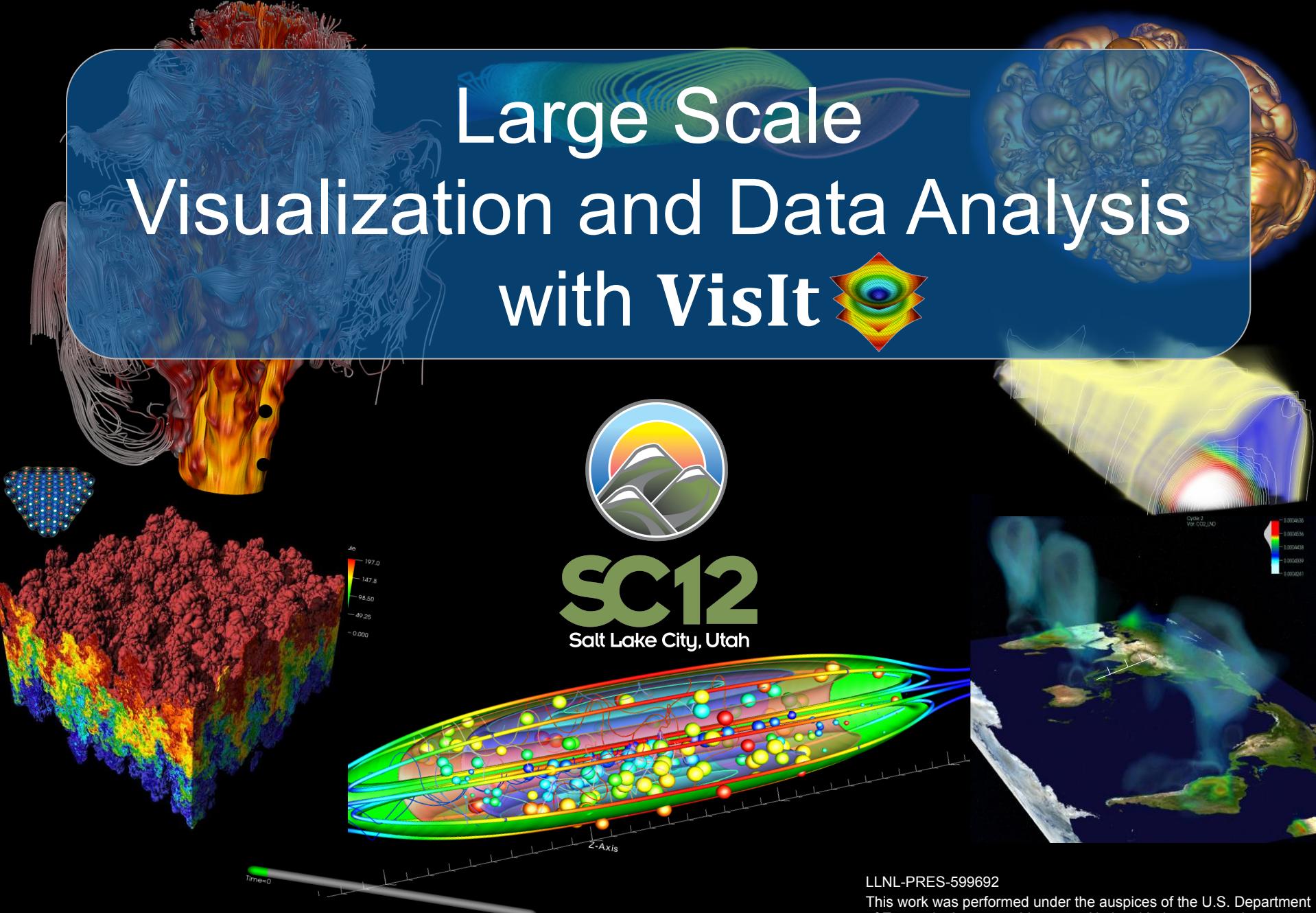


Large Scale Visualization and Data Analysis with VisIt

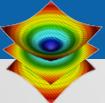


SC12
Salt Lake City, Utah



LLNL-PRES-599692

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC



Tutorial Speakers

Cyrus Harrison



Jean Favre



Hank Childs



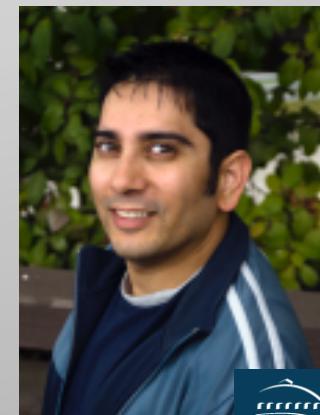
Dave Pugmire

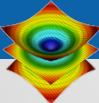


Brad Whitlock



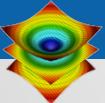
Hari Krishnan





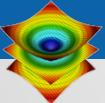
Morning Session (Introductory)

- VisIt Project Introduction (25 min)
- Tutorial Setup (10 min)
- Basics (30 min)
- Data Analysis (25 min)
- <*Break*> (30 min)
- Python Scripting (25 min)
- Movie Making (25 min)
- Alternate Data Representations (20 min)
- New Python Capabilities (10 min)
- Practical Tips for Using VisIt (10 min)



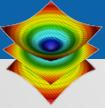
Afternoon Session (Advanced) [1/2]

- Client-server (10 min)
- Parallelism in VisIt (10 min)
- Advanced Topics:
 - Named Selections (10 min)
 - Streamline Computation (20 min)
 - Material Interface Reconstruction (5 min)
- <*Break*> (30 min)

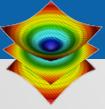


Afternoon Session (Advanced) [2/2]

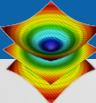
- **Customizing VisIt:**
 - Writing a file format reader (30 min)
 - Writing an analysis operator (15 min)
 - Creating a Python query (10 min)
 - Creating custom Python applications (15 min)
- **In situ processing** (35 min)
- **Wrap-up and Questions** (20 min)



Morning Session

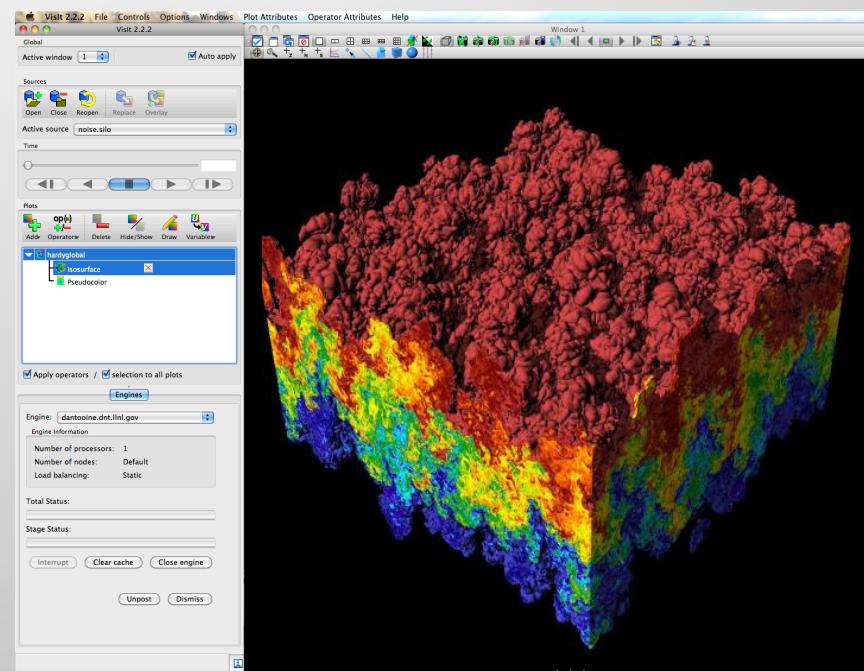


VisIt Project Introduction

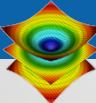


VisIt is an open source, turnkey application for data analysis and visualization of mesh-based data.

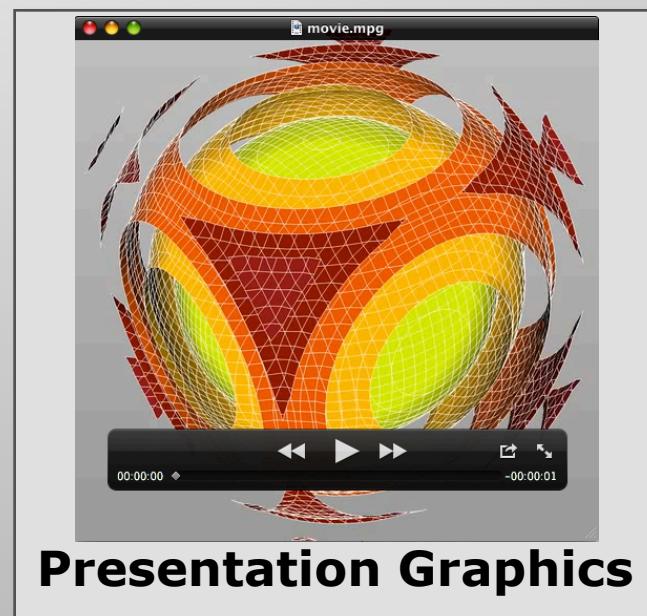
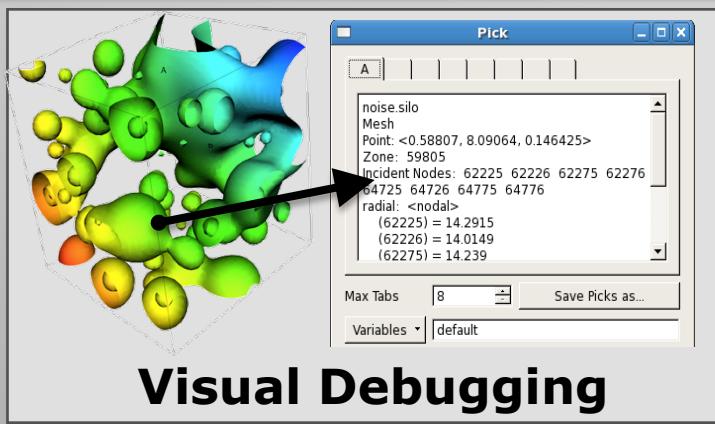
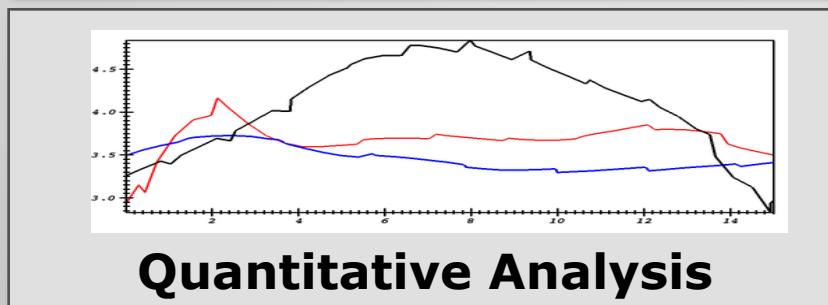
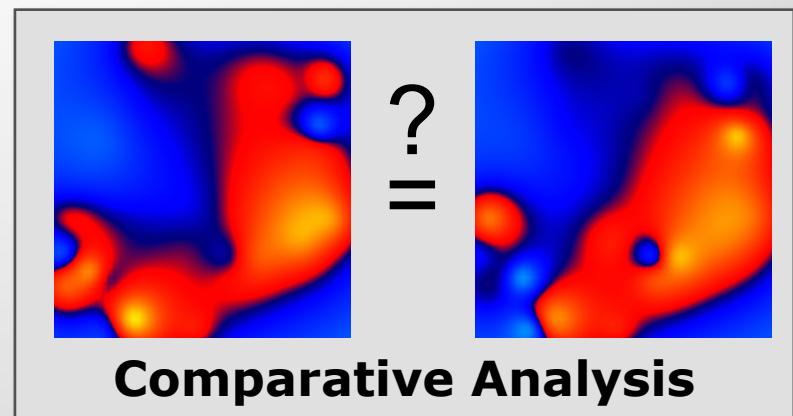
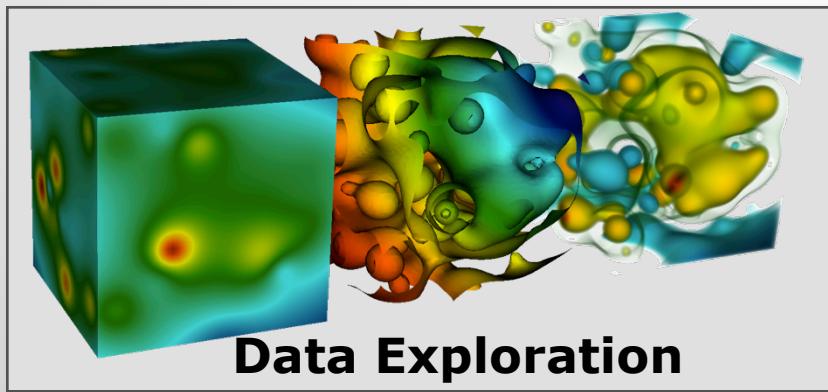
- Production end-user tool supporting scientific and engineering applications.
- Provides an infrastructure for parallel post-processing that scales from desktops to massive HPC clusters.
- Source released under a BSD style license.

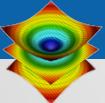


**Density Isovolum of a
3K³ (27 billion cell) dataset**

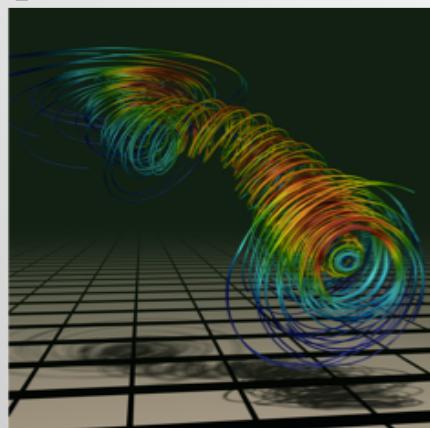


VisIt supports a wide range of use cases.

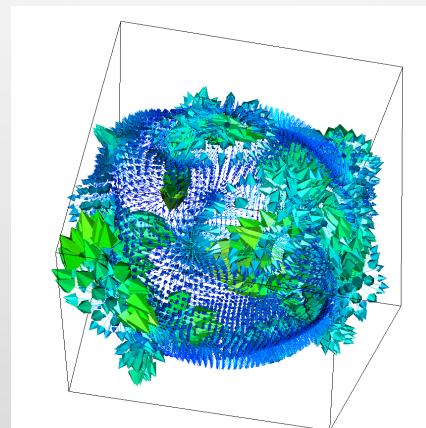




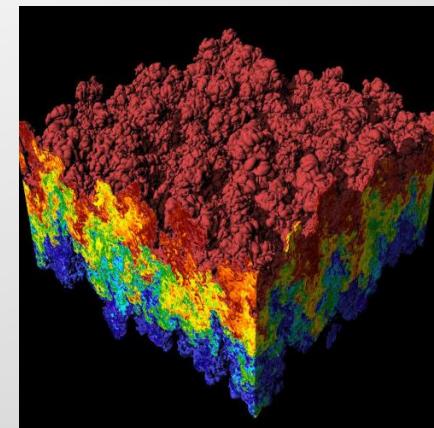
Examples of VisIt's visualization capabilities.



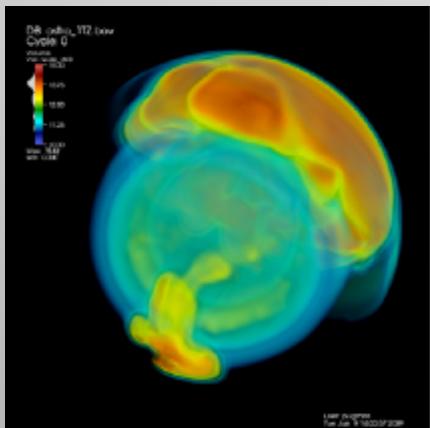
Streamlines



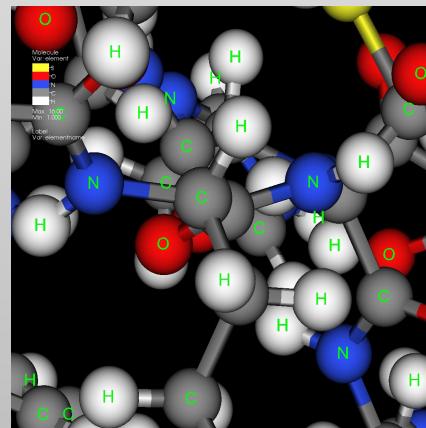
Vector / Tensor Glyphs



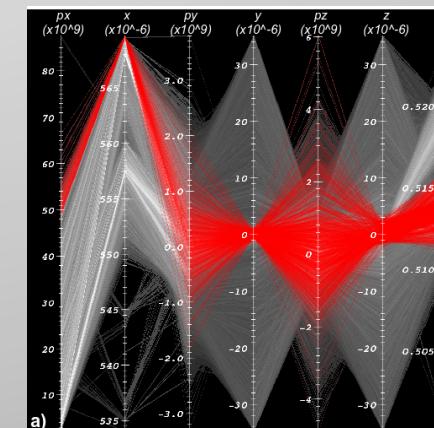
Pseudocolor Rendering



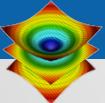
Volume Rendering



Molecular Visualization

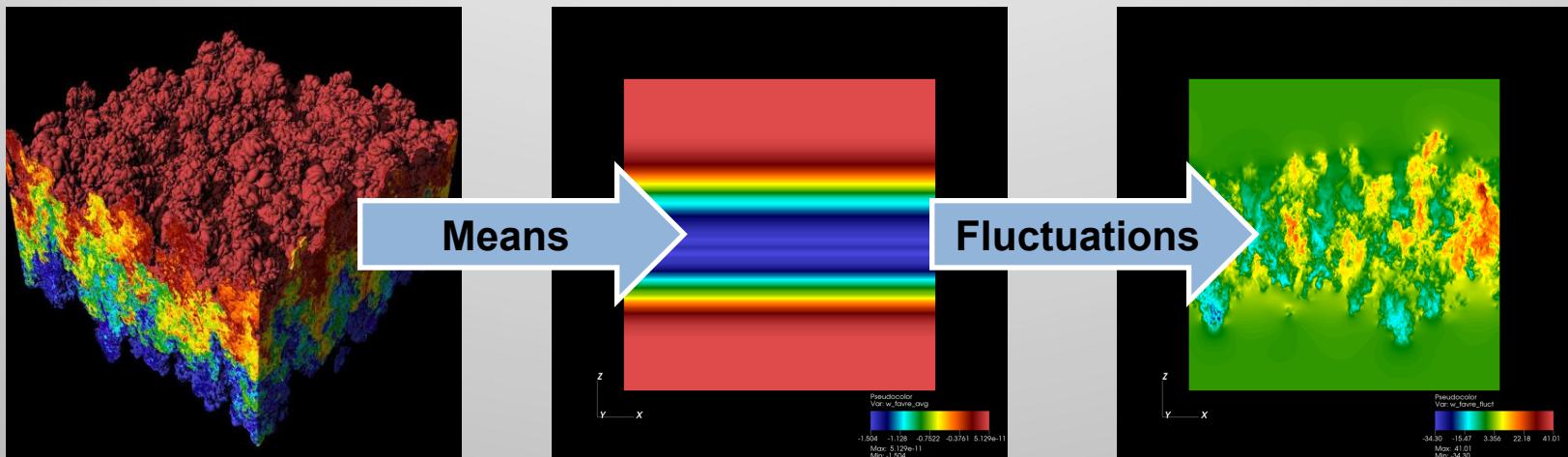


Parallel Coordinates



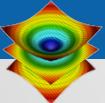
Analysis Example: Turbulence Operators

- **Goal:** Provide a one set of turbulence tools that can be used across multiple codes.
- **Application:** Validate RANS model parameters from high fidelity DNS simulations.



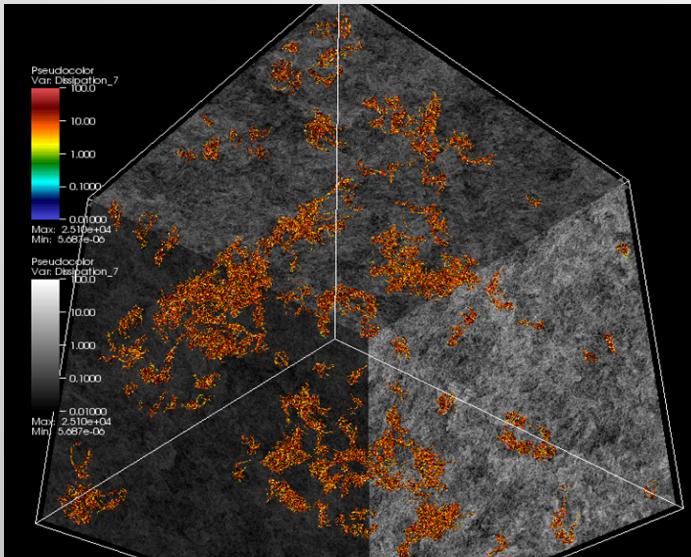
Joint work with Oleg Schilling, LLNL

We are developing scripted building blocks for flow analysis, including field means and fluctuations.



Analysis Example: Evolution of Vorticity

- **Goal:** Identify and track coherent vortical structures in turbulent flow as time evolves.



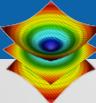
Isosurfaces of a
4K³ (68 billion cell) dataset



Labeled Connected
components

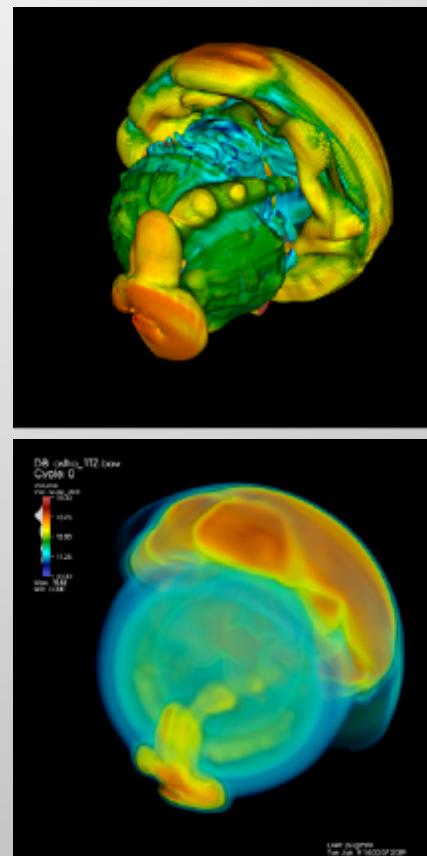
Collaboration with Kelly Gaither, TACC et al (IEEE CG&A July/August 2012)

VisIt was used to calculate isosurfaces, identify connected components, and extract component features.



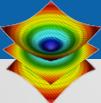
VisIt scales well on current HPC platforms.

Machine	Architecture	Problem Size	# of Cores
Graph	X86_64	20,001³ (8 T cells)	12K
Dawn	BG/P	15,871 ³ (4 T cells)	64K
Franklin	Cray XT4	12,596 ³ (2 T cells)	32K
JaguarPF	Cray XT5	12,596 ³ (2 T cells)	32K
Juno	X86_64	10,000 ³ (1 T cells)	16K
Franklin	Cray XT4	10,000 ³ (1 T cells)	16K
Ranger	Sun	10,000 ³ (1 T cells)	16K
Purple	IBM P5	8,000 ³ (0.5 T cells)	8K



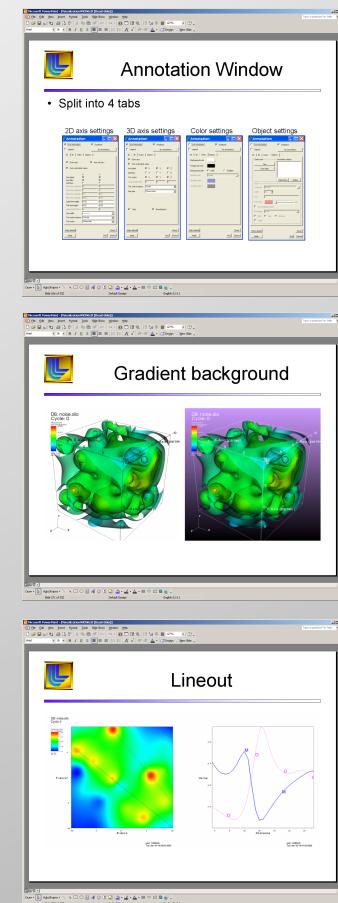
Scaling Studies of Isosurface Extraction and Volume Rendering (2009)

VisIt is also used daily by domain scientists.

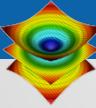


The VisIt team focuses on making a robust, usable product for end users.

- Regular releases (~ 6 / year)
 - Executables for all major platforms
 - End-to-end build process script ``build_visit''
- Customer Support and Training
 - visitusers.org, wiki for users and developers
 - Email lists: visit-users, visit-developers
 - Beginner and advanced tutorials
 - VisIt class with detailed exercises
- Documentation
 - “Getting data into VisIt” manual
 - Python interface manual
 - Users reference manual

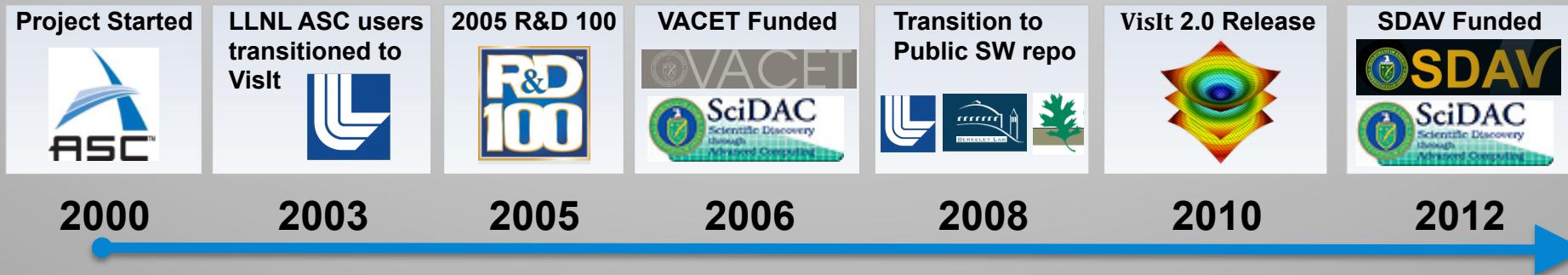


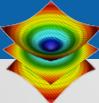
Slides from the VisIt class



VisIt is a vibrant project with many participants.

- The VisIt project started in 2000 to support LLNL's large scale ASC physics codes.
- The project grew beyond LLNL and ASC with research and development from DOE SciDAC and other efforts.
- VisIt is now supported by multiple organizations:
 - LLNL, LBNL, ORNL, UC Davis, Univ of Utah, ...
- Over 75 person years effort, 1.5+ million lines of code.





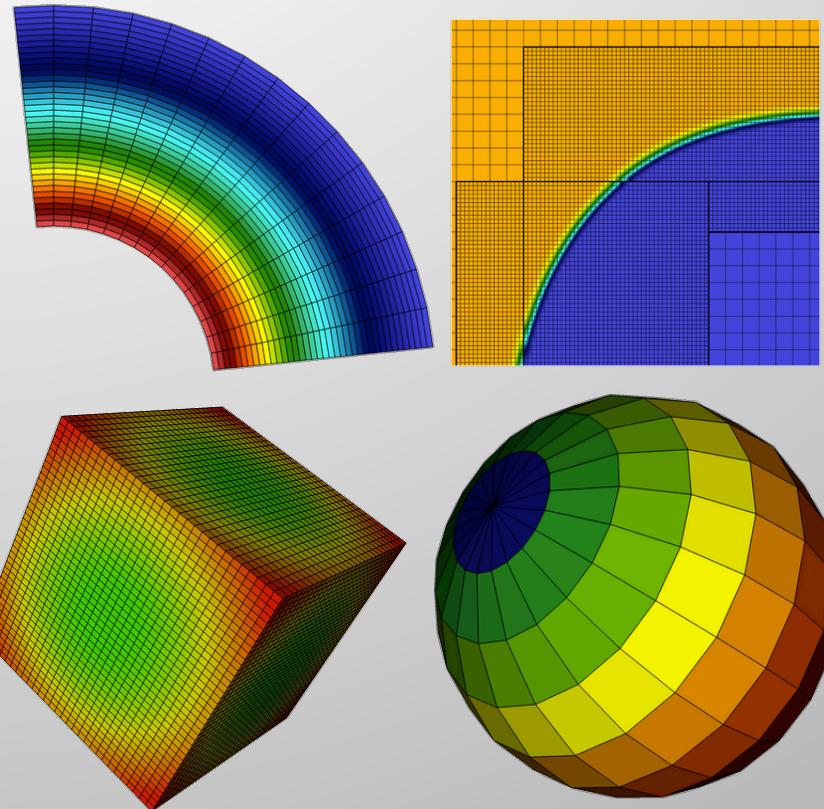
VisIt provides a flexible data model, suitable for many application domains.

- **Mesh Types:**

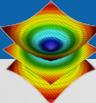
- Point, Curve, 2D/3D
Rectilinear, Curvilinear,
Unstructured
- Domain Decomposed, AMR
- Time Varying

- **Fields:**

- Scalar, Vector, Tensor,
Material volume fractions,
Species

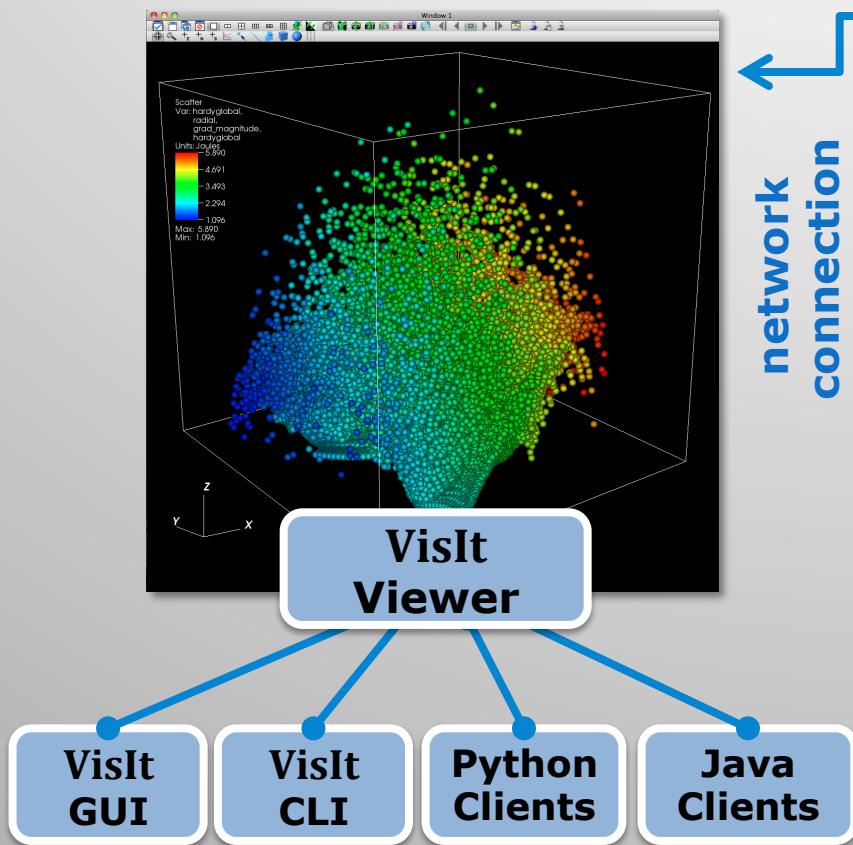


VisIt currently supports over 110 file formats.

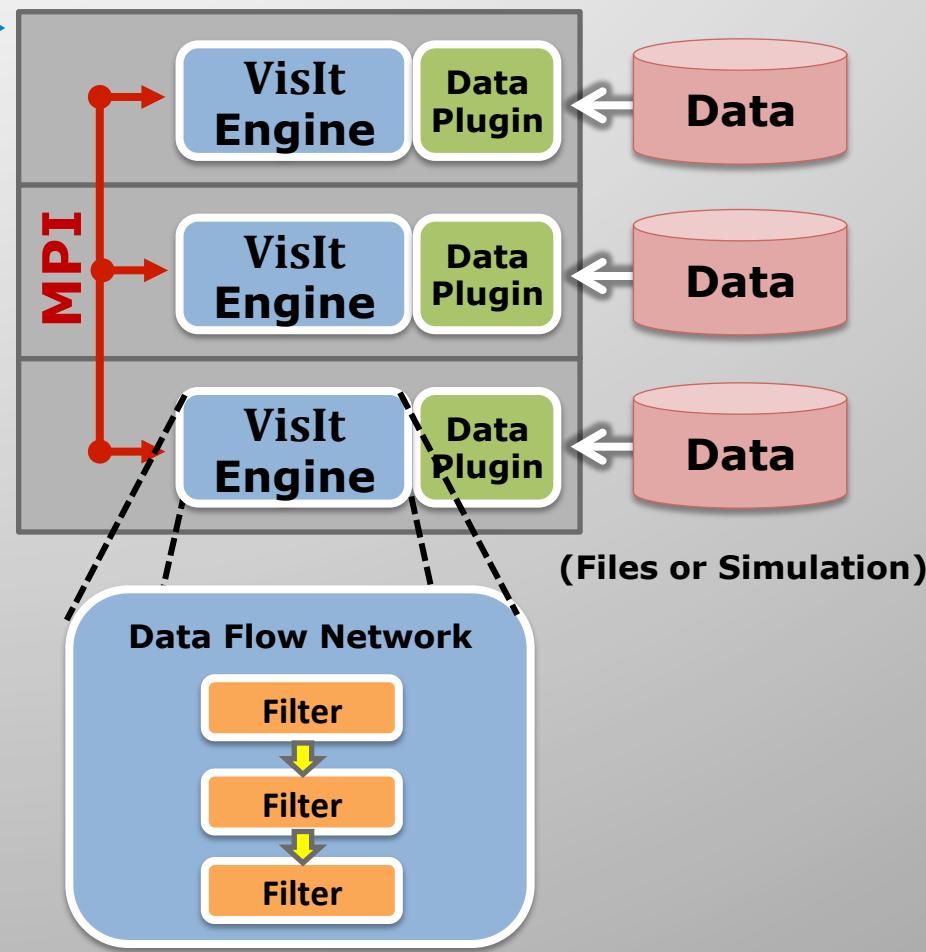


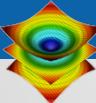
VisIt employs a parallelized client-server architecture.

Local Components

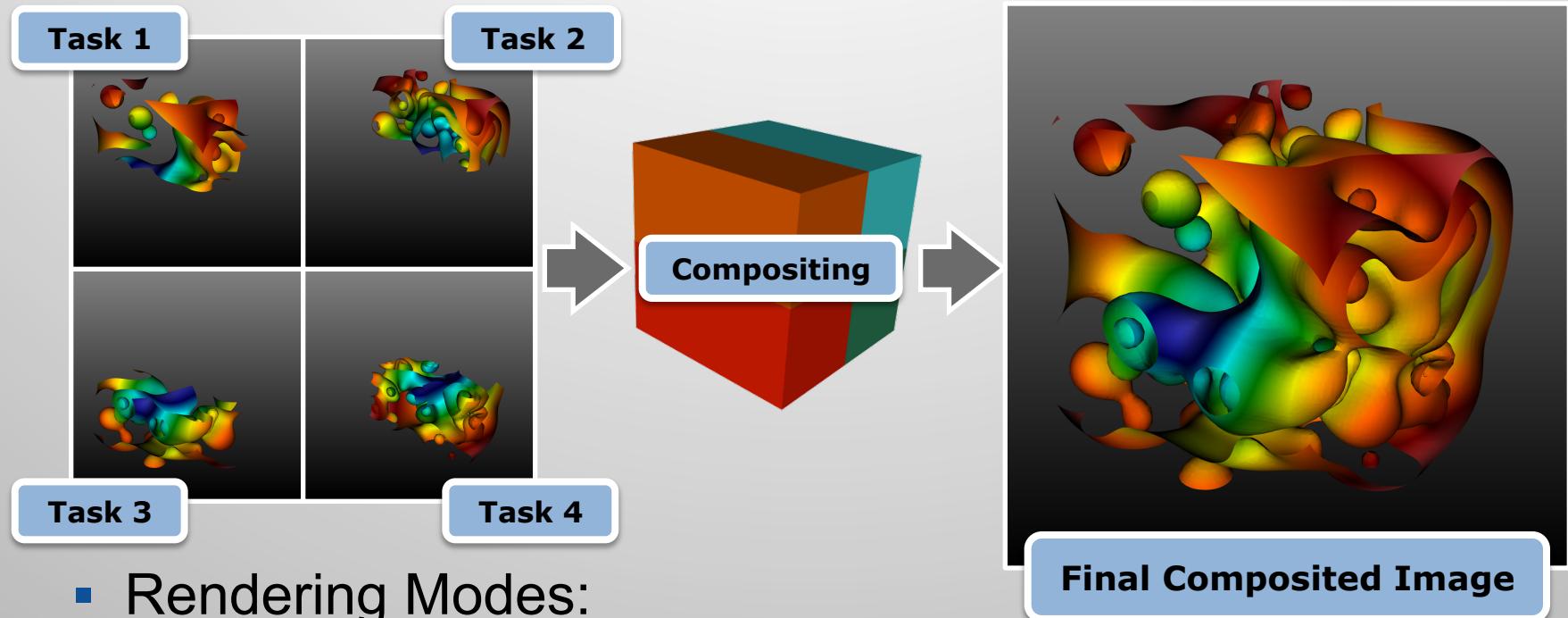


Parallel Cluster

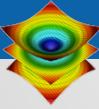




VisIt automatically switches to a scalable rendering mode for large data sets.

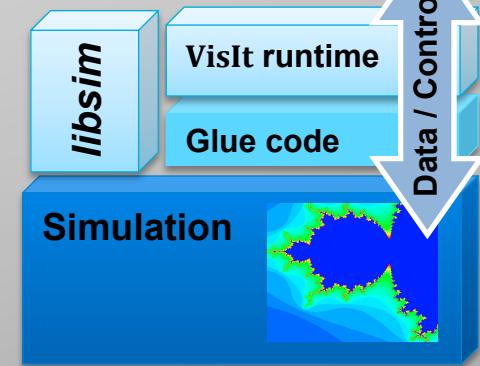
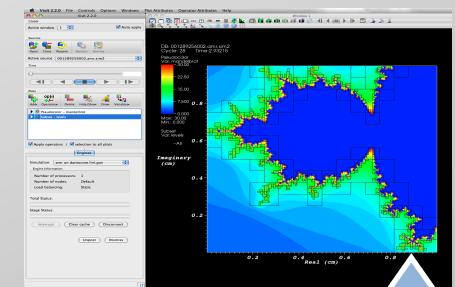


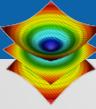
- Rendering Modes:
 - Local (hardware)
 - Remote (software or hardware)
- Beyond surfaces:
 - VisIt also provides scalable volume rendering.



VisIt's infrastructure provides a flexible platform for custom workflows.

- C++ Plugin Architecture
 - Custom File formats, Plots, Operators
 - Interface for custom GUIs in Python, C++ and Java
- Python Interfaces
 - Python scripting and batch processing
 - Data analysis via Python Expressions and Queries.
- *Libsim* library
 - Enables coupling of simulation codes to VisIt for in situ visualization.





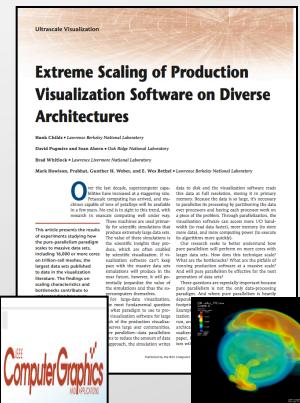
VisIt is used as a platform to deploy visualization research.

- #### ■ Research Collaborations:

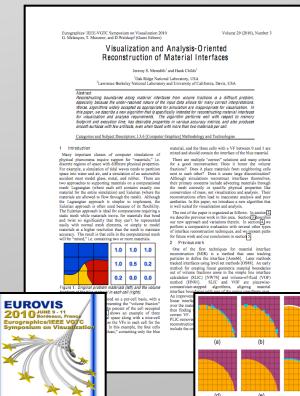


UT/TACC Subcontract

- Research Focus:
 - Next Generation Architectures
 - Parallel Algorithms



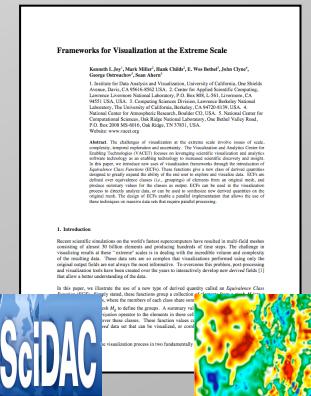
Scaling research: Scaling to 10Ks of cores and trillions of cells.



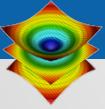
Algorithms research: Reconstructing material interfaces for visualization



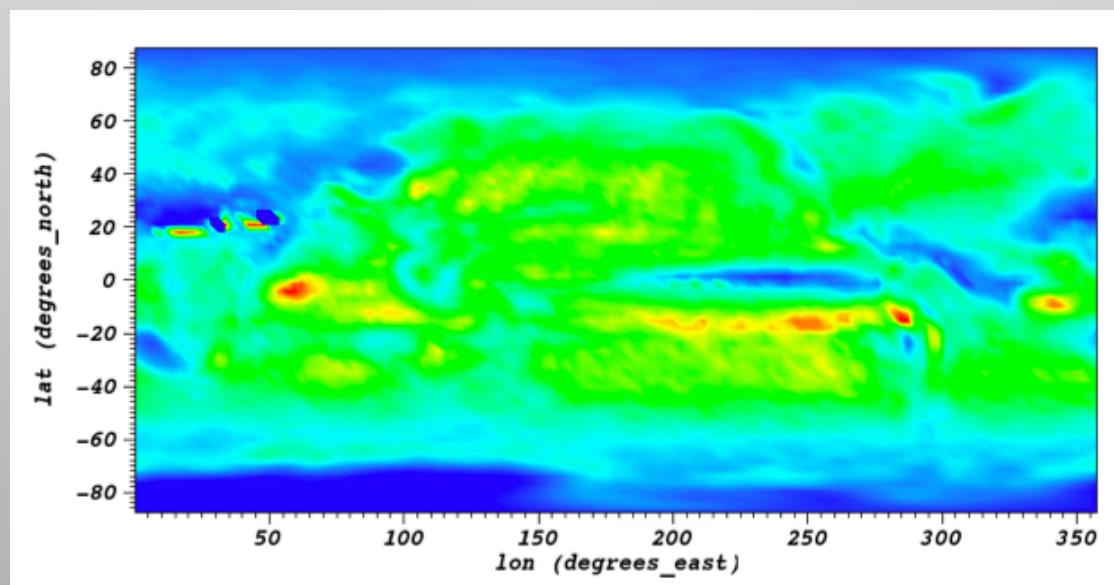
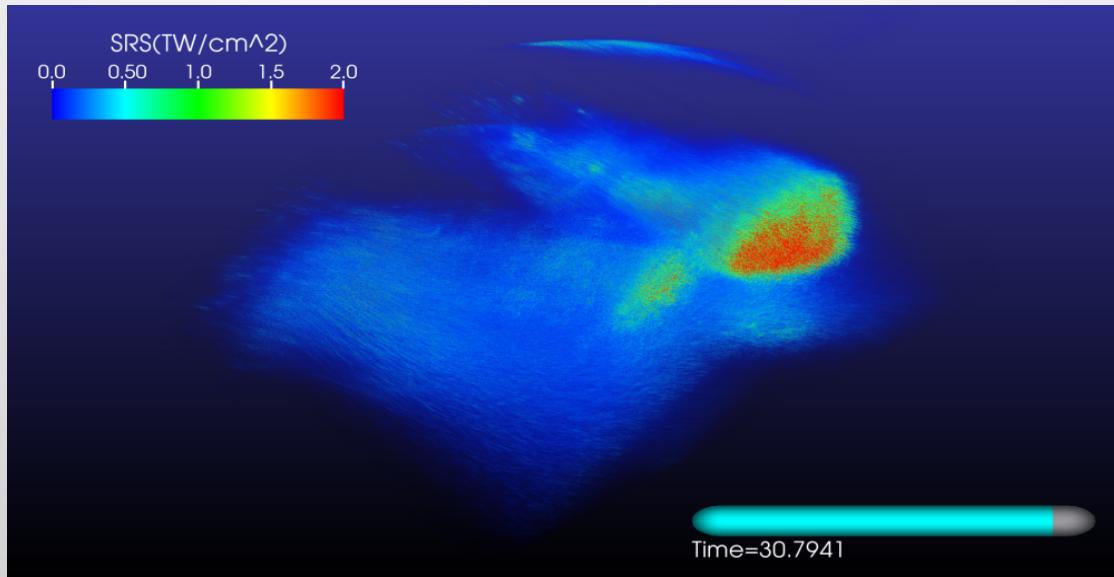
Algorithms research: How to efficiently calculate particle paths in parallel.

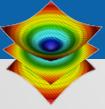


Methods research: How to incorporate statistics into visualization.



More examples of VisIt Usage



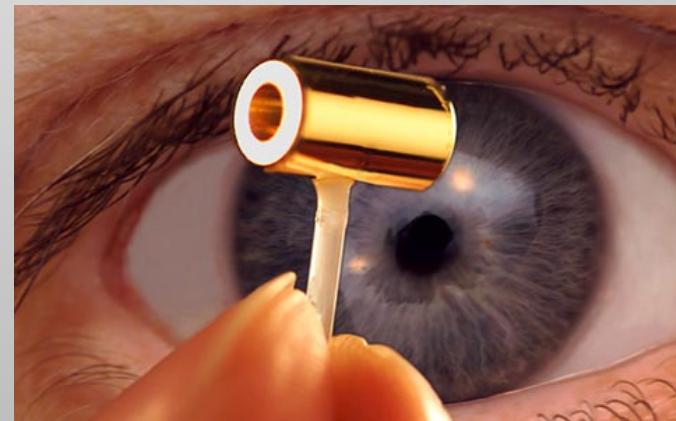


Visualizing a Nation Ignition Facility (NIF) Laser Simulation: NIF Background

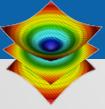
- NIF is laser facility at LLNL that is three football fields in size.
- Ignition Experiments:
 - 192 lasers deposit energy into a gold *hohlraum* which contains a target capsule.
 - The goal is to ignite the deuterium and tritium fuel in the target.



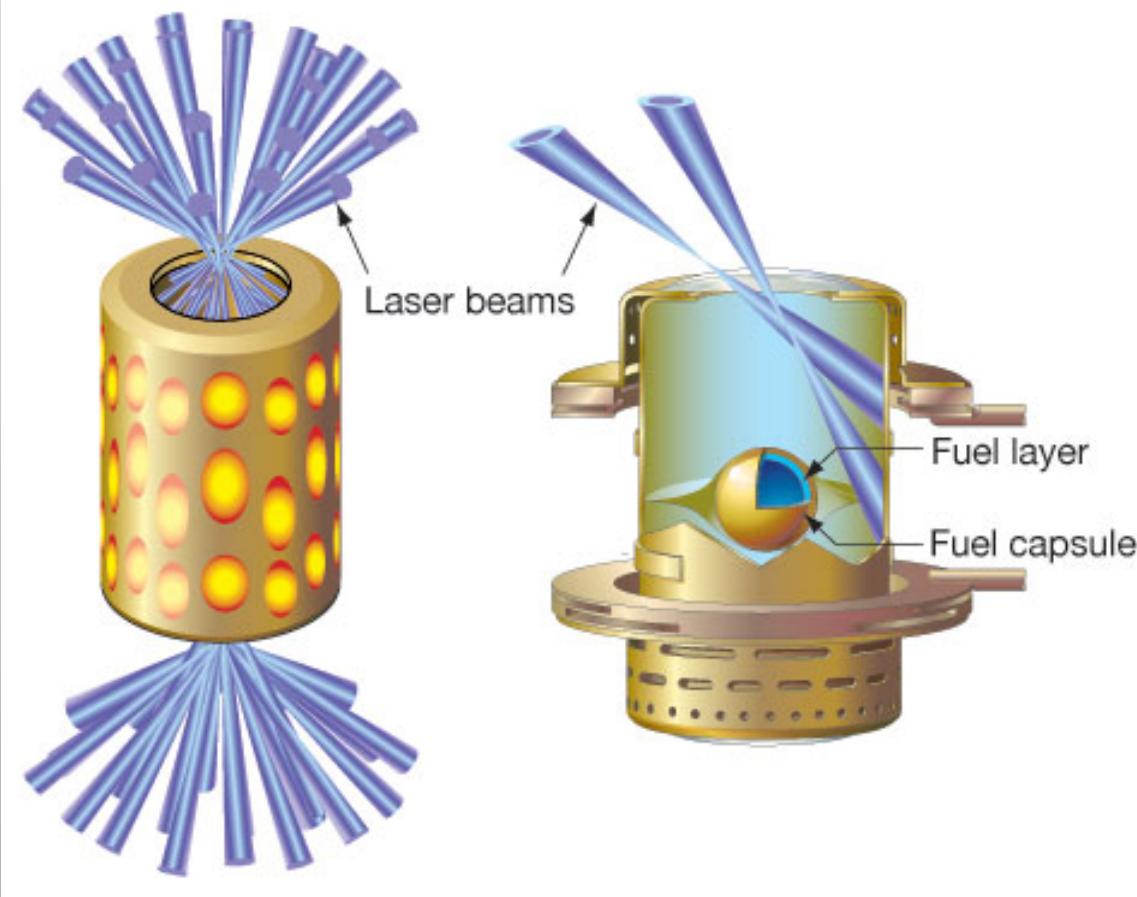
The Nation Ignition Facility



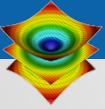
A gold *hohlraum*



NIF Target Physics Background

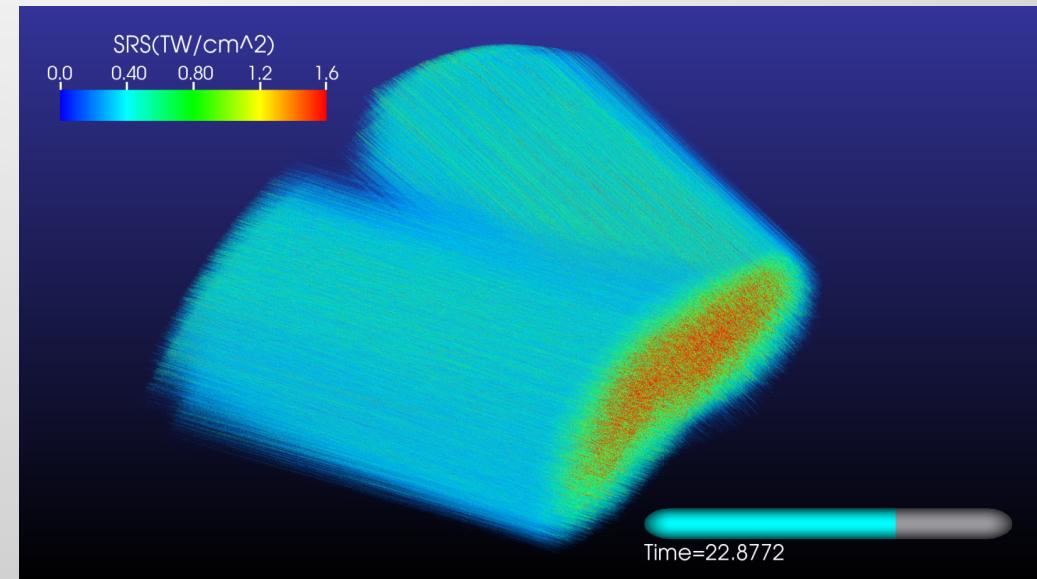


All of the energy of NIF's 192 beams is directed inside a gold cylinder called a hohlraum, which is about the size of a dime. A tiny capsule inside the hohlraum contains atoms of deuterium (hydrogen with one neutron) and tritium (hydrogen with two neutrons) that fuel the ignition process.



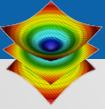
Simulating Two Laser Beams to Understand Back Scatter

- PF3D simulation using 32K CPUs
- Simulates two beams hitting a hohlraum, to quantify how much back scatter is present and the effect of multiple interacting beams.
- Each time step is a structured grid with 220 billion hexahedrons

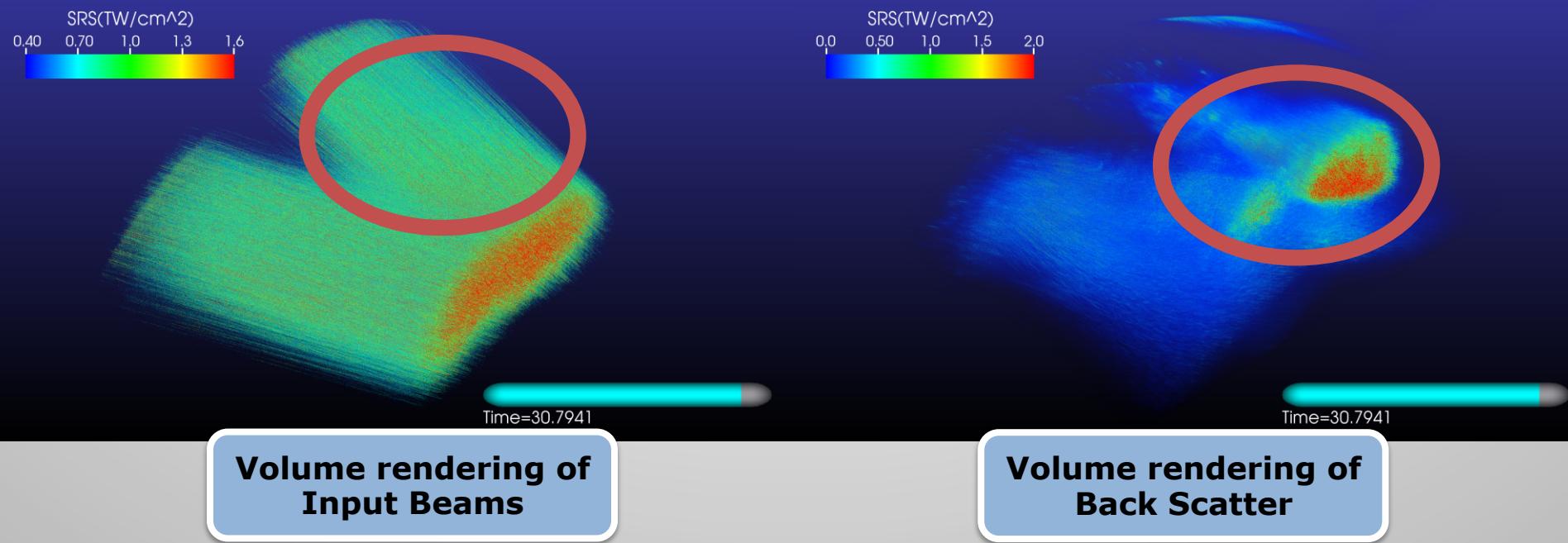


Visualization by Eric Brugger, LLNL

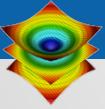
Volume rendering of Input Beams using 2K CPUs



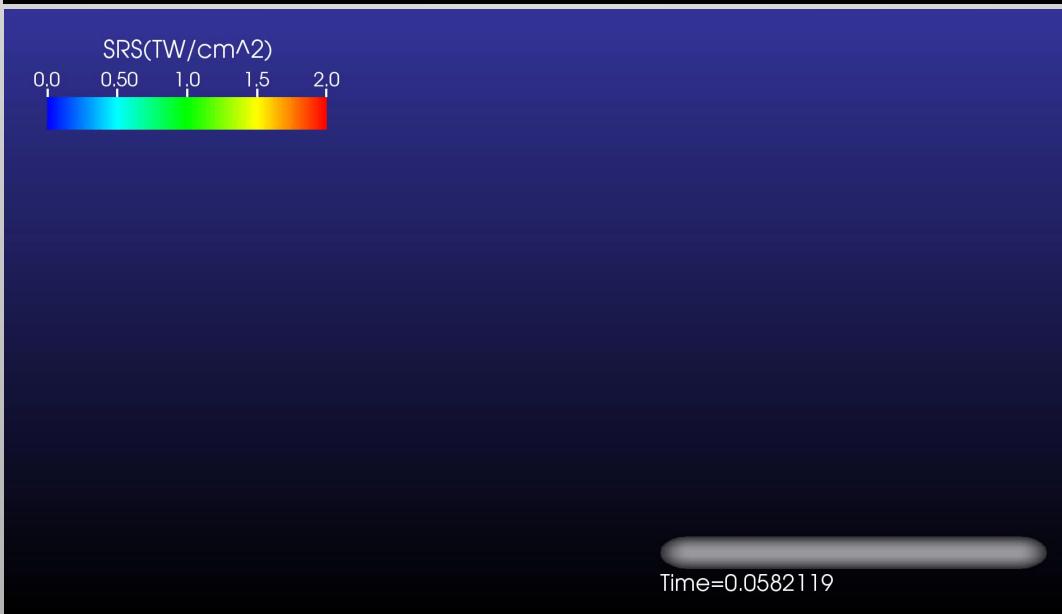
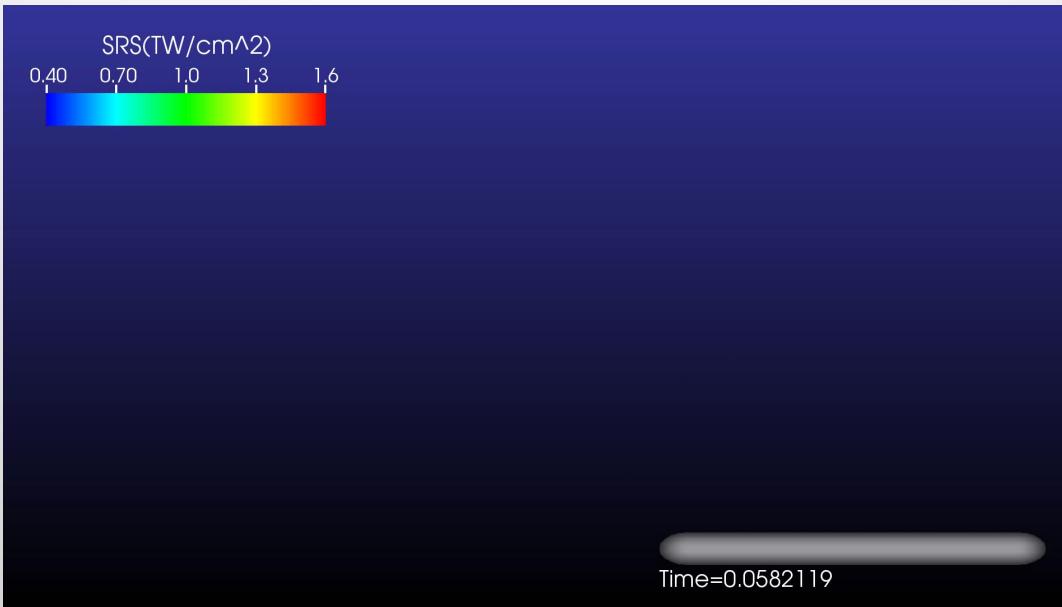
Effect of Back Scatter on Input Beams

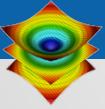


Reduced input beam intensity corresponds
with regions of high back scatter



Visualization of NIF Simulation

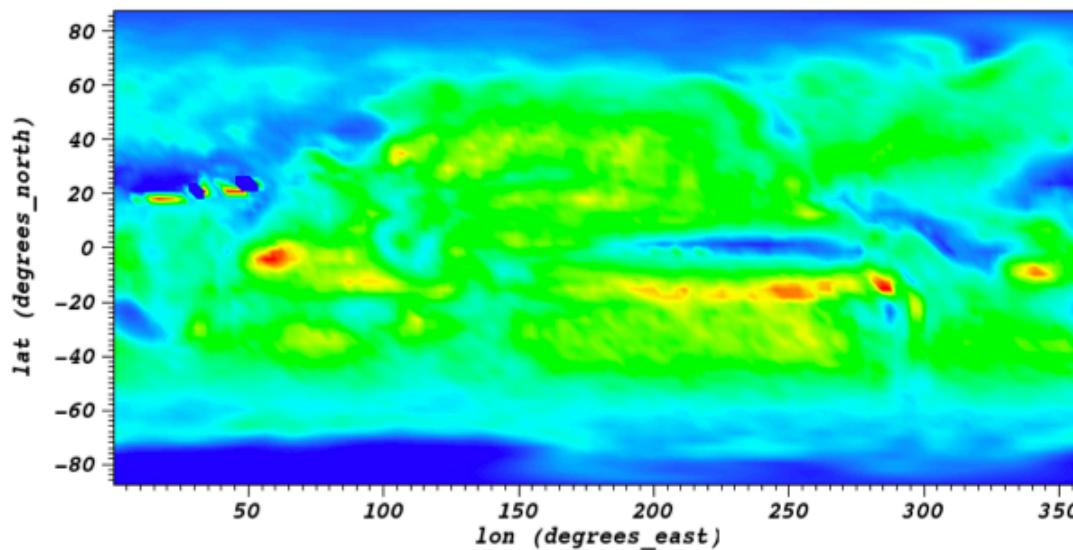




Statistical Analysis for Climate Science

- Couple sophisticated climate analysis techniques (written in R) with VisIt
- Analysis Scaling Example:

Peaks Over Threshold Analysis

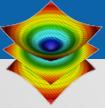


Scaling study

num CPUs	Annual Analysis	Monthly Analysis
1	600.53	7,904.97
2	320.95	5,954.16
4	181.40	4,948.32
8	112.20	2,589.61
16	67.49	2,121.38
32	34.36	1,317.06
64	19.51	673.61
128	10.33	558.18
256	5.99	576.66
512	5.10	317.34
1024	4.59	166.41

Courtesy Dave Pugmire, ORNL

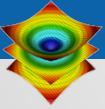
Excellent scalability allows semi-interactive analysis of 100 years of precipitation data.



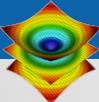
VisIt: What's the Big Deal?

- Everything works at scale
- Robust, usable tool
- Features that span the “power of visualization”:
 - Data Exploration
 - Confirmation
 - Communication
- Features for different kinds of users:
 - Visualization Experts
 - Code Developers
 - Code Consumers

Healthy future: Vibrant Developer and User Communities

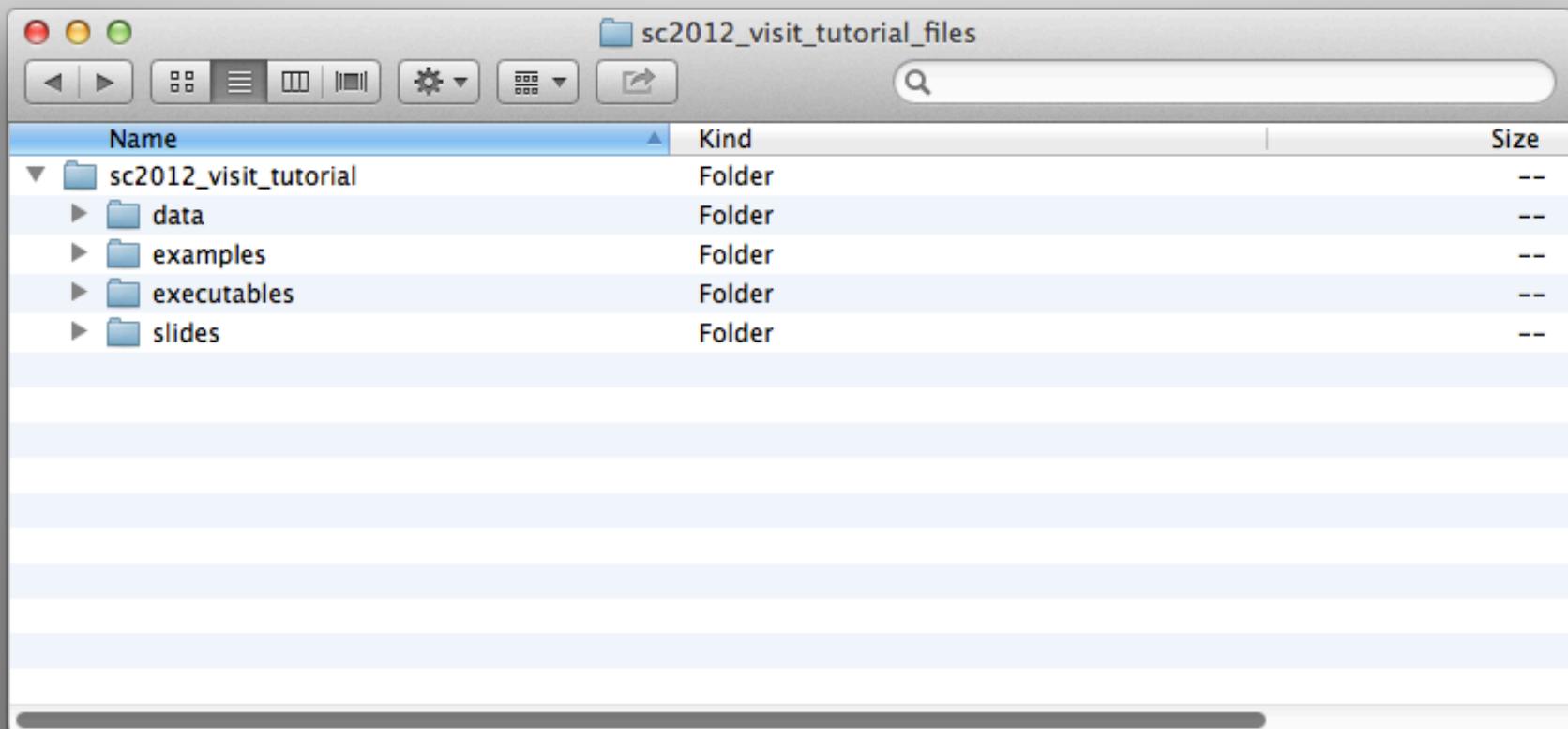


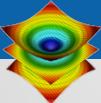
Tutorial Setup: VisIt installation and supporting files



Shared USB Drive

- Copy the **sc2012_visit_tutorial** folder





Shared USB Drive Contents: VisIt 2.5.2

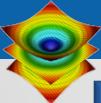
Release Binaries: executables/

Also available at:

<https://wci.llnl.gov/codes/visit/executables.html>

The screenshot shows a web browser window with the URL <https://wci.llnl.gov/codes/visit/executables.html>. The page has a purple header bar with the VisIt logo. Below the header, there are tabs for Home, Downloads (which is selected), Documentation, and What's New. A sidebar on the right includes Index and Contact links. The main content area is titled "VisIt Executables". It contains a paragraph about the page's purpose, mentioning links for Unix, Windows, and Mac OS X systems, and noting that Unix and Mac OS X executables require an install script. It also mentions MD5 and SHA1 checksums for verification. A section titled "Attention Windows users" provides instructions for fixing an uninstaller issue. Below this, a heading "VisIt 2.5.2" is followed by a bulleted list of links: VisIt release notes, VisIt install script, VisIt install notes, VisIt md5 checksums, VisIt sha1 checksums, VisIt sha256 checksums, and VisIt file sizes. At the bottom, there is a table with three columns: platform, architecture, and executable. The table lists five rows of download links for VisIt 2.5.2 executables:

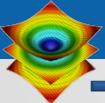
platform	architecture	executable
Linux - x86_32 bit Redhat Enterprise Linux 5, kickit.llnl.gov 2.6.18-308.1.1.el5PAE #1 SMP, gcc 4.1.2 Will work on most Linux x86 systems.	linux-intel	download
Linux - x86_64 64 bit Redhat Enterprise Linux 5, sidious.llnl.gov 2.6.18-308.4.1.el5 #1 SMP, gcc 4.1.2 Will work on most Linux x86_64 systems.	linux-x86_64	download
Linux - x86_64 64 bit Redhat Enterprise Linux 6, hoth.llnl.gov 2.6.32-220.13.1.el6.x86_64 #1 SMP, gcc 4.4.6	linux-x86_64	download
Linux - x86_64 64 bit Ubuntu 11.04, ubuntu1104-64.llnl.gov 2.6.38-8-generic #42-Ubuntu SMP, gcc 4.5.2	linux-x86_64	download
Linux - x86_64 64 bit		



Installing VisIt

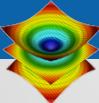
Select a binary for your platform from **executables/** and install:

- Linux/Unix:
 - Untar
- Mac:
 - Open DMG and copy VisIt app bundle to Desktop
- Windows:
 - Run installer program



Testing your VisIt install

- Linux/Unix:
 - >path/to/visit/bin/visit
- Mac:
 - Double click VisIt app bundle
- Windows:
 - Launch from start menu

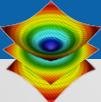


Shared USB Drive Contents: Example Datasets

Data Files:
data/

The screenshot shows a web browser window with the title 'Tutorial Data - Visitusers.org'. The URL in the address bar is 'www.visitusers.org/index.php?title=Tutorial_Data'. The page content includes the 'Visit users .org' logo, a navigation menu with links like 'Cyrus', 'my talk', 'my preferences', 'my watchlist', 'my contributions', and 'log out', and a main section titled 'Tutorial Data' which describes example data sets used in the SC2012 Visit Tutorial. It lists download links for a Tarball and a Zip file. Below this, there are two 3D visualization renderings: 'example.silo' on the left and 'sedov.visit' on the right, both showing complex, multi-colored volumes.

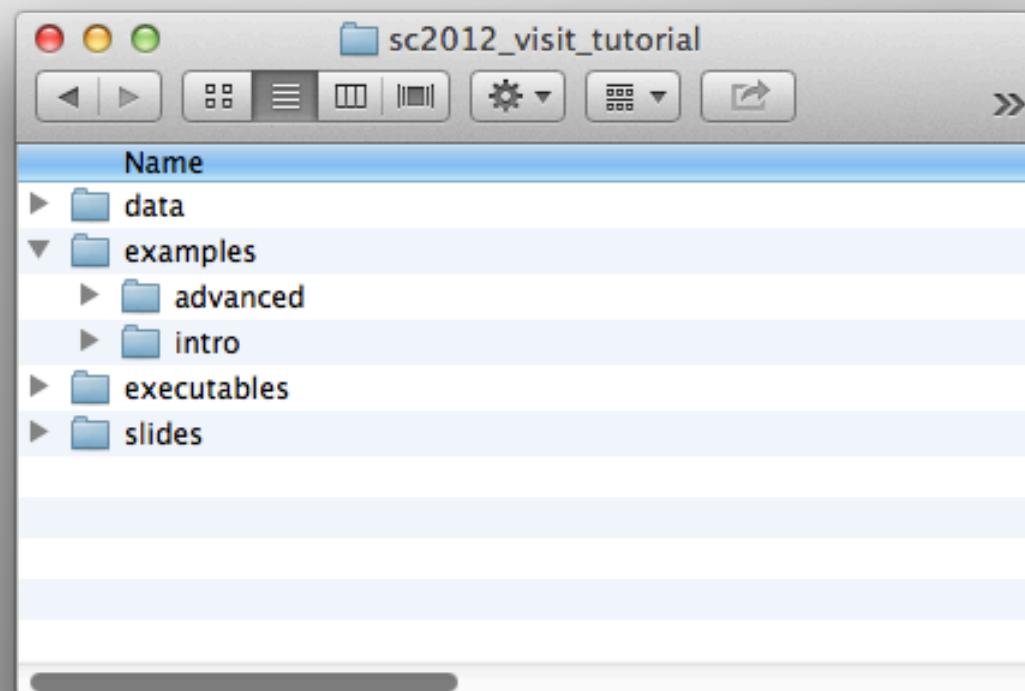
Also available at:
http://www.visitusers.org/index.php?title=Tutorial_Data



Shared USB Drive Contents: Example Code + Scripts

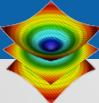
Example Files:

examples/



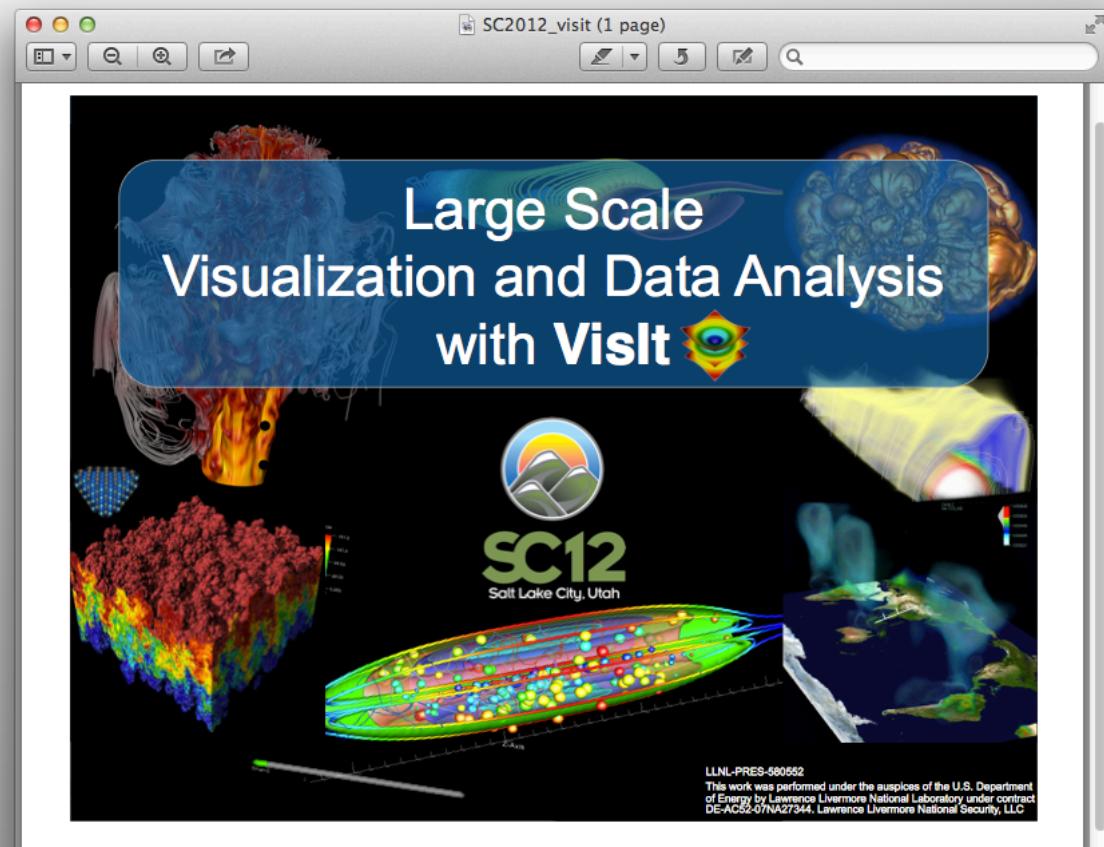
Also available at:

http://www.visitusers.org/index.php?title=Tutorial_Examples



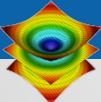
Shared USB Drive Contents: Tutorial Sides

Tutorial PDFs:
slides/



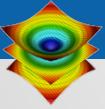
Also available at:

http://visitusers.org/index.php?title=Short_Tutorial

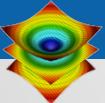


Before we begin ...

- Important: Ask questions any time!
- Tutorial script located at [visitusers.org](http://visitusers.org/index.php?title=Short_Tutorial) under “short tutorial”:
 - [http://visitusers.org/index.php?title=Short Tutorial](http://visitusers.org/index.php?title=Short_Tutorial)
- Reminder: We will discuss file format issues and how to get help at the end of the morning session.

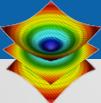


Basics

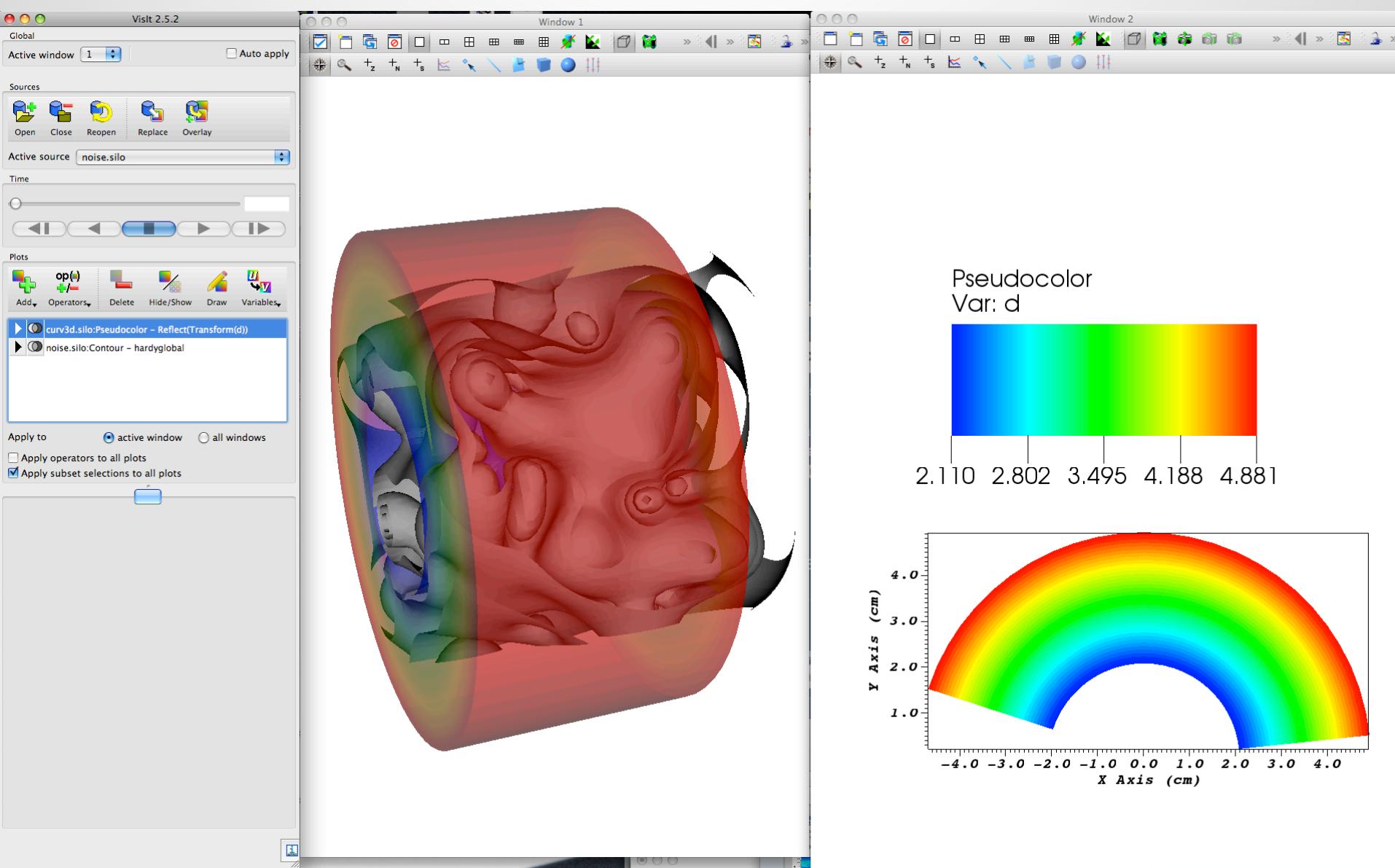


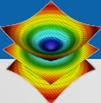
The key concepts of VisIt

- **Sources:** where you are getting data from
 - **Windows:** where visualizations are displayed
-
- **Plots:** how you render data
 - **Operators:** how you manipulate data
 - **Interactors:** what your mouse cursor does
 - **Tools:** advanced ways to control plots and operators
 - **Expressions:** mechanism for generating derived quantities
 - **Queries:** how to access quantitative information

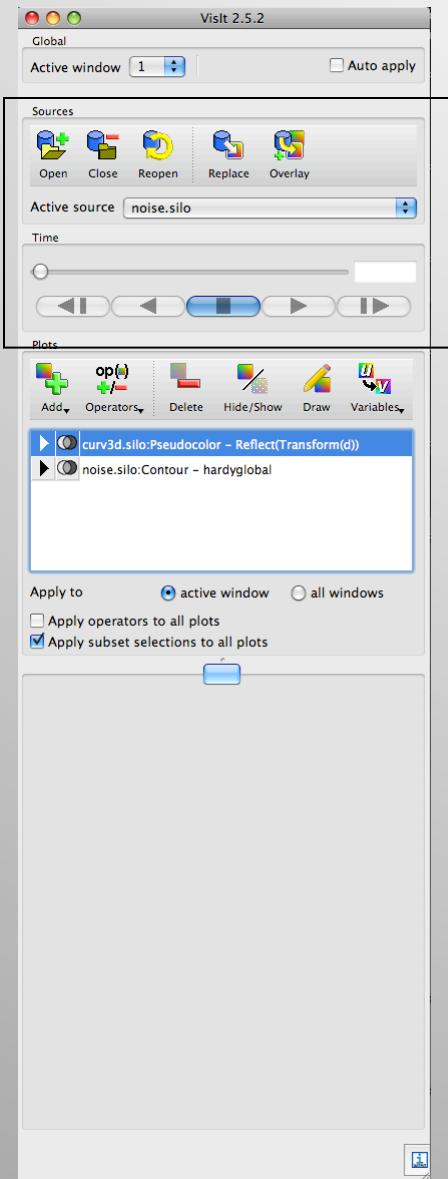


VisIt's GUI interface

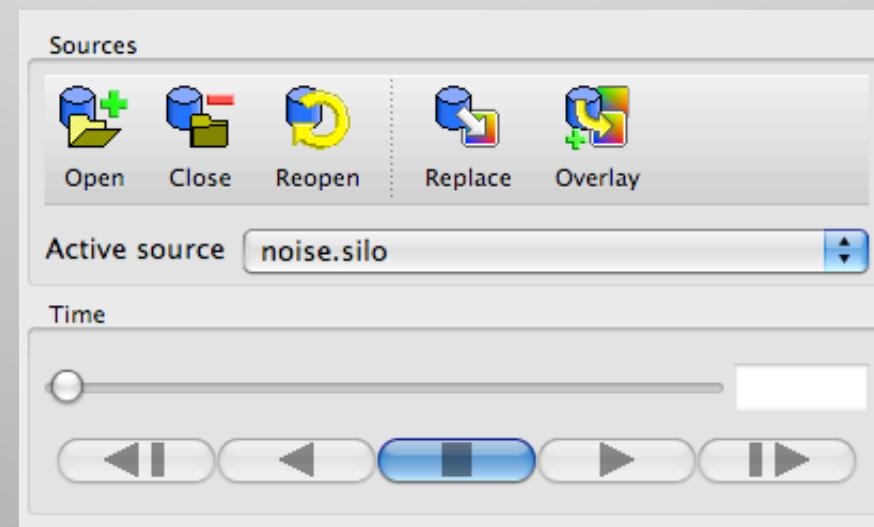


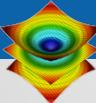


Sources Overview



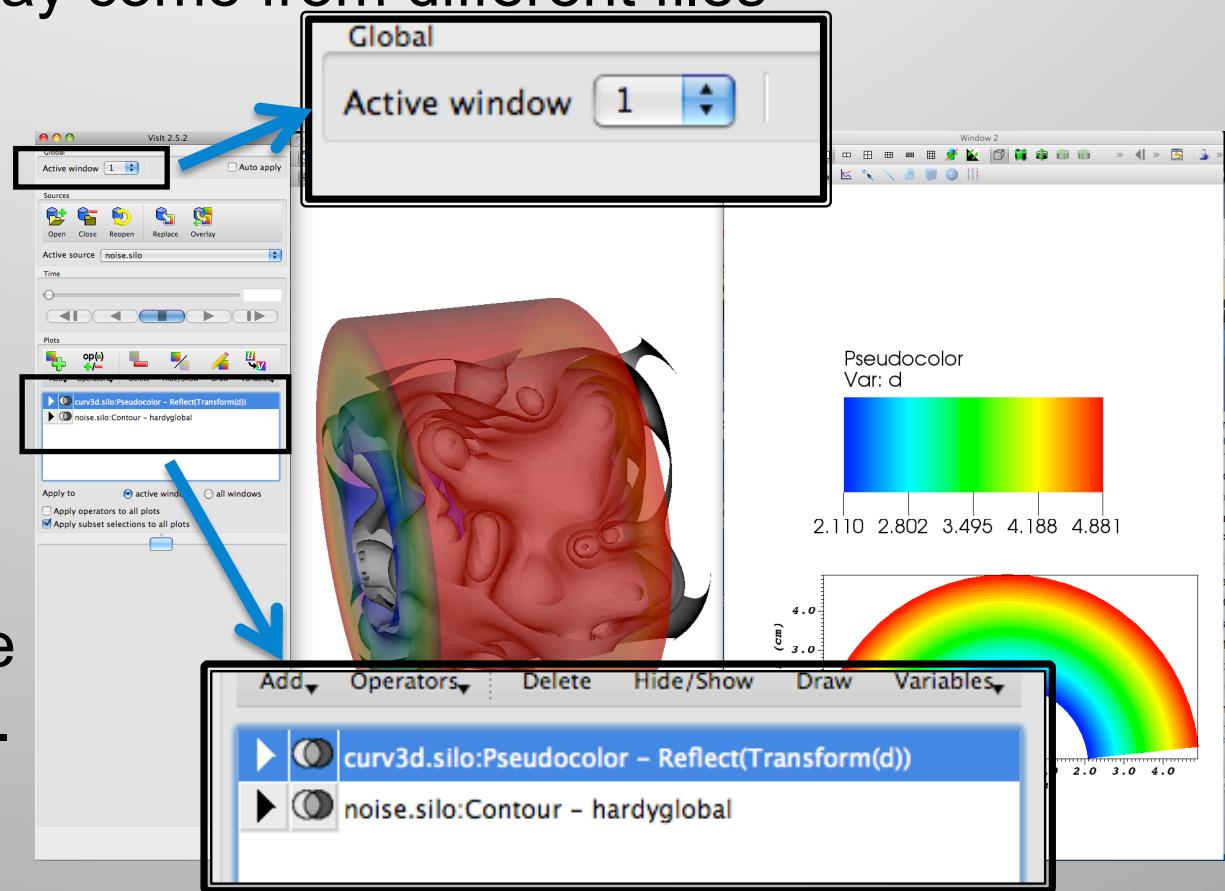
- Sources bring data in from files.
- You may use multiple sources.
- The “active source” is declares which file new plots should read from.
 - The active source can be changed at any time.
- If a source has multiple time slices, then the time slider is activated and may be changed.

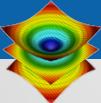




Windows Overview

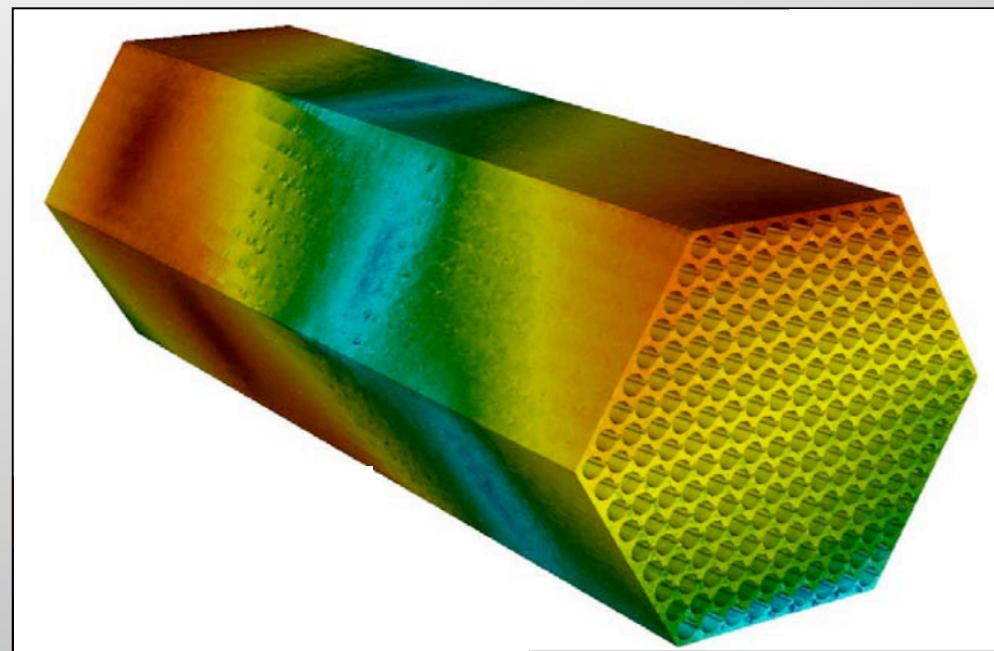
- You may have one or more windows
- Each window may have one or more plots
 - Those plots may come from different files
- Only one window is “active” at a time.
 - The user interface focuses on the active window.



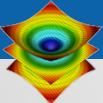


Pseudocolor plot

- Maps scalar fields (e.g., density, pressure, temperature) to colors.

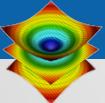


- Instructions for this mapping are encoded in the plot's *attributes*.

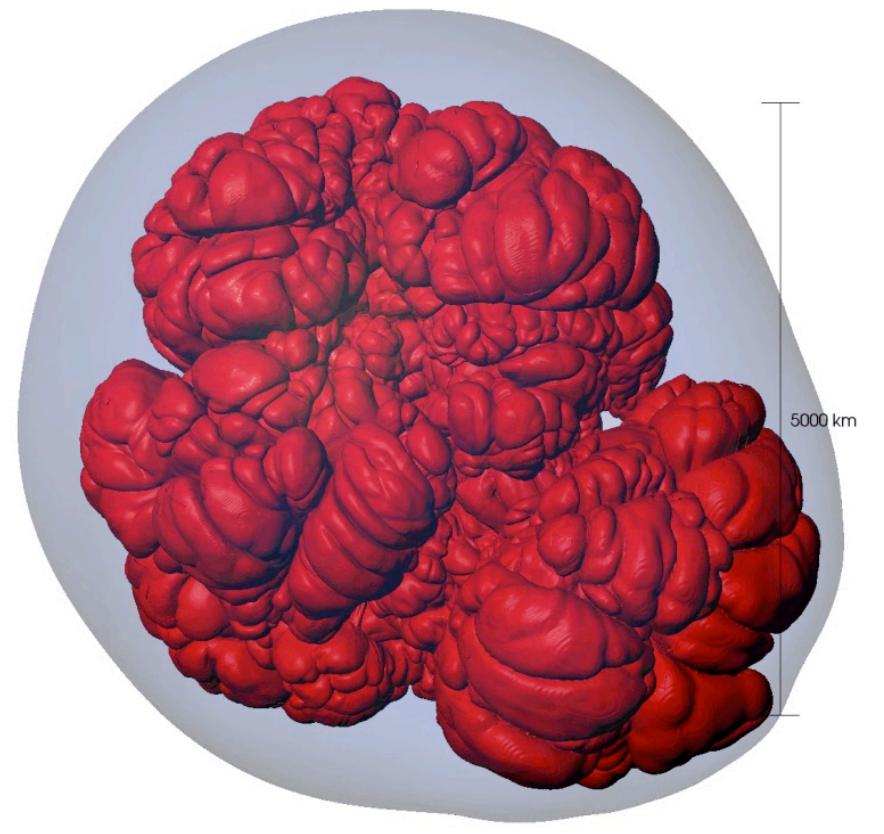


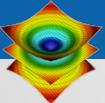
The key concepts of VisIt

- **Sources:** where you are getting data from
 - **Windows:** where visualizations are displayed
- **Plots:** how you render data
 - **Operators:** how you manipulate data
 - **Interactors:** what your mouse cursor does
 - **Tools:** advanced ways to control plots and operators
- **Expressions:** mechanism for generating derived quantities
 - **Queries:** how to access quantitative information

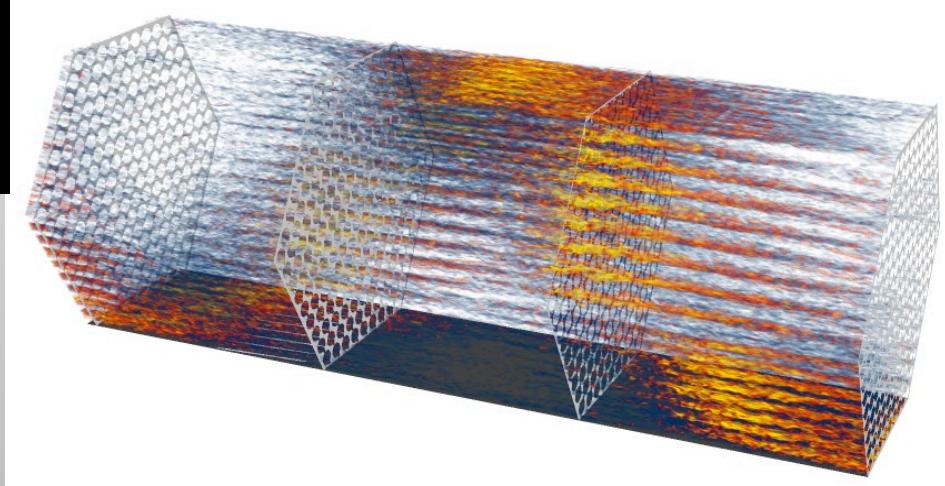
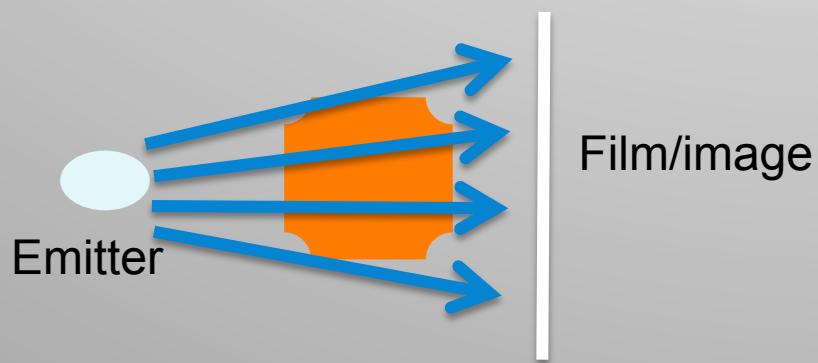
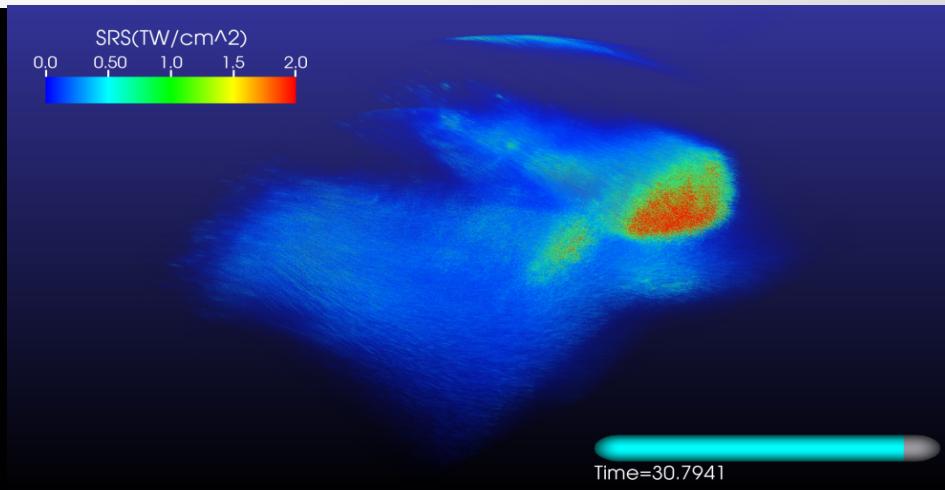
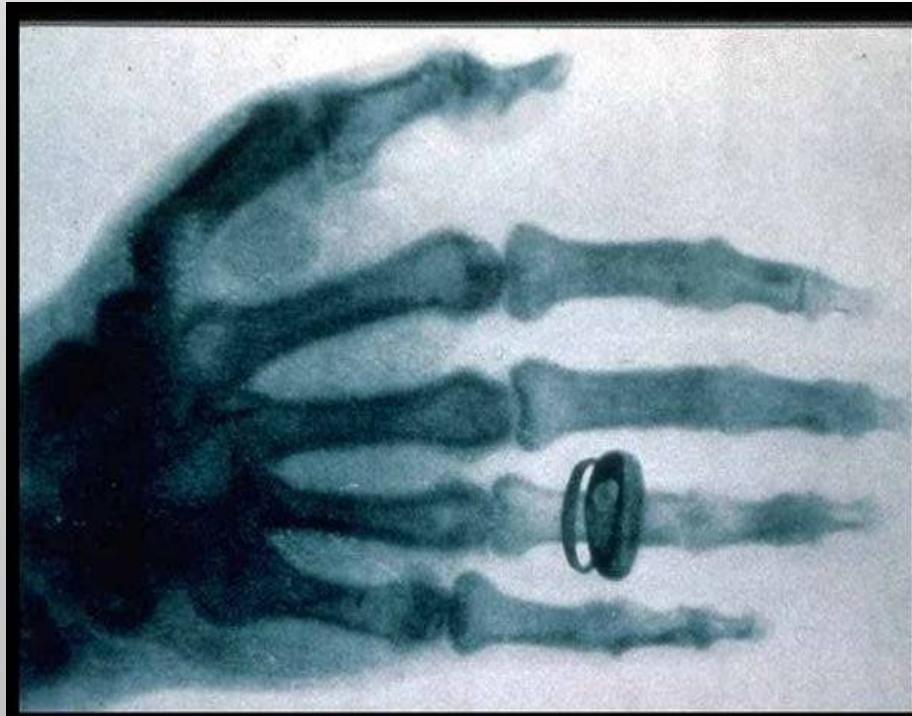


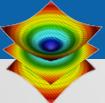
Contour / Isosurface





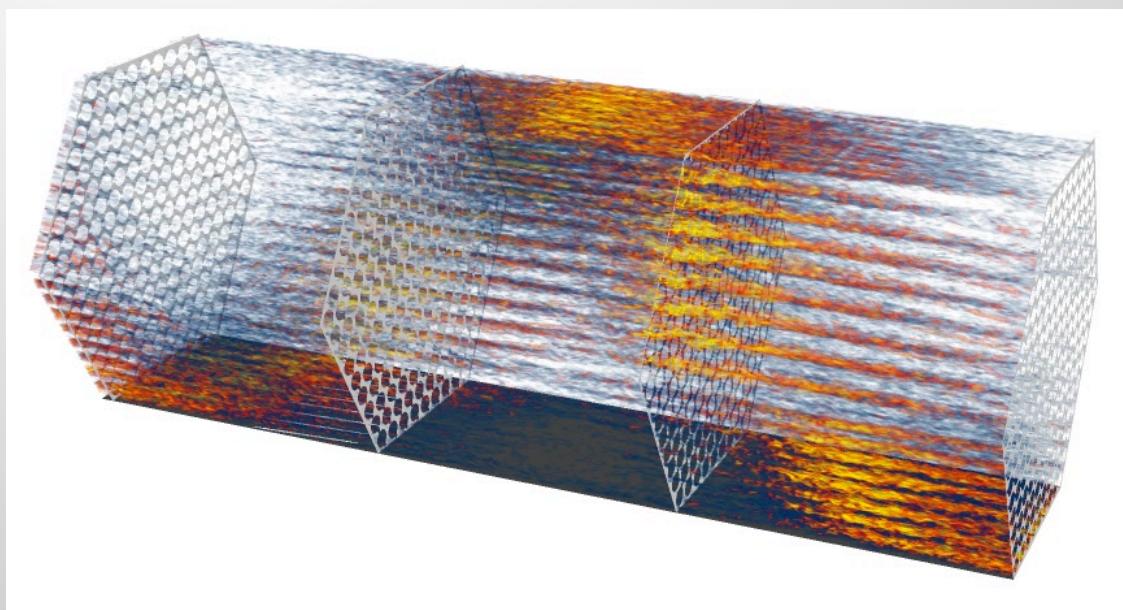
Volume rendering





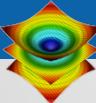
Visualizing a Nuclear Reactor Simulation

- Nek5000 simulation run on Argonne's Intrepid cluster
- Simulates coolant flowing around a nuclear reactor
- Each time step is an unstructured grid with 2 billion hexahedrons

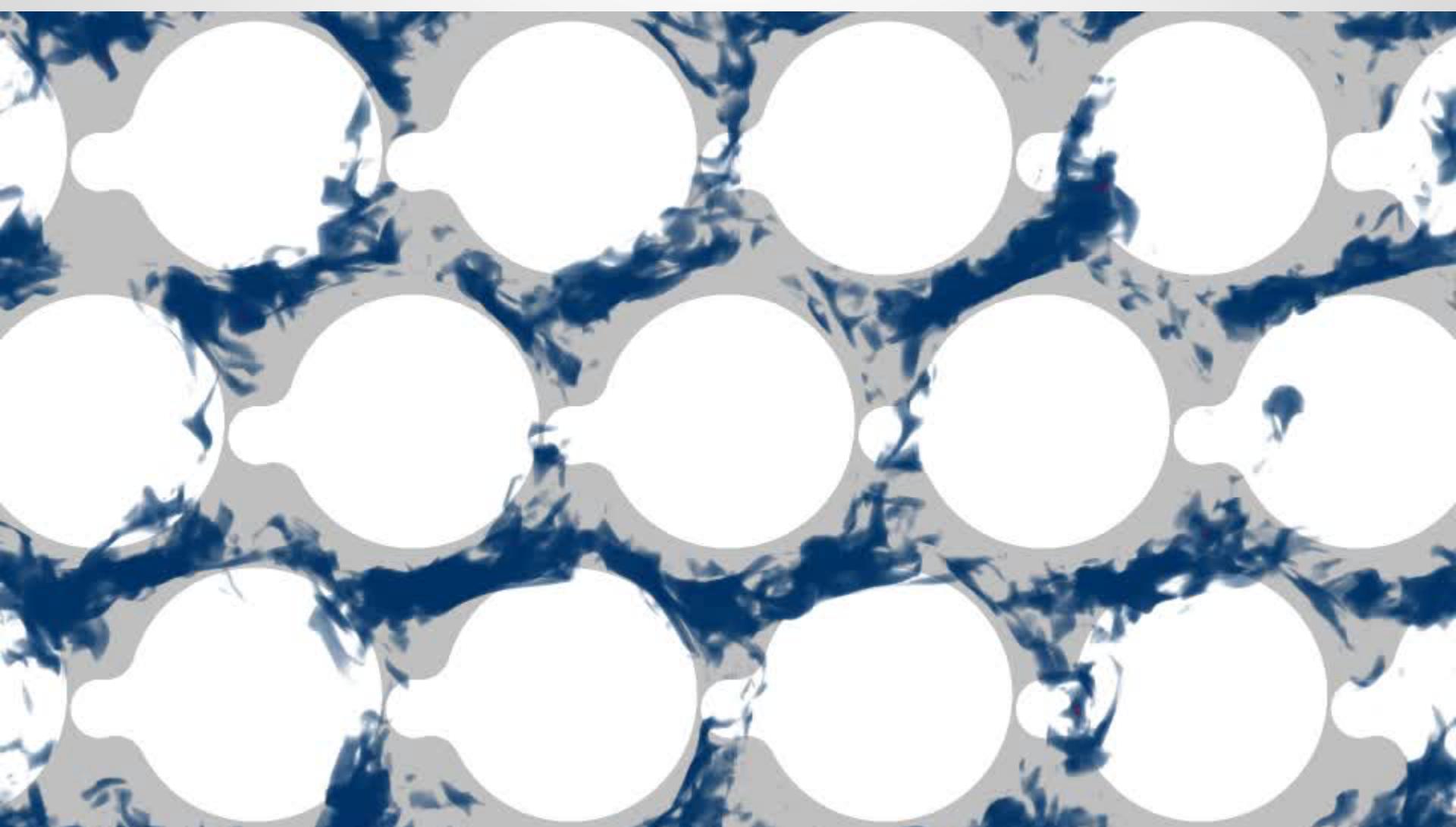


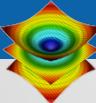
Visualization by Hank Childs, LBNL

**Volume rendering
using 128 CPUs**

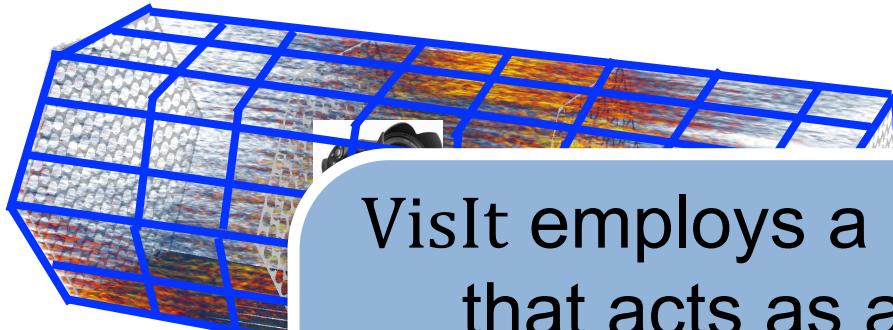


Nuclear Reactor Visualization

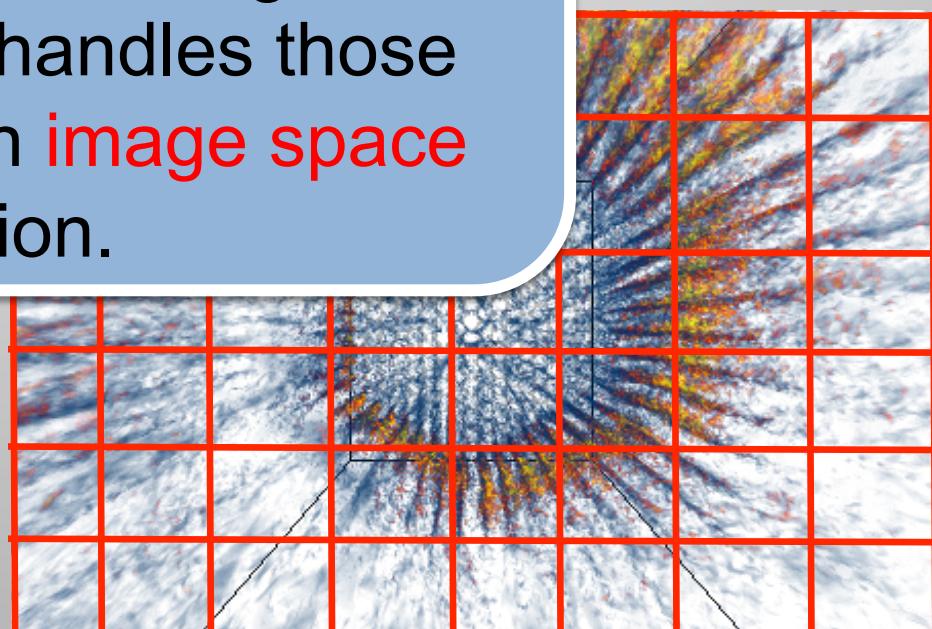


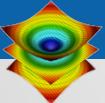


Volume rendering of massive data requires sophisticated algorithms.

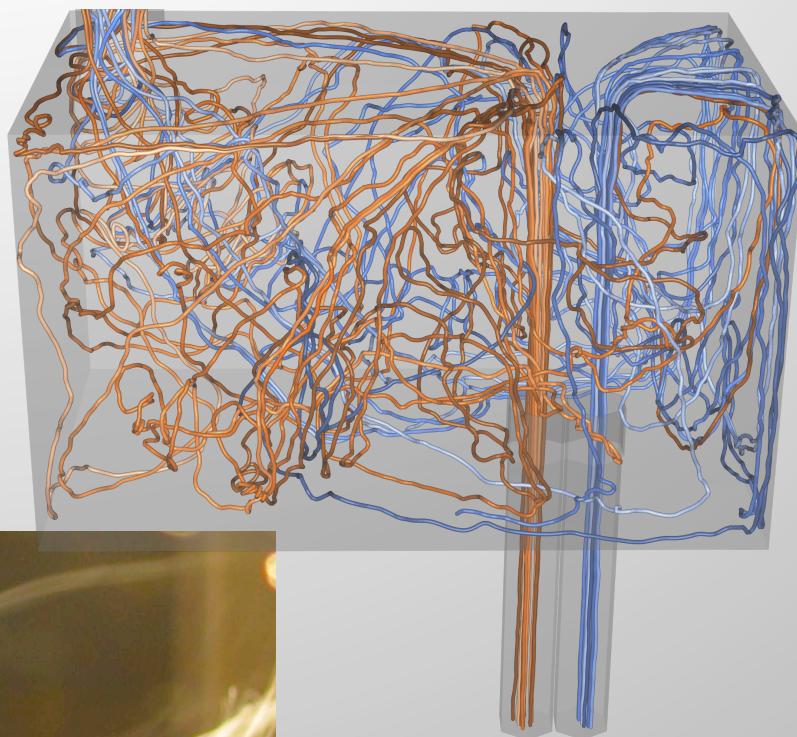
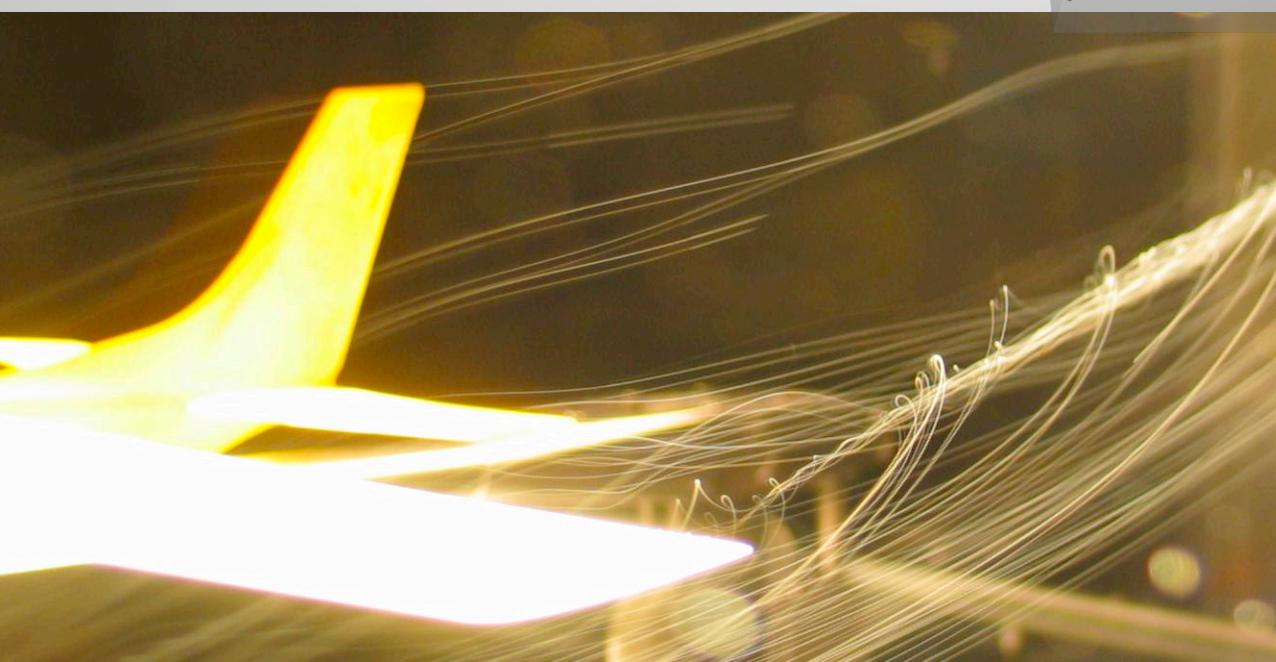


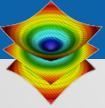
- “World Space” Partition
 - VisIt uses a hybrid approach to handle data imbalance
- “Image Space” Partition
 - Reduces memory usage
 - Each frame requires new partition



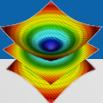


Streamlines



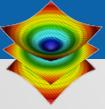


Data Analysis

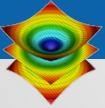


The key concepts of VisIt

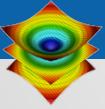
- **Sources:** where you are getting data from
 - **Windows:** where visualizations are displayed
 - **Plots:** how you render data
 - **Operators:** how you manipulate data
 - **Interactors:** what your mouse cursor does
 - **Tools:** advanced ways to control plots and operators
- **Expressions:** mechanism for generating derived quantities
 - **Queries:** how to access quantitative information



<Break>

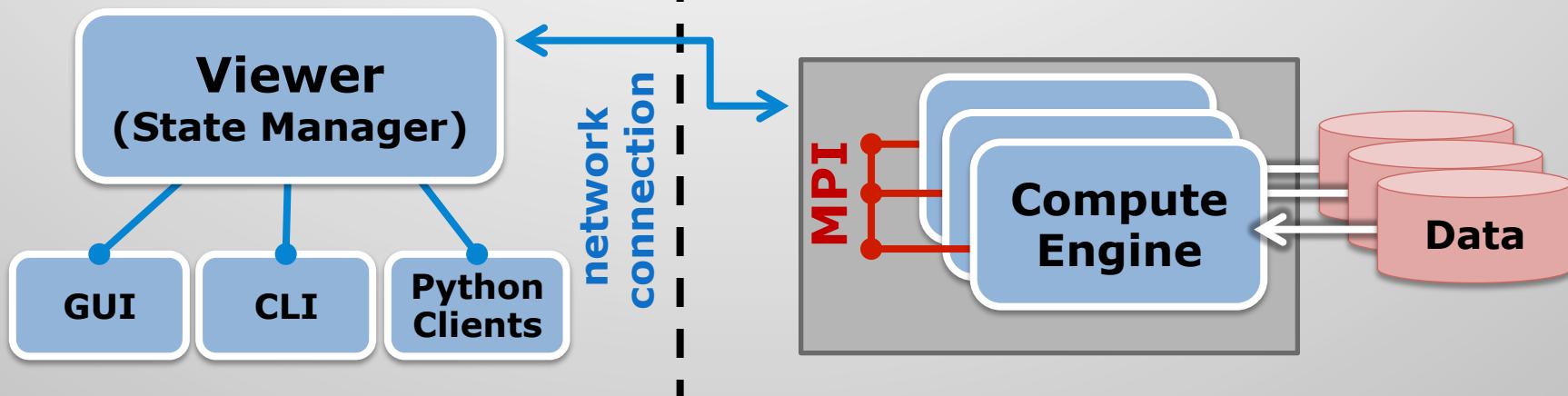


Python Scripting



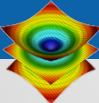
VisIt provides Python interfaces for state control and data manipulation.

Local Components | Parallel Cluster

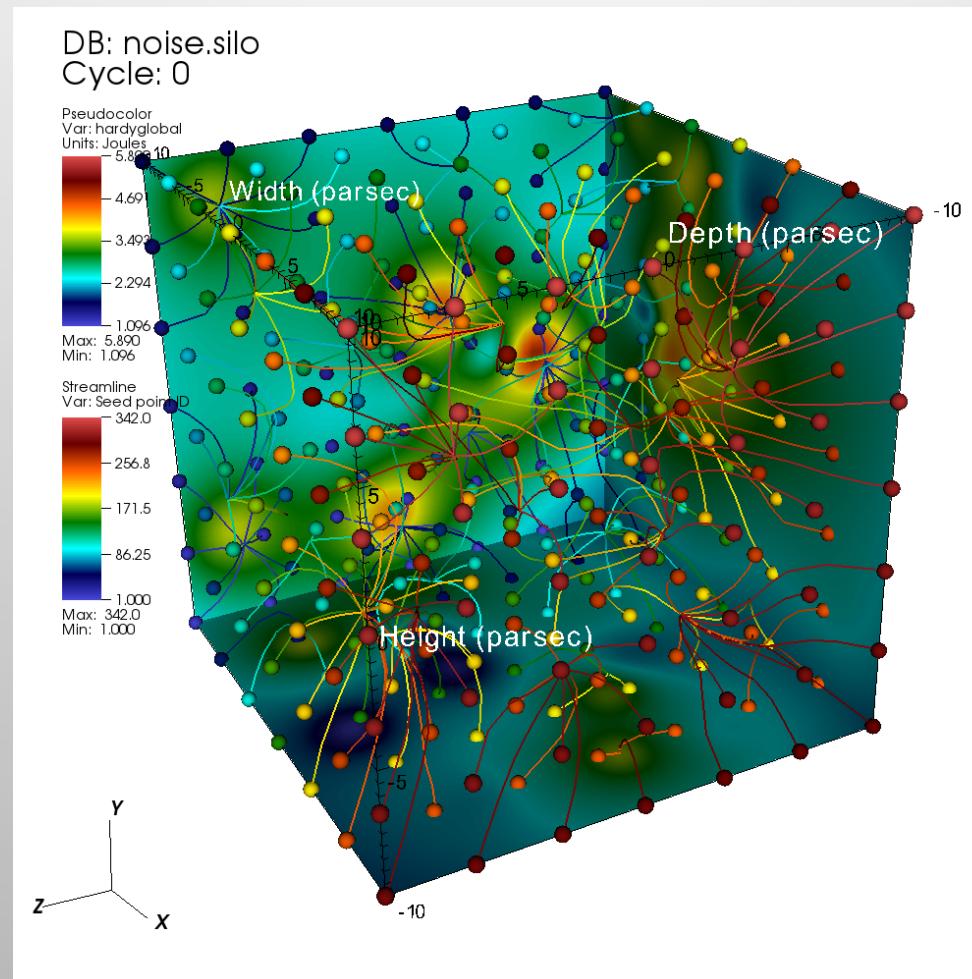


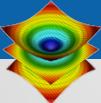
**Python Client Interface
(State Control)**

**Python Filter Runtime
(Direct Mesh Manipulation)**



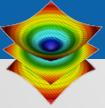
Python Client Interface Example Script: Using VisIt's Building Blocks



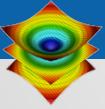


There are several ways to access VisIt's Python Client Interface.

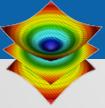
- Launch VisIt's CLI binary:
 - `visit -cli`
- Launch for windowless batch processing:
 - `visit -nowin -cli -s <script_file.py>`
- Control VisIt from a Python interpreter:
 - `import visit`
 - http://visitusers.org/index.php?title=Python_Module_Support
- Record GUI actions in to Python snippets:
 - Macro Recording provides a quick path to learn VisIt's Python Client API.



Movie Making



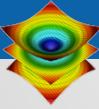
Alternate Data Representations



Special Topic:

New Python Capabilities

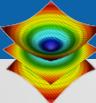
- Custom Python UIs



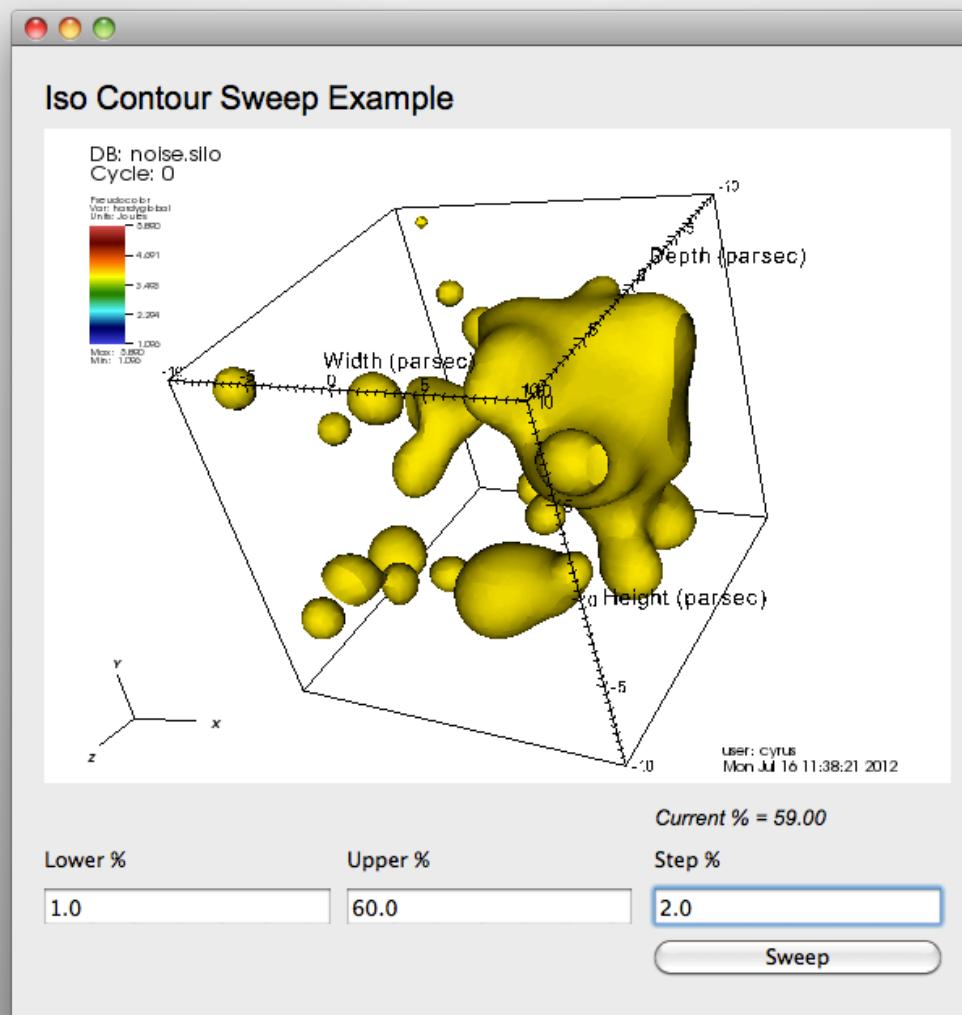
We extended VisIt's core C++ infrastructure to support Python custom UIs.

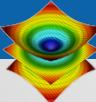
- We chose PySide, a LGPL Python Qt wrapper, as the primary UI framework.
- Current support includes the ability to:
 - *Embed VisIt's render windows.*
 - *Reuse VisIt's existing GUI widgets.*
 - *Design UIs via Qt Designer.*
- Full UI integration required combining ***client*** (CLI, GUI) and ***viewer*** processes.





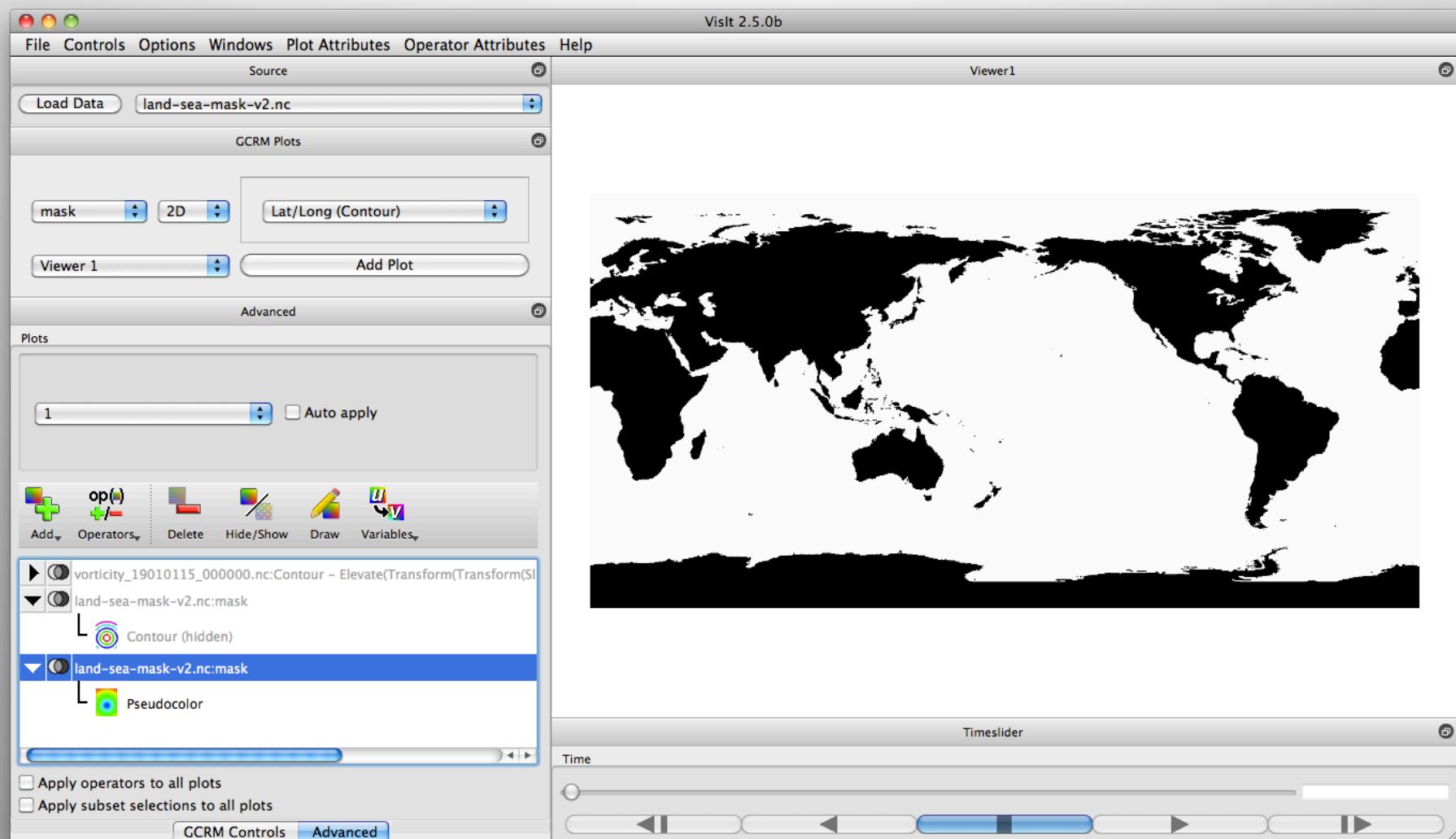
Custom Python UI Example Script: Isosurface Sweep

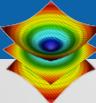




Custom UI Example: GCRM

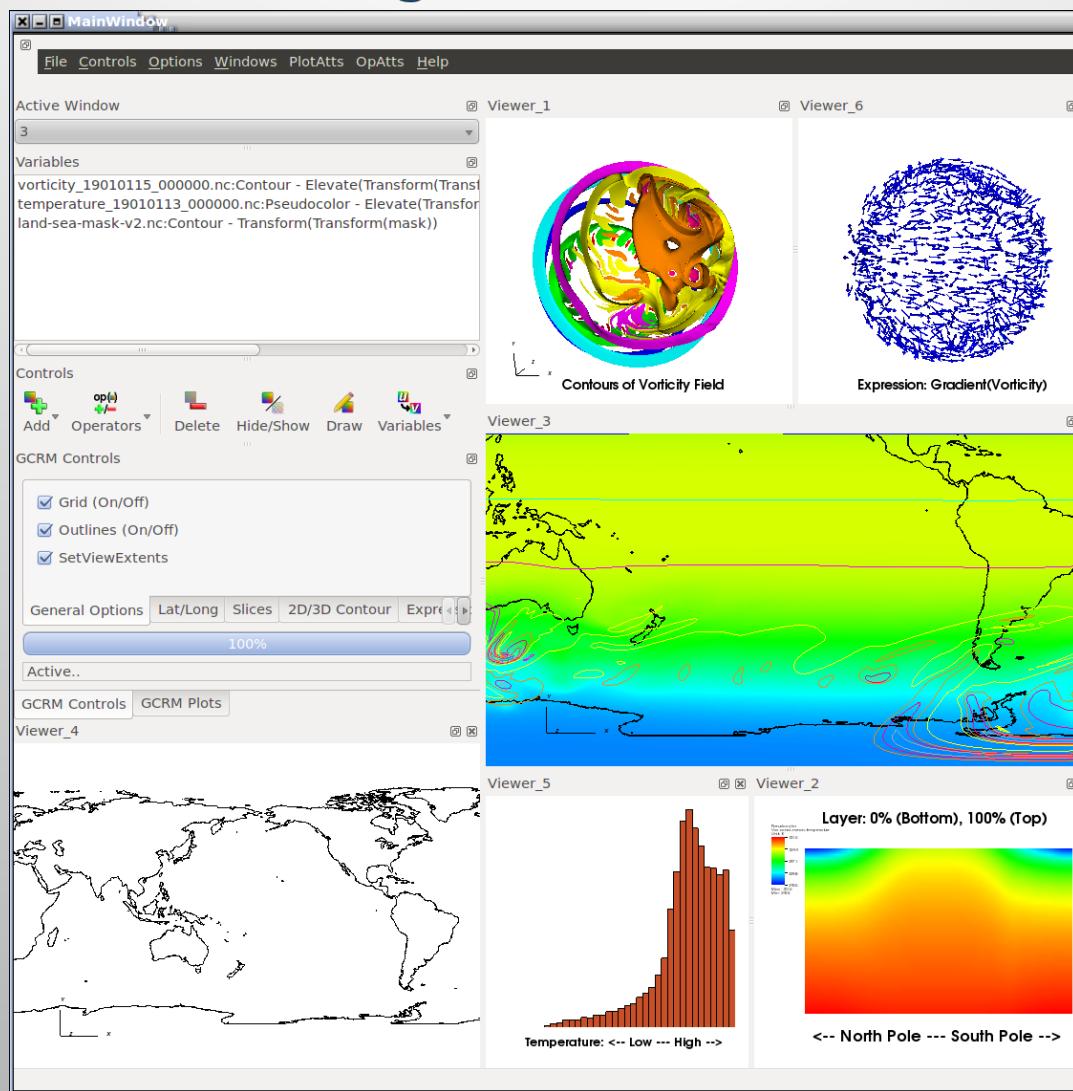
Global Cloud Resolving Model Viewer

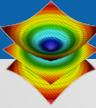




Custom UI Example: GCRM

Global Cloud Resolving Model Viewer





Custom UI Example: UV-CDAT

Ultrascale Visualization – Climate Data Analysis Tools

File View VisTrails Help Edit Tools PCMDITools Help

Projects

- Project 1*
 - untitled* @ A1
 - untitled* @ B1
 - untitled* @ B2
 - untitled* @ C1
 - untitled* @ C2
 - untitled* @ A2

Spreadsheet

Sheet 1

A **B** **C**

DB: pr_A2_710_729.nc Cycle: 0 Time: 209 (50)

Lat (degrees_north) Lon (degrees_east)

DB: BADC_BADC1.0.001.01.nc (APR 10 hour Simulation) (2015-1400)

Lat (degrees_north) Lon (degrees_east)

DB: yaetda.pak0apr.nc Cycle: 0 Time: 15

Lat (degrees_north) Lon (degrees_east)

Variables

- pr (7300, 64, 128)
- v (2, 80, 97)
- u (2, 80, 97)
- cld (120, 46, 72)
- ta (11, 17, 73, 144)
- precip (217, 288)
- temp (217, 288)
- TEMP (42, 2400, 3600)

Calculator

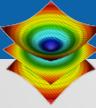
```
'pvvariablename': ('parameter', 232L, 'function', 196L, 32L), 'sliceOffset': ('parameter', 233L, 'function', 197L, 33L), 'col': ('parameter', 235L, 'function', 199L, 34L), 'row': ('parameter', 236L, 'function', 200L, 34L)}
```

```
addParameterChangesFromAliasesAction()
Aliases: {'pvfilename': 'work/pvclimate.nc', 'pvfilename.url': '', 'pvvariablename': 'TEMP', 'col': '1', 'pvvariablename.file': 'TEMP', 'row': '2'}
```

```
Pipeline Aliases: {'isoSurfaces': ('parameter', 234L, 'function', 198L, 33L), 'pvfilename': ('parameter', 231L, 'function', 195L, 32L), 'pvvariablename': ('parameter', 232L, 'function', 196L, 32L), 'sliceOffset': ('parameter', 233L, 'function', 197L, 33L), 'col': ('parameter', 235L, 'function', 199L, 34L), 'row': ('parameter', 236L, 'function', 200L, 34L)} <type 'vtkobject'>
```

Enter CDAT command and press Return

x^2	sqrt	1/x	x^y		
LN	LOG	e^x	10^x		
x<y	x>y	x<>y	x==y		
SIN	ARCSIN	COS	ARCCOS		
TAN	ARCTAN	STD	ABS		
REGRID	MASK	GET_MASK	GROWER		
Clear	7	8	9	*	(
Del	4	5	6	/)
Enter	1	2	3	+	PI
Plot	0	.	+/-	-	e



Custom UI Example: UV-CDAT

Ultrascale Visualization – Climate Data Analysis Tools

File View VisTrails Help Edit Tools PCMDITools Help

Projects

Project 1*

- untitled* @ A1
- untitled* @ B1
- untitled* @ B2
- untitled* @ C1
- untitled* @ C2
- untitled* @ A2

Spreadsheet

Sheet 1

A B C

1

2

VisIt VCS DV3D

Paraview VCS VisIt

Plots and Analyses

- DV3D
- PVClimate
- PV Climate Plot**
- VCS
- VisIt

Contour Plot
Extreme Value Analysis Plot
Pseudocolor Plot

Templates Plots and Analyses

Variables

- pr (7300, 64, 128)
- v (2, 80, 97)
- u (2, 80, 97)
- cld (120, 46, 72)
- ta (11, 17, 73, 144)
- precip (217, 288)
- temp (217, 288)
- TEMP (42, 2400, 3600)

Calculator

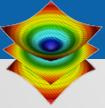
```

'pvvariablename': ('parameter', 232L, 'function',
196L, 32L), 'sliceOffset': ('parameter', 233L,
'function', 197L, 33L), 'col': ('parameter', 235L,
'function', 199L, 34L), 'row': ('parameter', 236L,
'function', 200L, 34L)}
addParameterChangesFromAliasesAction()
Aliases: {'pvfilename': 'work/pvclimate.nc',
'pvfilename.url': '', 'pvvariablename': 'TEMP', 'col':
'1', 'pvvariablename.file': 'TEMP', 'row': '2'}
Pipeline Aliases: {'isoSurfaces': ('parameter', 234L,
'function', 198L, 33L), 'pvfilename': ('parameter',
231L, 'function', 195L, 32L), 'pvvariablename':
('parameter', 232L, 'function', 196L, 32L),
'sliceOffset': ('parameter', 233L, 'function',
197L, 33L), 'col': ('parameter', 235L, 'function',
199L, 34L), 'row': ('parameter', 236L, 'function',
200L, 34L)}
<type 'vtkobject'>

```

Enter CDAT command and press Return

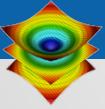
x^2	sqrt	1/x	x^y
LN	LOG	e^x	10^x
x<y	x>y	x<>y	x==y
SIN	ARCSIN	COS	ARCCOS
TAN	ARCTAN	STD	ABS
REGRID	MASK	GET_MASK	GROWER
Clear	7	8	9
Del	4	5	6
Enter	1	2	3
Plot	0	.	+/-
			-
			e



Special Topic:

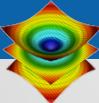
New Python Capabilities

- Python Filter Runtime

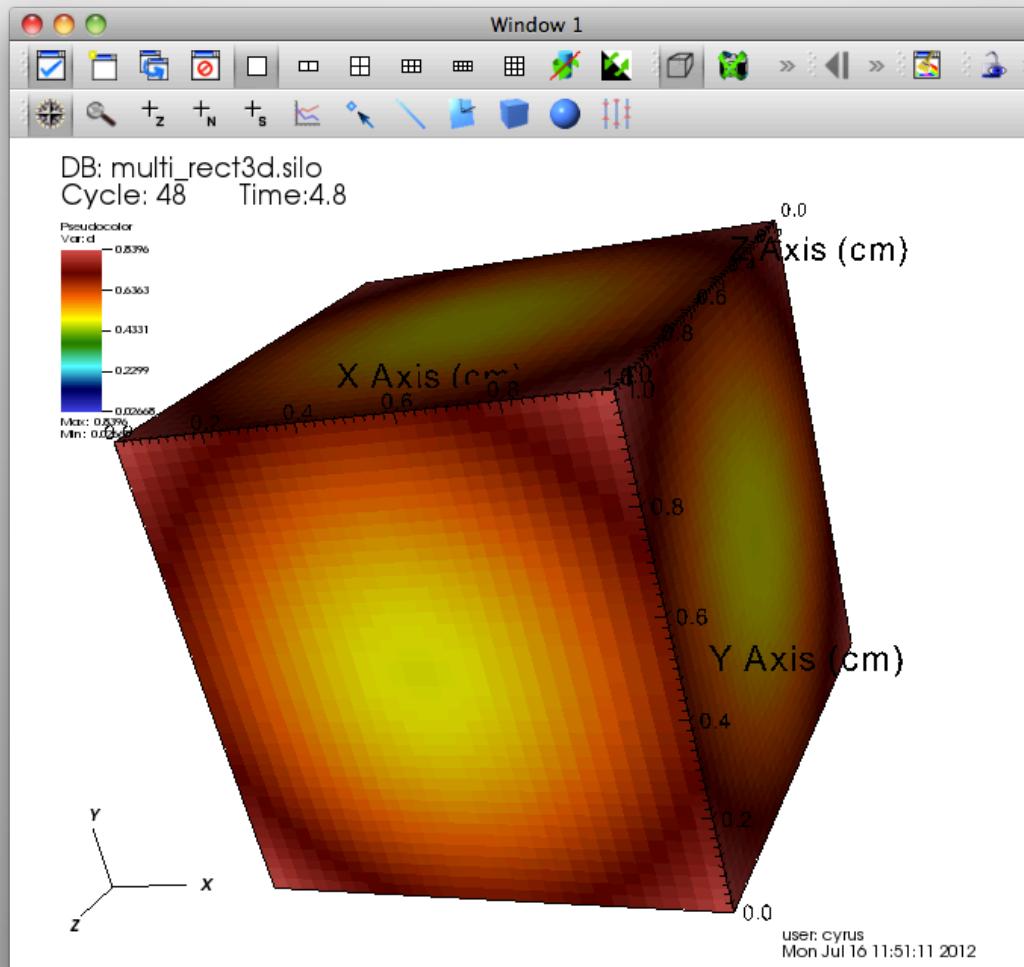


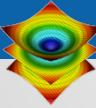
A Python Filter Runtime provides access to low-level mesh data structures.

- VisIt embeds a Python interpreter in each ***compute engine*** MPI process.
- C++/Python data exchanges utilize VTK's Python wrapper module.
 - Nice path to *numpy* and *scipy*.
- Parallel communication is provided via a simple MPI wrapper, named '*mpicom*'.
- We currently support Python Filters for VisIt's ***Expression*** and ***Query*** building blocks.



Python Filter Runtime Example Script: Cell Average Query





Python Filter Runtime Example: OpenCL Expression Framework Research

```

du = grad(vx,dims,x,y,z)
dv = grad(vy,dims,x,y,z)
dw = grad(vz,dims,x,y,z)

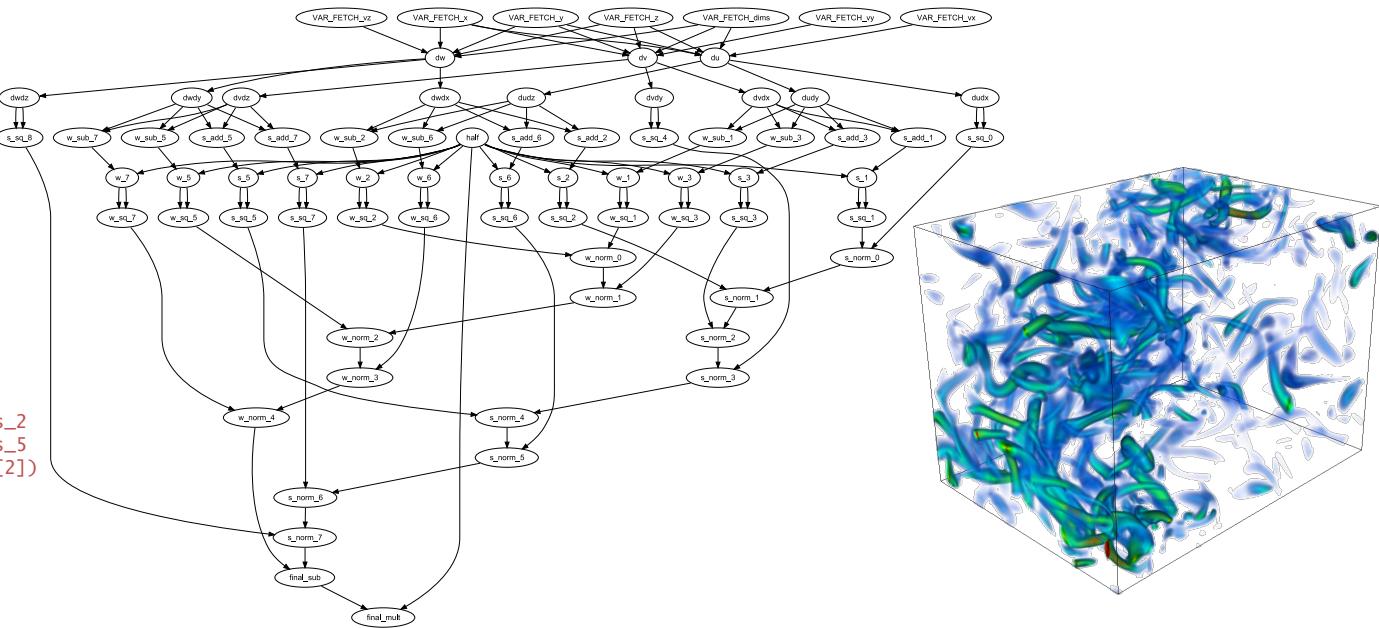
s_1 = (.5 * (du[1] + dv[0]))
s_2 = (.5 * (du[2] + dw[0]))
s_3 = (.5 * (dv[0] + du[1]))
s_5 = (.5 * (dv[2] + dw[1]))
s_6 = (.5 * (dw[0] + du[2]))
s_7 = (.5 * (dw[1] + dv[2]))
w_1 = (.5 * (du[1] - dv[0]))
w_2 = (.5 * (du[2] - dw[0]))
w_3 = (.5 * (dv[0] - du[1]))
w_5 = (.5 * (dv[2] - dw[1]))
w_6 = (.5 * (dw[0] - du[2]))
w_7 = (.5 * (dw[1] - dv[2]))

s_norm = (du[0]*du[0] + s_1*s_1 + s_2*s_2
          + s_3*s_3 + dv[1]*dv[1] + s_5*s_5
          + s_6*s_6 + s_7*s_7 + dw[2]*dw[2])

w_norm = (w_1*w_1 + w_2*w_2
          + w_3*w_3 + w_5*w_5
          + w_6*w_6 + w_7*w_7)

q_crit = (.5 * (w_norm - s_norm))

```



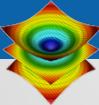
User Expression

Data Flow Network

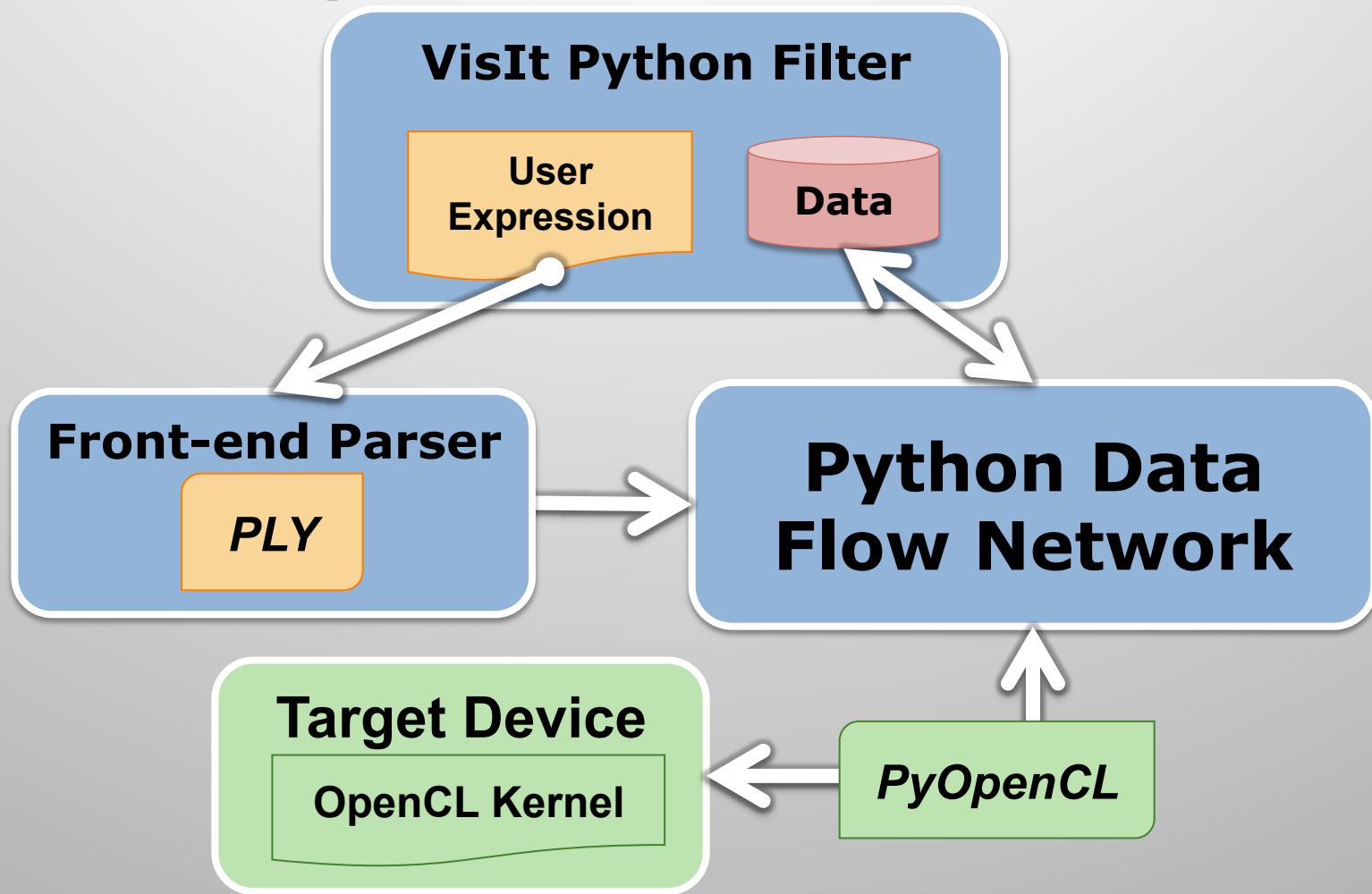
Volume Rendering
of GPU Result

Joint research with P. Navrátil (TACC), M. Jiang (LLNL) and M. Moussalem (UT-Austin)

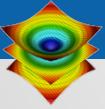
Illustration of an OpenCL expression framework executing a user defined Q-criterion expression, used for vortex core detection in flow vector fields.



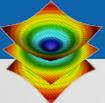
Python Filter Runtime Example: OpenCL Expression Framework Research



For more details: We are giving a talk the PyHPC 2012 Workshop (Friday)

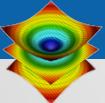


Practical Tips for Using VisIt



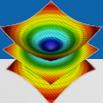
Practical Tips for Using VisIt

- How to get VisIt to read your data
- How to get help when you run into trouble



Practical Tips for Using VisIt

- **How to get VisIt to read your data**
- How to get help when you run into trouble

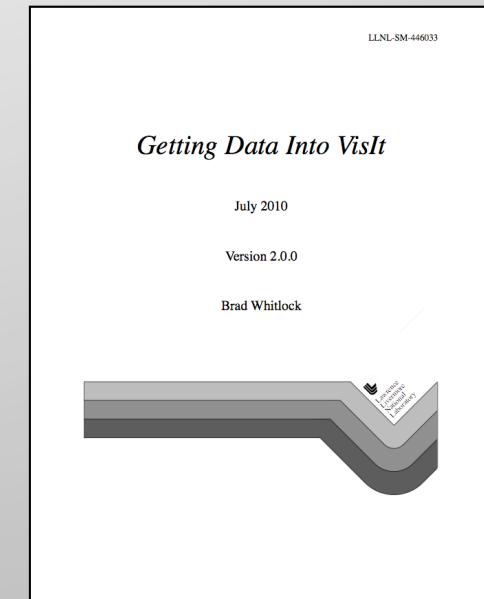


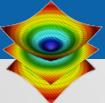
How to get VisIt to read your data.

- There is an extensive manual on this topic: “Getting Data Into VisIt”

<https://wci.llnl.gov/codes/visit/manuals.html>

- Three ways:
 - Use a known format
 - Write a file format reader
 - In situ processing



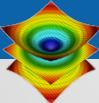


File formats that VisIt supports

- **110+ Total Readers:** ADIOS, **BOV**, Boxlib, CCM, CGNS, Chombo, CLAW, EnSight, ENZO, Exodus, FLASH, Fluent, GDAL, Gadget, Images (TIFF, PNG, etc), ITAPS/MOAB, LAMMPS, NASTRAN, **NETCDF**, Nek5000, OpenFOAM, PLOT3D, **PlainText**, **Pixie**, Shapefile, **Silo**, Tecplot, **VTK**, **Xdmf**, **Vs**, and many more

[http://www.visitusers.org/index.php?
title=Detailed list of file formats VisIt supports](http://www.visitusers.org/index.php?title=Detailed%20list%20of%20file%20formats%20VisIt%20supports)

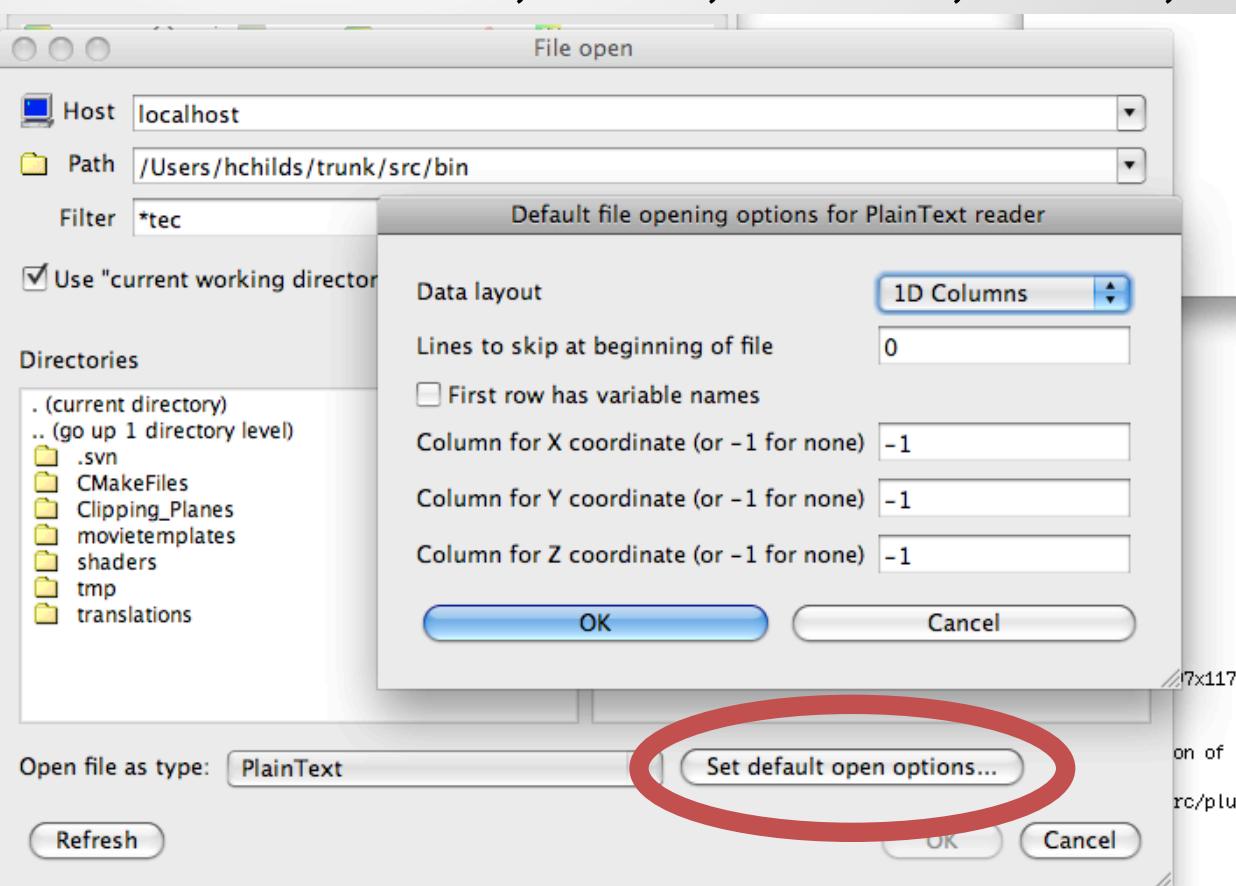
- Some readers are more robust than others.
 - For some formats, support is limited to flavors of a file a VisIt developer has encountered previously (e.g. Tecplot).



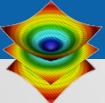
File formats that VisIt supports

- 110+ Total Readers: ADIOS, BOV, Boxlib, CCM, CGNS, Chombo, Exodus, FLASH, GMSH, (TIFF, PNG, etc.), NASTRAN, NetCDF, PLOT3D, PLAB, Tecplot, VTK, ...

http://www.visitucsb.org/tutorials/visit_tutorial.html#fileformats
title=Detailed list of supported file formats

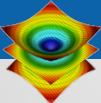


- Some readers support multiple file formats, supporting multiple file formats that a VisIt developer has encountered previously (e.g. Tecplot).



File formats that VisIt supports

- Common array writing libraries:
 - NETCDF
 - VisIt reader understands many (but not all) conventions
 - HDF5
 - Pixie is most general HDF5 reader
 - Many other HDF5 readers
- Xdmf: specify an XML file that describes semantics of arrays in HDF5 file
- VizSchema (Vs): add attributes to your HDF5 file that describes semantics of the arrays.



VTK File Format

- The VTK file format has both ASCII and binary variants.
 - Great documentation at:
<http://www.vtk.org/VTK/img/file-formats.pdf>
- Easiest way to write VTK files: use VTK modules
 - ... but this creates a dependence on the VTK library
- You can also try to write them yourself, but this is an error prone process.
- Third option: `visit_writer`



File Formats
for VTK Version 4.2
(Taken from The VTK User's Guide
Contact Kitware www.kitware.com to purchase)

VTK File Formats

The *Visualization Toolkit* provides a number of source and writer objects to read and write popular data file formats. The main reason for creating yet another data file format is to offer a common representation scheme for a variety of dataset types, and to provide a simple method to communicate data between software. Whenever possible, we recommend that you use formats that are more widely used. But if this is not possible, the *Visualization Toolkit* formats described here can be used instead. Note that these formats may not be supported by many other tools.

There are two different styles of file formats available in VTK. The simplest are the legacy, serial formats that are easy to read and write either by hand or programmatically. However, these formats are less flexible than the XML based file formats described later in this section. The XML formats support random access, parallel I/O, and portable data compression and are preferred to the serial VTK file formats whenever possible.

Simple Legacy Formats

The legacy VTK file formats consist of five basic parts.

1. The first part is the file version and identifier. This part contains the single line: # vtk DataFile Version x.x. The line must be exactly as shown with the exception of the version number x.x, which will vary with different releases of VTK. (Note: the current version number is 3.0. Version 1.0 and 2.0 files are compatible with version 3.0 files.)
2. The second part is the header. The header consists of a character string terminated by end-of-line character \n. The header is 256 characters maximum. The header can be used to describe the data and include any other pertinent information.
3. The next part is the file format. The file format describes the type of file, either ASCII or binary. On this line the single word ASCII or BINARY must appear.
4. The fourth part is the dataset structure. The geometry part describes the geometry and topology of the dataset. This part begins with a line containing the keyword DATASET followed by a keyword describing the type of dataset. The depending structure depends on the type of dataset. Other keywords define the internal structure of the data.
5. The final part consists of the data. This part begins with the keyword POINTS, CELLS, DATA, followed by an integer number specifying the number of points or cells, respectively. (It doesn't matter whether POINT, DATA or CELLS comes first.) Other keyword/data combinations then define the actual dataset attribute values (i.e., scalars, vectors, tensors, normals, texture coordinates, or field data).

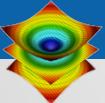
An overview of the file format is shown in [Figure 1](#). The first three parts are mandatory, but the other two are optional. Thus you have the flexibility of mixing and matching dataset attributes and geometry, either by operating system file manipulation or using VTK filters to merge data. Keywords are case insensitive, and may be separated by whitespace.

Before describing the data file formats please note the following:

- `dataType` is one of the types `bit`, `unsigned char`, `char`, `unsigned short`, `short`, `unsigned int`, `int`, `unsigned long`, `long`, `float`, or `double`. These keywords are used to describe the form of the data, both for reading from file, as well as constructing the appropriate internal object. Not all data types are supported for all classes.

LLNL-PRES-599692

79



VisIt Writer writes VTK files

- It is a “library” (actually a single C file) that writes VTK-compliant files.
 - The typical path is to link `visit_writer` into your code and write VTK files
- There is also Python binding for `visit_writer`.
 - The typical path is to write a Python program that converts from your format to VTK
- Both options are short term: they allow you to play with VisIt on your data. If you like VisIt, then you typically formulate a long term file format strategy.
- More information on `visit_writer`:
 - <http://visitusers.org/index.php?title=VisItWriter>

Python VisIt Writer in action

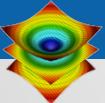
```
import visit_writer
import math
import sys

nX = 20
nY = 20
conn = []
for i in range(nX-1):
    for j in range(nY-1):
        pt1 = j*(nX) + i;
        pt2 = j*(nX) + i+1;
        pt3 = (j+1)*(nX) + i+1;
        pt4 = (j+1)*(nX) + i;
        conn.append([ "quad", pt1, pt2, pt3, pt4 ])

pts = []
rad = []
for i in range(nX):
    for j in range(nY):
        pts.extend([ float(i), float(j), 0 ])
        rad.append( math.sqrt(i*i + j*j) )

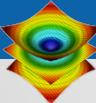
var_datum = [ "radius", 1, 1, rad ]
vars = [ var_datum ]
visit_writer.WriteUnstructuredMesh("ugrid.vtk", 0, pts, conn, vars)

sys.exit()
```



Silo file format

- Silo is a mature, self-describing file format that deals with multi-block data.
- It has drivers on top of HDF5 and “PDB”.
- Fairly rich data model
- More information:
 - <https://wci.llnl.gov/codes/silo/>



Silo features

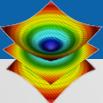
WCI | B Codes - SILO <https://wci.llnl.gov/codes/silo/>

Welcome to Silo

A mesh and field I/O library and scientific database

The screenshot displays the Silo software interface with the following components:

- Structured Rectilinear Mesh:** A 2D grid showing a color gradient from blue to red.
- Gridless Point Mesh:** A 3D cube containing scattered colored points.
- Structured (Curvilinear) Mesh:** A curved surface with a color gradient.
- Arbitrary Subsets:** Three 3D polyhedra with some faces colored red and green.
- Silex browser for Silo files:** A window showing a hierarchical tree of mesh components and a detailed table of object attributes.
- Constructive Solid Geometry (CSG) Mesh:** A 3D model of a torus with a central cutout.
- Unstructured Zoo (UCD) Mesh:** A complex 3D model composed of several different mesh types.
- Adaptive Mesh Refinement (AMR) Mesh:** A 3D model with a nested grid structure.
- Mixing Materials:** A 3D model showing different materials represented by different colors.
- Arbitrary Polyhedral Mesh:** A 3D cube with a complex, non-uniform internal mesh structure.
- XY Curve:** A 2D plot showing a curve starting at the origin and increasing exponentially.

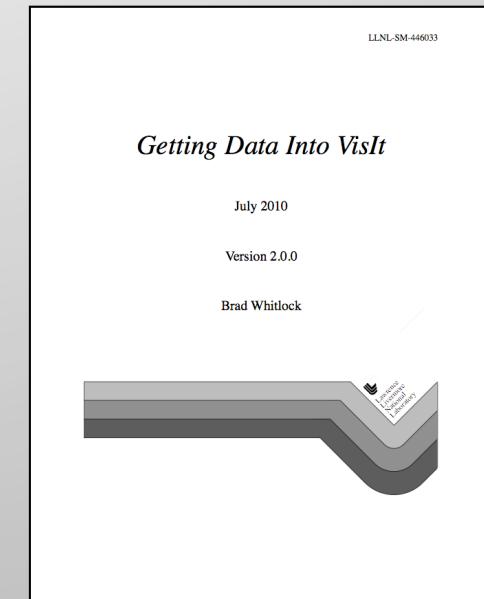


How to get VisIt to read your data.

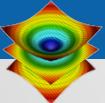
- There is an extensive manual on this topic: “Getting Data Into VisIt”

<https://wci.llnl.gov/codes/visit/manuals.html>

- Three ways:
 - Use a known format
 - **Write a file format reader**
 - **In situ processing**

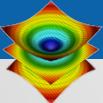


These topics will be covered in the afternoon session



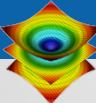
Practical Tips for Using VisIt

- How to get VisIt to read your data
- **How to get help when you run into trouble**



How to get help when you run into trouble

- FAQ
 - <http://visit.llnl.gov/FAQ.html>
- VisIt Users Mailing List
 - Address: visit-users@elist.ornl.gov
 - Info: <https://elist.ornl.gov/mailman/listinfo/visit-users>
 - Archive: <https://elist.ornl.gov/pipermail/visit-users/>
- VisIt Users Wiki
 - <http://www.visitusers.org>
- VisIt Users Forum
 - <http://visitusers.org/forum/YaBB.pl>
- Priority support for specific user groups:
 - VisIt-help-{XYZ} Mailing Lists
- Reference Manuals
 - <https://wci.llnl.gov/codes/visit/manuals.html>

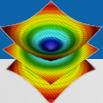


FAQ: <http://visit.llnl.gov/FAQ.html>

The screenshot shows a web browser window with the title "Frequently Asked Questions". The URL in the address bar is <https://wci.llnl.gov/codes/visit/FAQ.html>. The page content is titled "Frequently Asked Questions" and lists 32 numbered links related to VisIt usage. The browser interface includes a menu bar with items like Home, Downloads, Documentation, What's New, Index, and Contact.

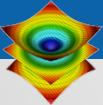
Frequently Asked Questions

1. [Contact information](#)
2. [Supported platforms](#)
3. [Optimal hardware/software](#)
4. [Debugging problems starting VisIt or opening files](#)
5. [Stereo rendering](#)
6. [VisIt won't run on Linux](#)
7. [Slow performance on Linux](#)
8. [Slow performance Using SSH](#)
9. [No output in visualization window](#)
10. [Accessing data on remote machine](#)
11. [Running VisIt in parallel](#)
12. [Supported data file formats](#)
13. [Getting your data into VisIt](#)
14. [Making a movie of your data](#)
15. [Setting your user name to connect to a remote machine](#)
16. [Cannot connect to a remote computer](#)
17. [Building VisIt on a Windows computer](#)
18. [Installing VisIt on a MacOS X computer](#)
19. [Hanging at 12% on Windows computers](#)
20. [Getting the Plugin Developer's Guide](#)
21. [Writing a plugin for VisIt](#)
22. [When new versions of VisIt are released](#)
23. [What is new in the latest version of VisIt](#)
24. [Compilers that can be used to build VisIt](#)
25. [VisIt's licensing agreement](#)
26. [Slow performance with ATI cards on Linux](#)
27. [Custom plugins with a downloaded VisIt binary](#)
28. [Getting HDF5 data into VisIt](#)
29. [Getting NETCDF data into VisIt](#)
30. [When I run VisIt on my Linux machine, I get a black screen](#)
31. [I get the message 'Publisher cannot be verified' when installing VisIt on Windows](#)
32. [Which libraries should I enable in build_visit?](#)



VisIt-users Mailing List

- You may only post to mailing list if you are also a subscriber.
- Approximately 400 recipients, approx. 300 posts per month.
- Developers monitor mailing list, strive for 100% response rate.
- Response time is typically excellent ($O(1 \text{ hour})$).
 - International community participates ... not unusual for a question from Australia to be answered by a European, while all US developers are asleep.
- List Address: visit-users@ornl.gov
- More information:
<https://email.ornl.gov/mailman/listinfo/visit-users>
- Archive: <https://email.ornl.gov/pipermail/visit-users/>



VisItusers.org

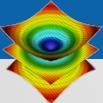
- Great source for VisIt tips and recipes.
- Users section has lots of practical advice:
 - “I solved this problem using this technique”
 - “Here’s my script to do this analysis”

Misc

[\[edit\]](#)

- Using VisIt in an mxterm
- Using derived data functions (DDFs)
- Using the command line interface
- How volume rendering works in VisIt
- Using cross-mesh field evaluations ... how to do differences, access other time slices, etc
- Keyframing example
- Exporting databases
- Directions for specific machines
- Using the VisIt Python API with a standard Python interpreter
- Pages that contain instructions specific to certain user groups and needs
- Issues related to running VisIt on Windows under cygwin
- VisIt's Camera model
- Using VisIt's mpeg2encode
- Molecular data features
- Extracting alpha
- (Very) High resolution rendering
- Elevating shapefiles
- Raytracing your visualizations with POV-Ray and a [tutorial](#)
- POV-Ray exporting example

VisItusers.org is the VisIt project's staging area for usage recipes and future formal documentation.



VisIt Users Forum

- <http://www.visitusers.org/forum>
- Increasingly popular option;
you can post without receiving
300 emails a month
 - But it is viewed by less people
and less well supported.
- Google indexes these pages.

Members viewing this topic (1): **Hank Childs**.

 pseudocolor plot legend attributes in python (Read 18 times)

Jennifer
YaBB Newbies
 Offline

Posts: 4
Fort Collins, CO

Hello. I want to set the attributes for a pseudocolor plot legend (turn off Let VisIt manage location of the legend (turn off Let VisIt manage number of Tic Marks, and the label appearance (set these properties in a python script? If so, how?

I tried to use the Command Control to record the "# Logging for AddAnnotationObject is not implemented # Logging for SetAnnotationObjectOptions is not implemented

Thanks,
Jennifer

 PM

Hank Childs
YaBB Moderator
 Online

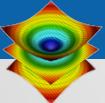
I use VisIt and I develop VisIt
Posts: 135
Davis, CA

Re: pseudocolor plot legend attributes in python
Reply #1 - 11/07/10 at 19:47:03

Hello Jennifer,

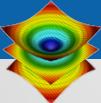
Each plot has an index and the plot's legend is re-

```
>>> GetAnnotationObjectNames()
('Plot0003',)
>>> a = GetAnnotationObject("Plot0003")
>>> a
active = 1
managePosition = 1
position = (0.05, 0.9)
xScale = 1
yScale = 1
```



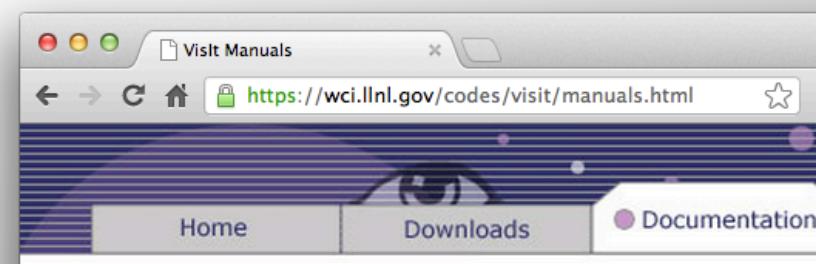
Visit-help-{XYZ}

- Some customer groups pay for priority VisIt support:
 - These customers can post directly to specific visit-help-{XYZ} support lists without subscribing.
 - The messages are received by all VisIt developers and supported collectively.
- Current Lists:
 - visit-help-asc, visit-help-scidac, visit-help-gnep, visit-help-ascem



Manuals & Other Documentation

- Getting Started Manual
- Users Manual
- Python Interface
- Getting Data Into VisIt
- VisIt Class Slides
- VisIt Class Exercises
- {Tutorials}



Manuals

VisIt Getting Started Manual (267 K pdf)

This [document](#) introduces you to the VisIt graphical user interface (GUI). It provides a brief overview on how VisIt works and directions on how to start VisIt and use the various controls in the VisIt main and popup windows.

VisIt User's Manual (15.8 MB pdf)

This [document](#) describes how to use the VisIt Graphical User Interface (GUI). It covers in detail all the controls available from the VisIt GUI, including such topics as working with files, plots, operators, saving and printing, visualization windows, quantitative analysis, making it pretty, animation and keyframing, interactive tools, multiple databases and windows, and remote visualization.

VisIt Python Interface Manual (3.6 MB pdf)

This [document](#) describes how to use the VisIt Python Interface. It covers in detail all of the functions and objects provided by the VisIt Python Interface and also gives some background on scripting with Python and common patterns that can be used when creating scripts for VisIt.