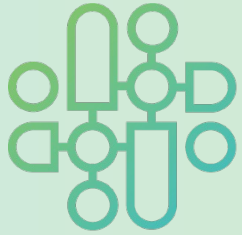


SC23
Denver, CO | i am hpc.

In Situ Analysis and Visualization with Ascent and ParaView Catalyst

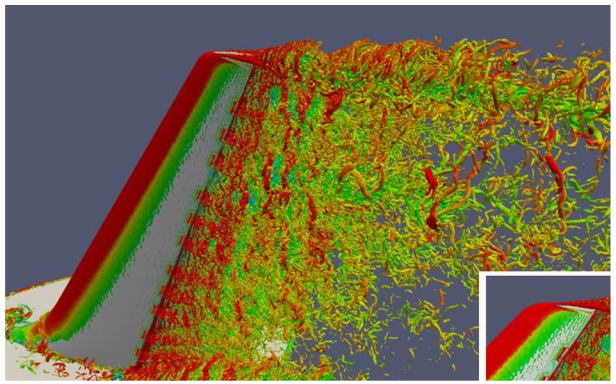
[Catalyst Project Overview]

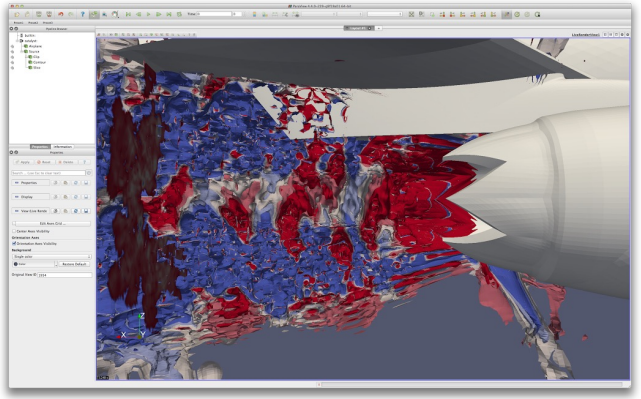
SC23 Tutorial
Monday November 13th, 2023

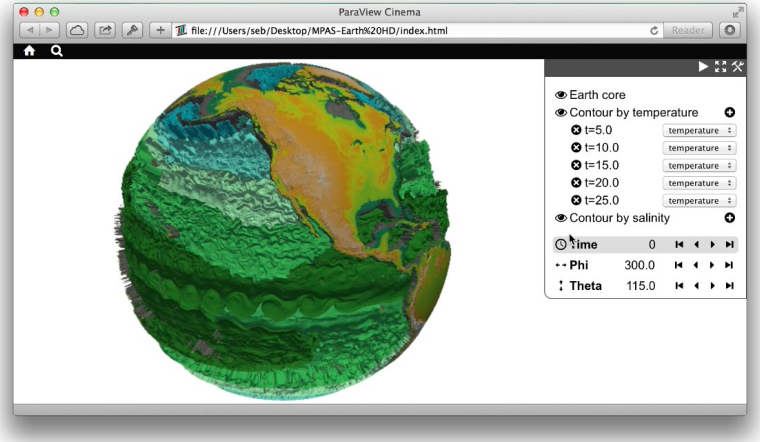
Corey Wetterer-Nelson, Kitware Inc.

ParaView Catalyst

ParaView Catalyst is an **in situ library**, with an adaptable application programming interface (API), that orchestrates the delicate alliance between simulation and analysis and/or visualization tasks.





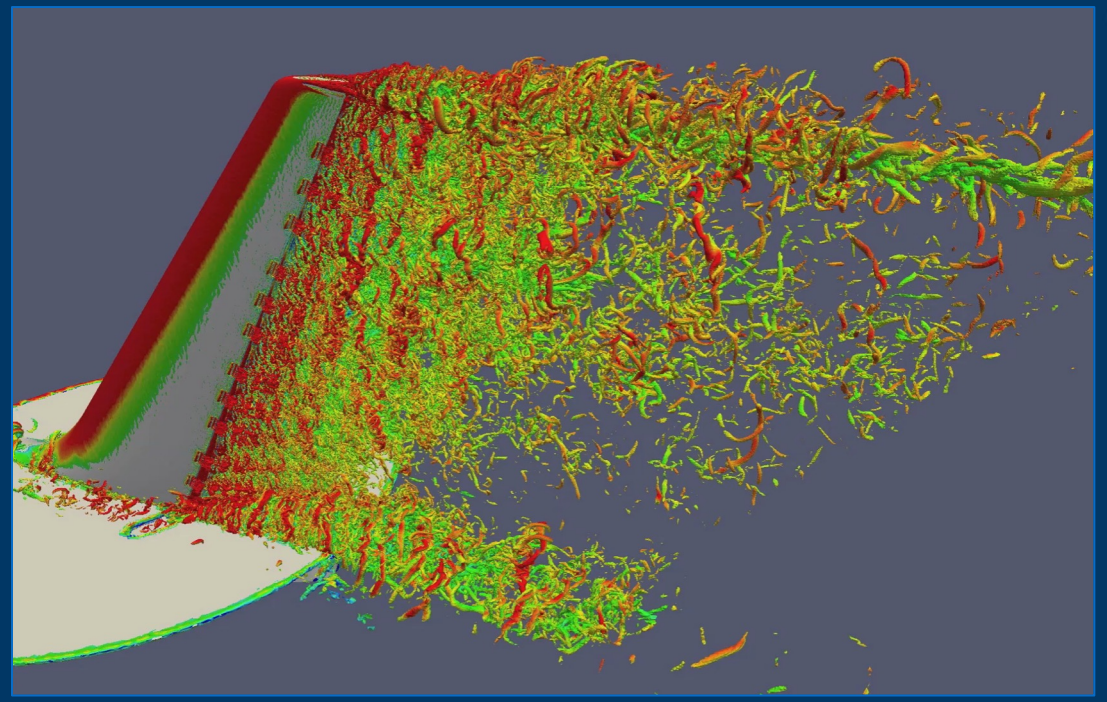


kitware

ParaView Catalyst background

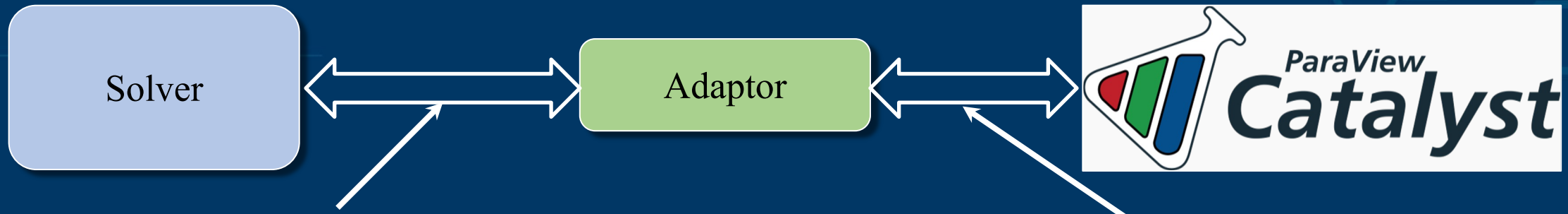
- Initial work started in 2008 from an Army SBIR
- Sep 2010 First ParaView release with CoProcessing (renamed to Catalyst in 2012)
- Sep 2014 Catalyst Live released
- Sep 2014 ParaView Cinema released
- Scaled to 1Mi MPI ranks on ALCF's Mira BG/Q
- SC16 visualization showcase winner generated animation using Catalyst
- Based on the ParaView libraries – full access to ParaView capabilities
- Batch and interactive in situ analysis and visualization
- In transit workflows done with standalone ParaView and ADIOS

>1M MPI ranks on Mira@ANL



kitware

Simulation, Adaptor and Catalyst Interactions



- Typically 3 calls between simulation code and adaptor
  - **catalyst_initialize()**
    - MPI communicator (optional)
    - Add analysis scripts
  - **catalyst_execute()**
    - Does the work
  - **catalyst_finalize()**

- Information provided by solver to adaptor
  - Time, time step, force output
  - Grids and fields

- **Information provided by adaptor**
  - Pipelines to execute
  - Time, time step, force output
  - Grid and fields when needed
  - MPI communicator
- **Information provided by Catalyst**
  - If co-processing needs to be done
  - What grids and fields are needed
- **User data can be shared both ways**

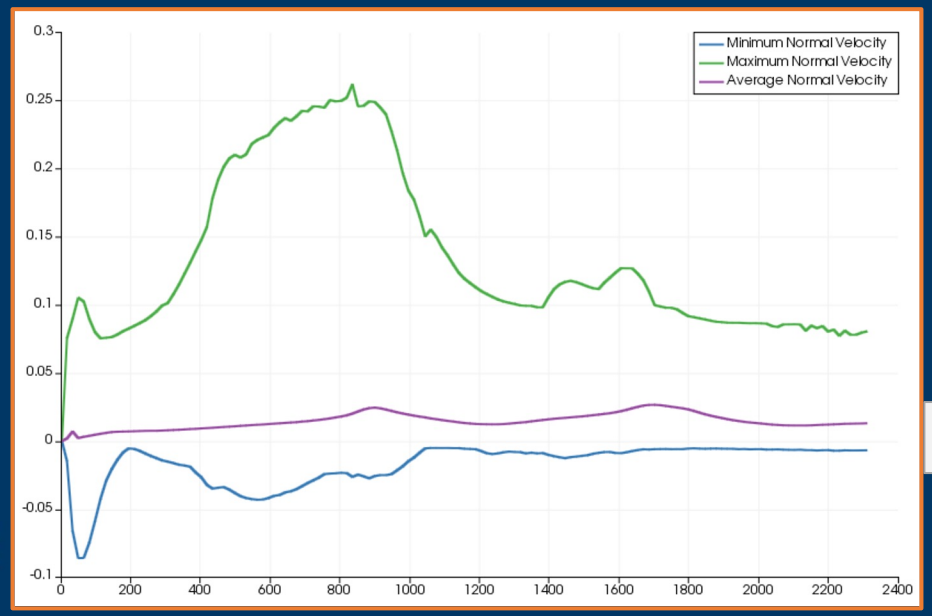
kitware

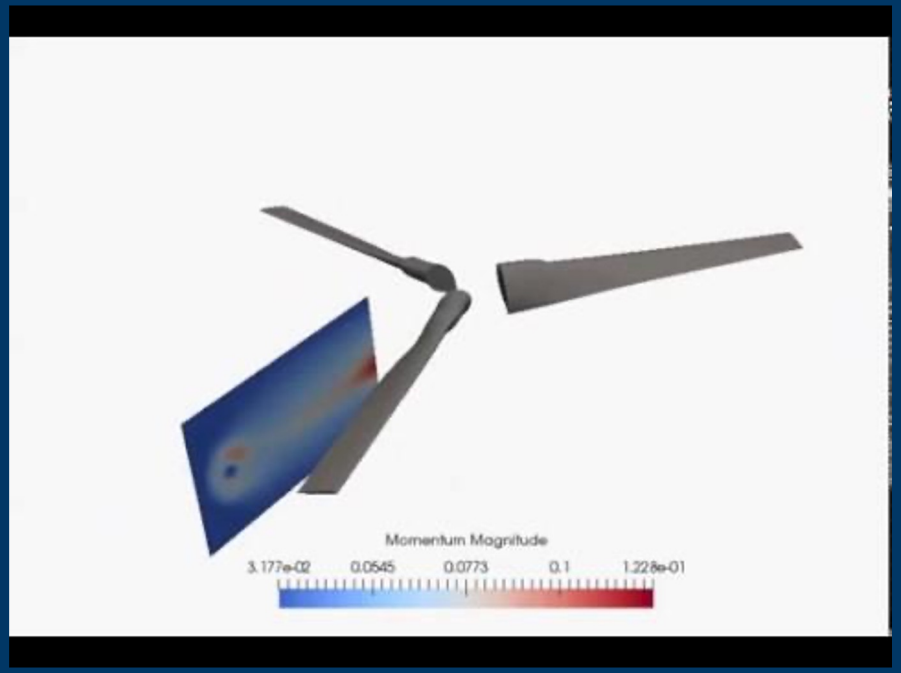
Performing In Situ Efficiently with Catalyst

- Small run-time overhead
  - Small initialization and finalization times
  - Scalable analysis and visualization algorithms
  - Reduced amount of IO (possibly at expense of more complex IO patterns)
- Small code footprint
  - Typically ~3 calls from simulation code to Catalyst (initialize, execute, finalize)
- Efficiency in computing and memory-management
  - On demand compute
  - Request only needed information
  - Flexible ways to represent different pieces of information
  - Zero-copy use of simulation data structures

Configurability

- Allow simulation specific output customization
- Edit Python scripts for Catalyst
  - Can change in situ output without any recompilation



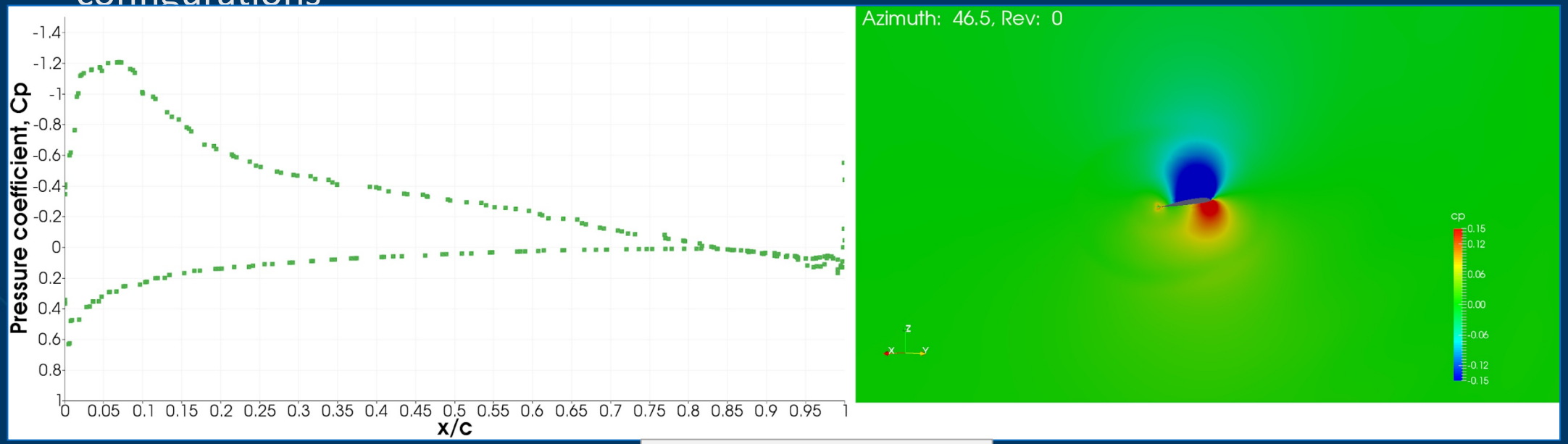


Results courtesy Rajneesh Singh & Ben Jimenez (ARL)

kitware

Combined Viz & Analysis

- Constructing relatively complex in situ pipelines requires both domain and viz experts
  - Increased interactions
  - Viz experts can help guide work early on
- Conceptually complex in situ output can be done through parameterized input configurations

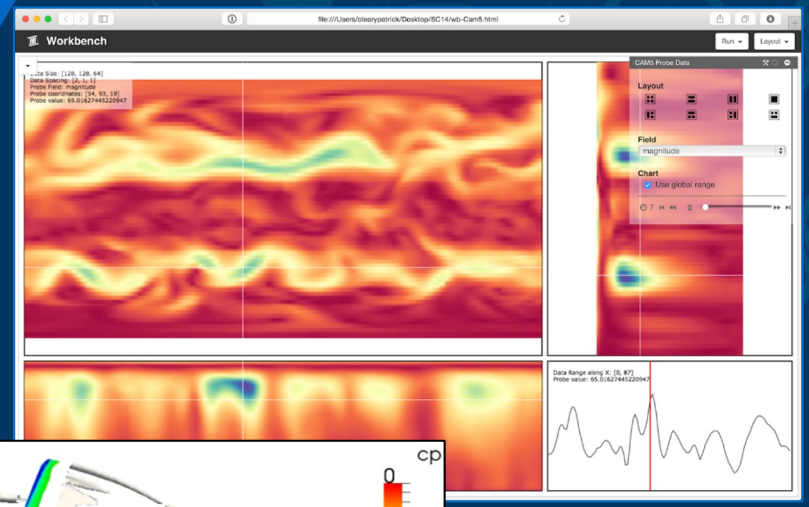


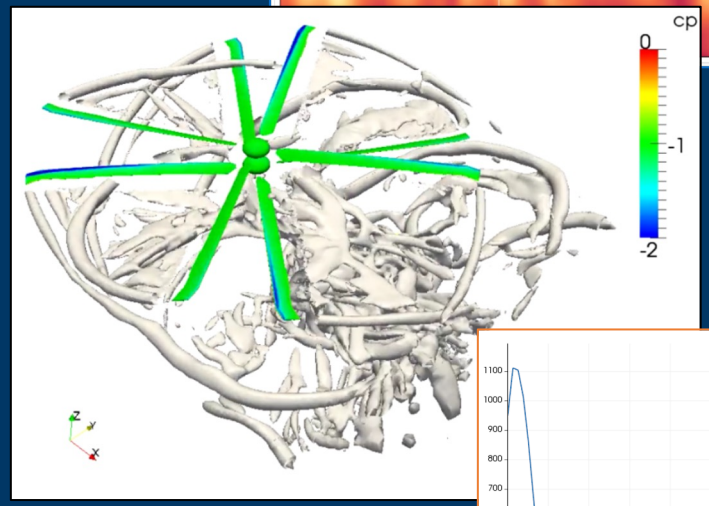
Singh & Jimenez (ARL)

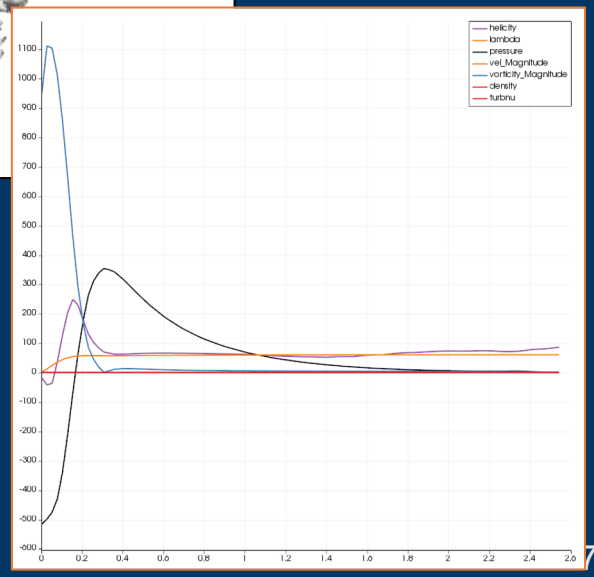
kitware

Multiple Output Types

- Images
  - Size is independent of grid size
  - Explorable through Cinema output (cinemascience.org)
  - Charts
- Data extracts
  - Large collection of file formats supported including VTK, ADIOS, HDF5 (in progress)
- ASCII output
  - Single quantities (e.g. field values at a point)
  - Statistics
  - Custom (e.g. value and location of highest temperature value)

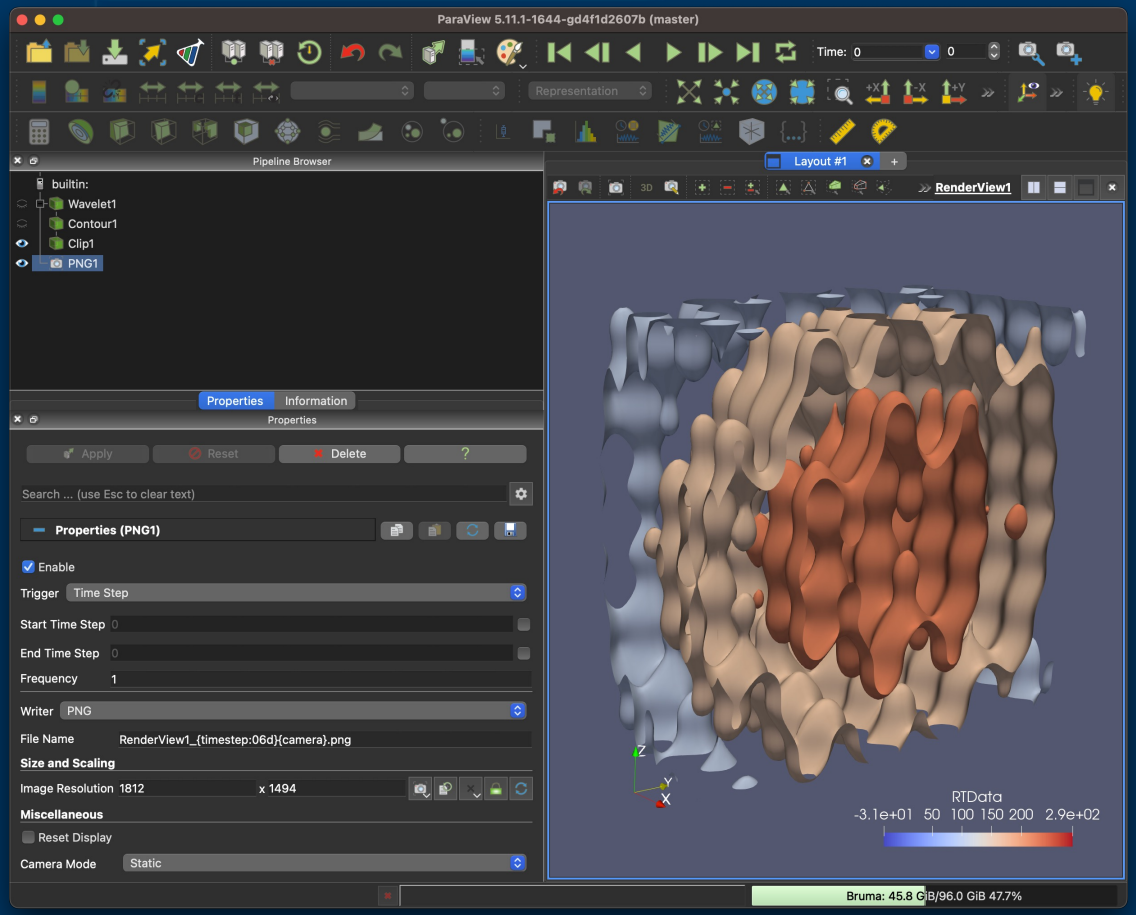






kitware

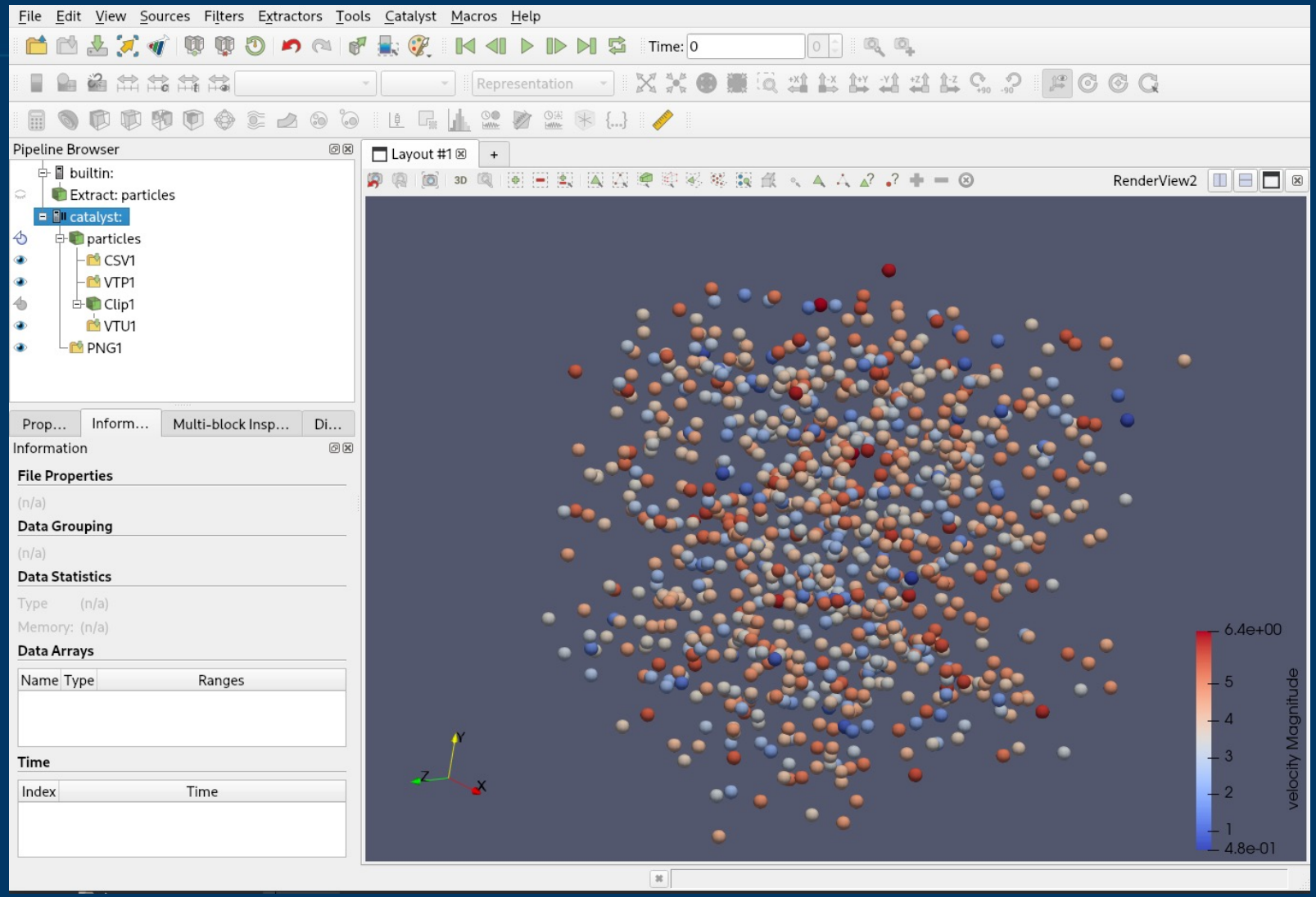
Automated Pipeline Generation in ParaView



```
Downloads — vi powerful-catalyst-script.py — 117×69
# script-version: 2.0
# Catalyst state generated using paraview version 5.11.1-1644-gd4f1d2607b
import paraview
paraview.compatibility.major = 5
paraview.compatibility.minor = 11

#### import the simple module from the paraview
from paraview.simple import *
#### disable automatic camera reset on 'Show'
paraview.simple._DisableFirstRenderCameraReset()

# ----------------------------------------------------------------
# setup views used in the visualization
# ----------------------------------------------------------------

# Create a new 'Render View'
renderView1 = CreateView('RenderView')
renderView1.ViewSize = [1270, 1494]
renderView1.AxesGrid = 'Grid Axes 3D Actor'
renderView1.CenterOfRotation = [50.0, 50.0, 50.0]
renderView1.StereoType = 'Crystal Eyes'
renderView1.CameraPosition = [246.58773794649193, -122.31188588172246, 141.7357239030381]
renderView1.CameraFocalPoint = [39.369554387900344, 37.086126761344865, 51.60179380913335]
renderView1.CameraViewUp = [-0.2798552621403546, 0.17043366851954914, 0.9447927798662107]
renderView1.CameraFocalDisk = 1.0
renderView1.CameraParallelScale = 86.60254037844386
renderView1.LegendGrid = 'Legend Grid Actor'

SetActiveView(None)

# ----------------------------------------------------------------
# setup view layouts
# ----------------------------------------------------------------

# create new layout object 'Layout #1'
layout1 = CreateLayout(name='Layout #1')
layout1.AssignView(0, renderView1)
layout1.SetSize(1270, 1494)

# ----------------------------------------------------------------
# restore active view
SetActiveView(renderView1)
# ----------------------------------------------------------------

# ----------------------------------------------------------------
# setup the data processing pipelines
# ----------------------------------------------------------------

# create a new 'Wavelet'
wavelet1 = Wavelet(registrationName='Wavelet1')
wavelet1.WholeExtent = [0, 100, 0, 100, 0, 100]
wavelet1.Center = [50.0, 50.0, 50.0]

# create a new 'Contour'
contour1 = Contour(registrationName='Contour1', Input=wavelet1)
contour1.ContourBy = ['POINTS', 'RTData']
contour1.Isosurfaces = [159.95820999145508, 33.951148986816406, 96.95467948913574, 159.95820999145508, 222.9617404937
744, 285.96527099609375]
contour1.PointMergeMethod = 'Uniform Binning'

# create a new 'Clip'
clip1 = Clip(registrationName='Clip1', Input=contour1)
clip1.ClipType = 'Plane'
clip1.HyperTreeGridClipper = 'Plane'
clip1.Scalars = ['POINTS', 'RTData']
clip1.Value = 191.45997619628906

# init the 'Plane' selected for 'ClipType'
```

kitware

ParaView Enables Live Visualization



kitware

Catalyst Runtime Selectable Backends

- **catalyst-stub**: no external dependencies, suitable for building
- **catalyst-paraview**: full in situ access to ParaView and VTK
- **catalyst-adios:** in transit workflows for asynchronous interactive analysis and visualization
- **catalyst-ascent** (you read that right!): Catalyst passthrough to Ascent platform
- Debug, replay, etc.
- your implementation!