



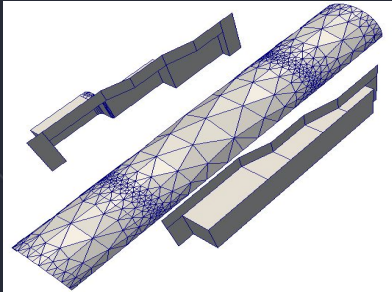
Uniform Rectilinear
(vtkImageData)



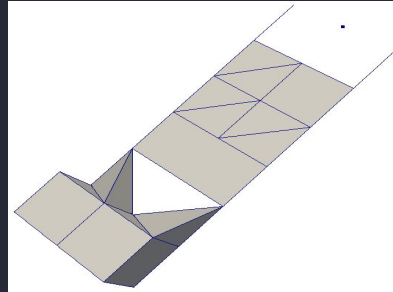
Non-Uniform Rectilinear
(vtkRectilinearData)



Curvilinear
(vtkStructuredData)



Polygonal
(vtkPolyData)



Unstructured Grid
(vtkUnstructuredGrid)

- Partitioned Dataset
- Partitioned Dataset Collection
- Adaptive Mesh Refinement (AMR)

Common Filters



Calculator



Contour



Clip



Slice



Threshold



Extract Subset



Glyph



Stream Tracer



Warp (vector)



Group Datasets



Extract Block

ParaView Demo

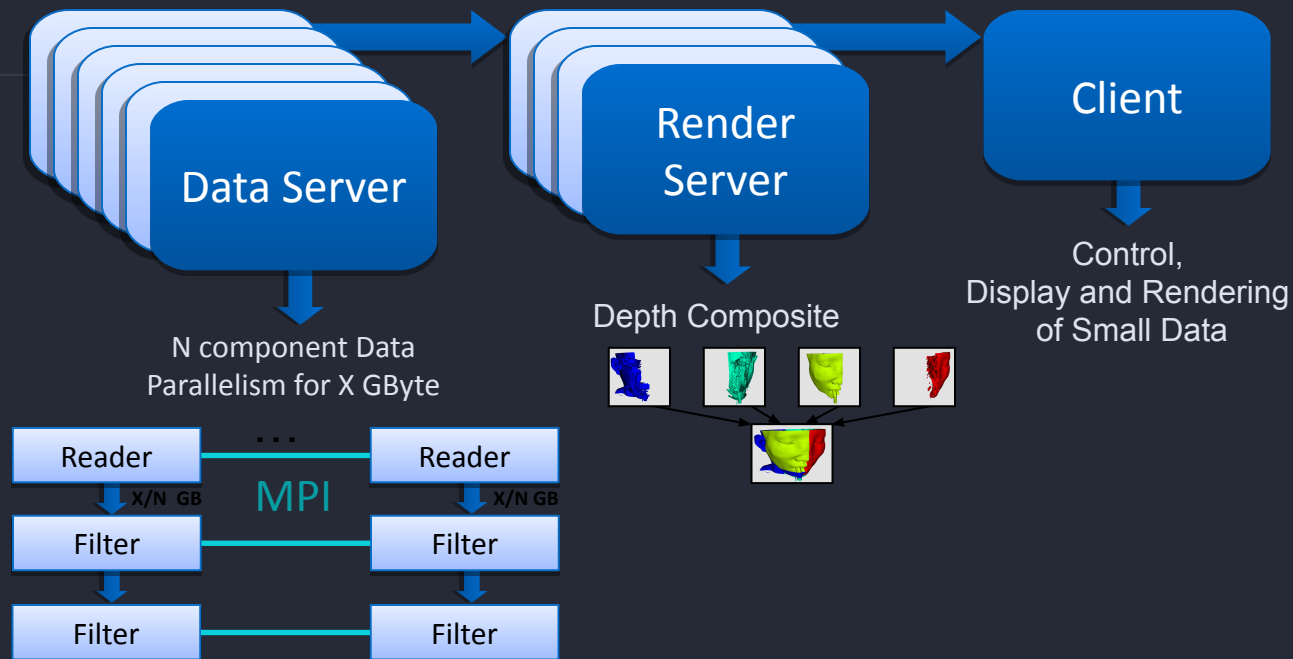
Visualizing Large Datasets with Distributed Memory Parallelism

ParaView's Running Modes

Builtin aka Standalone aka Serial		all components within one process (client may be GUI or pvpython) <code>paraview</code> or <code>pvpython</code>
Combined Server		data processing and parallel rendering in MPI job of combined processes. control from TCP connected client. <code>mpiexec -n x pvserver &; paraview #</code> or <code>pvpython # + Connect</code>
Batch		server is an MPI job which directly runs a python script <code>mpiexec -n x pvbatch \</code> <code>vis_script.py</code>

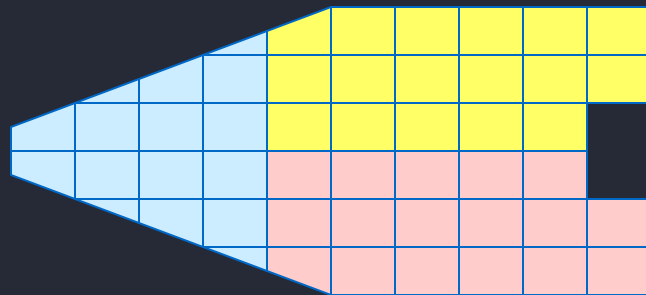
DS = data server

RS = render server



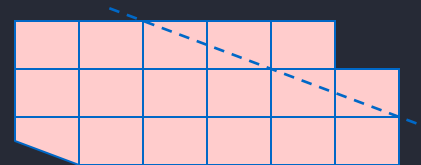
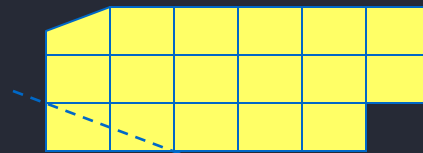
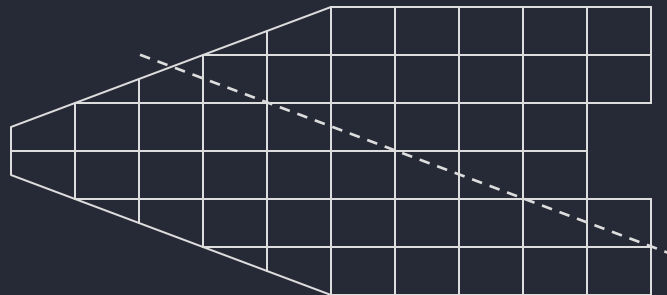
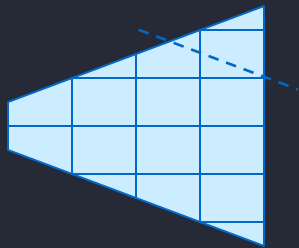
Data Parallel Pipelines

- Duplicate pipelines run independently on different partitions of data.



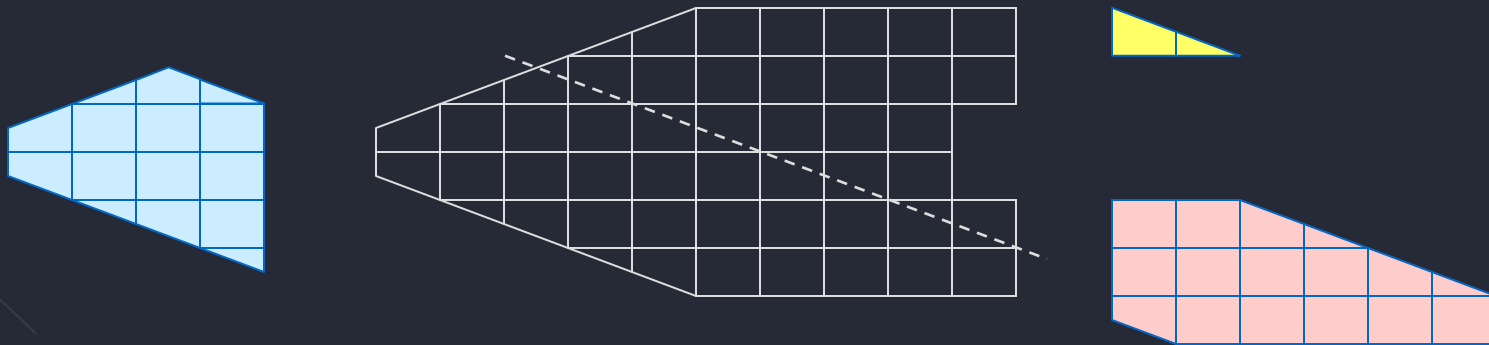
Data Parallel Pipelines

- Many operations will work regardless
 - Example: Clipping



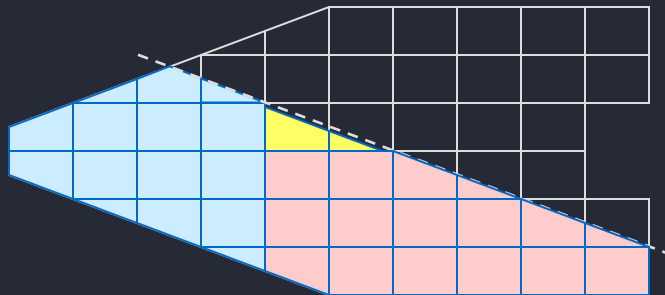
Data Parallel Pipelines

- Many operations will work regardless
 - Example: Clipping



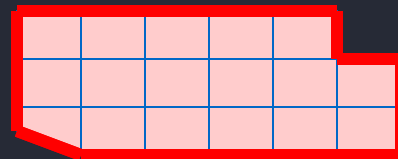
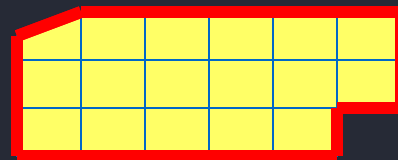
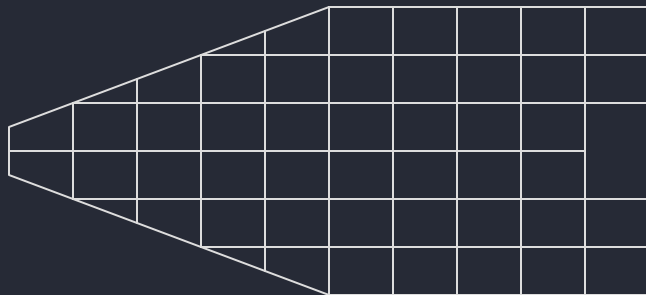
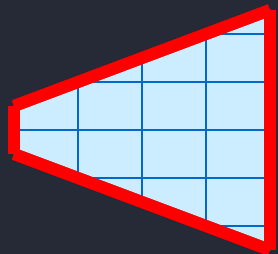
Data Parallel Pipelines

- Many operations will work regardless
 - Example: Clipping



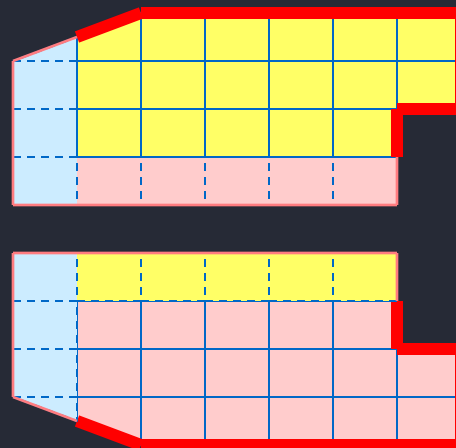
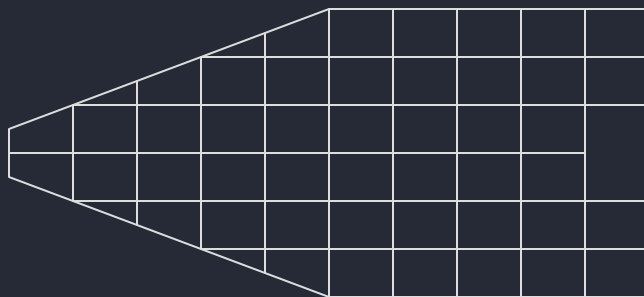
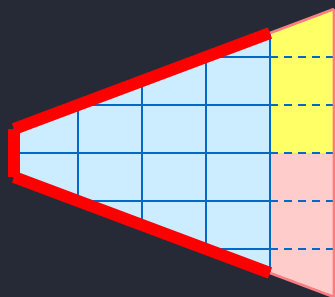
Advanced Data Parallel Pipelines

- Some operations will have problems
 - Example: External Faces



Advanced Data Parallel Pipelines

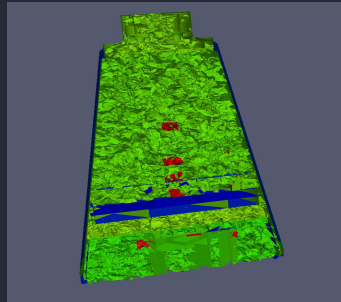
- Ghost cells can solve most of these problems



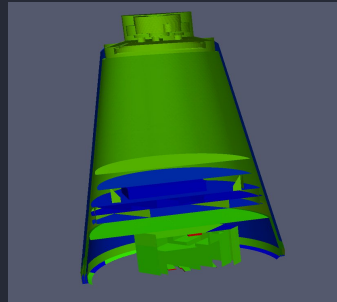
Ghost Cells filter

- Automatic when reading structured data
- Ghost Cells** filter: creates ghost cells (if data is partitioned on across ranks)

Extract Surface
without ghost cells



Extract Surface
after Ghost Cells
filter applied



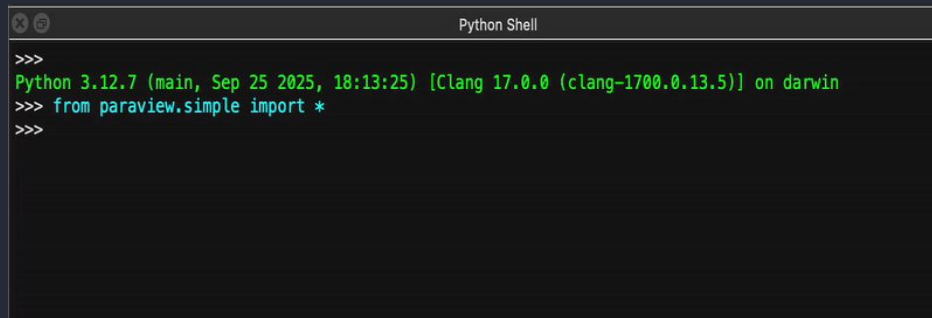
Python Scripting in ParaView

Python scripting of ParaView

- **ParaView is fully scriptable with Python**
 - Nearly all operations you can do in the ParaView GUI can be scripted instead
- **Opens up possibilities for automated analysis and visualization**
 - Run scripts instead of manually pointing and clicking
 - Increases reproducibility
 - Code generation with LLMs to drive visualization

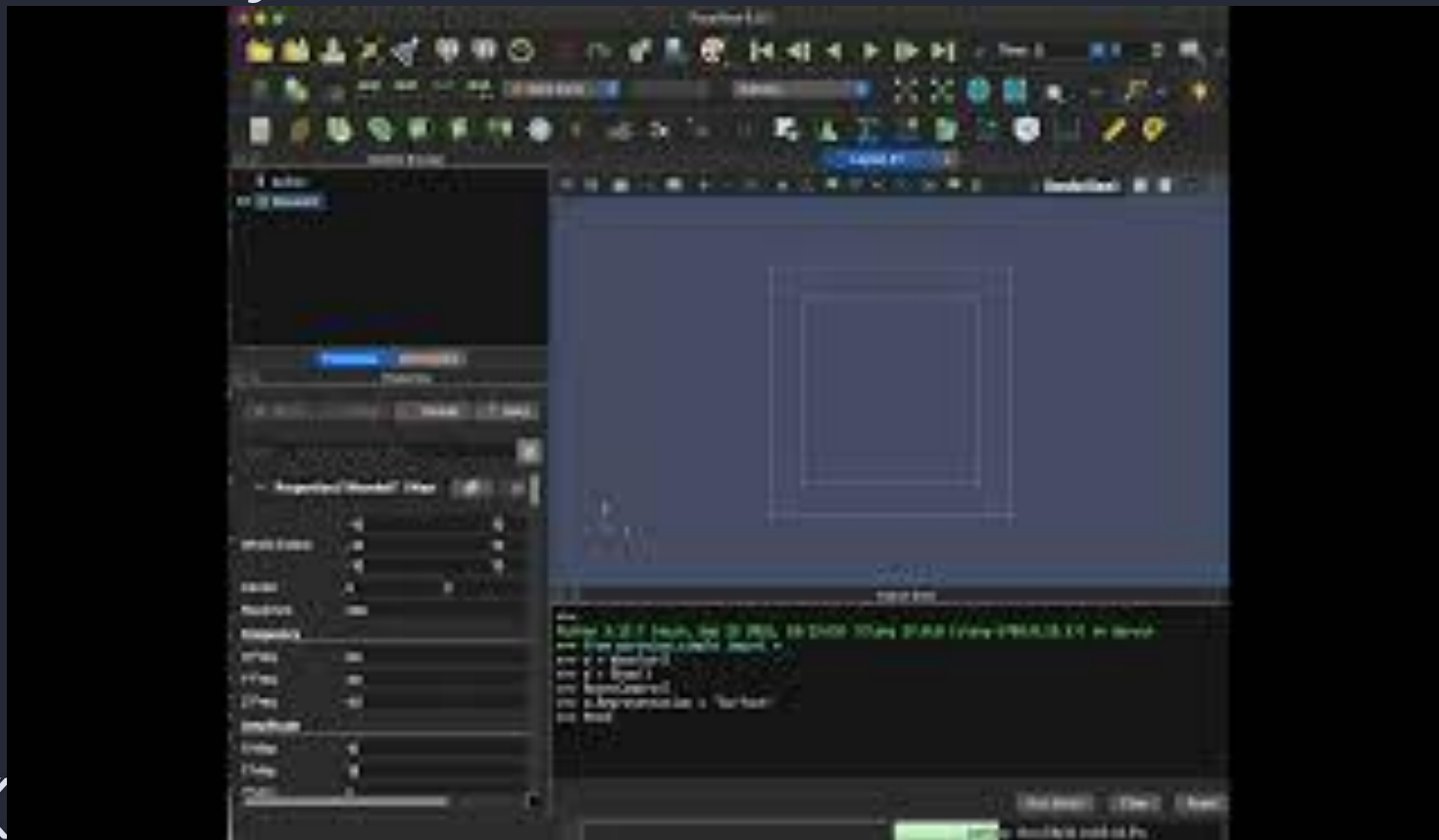
Python Shell

- The *Python Shell* is an interactive Python console in the ParaView GUI
 - Accessible through the **View** menu
-> **Python Shell**
- Can type in Python code or run Python scripts from a file with “Run Script” button
- Autocomplete available
- Buttons to “Clear” the shell and “Reset” the environment available



```
>>>
Python 3.12.7 (main, Sep 25 2025, 18:13:25) [Clang 17.0.0 (clang-1700.0.13.5)] on darwin
>>> from paraview.simple import *
>>>
```

Python Shell link to the ParaView GUI



Commands
executed in
Python Shell
update the
ParaView GUI



Getting help in the Python interpreter

- **help(paraview.simple)**
 - Lists all functions that paraview.simple gives you
 - Including all sources/filters you can create
- **help(Connect)** #paraview.simple prefix is implied
 - Ask for help on any particular function directly
- **help(Clip)**
 - Gives top level information about Clip (the filter itself)
- **help(myClip)**
 - Gives more details (ex properties you can call) when you give it a particular Clip object
- **dir(myClip)**
 - compact list of available functions and sometimes more complete alternative

Python Tracing

- Tools menu > **Start Trace**
- Build visualization pipeline with UI
- Tools menu > **End Trace**
- Save Python script - all actions recorded as a runnable Python script



In Situ Analysis at Scale with Catalyst

Compute has outrun the I/O stack

~3 TB/s

HBM3 on-device bandwidth

where your simulation data already
lives

~32 GB/s

PCIe host transfer rate

the rate at which you can move it
off-device

~100x

bandwidth discarded

every time you stage device memory
to host

In situ is not a feature request. It is the only way to keep pace with the data rate. The pipeline has to live where the data already is.

A lightweight framework beside the simulation

WHAT IT IS

A small C API that hands a simulation's data to analysis pipelines while it is still running. No intermediate file write.

HOW IT FITS

catalyst_initialize once, **catalyst_execute** each step, **catalyst_finalize** at the end. In-process by default, in-transit through the same Conduit data model.

DATA DESCRIPTION

Data arrays mapped to Blueprint conventions for Conduit.

SOLVER your code

CATALYST C API + Conduit Blueprint

PIPELINE VTK / Viskores filters

BSD-3 licensed. Available since 2014. Backwards-compatible C ABI since v2.

Deployed on leadership-class HPC, used by ECP applications and national labs.

ParaView Catalyst

Catalyst-instrumented code



Conduit



ParaView Catalyst
implementation



ParaView visualization
pipeline in Python

ParaView + AI

Plain English in, validated ParaView analysis out

NATURAL LANGUAGE, ONE TOOL

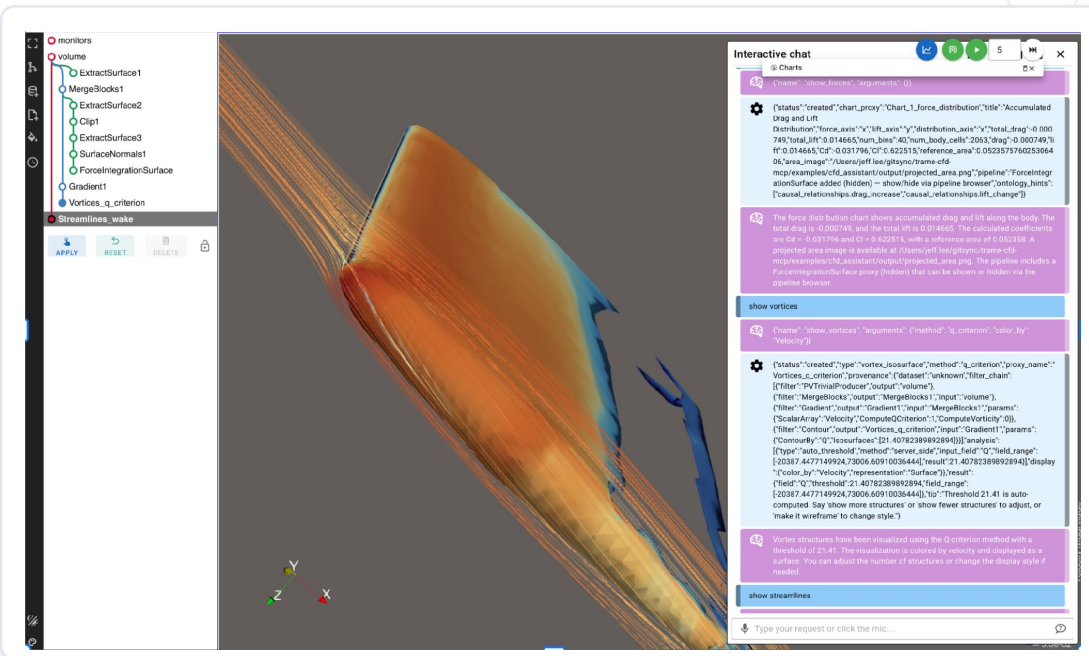
"show the vortices"
`show_vortices`

"streamlines in the wake"
`show_streamlines`

"explain the drag"
`show_forces`

"boundary layer at 50%"
`show_boundary_layer`

Every result returns its full filter chain as provenance.

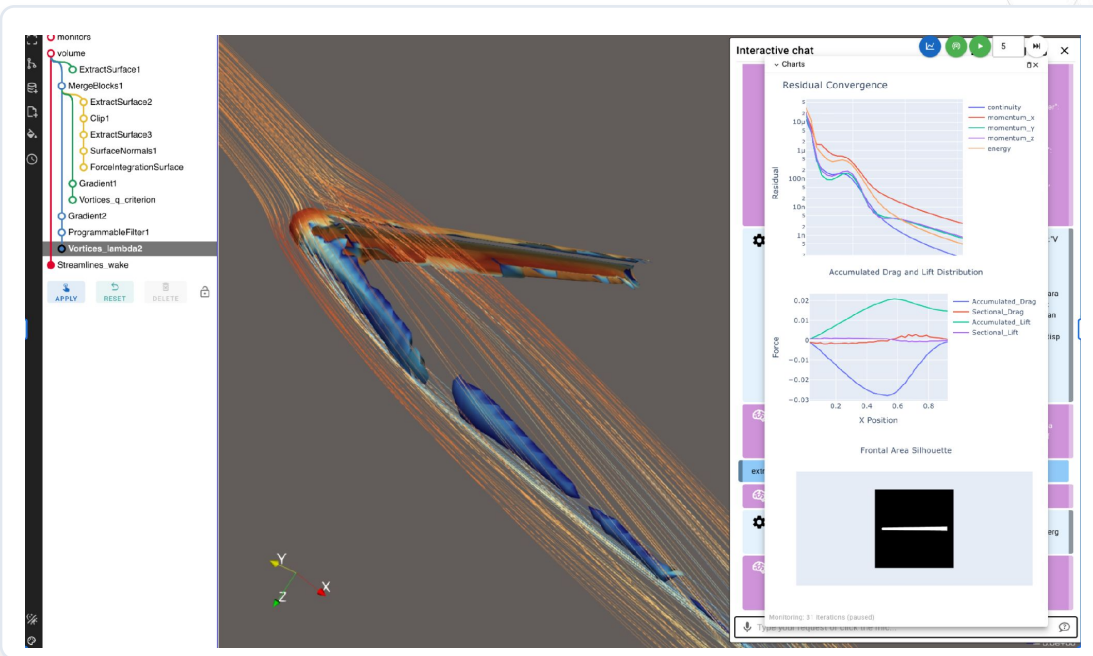


The same chat session watches, and can steer, the solver

LIVE, IN THE SAME CHAT

- Residual convergence streams as live Plotly charts
- Drag, lift, and frontal area update as iterations advance
- Ask about the run while it is still going

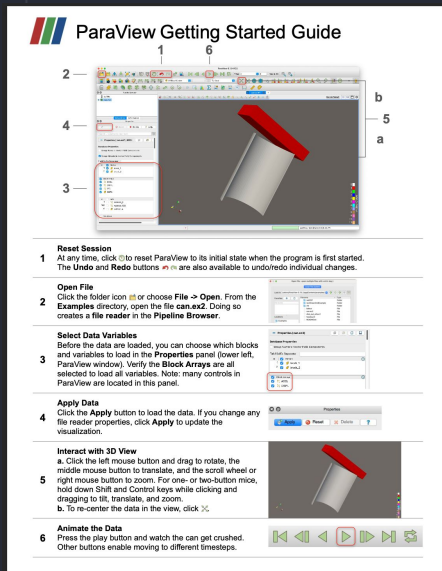
*Over a **Catalyst Live** link, the assistant reaches the running solver: the same in-situ path from Part One.*



Where to Learn More

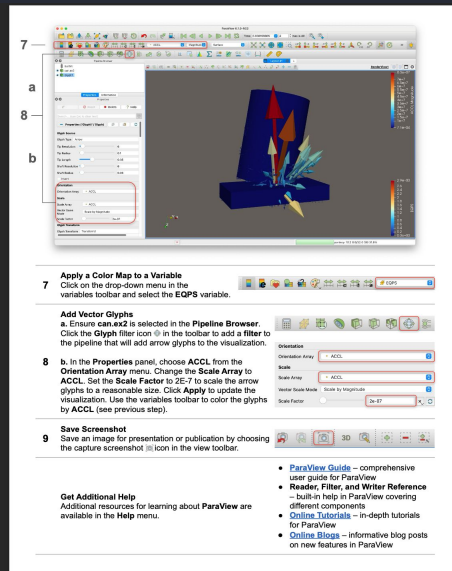
How to Learn More ParaView

Help menu -> Getting Started With ParaView



ParaView Getting Started Guide

- Reset Session**
At any time, click to reset ParaView to its initial state when the program is first started. The **Undo** and **Redo** buttons are also available to undo/redo individual changes.
- Open File**
Click the folder icon or choose **File > Open**. From the **Examples** directory, open the file **can.ex2**. Doing so creates a **file reader** in the **Pipeline Browser**.
- Select Data Variables**
Before the data are loaded, you can choose which blocks and variables to load in the **Properties** panel (lower left, ParaView window). Verify the **Block Arrays** are all selected to load all variables. Note: many controls in ParaView are located in this panel.
- Apply Data**
Click the **Apply** button to load the data. If you change any file reader properties, click **Apply** to update the visualization.
- Interact with 3D View**
a. Click the left mouse button and drag to rotate, the middle mouse button to translate, and the scroll wheel or right mouse button to zoom. For one- or two-button mice, hold down **Shift** and **Control** keys while clicking and dragging to tilt, translate, and zoom.
b. To re-center the data in the view, click .
- Animate the Data**
Press the play button and watch the can get crushed. Other buttons enable moving to different timesteps.

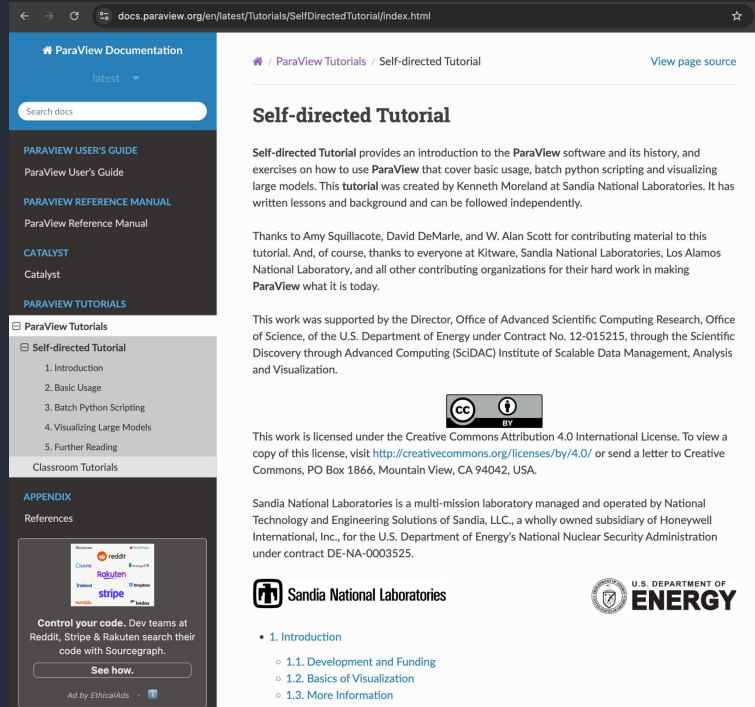


- Apply a Color Map to a Variable**
Click on the drop-down menu in the variables toolbar and select the **EQPS** variable.
- Add Vector Glyphs**
a. Ensure **can.ex2** is selected in the **Pipeline Browser**. Click the **Glyph** filter icon in the toolbar to add a **filter** to the pipeline that will add arrow glyphs to the visualization.
b. In the **Properties** panel, choose **ACCL** from the **Orientation Array** menu. Change the **Scale Array** to **ACCL**. Set the **Scale Factor** to **35.7** to scale the arrow glyphs to a reasonable size. Click **Apply** to update the visualization. Use the variables toolbar to color the glyphs by **ACCL** (see previous step).
- Save Screenshot**
Save an image for presentation or publication by choosing the capture screenshot icon in the view toolbar.

Get Additional Help
Additional resources for learning about ParaView are available in the **Help** menu.

- ParaView Guide** – comprehensive user guide for ParaView
- Reader, Filter, and Writer Reference** – built-in help in ParaView covering different components
- Online Tutorials** – in-depth tutorials for ParaView
- Online Blogs** – informative blog posts on new features in ParaView

docs.paraview.org



ParaView Documentation

latest

Search docs

- PARAVIEW USER'S GUIDE
ParaView User's Guide
- PARAVIEW REFERENCE MANUAL
ParaView Reference Manual
- CATALYST
Catalyst
- PARAVIEW TUTORIALS
Self-directed Tutorial

Self-directed Tutorial

Self-directed Tutorial provides an introduction to the ParaView software and its history, and exercises on how to use ParaView that cover basic usage, batch python scripting and visualizing large models. This tutorial was created by Kenneth Moreland at Sandia National Laboratories. It has written lessons and background and can be followed independently.

Thanks to Amy Squillacote, David DeMarle, and W. Alan Scott for contributing material to this tutorial. And, of course, thanks to everyone at Kitware, Sandia National Laboratories, Los Alamos National Laboratory, and all other contributing organizations for their hard work in making ParaView what it is today.

This work was supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. 12-015215, through the Scientific Discovery through Advanced Computing (SciDAC) Institute of Scalable Data Management, Analysis and Visualization.

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY

1. Introduction

- 1.1. Development and Funding
- 1.2. Basics of Visualization
- 1.3. More Information

Control your code. Dev teams at Reddit, Stripe & Rakuten search their code with Sourcegraph.

See how.

Ad by EthicalAds

Running ParaView on ALCF machines

Running ParaView server on Polaris

ALCF ParaView tutorial

This screenshot shows the 'ParaView on Polaris' page on the ALCF website. The page title is 'ParaView on Polaris'. The main content area explains that the recommended way to run ParaView on Polaris is in client/server mode, consisting of running the ParaView client on a local resource and the ParaView server on the Polaris compute nodes. It notes that the ParaView client needs to be installed on the local resource and must match the version of ParaView currently available on Polaris. A code block shows the command: `module use /soft/modulefiles module avail paraview`. Below this, it states that binary and source packages of the ParaView client for Linux, macOS, and Windows are available from the ParaView Download Page. The 'Connecting to the ParaView server on Polaris' section describes how to launch the ParaView server on Polaris from a local ParaView client. It includes a sub-section 'Start ParaView Client' which instructs the user to first launch the ParaView client on their local resource, then configure some server settings in the client. This initial setup should only need to be done once and can be reused each time you want to run ParaView on Polaris. The 'Server Configuration' section is also visible at the bottom. The left sidebar contains a navigation menu with links like 'ALCF User Guides', 'Chet', 'Polaris', 'Getting Started', 'Known Issues', 'System Updates', 'Compiling and Linking', 'Build Tools', 'Running Jobs', 'Applications and Libraries', 'Containers', 'Data Science', 'Programming Models', 'Debugging Tools', 'Performance Tools', 'Visualization', 'ParaView (Launch from Client)', 'ParaView (Manual Launch)', 'Visit', 'FFmpeg', 'ImageMagick', 'ParaView Tutorial', 'VNC', and 'Workflows'. The right sidebar contains a 'Table of contents' with links to 'Overview', 'Data', '1. Load Multi-component Dataset', '2. Select which data to view', '3. Manipulating the Color Map', '4. Data Representation', '5. Generate Streamlines', '6. Streamlines as Tubes', '7. Cutting Planes (Slices)', '8. Data Representation: Opacity', '9. Animating Simulation Data', '10. Animations', '11. Particles as Glyphs', '12. Enter: Red Blood Cells', '13. Using Color to Differentiate Data', '14. Further Exploration: Highlight the Mesh', '15. Further Exploration: Highlight the Vertices', '16. Further Exploration: Color by Variable', and '17. Background Color'. There is also a link to 'Does this doc need an update? Open an issue on GitHub'.

This screenshot shows the 'ParaView Tutorial Overview' page on the ALCF website. The page title is 'ParaView Tutorial Overview'. The main content area states that this tutorial is intended to be a hands-on resource for users interested in learning the basic concepts of ParaView. The examples can easily be run on a laptop, using the example data set provided. A list of bullet points includes: 'Tour of ParaView', 'Show range of visualization methods', 'Walk through various visualization techniques, hopefully illustrating how these can apply to your own data', 'Feel for ParaView "way"', and 'Terminology and step-by-step process peculiar to ParaView, which may differ from other packages, e.g., Visit'. Below the list is a large image showing a 3D visualization of a flow field, likely a blood flow simulation, with various colored regions and streamlines. The caption below the image reads 'Bloodflow Visualization by Joe Inley, ALCF'. The left sidebar contains a navigation menu with links like 'ALCF User Guides', 'Build Tools', 'Running Jobs', 'Applications and Libraries', 'Containers', 'Data Science', 'Programming Models', 'Debugging Tools', 'Performance Tools', 'Visualization', 'ParaView (Launch from Client)', 'ParaView (Manual Launch)', 'Visit', 'FFmpeg', 'ImageMagick', 'ParaView Tutorial', 'VNC', 'Workflows', 'Sophia', 'Running Jobs with PBS', 'Example Job Scripts', 'Common PBS Issues', 'Machine Reservations', 'Data Management', and 'File System & Storage'. The right sidebar contains a 'Table of contents' with links to 'Overview', 'Data', '1. Load Multi-component Dataset', '2. Select which data to view', '3. Manipulating the Color Map', '4. Data Representation', '5. Generate Streamlines', '6. Streamlines as Tubes', '7. Cutting Planes (Slices)', '8. Data Representation: Opacity', '9. Animating Simulation Data', '10. Animations', '11. Particles as Glyphs', '12. Enter: Red Blood Cells', '13. Using Color to Differentiate Data', '14. Further Exploration: Highlight the Mesh', '15. Further Exploration: Highlight the Vertices', '16. Further Exploration: Color by Variable', and '17. Background Color'. There is also a link to 'Does this doc need an update? Open an issue on GitHub'.

ParaView Office Hours

- Join us for **ParaView Office Hours**, Thursdays 3:30-5:00pm ET (New York)
- Hands-on guidance from the ParaView experts in an informal, interactive session
- Sessions every two weeks, excluding holidays

