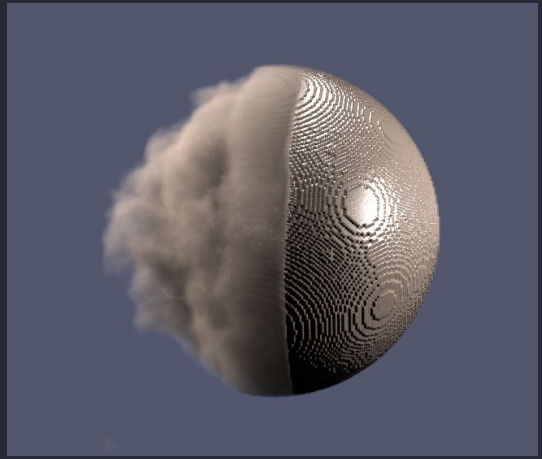
Large Scale Visualization with ParaView

Cory Quammen and Patrick Avery

kitware

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

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

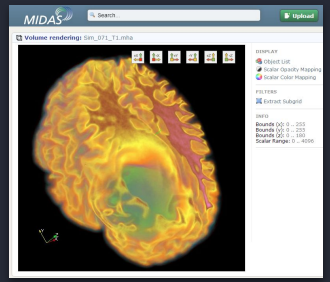
25+ years of expertise

Kitware USA, 1998
Kitware Europe, 2010

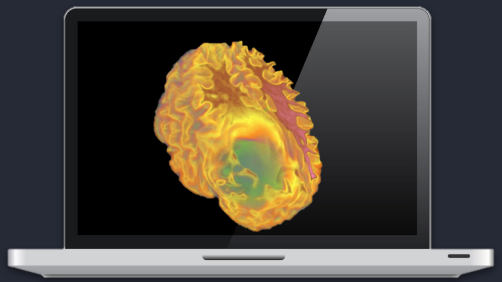
Founded on Open Source

Vibrant, enduring platform/communities
Strong commitment to Open Science

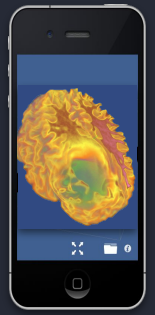
Applications / Universal Platforms



Web



Desktop



Mobile

Cloud /HPC

kitware Platforms

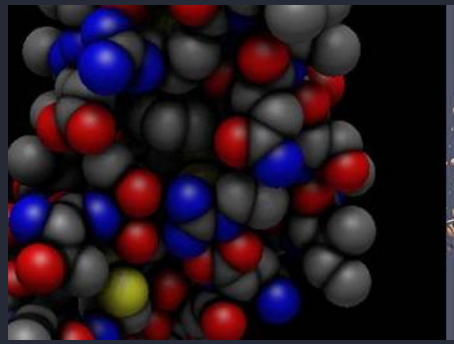
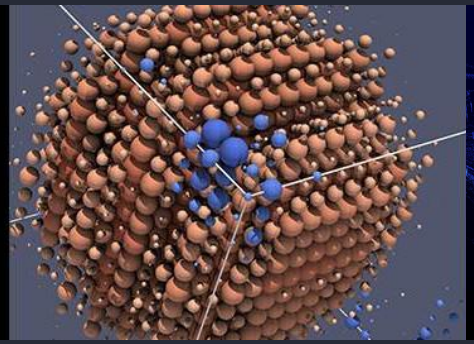
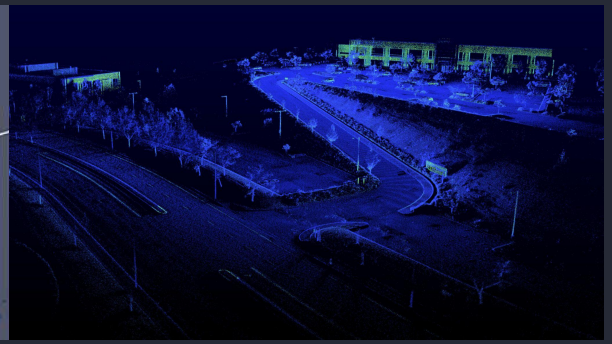
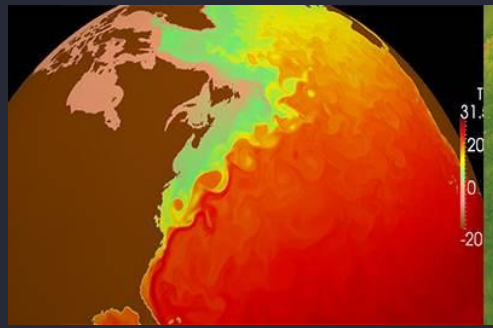
3D Slicer · ParaView · KWIVER · TeleSculptor · tomviz

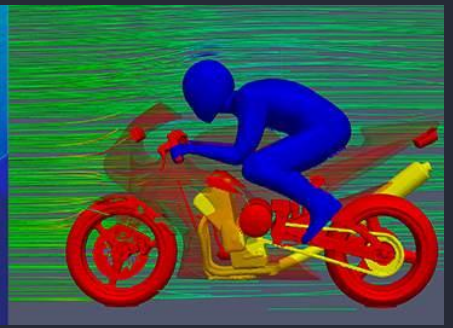
VTK.JS · LidarView · Slicer SALT · Resonant · VolView

Pulse Physiology Engine · CMake · trame · CMB · VTK · ITK

Kitware – HPC and Visualization

VTK ParaView CMB tomviz LidarView Trame

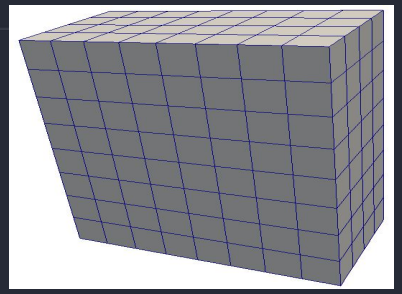



kitware

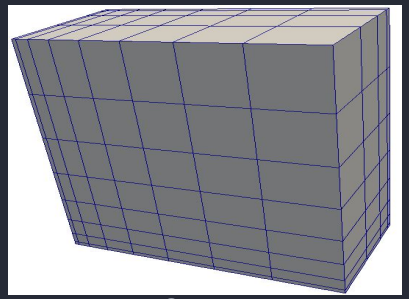
What is ParaView?

- 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**

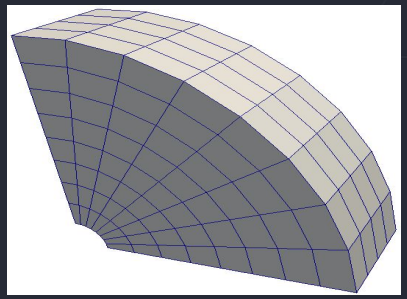
ParaView (VTK) Data Types



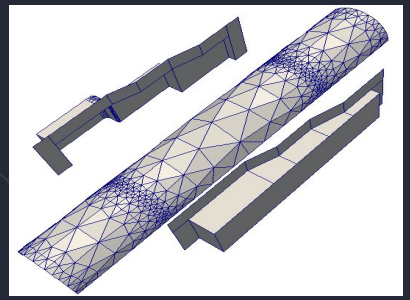
Uniform Rectilinear
(vtkImageData)



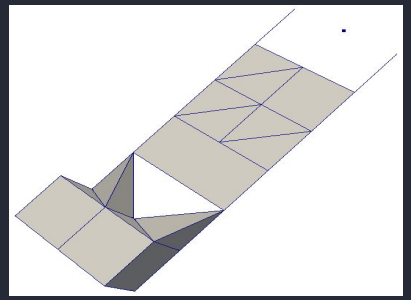
Non-Uniform Rectilinear
(vtkRectilinearData)



Curvilinear
(vtkStructuredData)



Polygonal
(vtkPolyData)



Unstructured Grid
(vtkUnstructuredGrid)

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

kitware

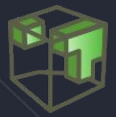
Common Filters


Calculator


Contour


Clip


Slice


Threshold


Extract Subset


Glyph


Stream Tracer

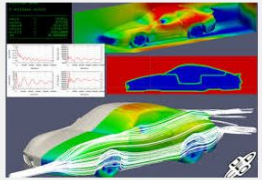
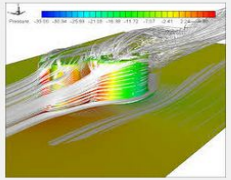
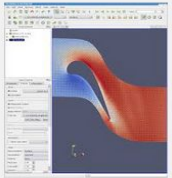
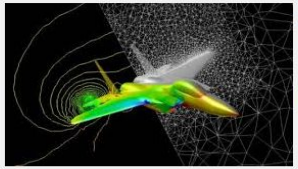
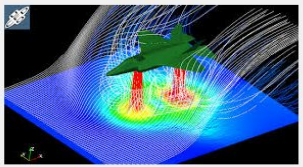
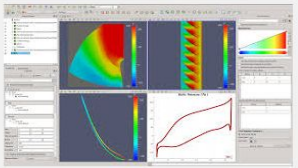
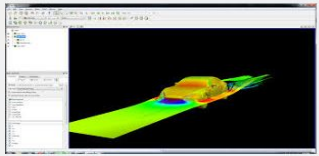
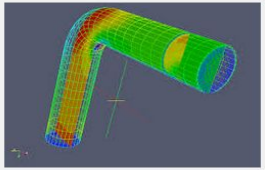
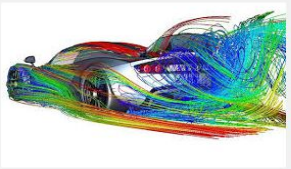
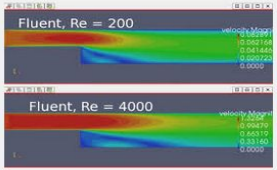

Warp (vector)

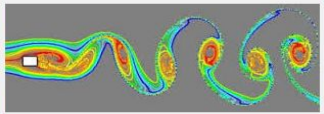
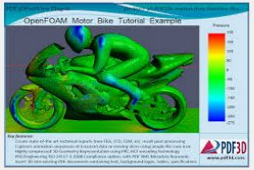

Group Datasets

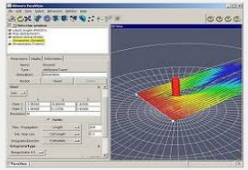
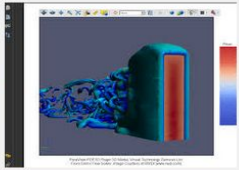
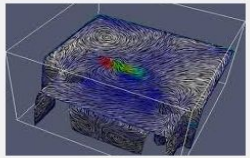
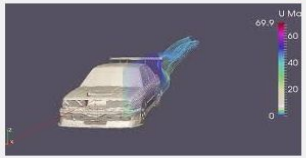
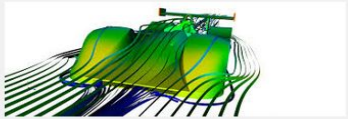
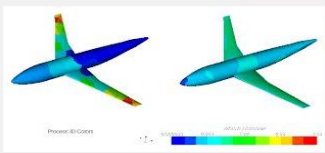
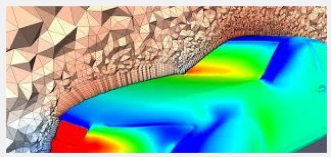
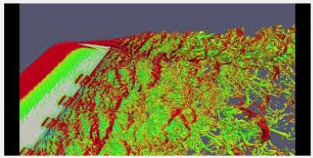
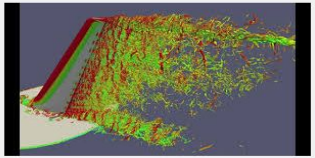
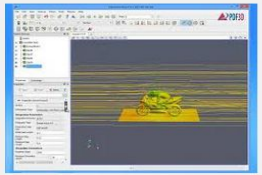
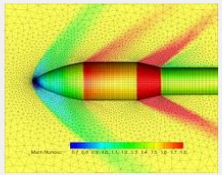
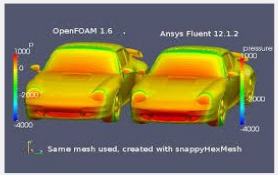

Extract Block

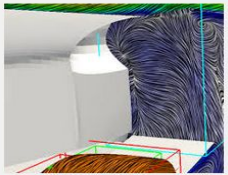
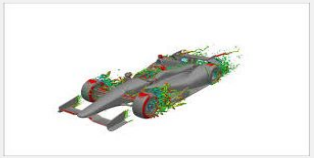
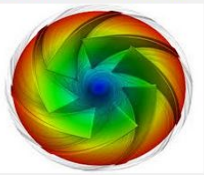
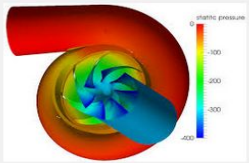
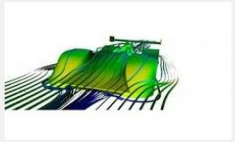
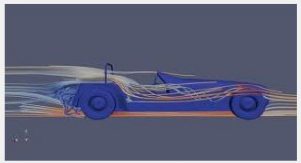
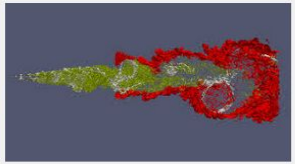
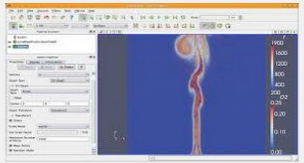
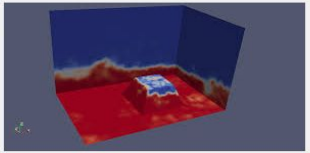
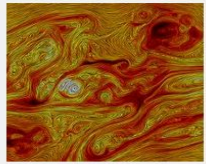
kitware

Scalable Post Hoc Scientific Analysis & Visualization

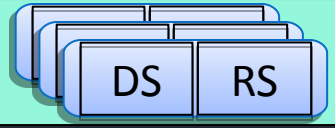











ParaView Demo

Visualizing Large Datasets with Distributed Memory Parallelism

ParaView's Running Modes

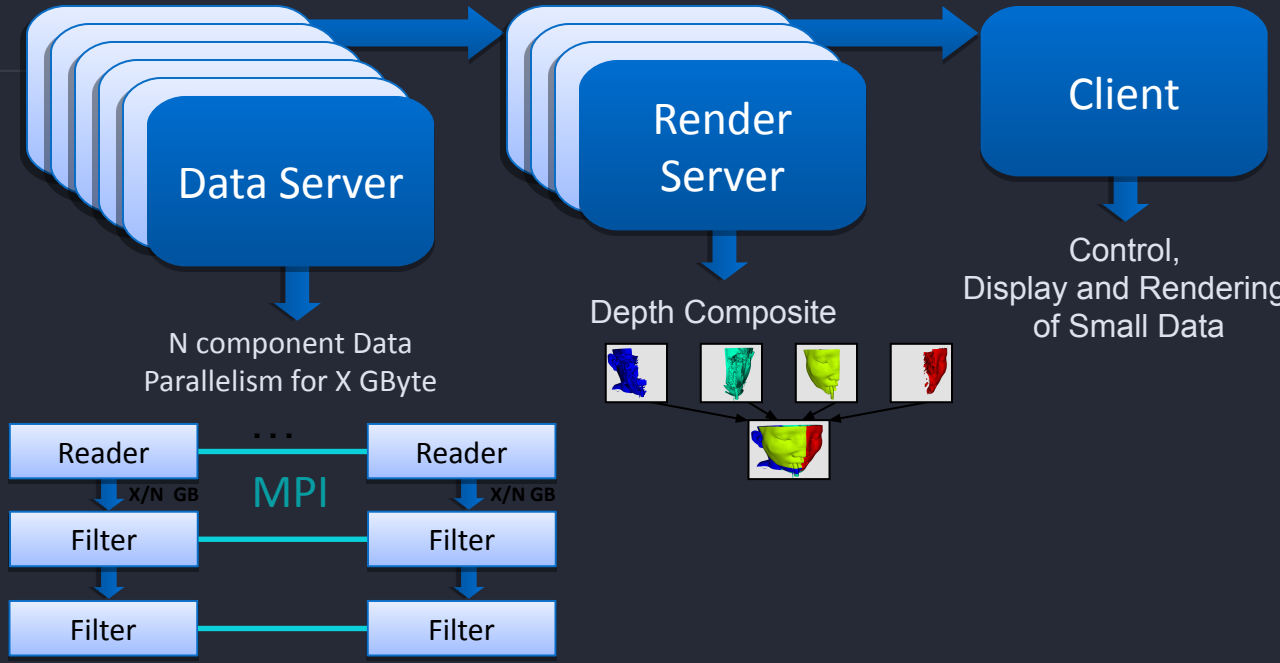
Builtin aka Standalone aka Serial		all components within one process (client may be GUI or pvpython) `paraview` or `pvpython`
Combined Server		data processing and parallel rendering in MPI job of combined processes. control from TCP connected client. `mpiexec -n x pvserver &; paraview #` or `pvpython # + Connect`

Batch		server is an MPI job which directly runs a python script
		`mpiexec -n x pvbatch \`
		`vis_script.py`

DS = data server

RS = render server

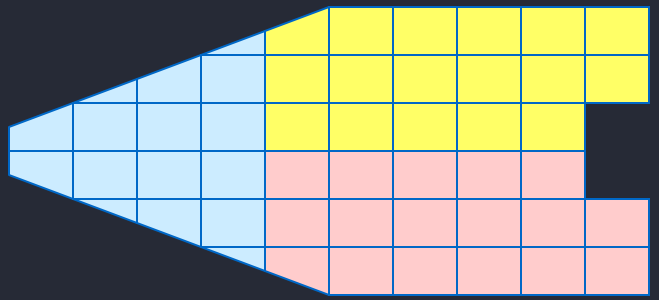
kitware


Data Server
Render Server
Client
Control,
Display and Rendering
of Small Data
Depth Composite
N component Data
Parallelism for X GByte
Reader
Reader
MPI
X/N GB
X/N GB
Filter
Filter
Filter
Filter

kitware

Data Parallel Pipelines

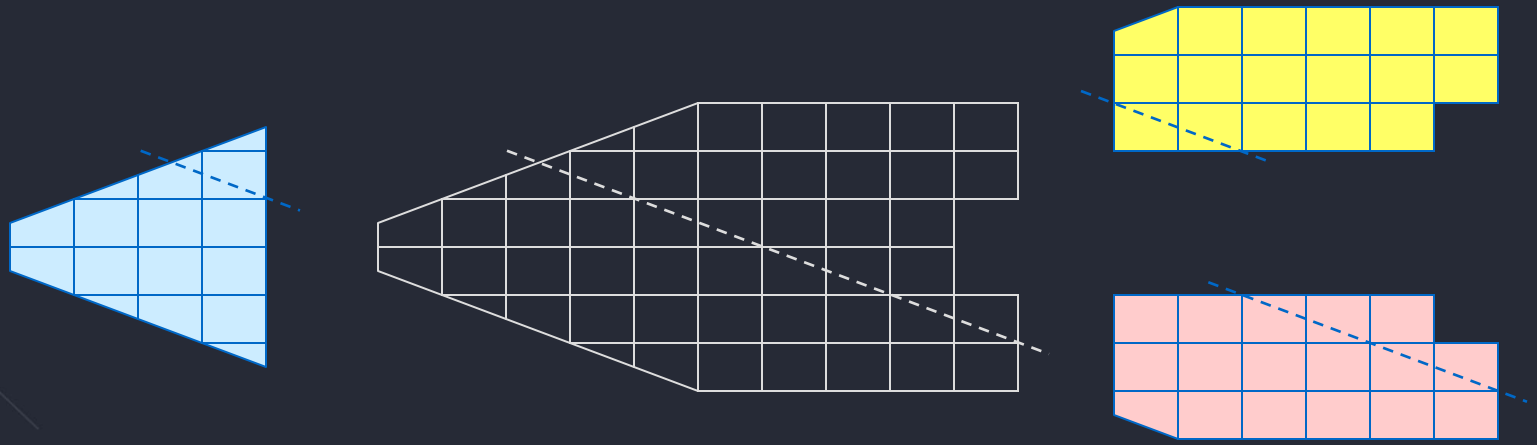
- Duplicate pipelines run independently on different partitions of data.



kitware

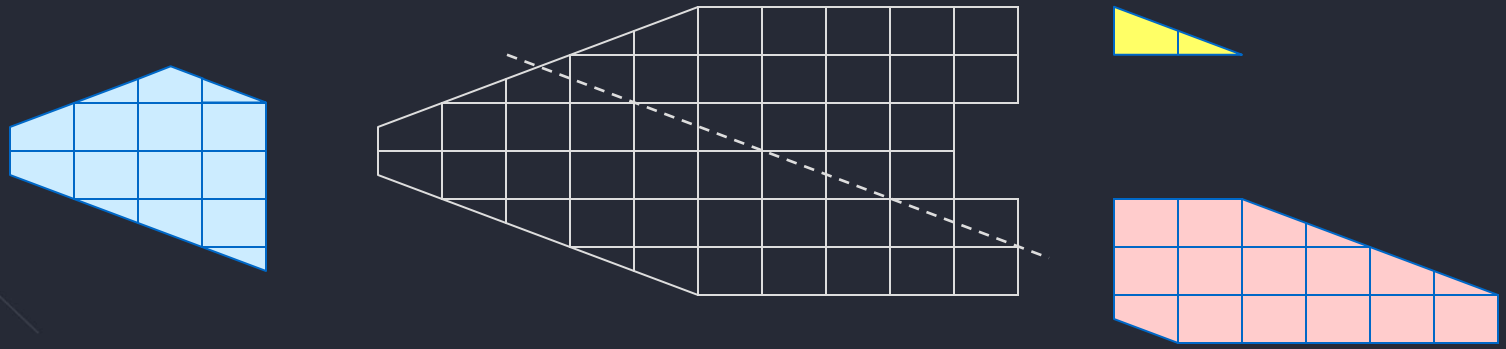
Data Parallel Pipelines

- Many operations will work regardless
  - Example: Clipping



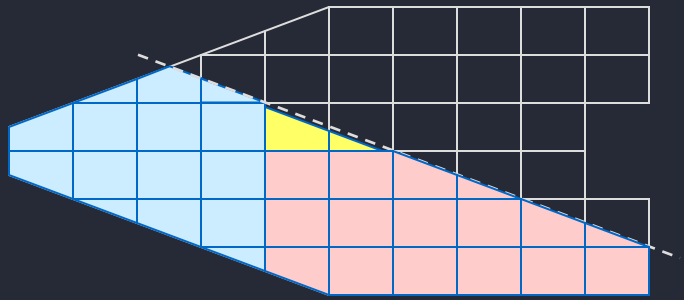
Data Parallel Pipelines

- Many operations will work regardless
  - Example: Clipping



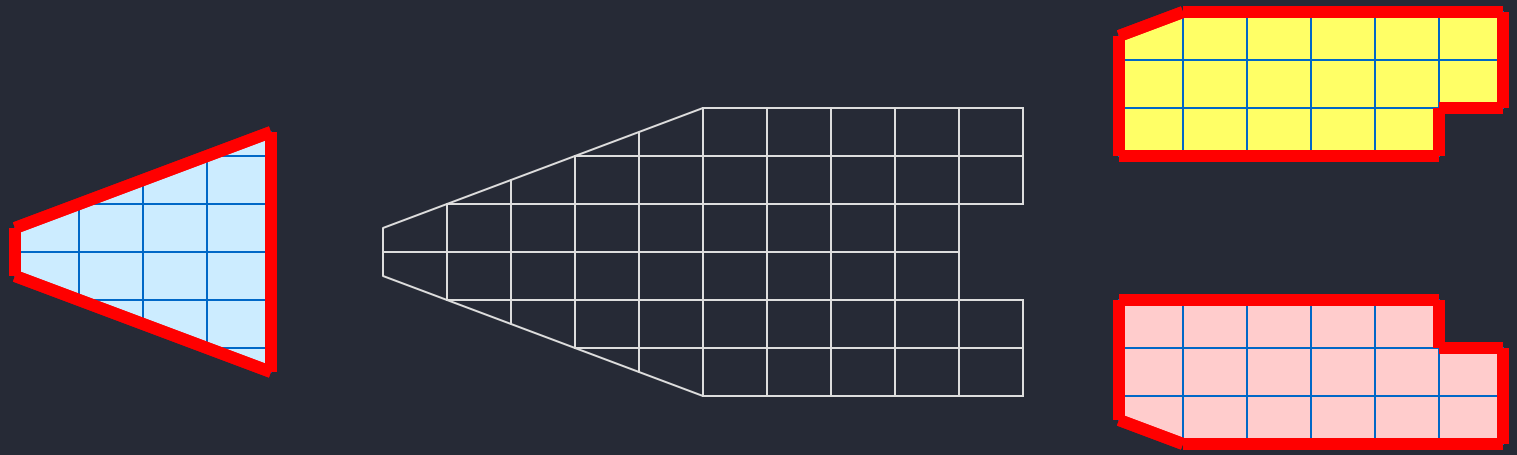
Data Parallel Pipelines

- Many operations will work regardless
  - Example: Clipping



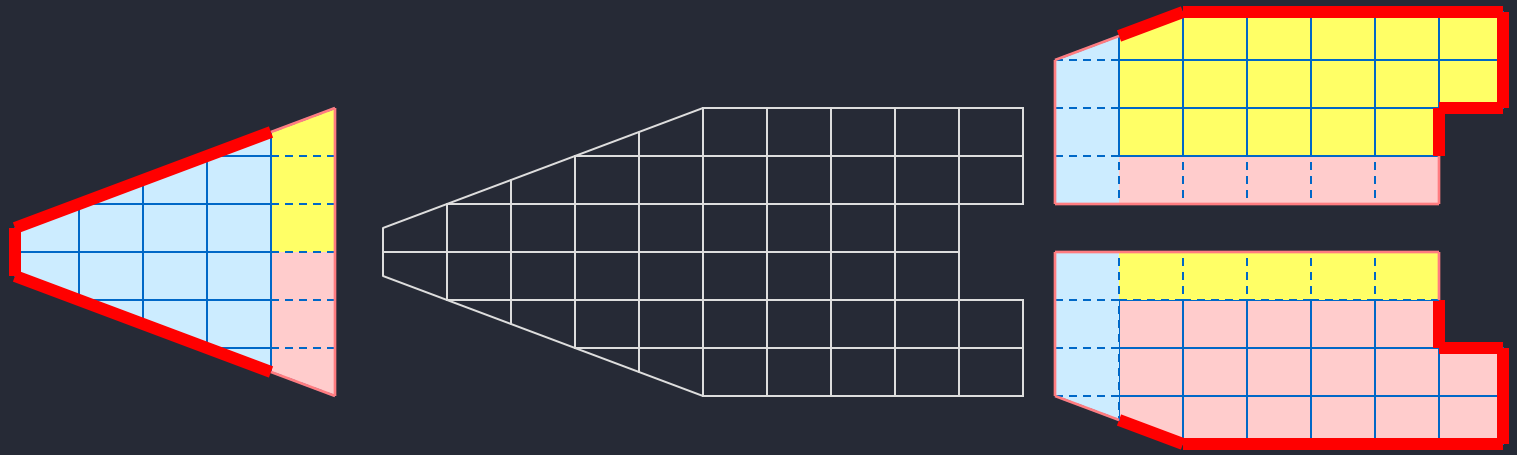
Advanced Data Parallel Pipelines

- Some operations will have problems
  - Example: External Faces



Advanced Data Parallel Pipelines

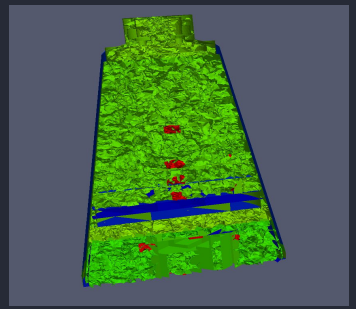
- Ghost cells can solve most of these problems

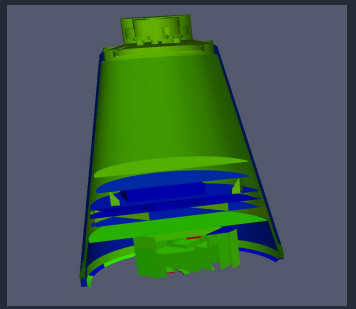


Ghost Cells filter

- 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

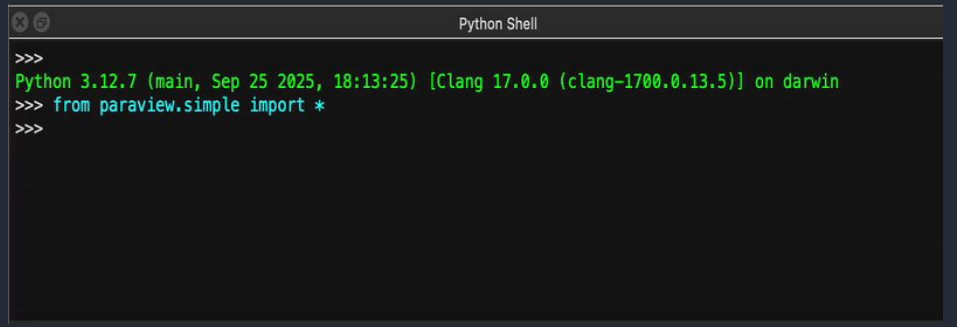
kitware

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

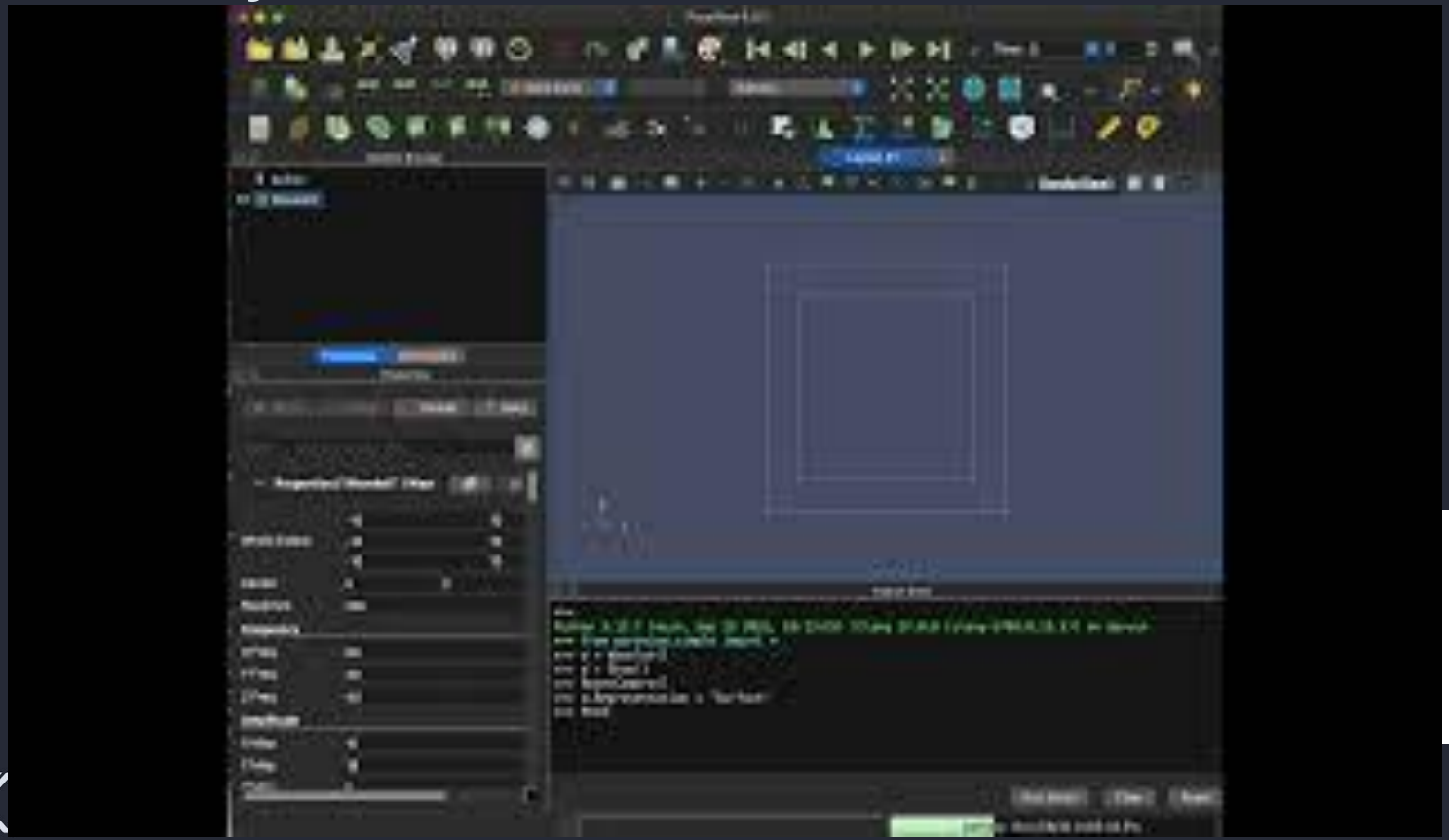
Python Shell

- **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**



kitware

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

kitware

In Situ Analysis at Scale with Catalyst

kitware

WHY IN SITU

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.

kitware

CATALYST IN ONE SLIDE

A lightweight framework beside the simulation

WHAT IT IS

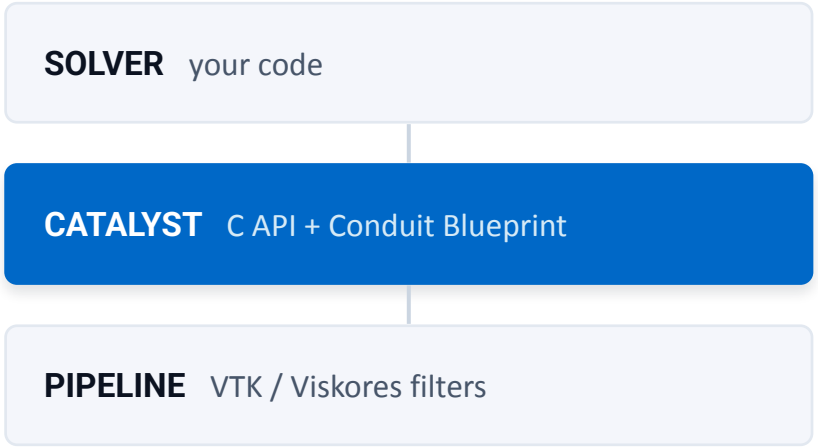
A small C API that hands a simulation's data to analysis pipelines while it is still running. No intermediate file write.

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

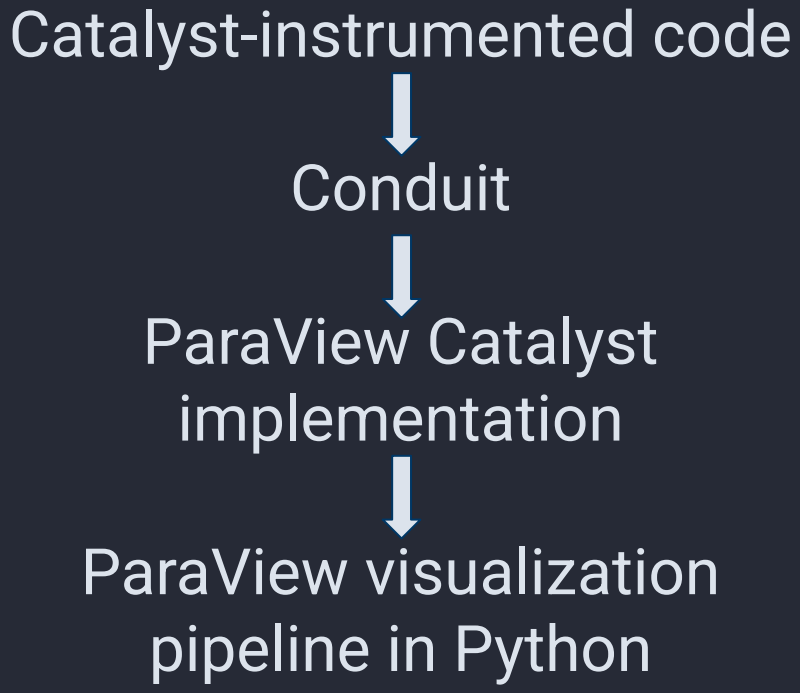
Data arrays mapped to Blueprint conventions for Conduit.



BSD-3 licensed. Available since 2014. Backwards-compatible C ABI since v2.
Deployed on leadership-class HPC, used by ECP applications and national labs.

kitware

ParaView Catalyst



kitware

ParaView + AI

Plain English in, validated ParaView analysis out

NATURAL LANGUAGE, ONE TOOL

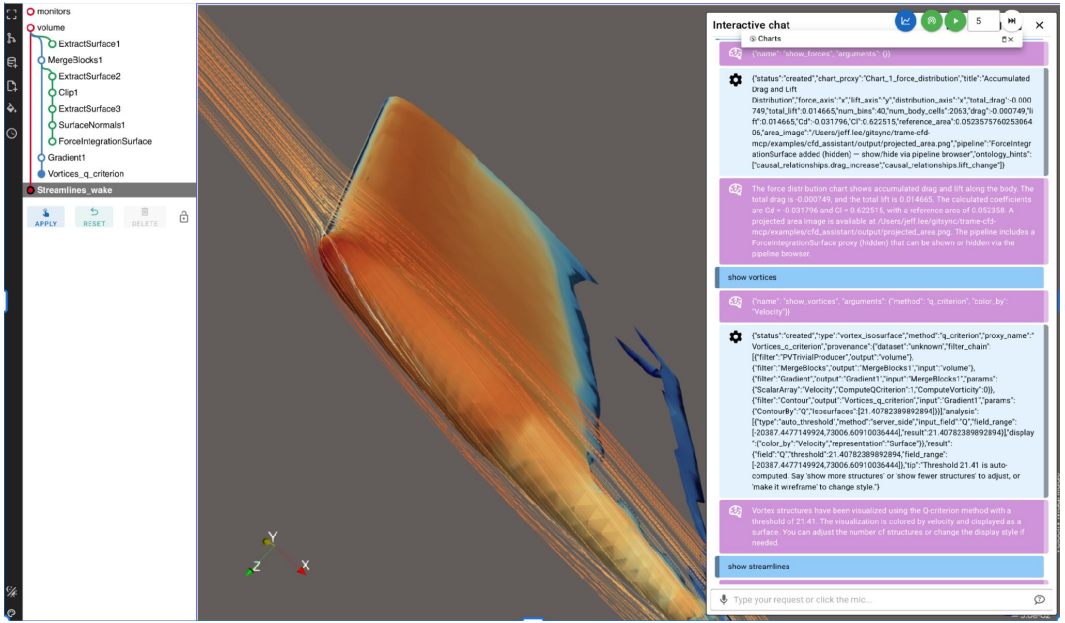
"show the vortices"
`show_vortices`

"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.



kitware

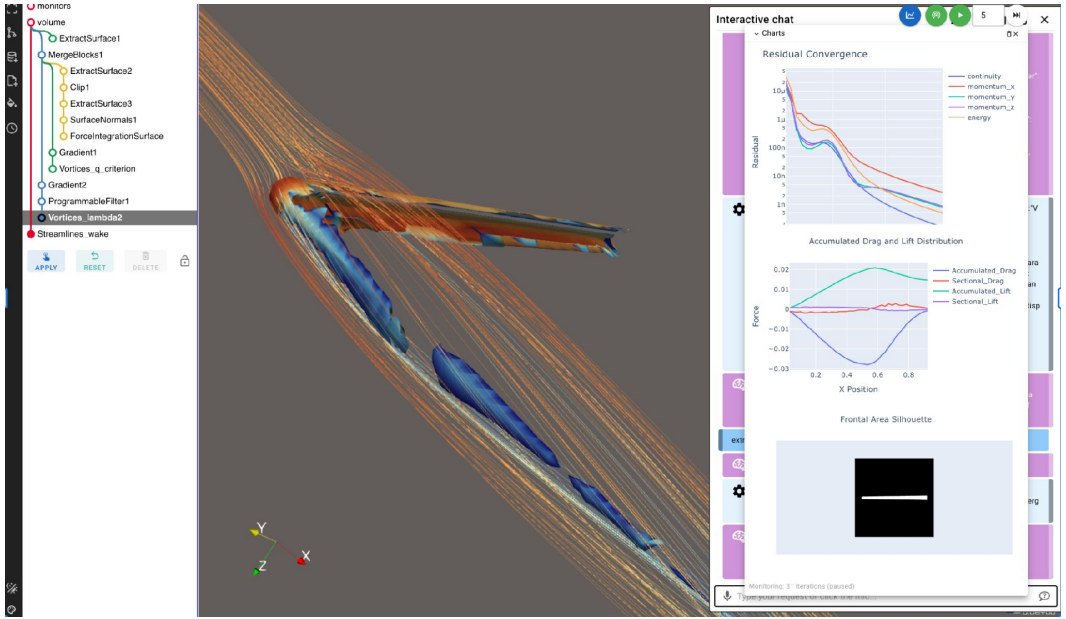
MONITORING A LIVE RUN

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.*



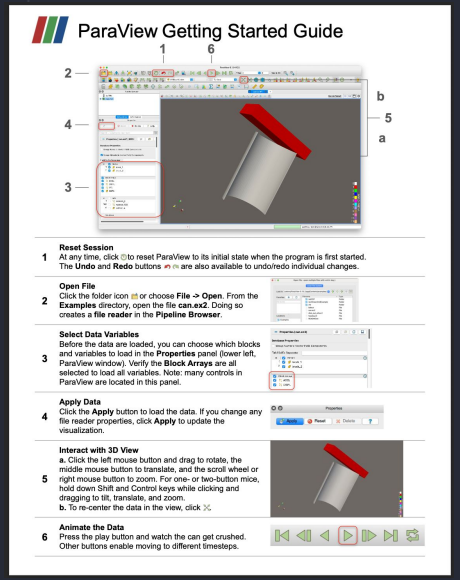
kitware

Where to Learn More

How to Learn More ParaView

docs.paraview.org

Help menu ->
Getting Started With ParaView



ParaView Getting Started Guide

Reset Session
1 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.

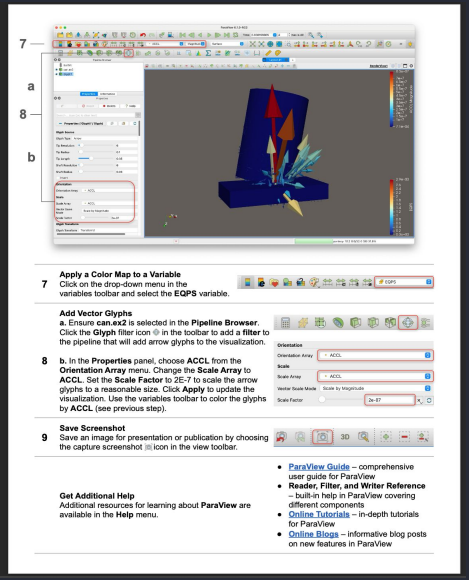
Open File
2 Click the folder icon or choose **File -> Open**. From the **Examples** directory, open the file **can.ex2**. Doing so creates a file reader in the **Pipeline Browser**.

Select Data Variables
3 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
4 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
5 **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
6 Press the play button and watch the can get crushed. Other buttons enable moving to different timesteps.



Apply a Color Map to a Variable
7 Click on the drop-down menu in the variables toolbar and select the **EQPS** variable.

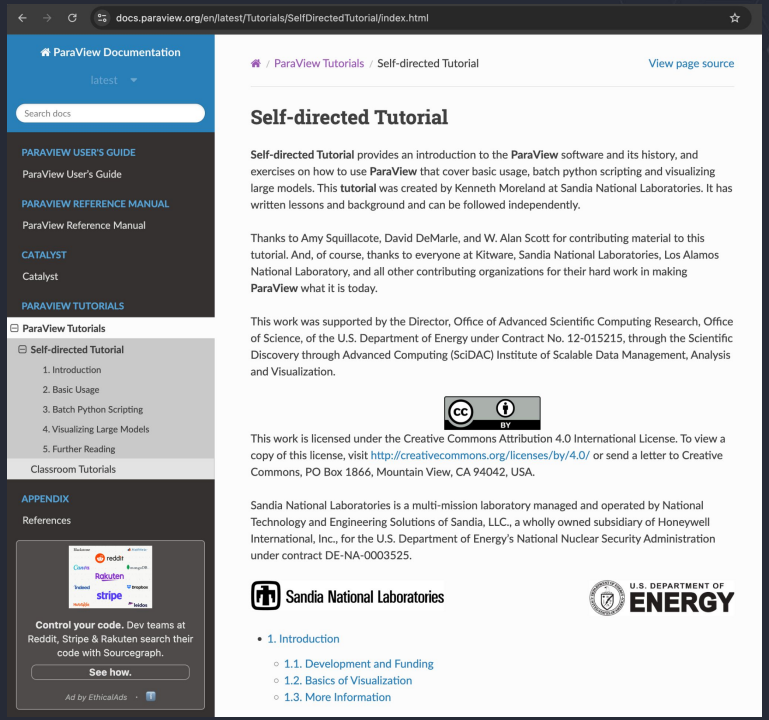
Add Vector Glyphs
8 **a.** Ensure **can.ex2** is selected in the **Pipeline Browser**. Click the **Glyph** filter icon in the toolbar to add a **filter** to the pipeline that will add arrow glyphs to the visualization.

b. In the **Properties** panel, choose **ACCL** from the **Orientation Array** menu. Change the **Scale Array** to **ACCL**. Set the **Scale Factor** to 2E-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
9 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/en/latest/Tutorials/SelfDirectedTutorial/index.html

ParaView Documentation
latest
Search docs

PARAVIEW USER'S GUIDE
ParaView User's Guide
PARAVIEW REFERENCE MANUAL
ParaView Reference Manual
CATALYST
Catalyst
PARAVIEW TUTORIALS
ParaView Tutorials
Self-directed Tutorial
1. Introduction
2. Basic Usage
3. Batch Python Scripting
4. Visualizing Large Models
5. Further Reading
Classroom Tutorials
APPENDIX
References

Control your code. Dev teams at Reddit, Stripe & Rakuten search their code with Sourcegraph.
See how.
Ad by EthicalAds

ParaView Tutorials / Self-directed Tutorial — View page source

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.

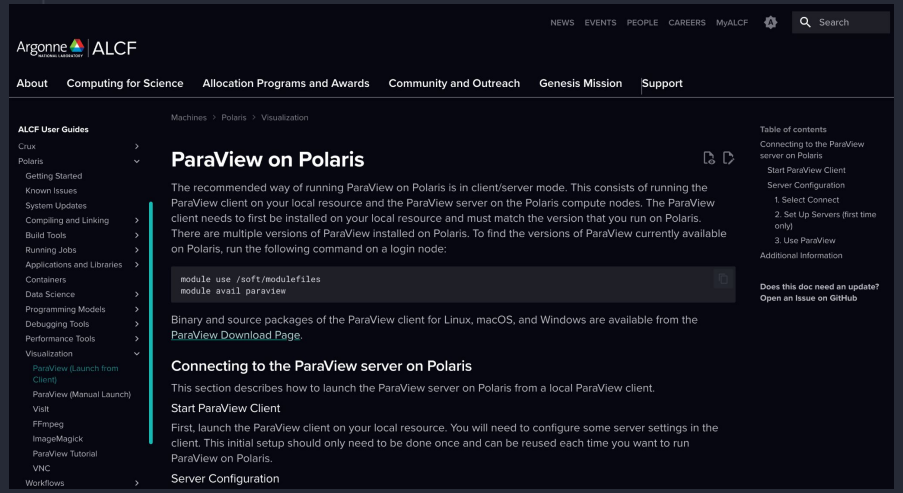
Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Sandia National Laboratories — U.S. DEPARTMENT OF ENERGY

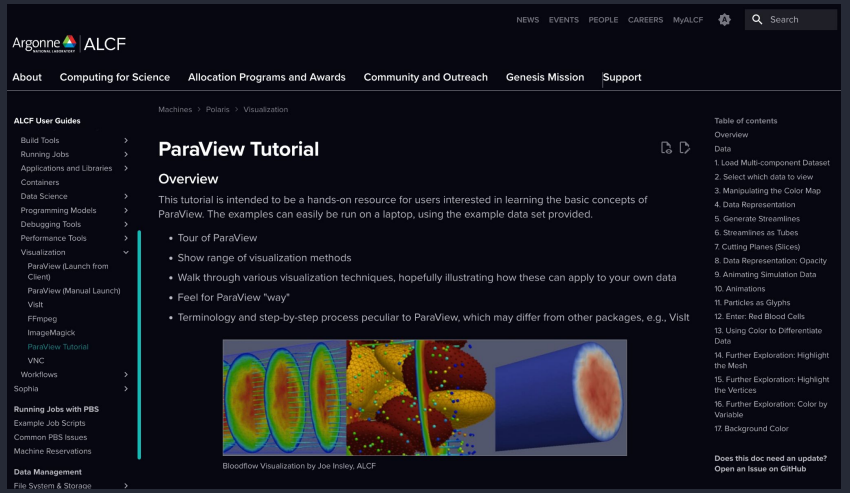
kitware

Running ParaView on ALCF machines

Running ParaView server on Polaris



ALCF ParaView tutorial



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

kitware

kitware.com/paraview-office-hours/