

Explore



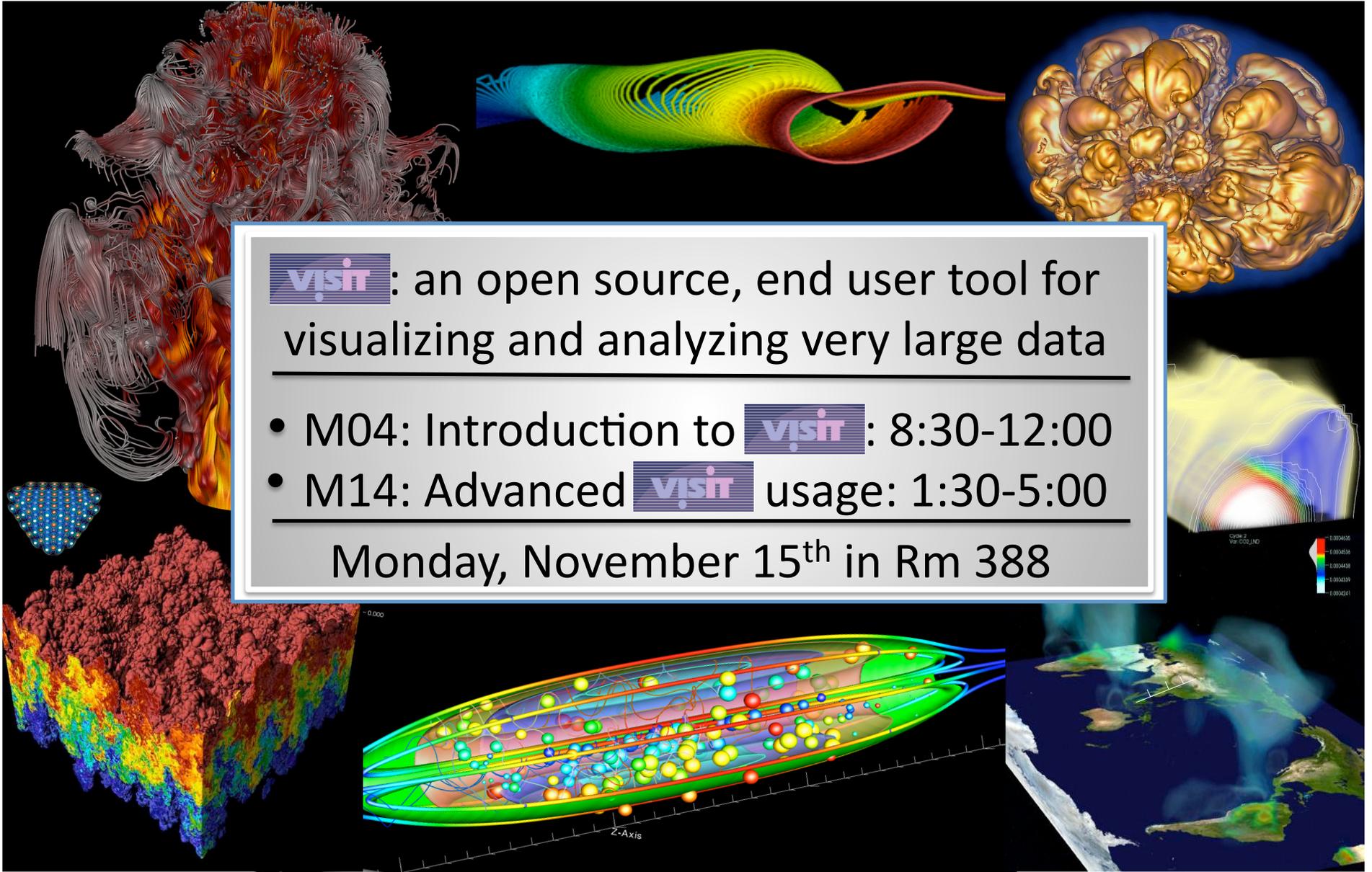
at



visit: an open source, end user tool for visualizing and analyzing very large data

- M04: Introduction to **visit** : 8:30-12:00
- M14: Advanced **visit** usage: 1:30-5:00

Monday, November 15th in Rm 388

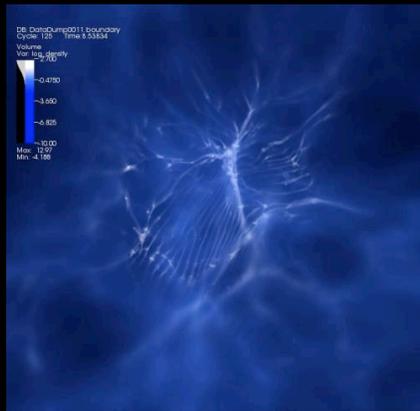


Introduction to

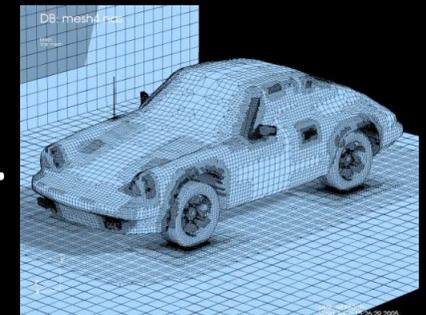
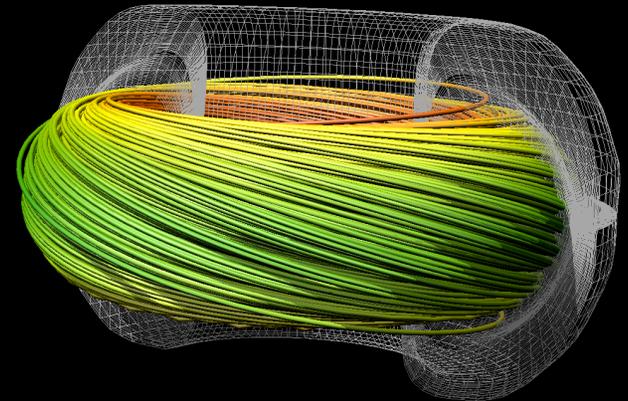


Tutorial M04

8:30 AM Monday, November 15th in Rm 388

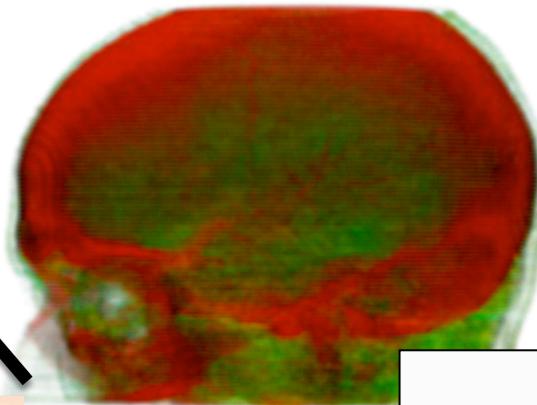
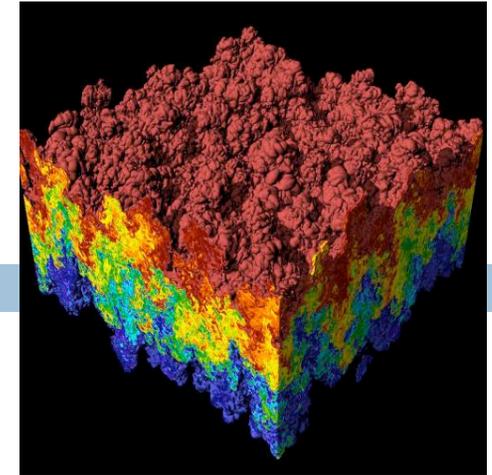


- Basic usage
- Data analysis
- Derived quantities
- Basic scripting
- Basic moviemaking
- How to get started:
 - Installing, file formats, etc.
- + much more!

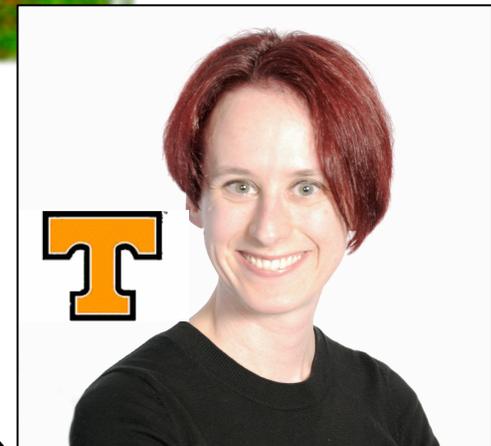


Tutorial speakers

- Hank Childs
- Amy Szczepanski
- Dave Pugmire



(Really Amy's head)



Tutorial Schedule



- 8:30-8:45: Vislt project overview (powerpoint)
- 8:45-9:30: Vislt basics: plots, operators, and more...
- 9:30-10:00: Queries and Expressions
- 10:00-10:15: Special Topic: Streamlines
- 10:15-10:35: Break
- 10:35-11:00: Scripting
- 11:00-11:20: Moviemaking
- 11:20-11:50: How to Succeed With Vislt After This Tutorial
(powerpoint)
- 11:50-12:00: Questions & Wrapup

Course Materials

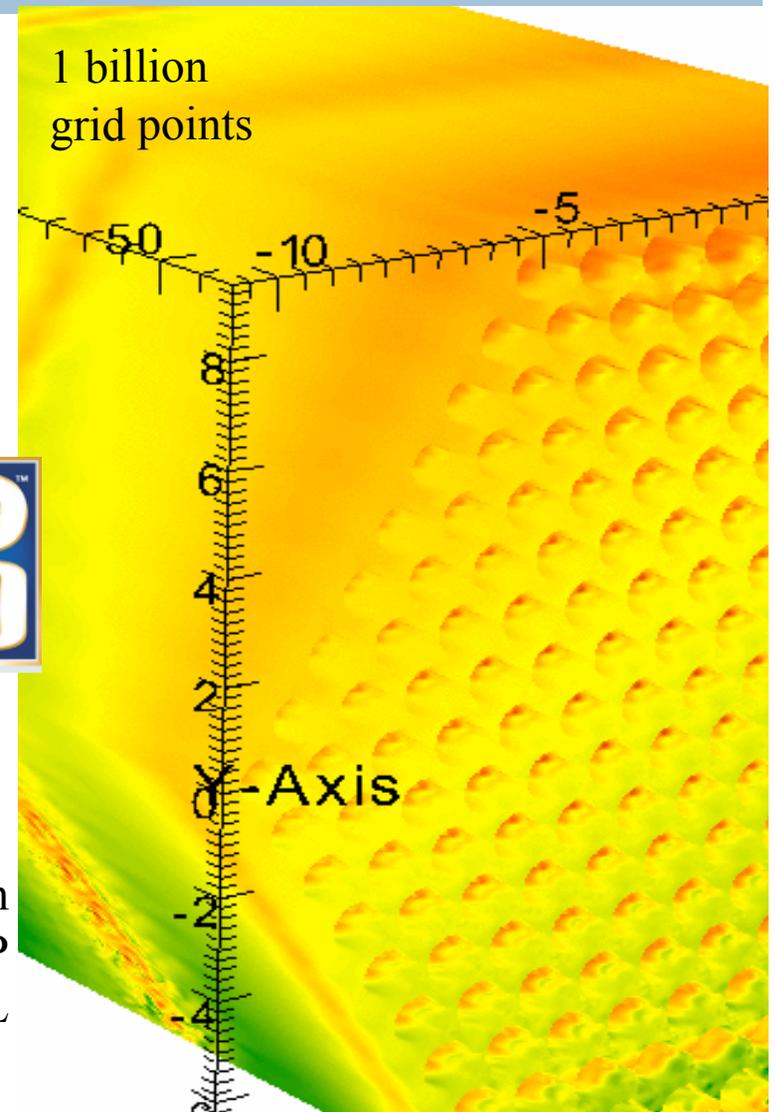
- All materials (Powerpoint & “script”) are available at visitusers.org.
- Two ways to find it:
 - 1) Go to visitusers.org and click on the link for SC10 tutorial
 - 2) Go to http://www.visitusers.org/index.php?title=SC10_Tutorial

VisIt is an open source, richly featured, turn-key application for large data.

- Used by:
 - Visualization experts
 - Simulation code developers
 - Simulation code consumers
- Popular
 - R&D 100 award in 2005
 - Used on many of the Top500
 - >>>100K downloads

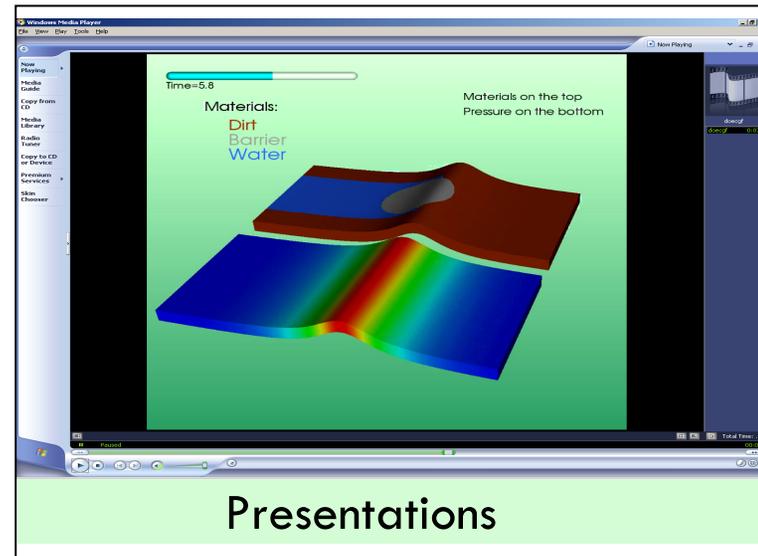
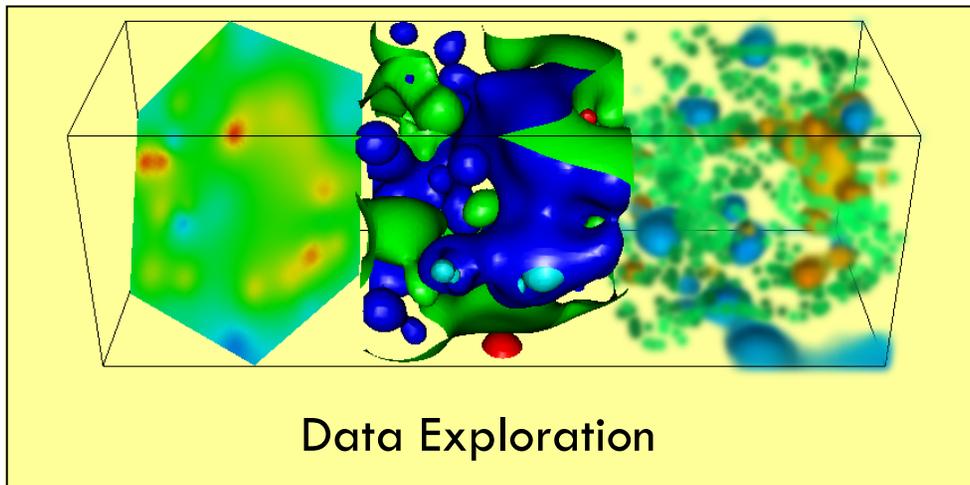
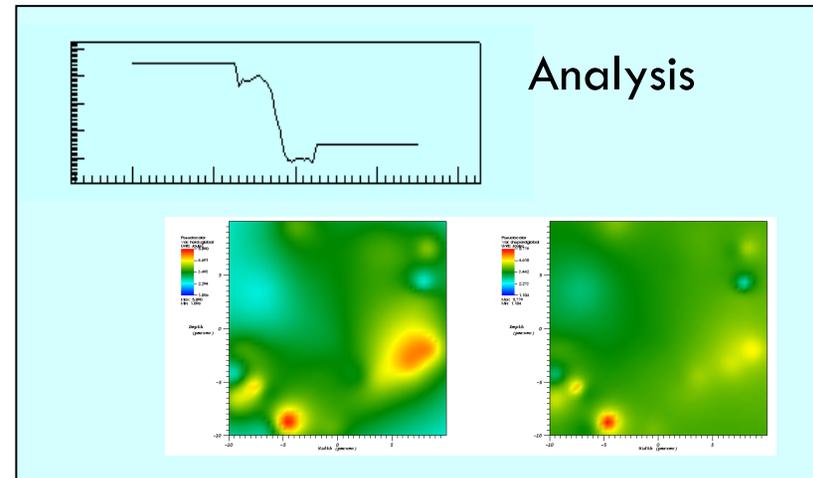
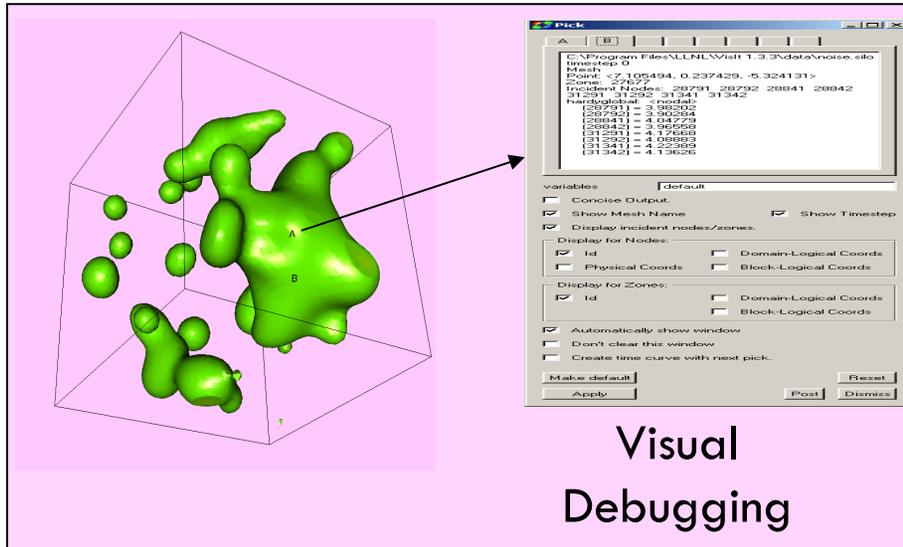


217 pin reactor cooling simulation
Run on 1/4 of Argonne BG/P
Image credit: Paul Fischer, ANL



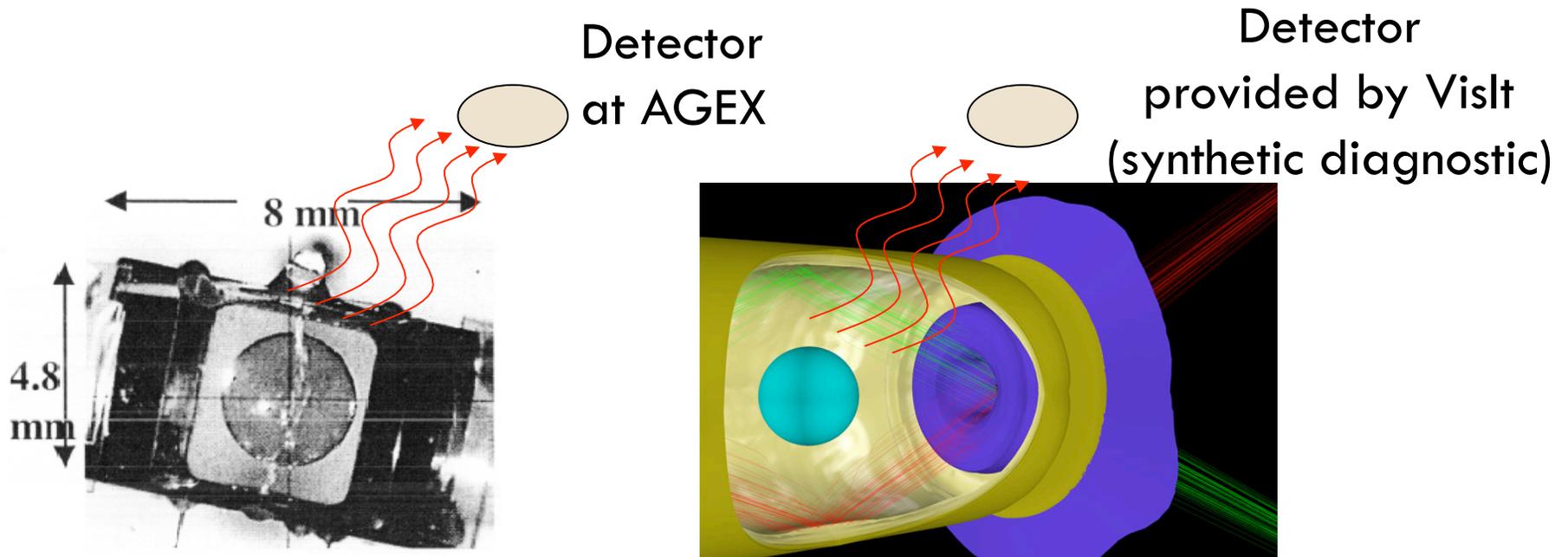
Terribly Named!!!

... intended for much more than just visualization



What sort of analysis is appropriate for VisIt?

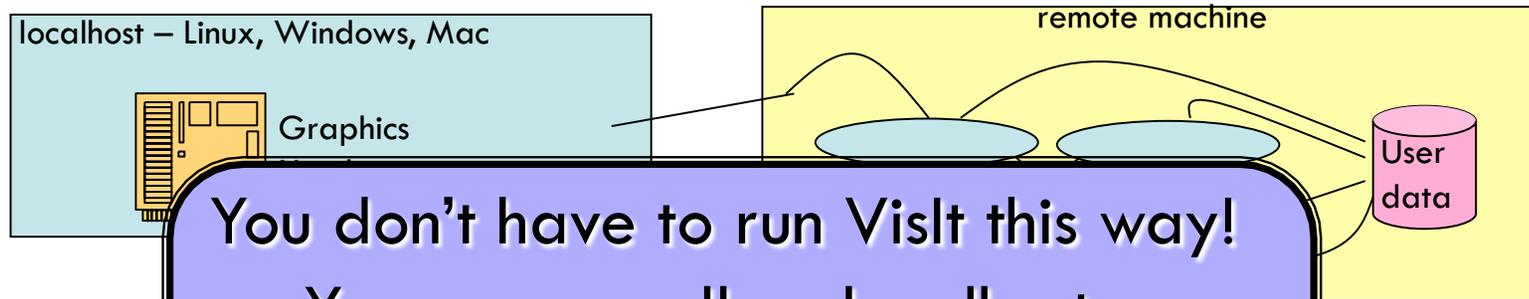
- Techniques that span scientific domains (e.g. integration, volumes, surface areas, fluxes, connected components, chord length distributions)
- Specialized analysis (e.g. hohlraum flux at AGEX)



VisIt has a rich feature set.

- Meshes: rectilinear, curvilinear, unstructured, point, AMR
- Data: scalar, vector, tensor, material, species
- Dimension: 1D, 2D, 3D, time varying
- Rendering (~15): pseudocolor, volume rendering, hedgehogs, glyphs, mesh lines, etc...
- Data manipulation (~40): slicing, contouring, clipping, thresholding, restrict to box, reflect, project, revolve, ...
- File formats (~115)
- Derived quantities: >100 interoperable building blocks
 - ▣ +, -, *, /, gradient, mesh quality, if-then-else, and, or, not
- Many general features: position lights, make movie, etc
- Queries (~50): ways to pull out quantitative information, debugging, comparative analysis

VisIt employs a parallelized client-server architecture.



You don't have to run VisIt this way!
You can run all on localhost
(like this tutorial!)
You can tunnel through ssh and
run all on the remote machine

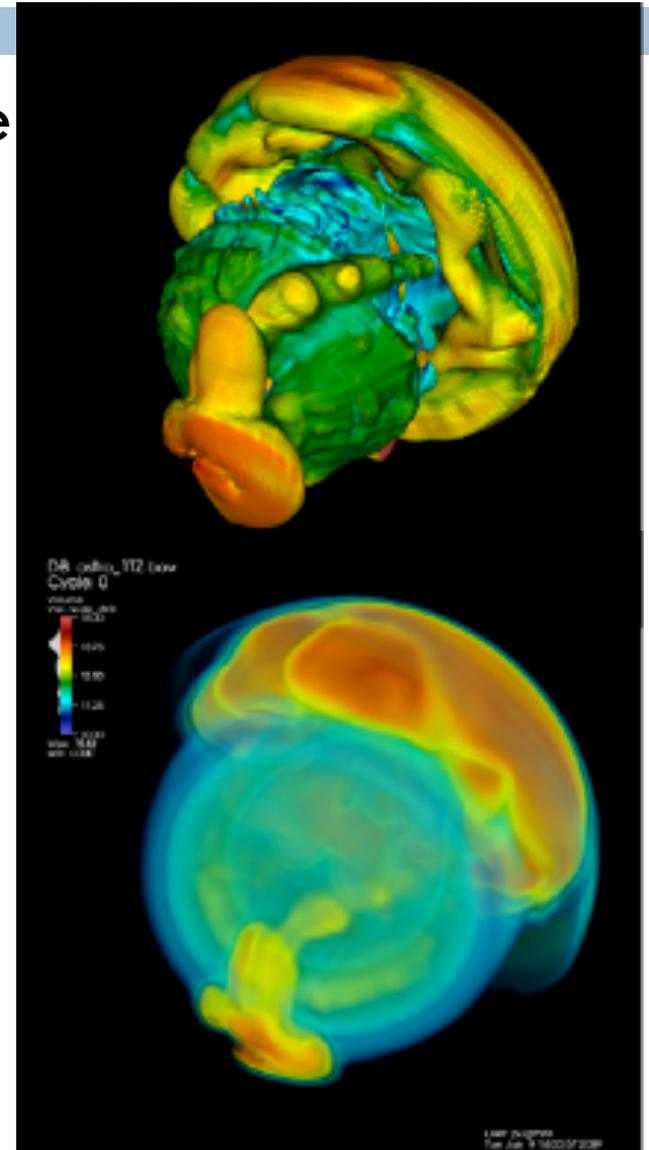
- Client-server architecture
- observability
- Good visualization
- Leverages available resources
- Scales well
- No need to move data
- Heavy use of VTK
- Multiple UIs: GUI (Qt), CLI (Python), more...

Visit recently demonstrated good performance at unprecedented scale.

- Weak scaling study: $\sim 62.5\text{M}$ cells/core

Machine	Model	Problem Size	#cores
Franklin	Cray XT4	1T, 2T	16K, 32K
Dawn	BG/P	4T	64K
JaguarPF	Cray XT5	2T	32K
Juno	X86_64	1T	16K
Purple	IBM P5	0.5T	8K
Ranger	Sun	1T	16K

Two trillion cell data set,
rendered in Visit by
David Pugmire on ORNL
Jaguar machine



It takes a lot of research to make VisIt work

A Contract Based System For Large Data Visualization*

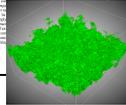
Steve Chong
 University of California, Lawrence Livermore National Laboratory
 D.J. Ragan, Karlton Swartz, James Peterson, Nicholas Miller, and Brad Whitlock

Abstract

The new generation of visualization systems is being developed in a contract-based environment. This paper describes the design and implementation of a contract-based system for large data visualization. The system is designed to be flexible and scalable, and to support a wide range of visualization techniques. It is implemented as a set of services that can be used by a variety of visualization applications. The system is designed to be easy to use and to integrate with existing visualization systems.

1. Introduction

The new generation of visualization systems is being developed in a contract-based environment. This paper describes the design and implementation of a contract-based system for large data visualization. The system is designed to be flexible and scalable, and to support a wide range of visualization techniques. It is implemented as a set of services that can be used by a variety of visualization applications. The system is designed to be easy to use and to integrate with existing visualization systems.

Extreme Scaling of Production Visualization Software on Diverse Architectures

Mark Chiba • Lawrence Berkeley National Laboratory
 David Papoušek and Scott Adams • Oak Ridge National Laboratory
 Mark Whittaker • Lawrence Livermore National Laboratory
 Mark Anderson, Franka Guderer, R. Wicker, and E. Wray Bell • Lawrence Berkeley National Laboratory

Abstract

Our goal is to build visualization software that can scale to a large number of processors. This is a challenging task because of the diverse architectures and the need to support a wide range of visualization techniques. We describe our approach to this problem, which involves building a system that can be scaled to a large number of processors and that can support a wide range of visualization techniques.




Fast, Memory-Efficient Cell Location in Unstructured Grids for Visualization

David J. Silberman, et al.

Abstract

This paper describes a fast and memory-efficient algorithm for cell location in unstructured grids. The algorithm is designed to be efficient and to support a wide range of visualization techniques. It is implemented as a set of services that can be used by a variety of visualization applications. The system is designed to be easy to use and to integrate with existing visualization systems.




A Scalable, Hybrid Scheme for Volume Rendering Massive Data Sets*

Mark Whittaker • Lawrence Livermore National Laboratory
 David J. Silberman • Lawrence Livermore National Laboratory
 David J. Silberman • Lawrence Livermore National Laboratory

Abstract

This paper describes a scalable and hybrid scheme for volume rendering massive data sets. The scheme is designed to be efficient and to support a wide range of visualization techniques. It is implemented as a set of services that can be used by a variety of visualization applications. The system is designed to be easy to use and to integrate with existing visualization systems.

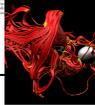



Scalable Computation of Streamlines on Very Large Datasets

David J. Silberman • Lawrence Livermore National Laboratory
 David J. Silberman • Lawrence Livermore National Laboratory
 David J. Silberman • Lawrence Livermore National Laboratory

Abstract

This paper describes a scalable and efficient algorithm for computing streamlines on very large datasets. The algorithm is designed to be efficient and to support a wide range of visualization techniques. It is implemented as a set of services that can be used by a variety of visualization applications. The system is designed to be easy to use and to integrate with existing visualization systems.

Systems research:
 Adaptively applying algorithms in a production env.

Don't be scared! ...
 VisIt developers mostly wear their software engineering hats – not their research hats.

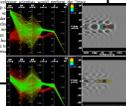
Algorithms research:
 How to efficiently calculate particle paths in parallel.

High Performance Multivariate Visual Data Exploration for Extremely Large Data

David J. Silberman • Lawrence Livermore National Laboratory
 David J. Silberman • Lawrence Livermore National Laboratory
 David J. Silberman • Lawrence Livermore National Laboratory

Abstract

This paper describes a high performance and multivariate visual data exploration system for extremely large data. The system is designed to be efficient and to support a wide range of visualization techniques. It is implemented as a set of services that can be used by a variety of visualization applications. The system is designed to be easy to use and to integrate with existing visualization systems.

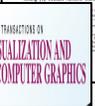
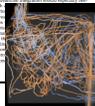
Systems research:
 Using smart DB technology to accelerate processing

Transitions in Visualization and Computer Graphics

David J. Silberman • Lawrence Livermore National Laboratory
 David J. Silberman • Lawrence Livermore National Laboratory
 David J. Silberman • Lawrence Livermore National Laboratory

Abstract

This paper describes transitions in visualization and computer graphics. The system is designed to be efficient and to support a wide range of visualization techniques. It is implemented as a set of services that can be used by a variety of visualization applications. The system is designed to be easy to use and to integrate with existing visualization systems.

Architectural research:
 Hybrid parallelism + particle advection

hpv

David J. Silberman • Lawrence Livermore National Laboratory
 David J. Silberman • Lawrence Livermore National Laboratory
 David J. Silberman • Lawrence Livermore National Laboratory

Abstract

This paper describes the hpv system. The system is designed to be efficient and to support a wide range of visualization techniques. It is implemented as a set of services that can be used by a variety of visualization applications. The system is designed to be easy to use and to integrate with existing visualization systems.



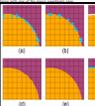

Architectural research:
 Parallel GPU volume rendering

EUROVIS 2010

David J. Silberman • Lawrence Livermore National Laboratory
 David J. Silberman • Lawrence Livermore National Laboratory
 David J. Silberman • Lawrence Livermore National Laboratory

Abstract

This paper describes the EUROVIS 2010 system. The system is designed to be efficient and to support a wide range of visualization techniques. It is implemented as a set of services that can be used by a variety of visualization applications. The system is designed to be easy to use and to integrate with existing visualization systems.

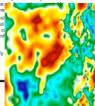
Algorithms research:
 Reconstructing material interfaces for visualization

Frameworks for Visualization at the Extreme Scale

David J. Silberman • Lawrence Livermore National Laboratory
 David J. Silberman • Lawrence Livermore National Laboratory
 David J. Silberman • Lawrence Livermore National Laboratory

Abstract

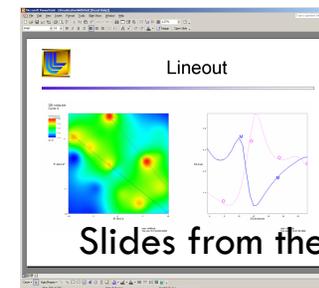
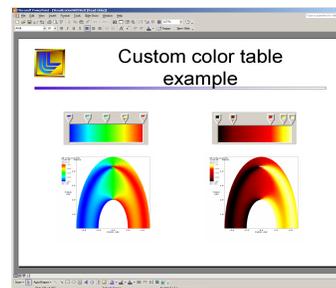
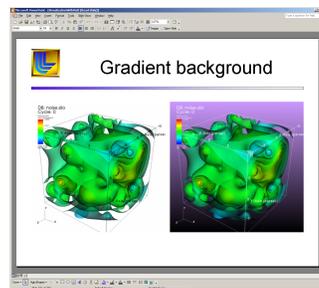
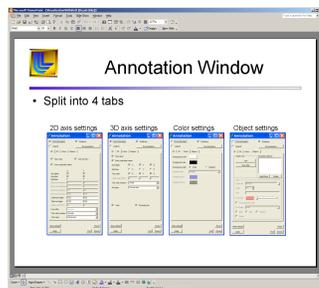
This paper describes frameworks for visualization at the extreme scale. The system is designed to be efficient and to support a wide range of visualization techniques. It is implemented as a set of services that can be used by a variety of visualization applications. The system is designed to be easy to use and to integrate with existing visualization systems.

Methods research:
 How to incorporate statistics into visualization.

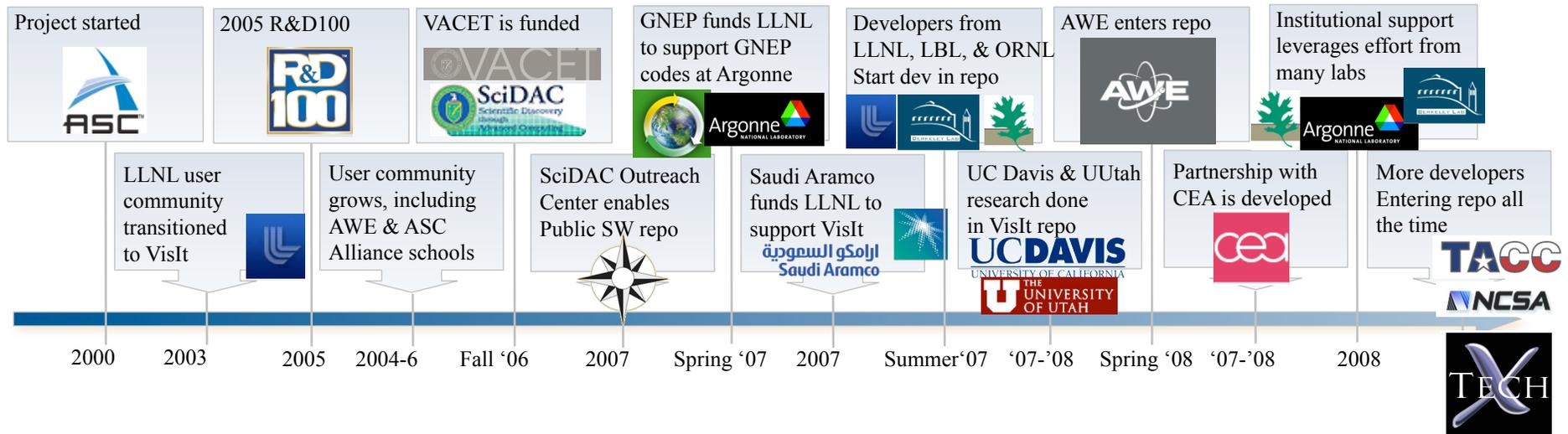
The VisIt team focuses on making a robust, usable product for end users.

- Manuals
 - 300 page user manual
 - 200 page command line interface manual
 - “Getting your data into VisIt” manual
- Wiki for users (and developers)
- Revision control, nightly regression testing, etc
- Executables for all major platforms
- Day long class, complete with exercises



VisIt is a vibrant project with many participants.

- Over 75 person-years of effort
- Over 1.5 million lines of code
- Partnership between: Department of Energy's Office of Science, National Nuclear Security Agency, and Office of Nuclear Energy, the National Science Foundation XD centers (Longhorn XD and RDAV), and more....



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:
 - ▣ Vis experts
 - ▣ Code developers
 - ▣ Code consumers
- Healthy future: vibrant developer and user communities

Before we begin...



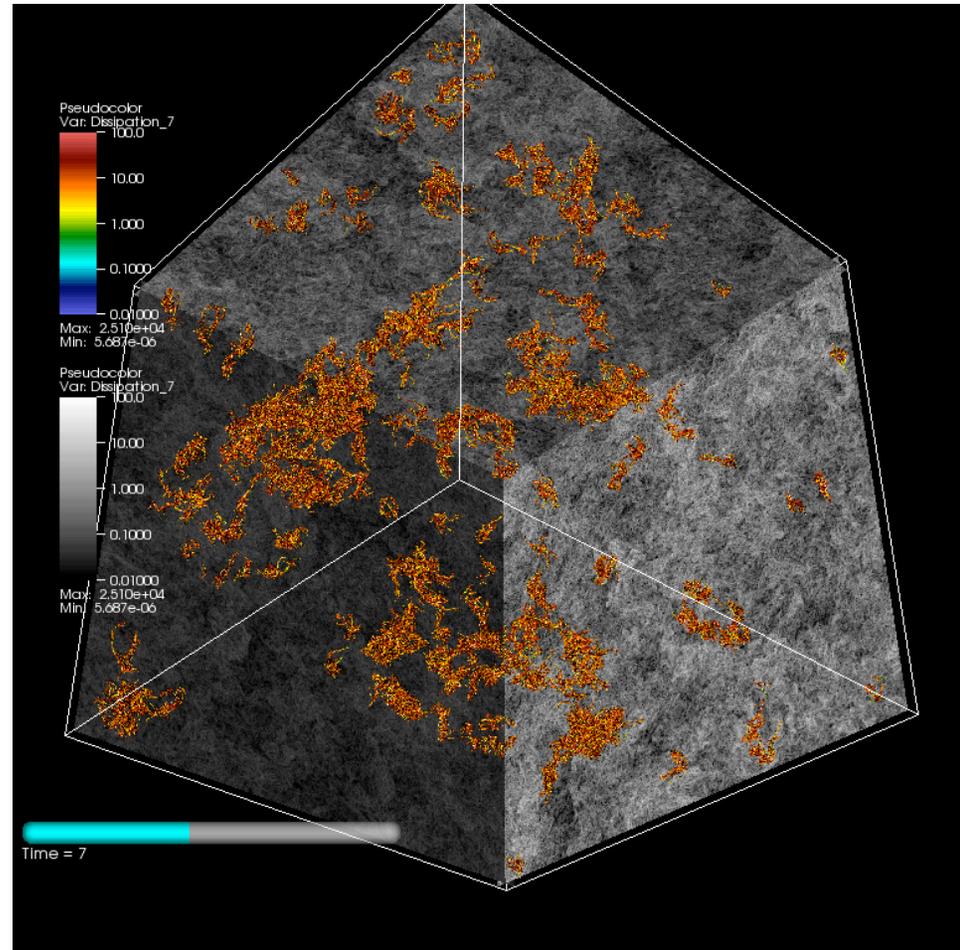
- Reminder: I will discuss file format issues, installation issues, and how to get help at the end of the tutorial
- Important: ask questions any time!
- Course materials: http://www.visitusers.org/index.php?title=SC10_Tutorial

<demonstration begins>

- Visit basics
- Queries and expressions
- Streamlines

Visualizing and Analyzing Large-Scale Turbulent Flow

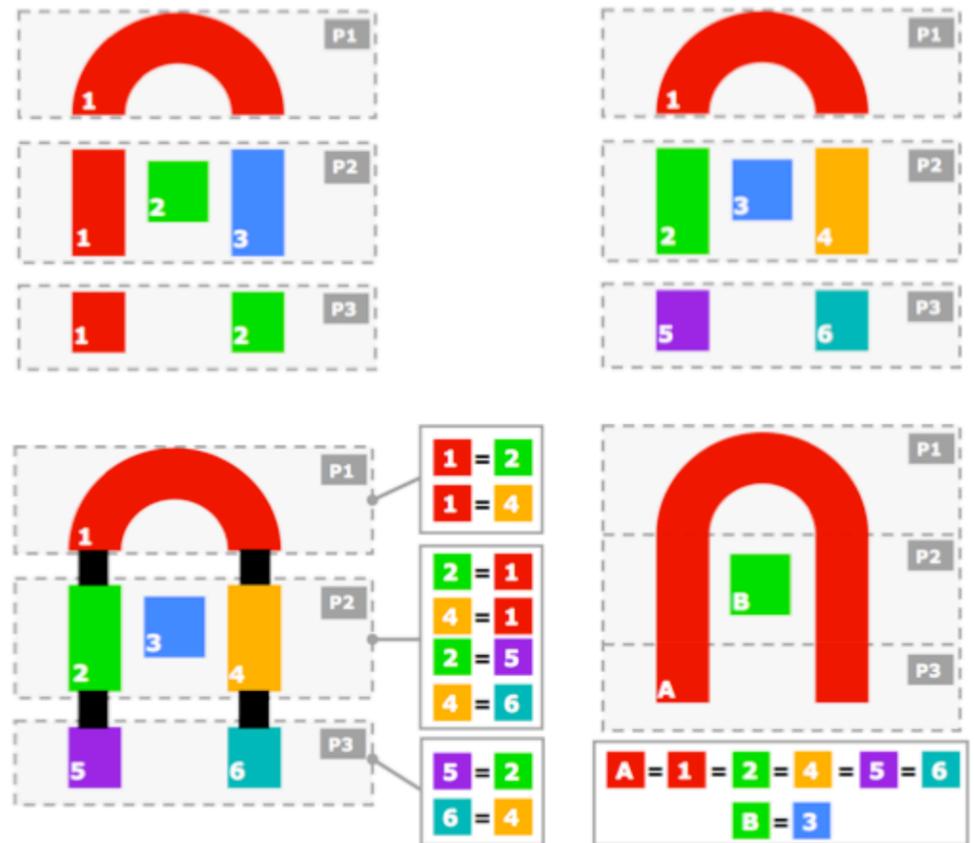
- Detect, track, classify, and visualize features in large-scale turbulent flow.
- Analysis effort by Kelly Gaither (TACC), Hank Childs (LBNL), & Cyrus Harrison (LLNL).
- Stresses two algorithms that are difficult in a distributed memory parallel setting:
 1. Can we identify connected components?
 2. Can we characterize their shape?



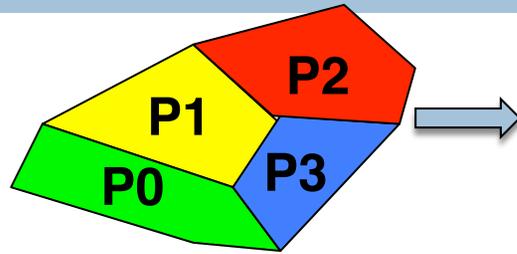
VisIt calculated connected components on a $4K^3$ turbulence data in parallel using TACC's Longhorn machine. 2 million components were initially identified and then the map expression was used to select only the components that had total volume greater than 15. Data courtesy of P.K. Yeung & and Diego Donzis

Identifying connected components in parallel is difficult.

- Hard to do efficiently
- Tremendous bookkeeping problem.
- 4 stage algorithm that finds local connectivity and then merges globally.



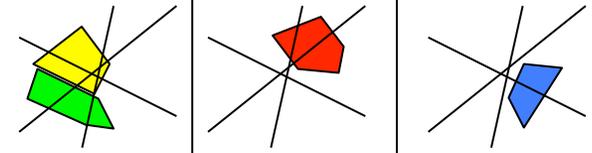
We used shape characterization to assist our temporal tracking.



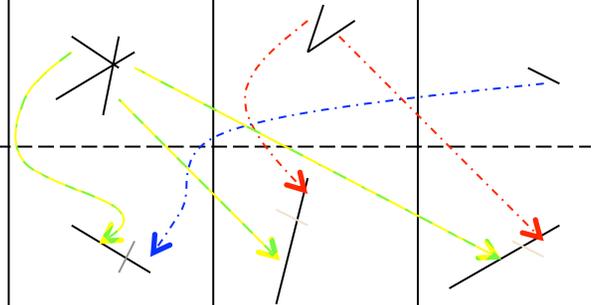
- Shape characterization metric: chord length distribution
- Difficult to perform efficiently in a distributed memory setting

Line Scan Filter

1) Choose Lines



2) Calculate Intersections

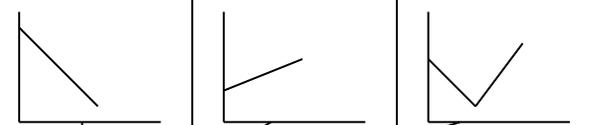


3) Segment redistribution



Line Scan Analysis Sink

4) Analyze lines



5) Collect results



<demonstration continues>

- Scripting
- Moviemaking

“How to make VisIt work after you get home”

- How to get VisIt running on your machine
 - Downloading and installing VisIt
 - Building VisIt from scratch
- How to get VisIt to read your data
- How to get help when you run into trouble
- I like the power of VisIt, but I hate the interface
- How to run client-server

“How to make VisIt work after you get home”

- How to get VisIt running on your machine
 - Downloading and installing VisIt
 - Building VisIt from scratch
- How to get VisIt to read your data
- How to get help when you run into trouble
- I like the power of VisIt, but I hate the interface
- How to run client-server

Can I use a pre-built VisIt binary or do I need to build it myself?

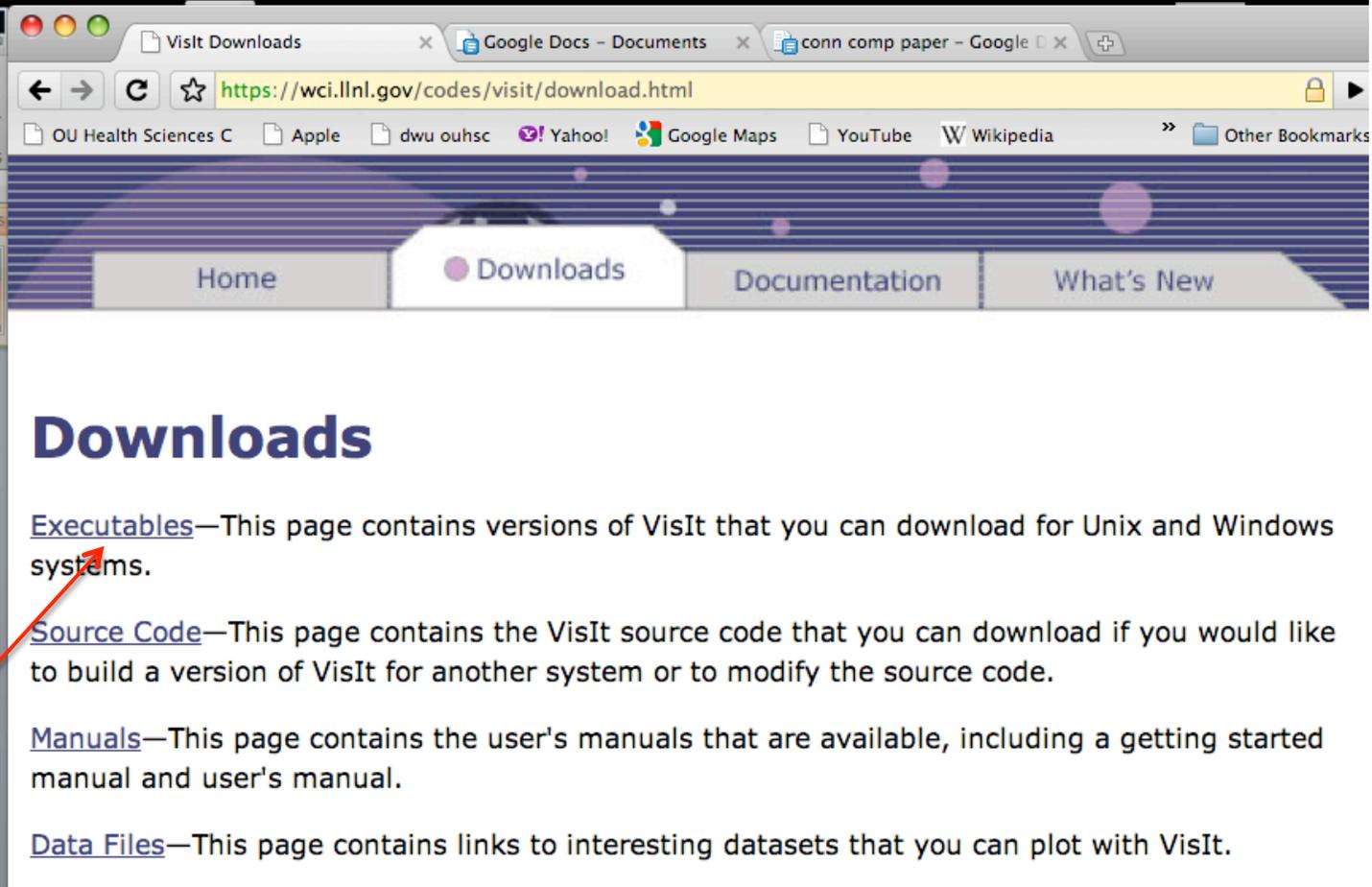
- Pre-built binaries work on most modern machines.
- ... but pre-built binaries are serial only.
 - ▣ Why the VisIt team can't offer parallel binaries:
Your MPI libraries, networking libraries are unlikely to match ours
- Recommendation: try to use the pre-builts first and build VisIt yourself if they don't work.
- Also: all VisIt clients run serial-only. If you want to install VisIt on your desktop to connect to a remote parallel machine, serial is OK.

How do I use pre-built VisIt binaries?

- A: Go to <http://www.llnl.gov/visit>



How do I use pre-built VisIt binaries?



Downloads

[Executables](#)—This page contains versions of VisIt that you can download for Unix and Windows systems.

[Source Code](#)—This page contains the VisIt source code that you can download if you would like to build a version of VisIt for another system or to modify the source code.

[Manuals](#)—This page contains the user's manuals that are available, including a getting started manual and user's manual.

[Data Files](#)—This page contains links to interesting datasets that you can plot with VisIt.

How do I use pre-built VisIt binaries?

VisIt Executables

This page contains links to download VisIt executables for Unix, Windows, and Mac OS X systems. The page contains several versions of VisIt, organized from the most recent to the oldest. The unix and Mac OS X executables require downloading an install script along with the file containing the executable. The Windows executables are packaged in a self contained installer. Instructions for installing VisIt can be found in the install notes. Md5 and sha1 checksums, as well as file sizes are provided for checking that the files were properly downloaded if corruption of the files is suspected during the download process.

VisIt 2.1.0

- [VisIt release notes](#)
- [VisIt install script](#)
- [VisIt install notes](#)
- [VisIt md5 checksums](#)
- [VisIt sha1 checksums](#)
- [VisIt file sizes](#)

Important

platform	executable
Linux - x86 32 bit Redhat Enterprise Linux 3, hoth.llnl.gov 2.4.21-27.0.2c.ELsmp, gcc 3.2.3 Will work on most Linux x86 systems.	 download
Linux - x86_64 64 bit Ubuntu 8.04, pion.ornl.gov 2.6.24-19, gcc 4.2.4	 download
Linux - x86_64 64 bit Redhat Enterprise Linux 4, photon.ornl.gov 2.6.9-89.0.20.ELsmp, gcc 3.4.6 Will work on most Linux x86_64 systems.	 download

How do I use pre-built VisIt binaries?

Linux - x86_64 64 bit Ubuntu 8.04, pion.ornl.gov 2.6.24-19, gcc 4.2.4	 download
Linux - x86_64 64 bit Redhat Enterprise Linux 4, photon.ornl.gov 2.6.9-89.0.20.ELsmp, gcc 3.4.6 Will work on most Linux x86_64 systems.	 download
Linux - x86_64 64 bit Redhat Enterprise Linux 5, yana.llnl.gov 2.6.18-76chaos, gcc 4.1.2 Will work on most Linux x86_64 systems.	 download
Linux - x86_64 64 bit Scientific Linux SL release 5.4, euclid.nersc.gov 2.6.18-164.9.1.el5-bsdvs3, gcc 4.1.2	 download
Windows (Xp / Vista / 7) 32 bit MSVC8, Visual Studio 2005	 download
Mac OS X - Intel Darwin 10.5, Darwin Kernel Version 9.7.0, gcc 4.0.1, OpenMPI <i>(Includes parallel VisIt compatible with MacOS X 10.5's default MPI)</i>	 download
Mac OS X - Intel 64 bit Darwin 10.6.3, Darwin Kernel Version 10.3.0, gcc 4.2.1, OpenMPI <i>(Includes parallel VisIt compatible with MacOS X 10.6's default MPI)</i>	 download
Mac OS X - Intel Darwin 10.4	 download
AIX - 32 bit AIX 5.3, up.llnl.gov 00C5D6DD4C00, xlc	 download
AIX - 64 bit AIX 5.3, up.llnl.gov 00C5D6DD4C00, xlc	 download
Java client library (jar file, compiled classes, source code, examples)	 download

How do I use the pre-built VisIt binaries?

□ Unix & Mac:

- Download install script
- Download binary
- Run install script
- --or—
- Download binary
- Untar



Good for host profiles, maintaining multiple versions, multiple OSs



Quick & easy

□ Windows:

- Download installer program & run

□ Full install notes:

- https://wci.llnl.gov/codes/visit/2.1.0/INSTALL_NOTES

Important step: choosing host profiles

- Many supercomputing sites have set up “host profiles”.
 - ▣ These files contain all the information about how to connect to their supercomputers and how to launch parallel jobs there.
- You select which profiles to install when you install VisIt.
- Profiles that come with VisIt:
 - ▣ NERSC, LLNL Open, LLNL Closed, ORNL, Argonne, TACC, LBNL desktop network, Princeton, UMich CAC
- Other sites maintain profiles outside of VisIt repository.
 - ▣ If you know folks running VisIt in parallel at a site not listed above, ask them for their profiles.

“How to make VisIt work after you get home”

- How to get VisIt running on your machine
 - Downloading and installing VisIt
 - Building VisIt from scratch
- How to get VisIt to read your data
- How to get help when you run into trouble
- I like the power of VisIt, but I hate the interface
- How to run client-server

Building VisIt from scratch

- Building VisIt from scratch on your own is very difficult.
- ... but the “build_visit” script is fairly reliable.

Automatically build VisIt with the *build_visit* script!

[Download build_visit script here.](#)

VisIt can now be built automatically using the [build_visit](#) script on many Linux, MacOS X, and AIX platforms (*more to come*). The [build_visit](#) script takes care of downloading relevant VisIt and 3rd party source code, configuring, and building it all using your C++ compiler. We encourage users to build VisIt using the [build_visit](#) script when our binary distributions have trouble running on some systems. We also recommend using the [build_visit](#) script on your system if you plan to:

- Modify the VisIt source code.
- Run a parallel compute engine. Building a parallel version of VisIt on your system allows you to configure VisIt so it uses your MPI library, avoiding incompatibilities.
- Create your own VisIt plugins. Building VisIt on your system ensures that it is built with the same C++ compiler that you will use to develop your plugin, minimizing the chance for runtime library incompatibilities.



What “build_visit” does



- Downloads third party libraries
- Patches them to accommodate OS quirks
- Builds them
- Creates “config-site” file, which communicates information about where 3rd party libraries live to VisIt’s build system.
- Downloads VisIt source code
- Builds VisIt

“build_visit” details



- There are two ways to use build_visit:
 - Curses-style GUI
 - Command line options through `–console`
 - Developers all use `–console` and it shows!!
- Tips:
 - Don't build every third party library unless you really need to.
 - Set up a “`—thirdparty-path`”.

“build_visit” details



- Q: How long does build_visit take? A: hours
- Q: I have my own Qt/VTK/Python, can I use those?
 - ▣ Hank highly recommends against
- Q: What happens after build_visit finishes?
 - ▣ A1: you can run directly in the build location
 - ▣ A2: you can make a package and do an install like you would with the pre-built binaries

“build_visit” details

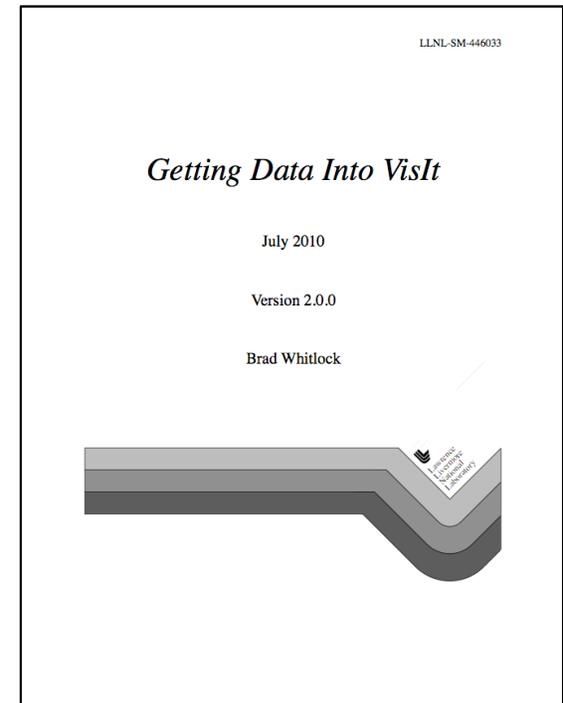
- Most common build_visit failures:
 - gcc is not installed
 - X11 development package is not installed
 - OpenGL development package is not installed
- Most common VisIt runtime failure: really antique OpenGL drivers.
 - Hank runs SUSE 9.1 (from 2005) at home.
- Build process for Windows is very different. Rarely a need to build on Windows, aside from VisIt development.

“How to make VisIt work after you get home”

- How to get VisIt running on your machine
 - Downloading and installing VisIt
 - Building VisIt from scratch
- How to get VisIt to read your data
- How to get help when you run into trouble
- I like the power of VisIt, but I hate the interface
- How to run client-server

How to get your data into VisIt

- There is an extensive (and up-to-date!) manual on this topic: “Getting Your Data Into VisIt”
- Three ways:
 - Use a known format
 - Write a file format reader
 - In situ processing
 - Latter two covered in afternoon course

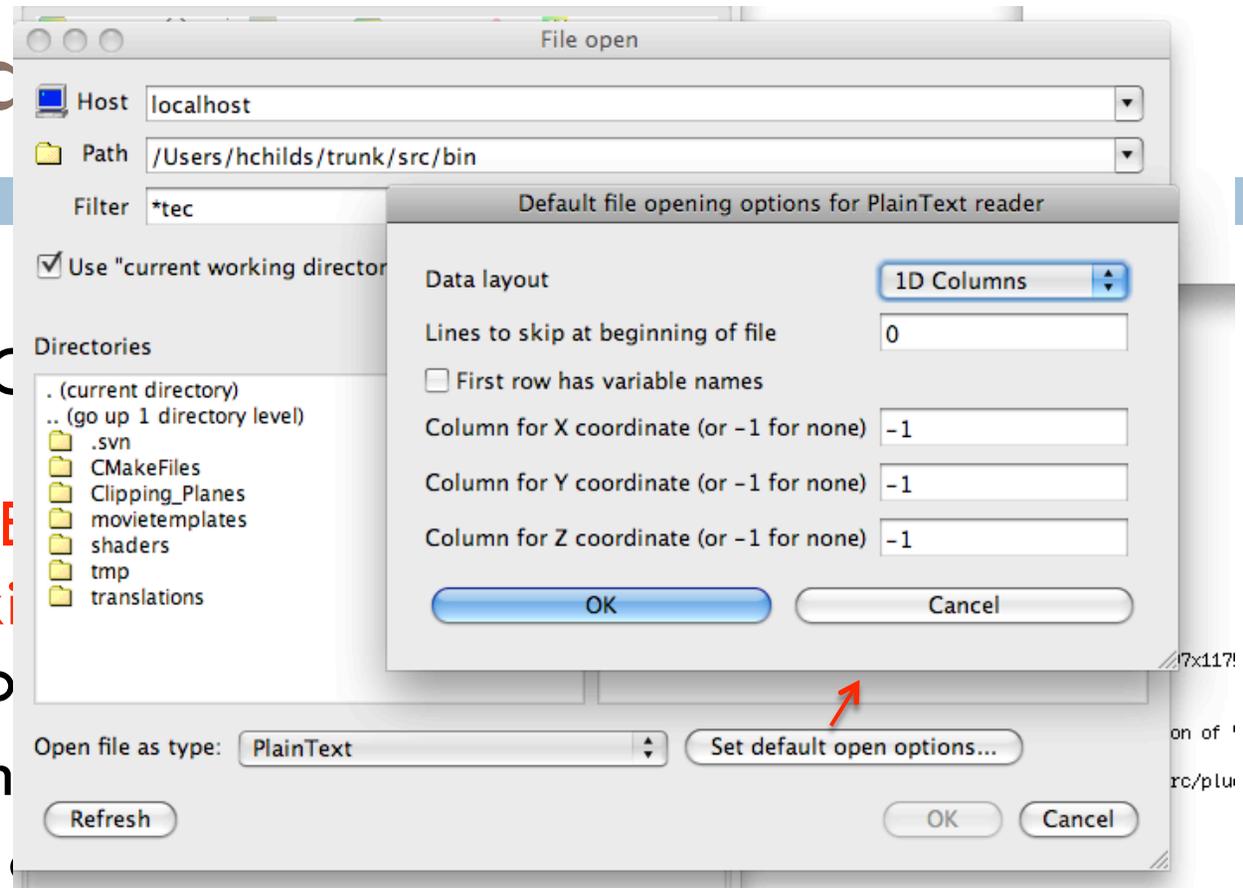


File formats that VisIt supports

- 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
 - 113 total readers
- 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 format

- ADIOS, BOV, EnSight, ENZO, Images (TIFF, NASTRAN, NE, PlainText, Pixar) and many more
- BOV: raw binary
 - → you have a file that describes dimensions
- PlainText: reads space delimited columns.
 - Controls for specifying column purposes



File formats that VisIt supports

- 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
- NETCDF: VisIt reader understands many (but not all) conventions
- Pixie: most general HDF5 reader
 - ▣ Many other HDF5 readers

File formats that VisIt supports

- 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
- 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.

File formats that VisIt supports

- 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
- VTK: not built for performance, but it is great for getting into VisIt quickly
- Silo: higher barriers to entry, but performs well and fairly mature

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 Visualization Toolkit also provides some of its own file formats. The main reason for creating yet another data file format is to provide a consistent data representation for a variety of dataset types, and to provide a simple method to communicate it 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 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: `# vtkDataFile Version x.x`. This line must be exactly followed by the specification of the version number `x.x`, which will vary with different releases of VTK. Note: the current version number is 3.0. Versions 3.0 files are compatible with version 3.0.
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 string `ASCII` or `BINARY` must appear.
4. The fourth part is the data itself. The data describes the geometry and data of the dataset. This part begins with a keyword that is followed by the keyword `POINTS` and the type of dataset. Then, depending upon the type of dataset, other keyword/data combinations define the actual data.
5. The final part describes the dataset attributes. This part begins with the keywords `POINT_DATA` or `CELL_DATA`, followed by an integer number specifying the number of points or cells, respectively. (It doesn't matter whether `POINT_DATA` or `CELL_DATA` 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 objects. Not all data types are supported for all classes.

VisItWriter 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 VisitWriter 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

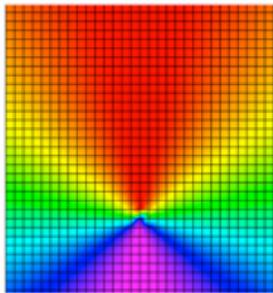


- Silo is a mature, self-describing file format that deals with multi-block data.
- It has drivers on top of HDF5, NetCDF, and “PDB”.
- Fairly rich data model
- More information:
 - ▣ <https://wci.llnl.gov/codes/silo/>

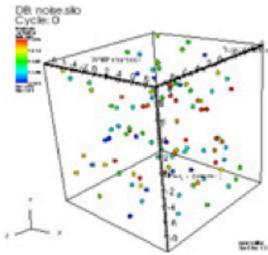
Silo features

Welcome to Silo

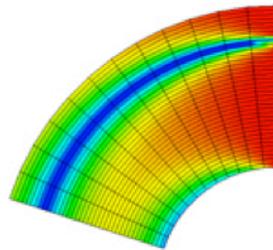
A mesh and field I/O library and scientific database



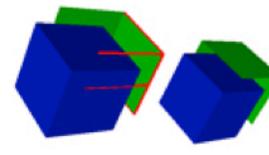
Structured Rectilinear Mesh



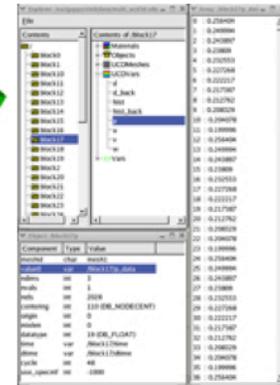
Gridless Point Mesh



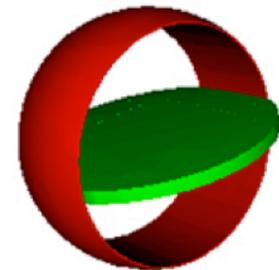
Structured (Curvilinear) Mesh



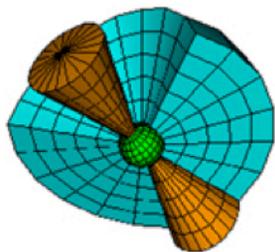
Arbitrary Subsets



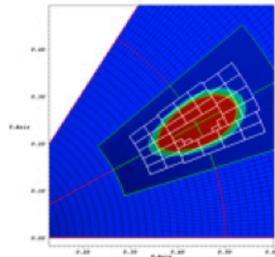
Silex browser for Silo files



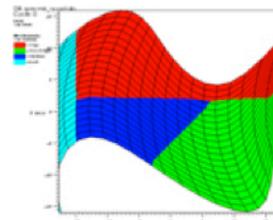
Constructive Solid Geometry (CSG) Mesh



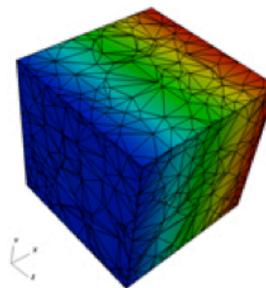
Unstructured Zoo (UCD) Mesh



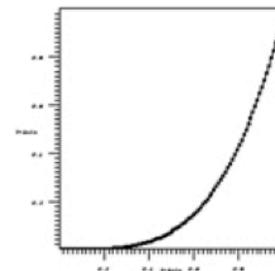
Adaptive Mesh Refinement (AMR) Mesh



Mixing Materials



Arbitrary Polyhedral Mesh



XY Curve

“How to make VisIt work after you get home”

- How to get VisIt running on your machine
 - Downloading and installing VisIt
 - Building VisIt from scratch
- How to get VisIt to read your data
- How to get help when you run into trouble
- I like the power of VisIt, but I hate the interface
- How to run client-server

How to get help when you run into trouble

□ Six options:

□ FAQ

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

□ Documentation

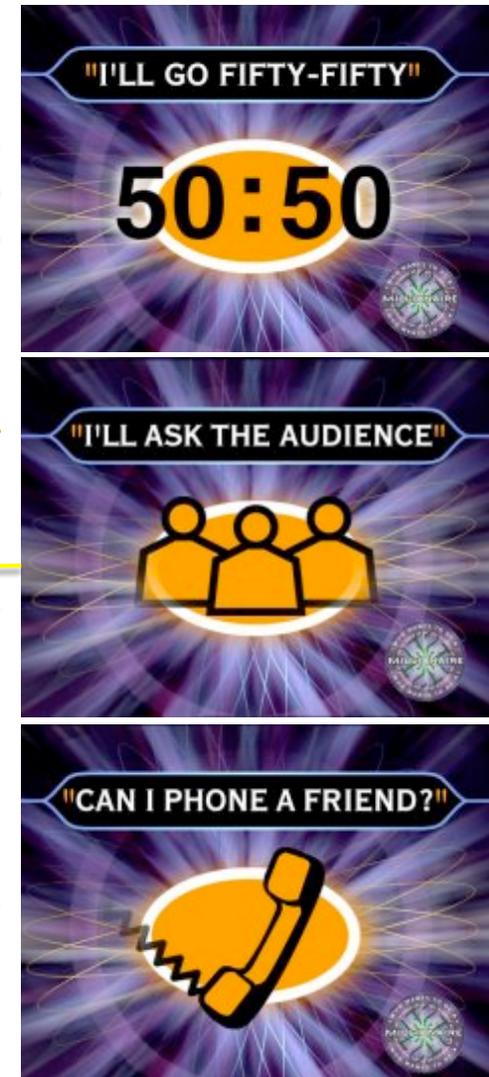
- <https://wci.llnl.gov/codes/visit/doc.html>
- <http://www.visitusers.org>

□ Vislt-users mailing list

□ Vislt-users archives

□ Vislt users forum

□ Vislt-help-XYZ mailing list



Manuals & documentation



- Getting started manual
- Users manual (old, but still useful)
- Python interface (to be updated in two weeks)
- Getting Data Into VisIt
- VisIt Class Slides
- VisIt Class Exercises
- This Tutorial

Visitusers.org

- Users section has lots of practical tips:
 - “I solved this problem with this technique”
 - “Here’s my script to do this functionality”
- In practical terms, this is a staging area for formal documentation in the future.

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](#)

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



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] Building Parallel Visit: Issue w/ Qt

Vedran Coralic [vcoralic at caltech.edu](mailto:vcoralic@caltech.edu)

Wed Nov 3 00:21:37 EDT 2010

- Previous message: [\[visit-users\] Building Parallel Visit: Issue w/ Qt](#)
- Next message: [\[visit-users\] makemili](#)
- Messages sorted by: [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#)

Thank you very much Jeremy! That seemed to do the trick. I have now finished successfully building VisIt.

2010/11/2 Meredith, Jeremy S. <[jsmeredith at ornl.gov](mailto:jsmeredith@ornl.gov)>

> Here's what I did to work around this problem:

> - when the Qt build fails, cd into the Qt directory and type "make install"

> - this appears to immediately start putting the libraries in the installation location even though the build "failed"

> - as soon as it's put the libQCLucene stuff into the installation location, kill the build

> - now type "make", and it will finish building successfully

> - and when it's done, type "make install" and it will finish installing

Archives by thread

[\[subject \]](#) [\[author \]](#) [\[date \]](#)

EDT 2010

ST 2010

[Building Parallel Visit: Issue w/ Qt](#) Vedran Coralic
[Building Parallel Visit: Issue w/ Qt](#) Meredith, Jeremy S.
[Building Parallel Visit: Issue w/ Qt](#) Vedran Coralic
Daniel, James L ERDC-GSL-MS
[li Seipel, William F NWO](#)
[makemili Seipel, William F NWO](#)
[se/RCP](#) Leguay Romain
[Eclipse/RCP](#) Hank Childs
[of recorded macros?](#) Cyrus Harrison
[hardware acceleration problems](#) Patrick Shinpaugh
[on a CentOS server](#) Katie Boyle

- [\[visit-users\] Running Visit on a CentOS server](#) Cyrus Harrison
- [\[visit-users\] Running Visit on a CentOS server](#) J.S. van Bethlehem
- [\[visit-users\] Controlling Annotation objects through cli](#) Shriram Jagannathan
- [\[visit-users\] Controlling Annotation objects through cli](#) Cyrus Harrison
 - [\[visit-users\] Controlling Annotation objects through cli](#) Shriram Jagannathan
 - [\[visit-users\] Controlling Annotation objects through cli](#) J.S. van Bethlehem

question from Australia to be answered by a European in white
I'm asleep

- ❑ List: visit-users@ornl.gov
- ❑ More information:
<https://email.ornl.gov/mailman/listinfo/visit-users>
- ❑ Archive: <https://email.ornl.gov/pipermail/visit-users/>

Board Topics							
		CGNS OversetHoles « Pages 1 2 »		tpg2114	16	160	→ 11/11/10 at 22:39:11 By: cean
		3d vector on 2d mesh?		tsch	2	34	→ 11/10/10 at 09:26:36 By: Jeremy Meredith
		Image messed up when save		Pinpin	13	150	→ 11/09/10 at 12:14:29 By: BradWhitlock
		pseudocolor plot legend attributes in python		Jennifer	2	17	→ 11/07/10 at 22:11:27 By: Jennifer
		graph along 2D					
		threshold variable c					
		Python compatibility					
		python interface ...					
		Mesa: 'make' f					
		failed					
		applyOperator					
		how to get cycle on					
		annotation?					
		« Pages 1 2 »					
		Averaging 2D s					
		databases					
		smooth operator					
		Appearance of lines					
		Add and read p					
		No image was					

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

pseudocolor plot legend attributes in python
11/07/10 at 19:06:30

Hello. I want to set the attributes for a pseudocolor plot legend in a python script such as the location of the legend (turn off Let VisIt manage legend position), the X-scale & Y-scale, the number of Tic Marks, and the label appearance (number format, font height). Is it possible to set these properties in a python script? If so, how can I do this?

I tried to use the Command Control to record these changes, but the output states:
"# Logging for AddAnnotationObject is not implemented yet."
"# Logging for SetAnnotationObjectOptions is not implemented yet."

Thanks,
Jennifer

[Back to top](#) IP Logged

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 referred to through that same index.

```
>>> 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 Visit funding and get direct support.
 - ▣ These customers can post directly to visit-help-xyz without being a subscriber
 - ▣ The messages are received by all Visit developers and supported communally
- Lists:
 - ▣ Visit-help-asc, visit-help-scidac, visit-help-gnep, visit-help-ascem

“How to make VisIt work after you get home”

- How to get VisIt running on your machine
 - Downloading and installing VisIt
 - Building VisIt from scratch
- How to get VisIt to read your data
- How to get help when you run into trouble
- I like the power of VisIt, but I hate the interface
- How to run client-server

It is possible (although non-trivial) to write a custom user interface to VisIt

The screenshot displays the VORPAL Composer software interface, which is a custom user interface for VisIt. The interface is divided into several panels:

- CONTROLS:** A panel on the left containing a list of variables under categories: <Scalars>, <Vectors>, and <Meshes>. The <Scalars> list includes SumRho_0 through SumRho_3, Rho, J_magnitude, YeeElecField_0 through YeeElecField_2, E_magnitude, YeeMagField_0 through YeeMagField_2, B_magnitude, pycavity_0 through pycavity_1, and universe_0 through universe_1. The <Vectors> list includes J, E, and B. Below the list are options for Clip Mode, Slice (set to YeeElecField_1), Align slice/clip to axis (X, Y, Z), Annotation Level (4 - all), and Primary Window (3d view). Buttons for Reset Views and Save Image are at the bottom.
- VISUALIZATION:** A central 3D view showing a simulation of a plasma structure. The structure is rendered in green and is surrounded by a red wireframe box. A red arrow labeled "Normal <0 0 1>" points downwards. A black arrow labeled "Ugh <0 0 0>" points to the left. The axes are labeled X, Y, and Z. A color scale legend for "Pseudocolor Var: YeeElecField_1" is shown on the left, with values ranging from -433.9 to 202.5. The text "DB: c:\src\ovp\bin\ovp, YeeElecField_1.v0" is visible at the top of the 3D view.
- 2-d view:** A panel on the right showing a 2D projection of the simulation data, with a color scale legend.
- Line-out:** A panel on the right showing a line plot of the simulation data, with a color scale legend.

At the bottom right of the VORPAL Composer window, the text "user: mduvant Fri Nov 12 17:14:20 2010" is displayed.

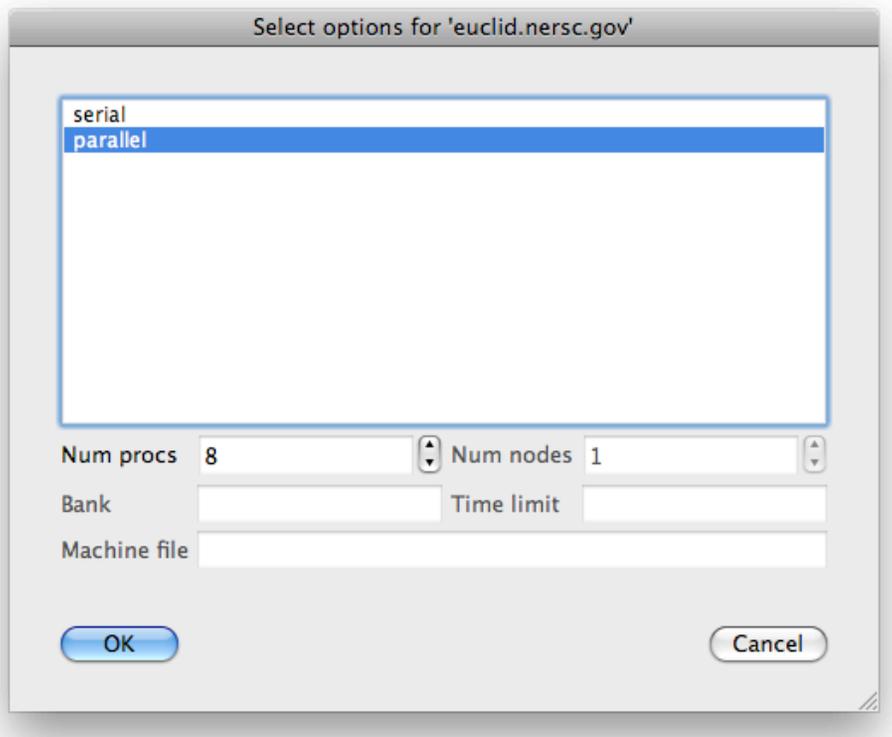
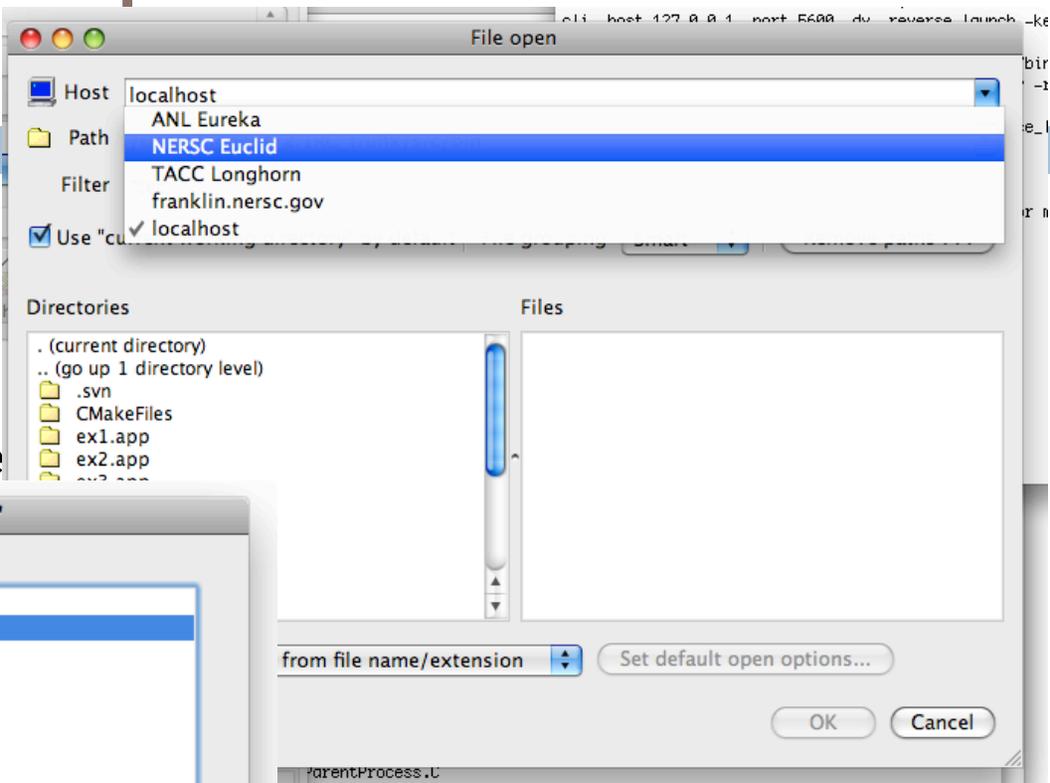


“How to make VisIt work after you get home”

- How to get VisIt running on your machine
 - Downloading and installing VisIt
 - Building VisIt from scratch
- How to get VisIt to read your data
- How to get help when you run into trouble
- I like the power of VisIt, but I hate the interface
- **How to run client-server**

How to run cli

- There are two critical steps:
 - Connecting to the host
 - Getting an engine



Thank you for coming!!

- Hank Childs, hchilds@lbl.gov
- Amy Szczepanski, aszczepa@utk.edu
- Dave Pugmire, pugmire@ornl.gov
- User resources:
 - Main website: <http://www.llnl.gov/visit>
 - Wiki: <http://www.visitusers.org>
 - Email: visitusers@ornl.gov
 - Email: visit-help-scidac@ornl.gov
- Development resources:
 - Email: visit-developers@ornl.gov
 - SVN: <http://portal.nersc.gov/svn/visit>

