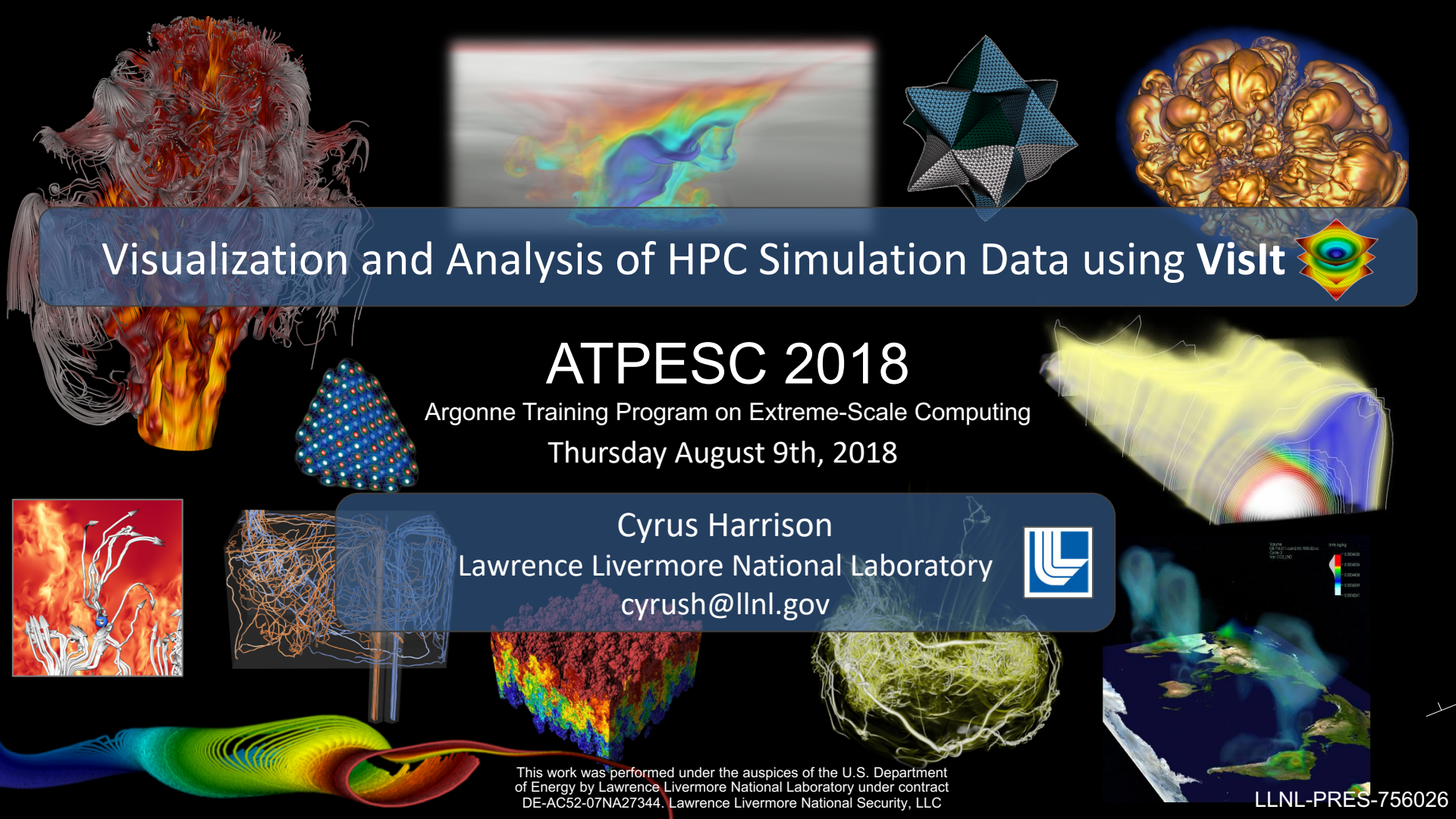


Visualization and Analysis of HPC Simulation Data using **VisIt**

ATPESC 2018

Argonne Training Program on Extreme-Scale Computing

Thursday August 9th, 2018

Cyrus Harrison

Lawrence Livermore National Laboratory

cyrush@llnl.gov

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

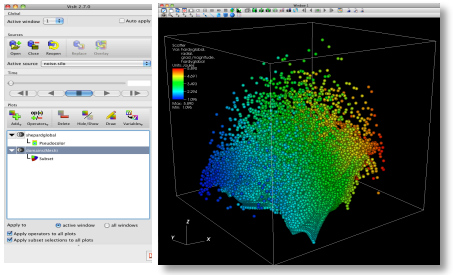
LLNL-PRES-756026

ATPESC 2018 Outline

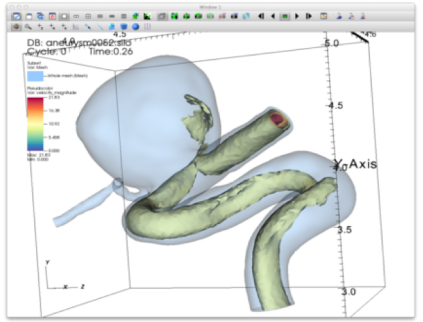
- VisIt Project Introduction (30 min)

  {Lunch!}

- Hands-on: (1.5 hours)
  - Guided tour of VisIt
  - Visualization of an Aneurysm (Blood Flow) Simulation



Guided Tour of VisIt



Hands-on Exploration

Tutorial Resources

- **Tutorial Materials**
  - http://visitusers.org/index.php?title=VisIt_Tutorial
  - **Tutorial Preparation**
    - http://visitusers.org/index.php?title=Tutorial_Preparation
  - **VisIt Binaries**
    - https://wci.llnl.gov/codes/visit/executables.html
  - **Example Datasets**
    - http://visitusers.org/index.php?title=Tutorial_Data

Tutorial Data Acknowledgements

Aneurysm Simulation Dataset

Simulated using the LifeV (http://www.lifev.org/) finite element solver.

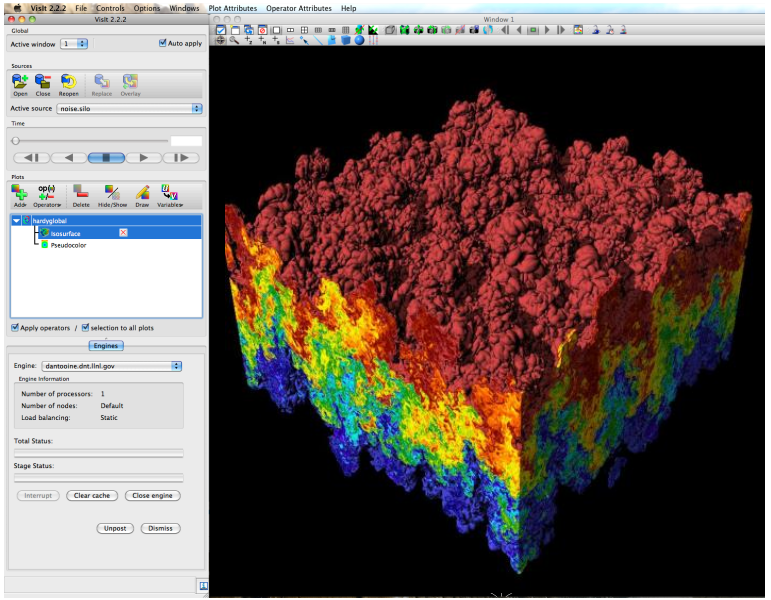
Available thanks to:

- Gilles Fourestey and Jean Favre
  Swiss National Supercomputing Centre (http://www.cscs.ch/)

VisIt Project Introduction

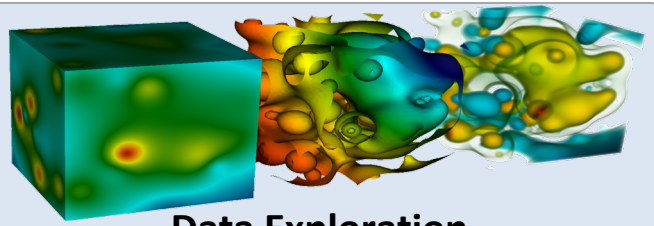
VisIt is an open source, turnkey application for data analysis and visualization of mesh-based data

- 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.

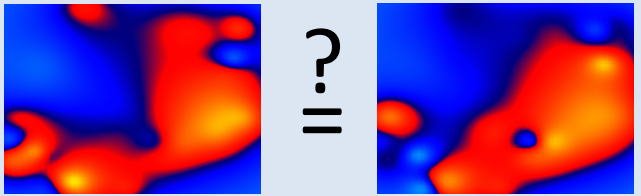


Pseudocolor plot of Density
(27 billion element dataset)

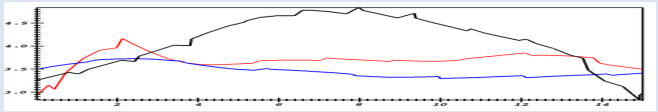
VisIt supports a wide range of use cases



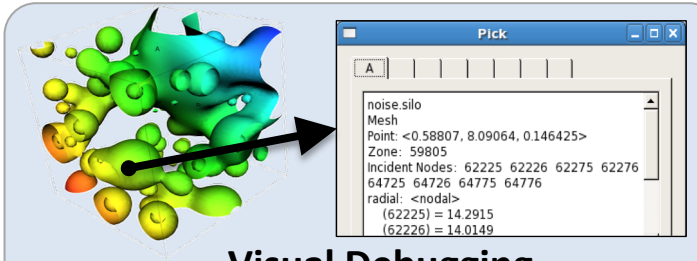
Data Exploration



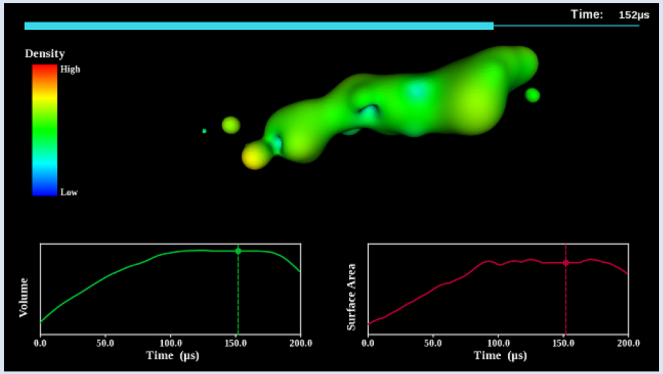
Comparative Analysis



Quantitative Analysis

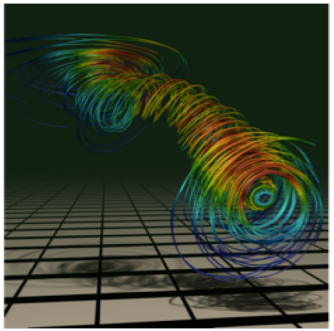


Visual Debugging

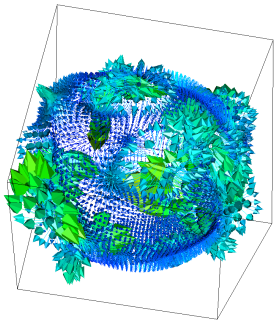


Presentation Graphics

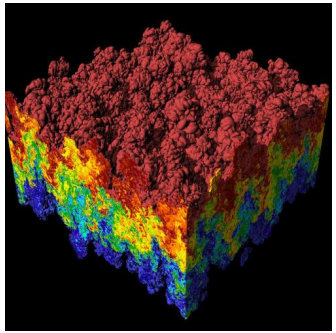
VisIt provides a wide range of plotting features for simulation data across many scientific domains



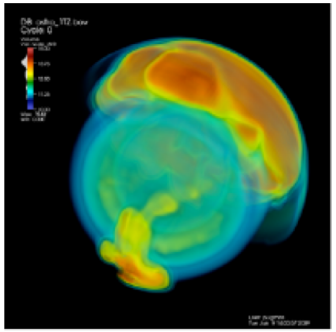
Streamlines / Pathlines



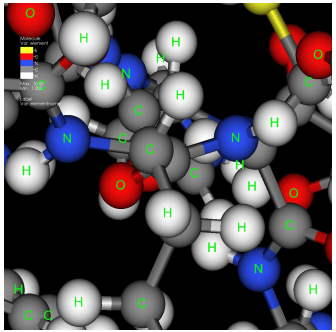
Vector / Tensor Glyphs



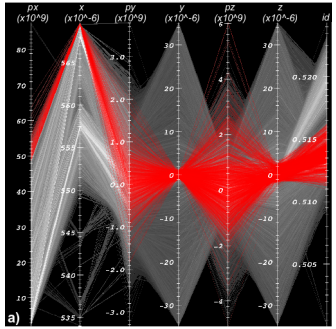
Pseudocolor Rendering



Volume Rendering

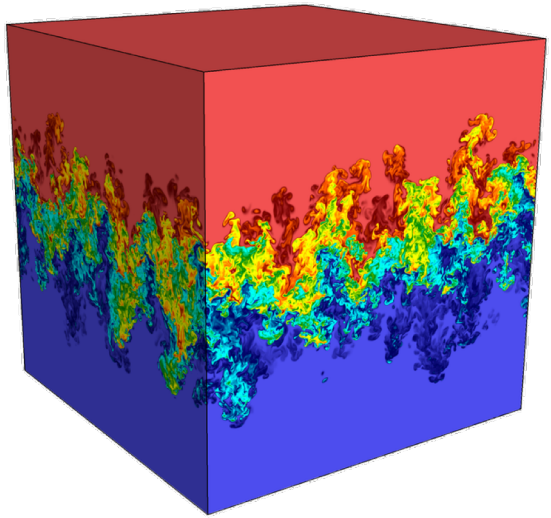


Molecular Visualization

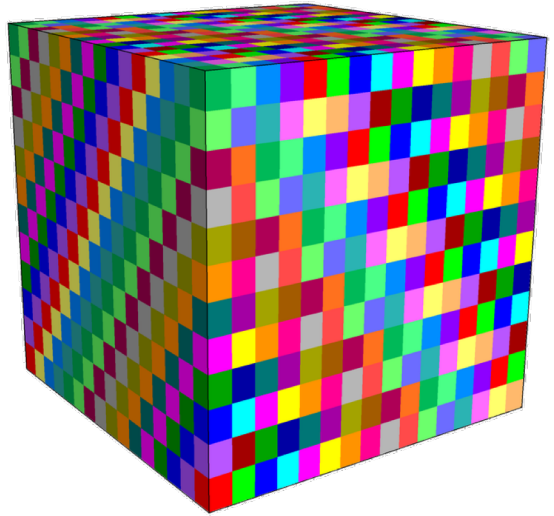


Parallel Coordinates

VisIt uses MPI for distributed-memory parallelism on HPC clusters



Full Dataset
(27 billion total elements)



3072 sub-grids
(each 192x129x256 cells)

We are enhancing VisIt's pipeline infrastructure to support threaded processing and many-core architectures

VisIt is a vibrant project with many participants

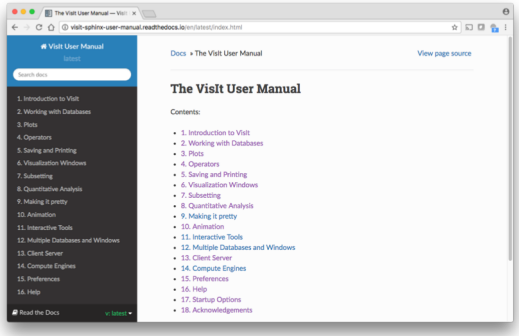
- The VisIt project started in 2000 to support LLNL's large scale ASC physics codes.
- The project grew beyond LLNL and ASC with development from DOE SciDAC and other efforts.
- VisIt is now supported by multiple organizations:
  - LLNL, LBNL, ORNL, Univ of Oregon, Univ of Utah, Intelligent Light, …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

Project Started	LLNL ASC users transitioned to VisIt	2005 R&D 100	VACET Funded	Transition to Public SW repo	VisIt 2.0 Release	R&D Collaborations
2000	2003	2005	2006	2008	2010	2012 - 2020

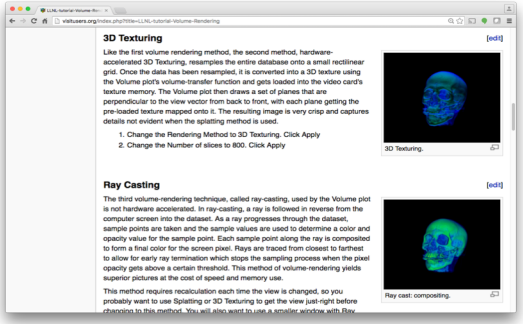
The VisIt team focuses on making a robust, usable product for end users

- **Regular Releases (~ 6 / year)**
  - Binaries for all major platforms
  - End-to-end build process script ``build_visit''
- **User 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**
  - Users Reference Manual

  http://visit-sphinx-user-manual.readthedocs.io/en/latest/index.html
  - Getting Data Into VisIt Manual
  - Python Interface Manual



VisIt User Manual



Tutorials on visitusers.org

The VisIt team releases binaries for several platforms and a script that automates the build process

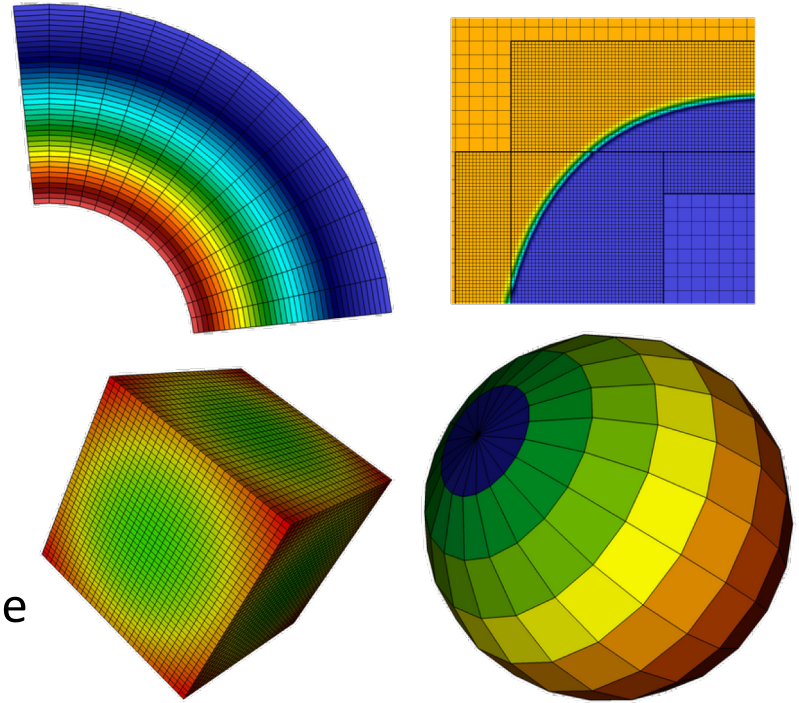
“How do I obtain VisIt?”

- Use an existing build:
  - For your Laptop or Workstation:
    - Binaries for Windows, OSX, and Linux (RHEL + Ubuntu): (https://wci.llnl.gov/simulation/computer-codes/visit/executables)
  - For your favorite HPC Cluster:
    - Several HPC centers have VisIt installed

- Build VisIt yourself:
  - “build_visit” is a script that automates the process of building VisIt and its third-party dependencies. (also at: https://wci.llnl.gov/simulation/computer-codes/visit/executables)
  - Fledgling support for building via spack (https://github.com/spack/spack)

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
  - Primarily linear element support, limited quadratic element support
- **Field Types**
  - Scalar, Vector, Tensor, Material Volume Fractions, Species

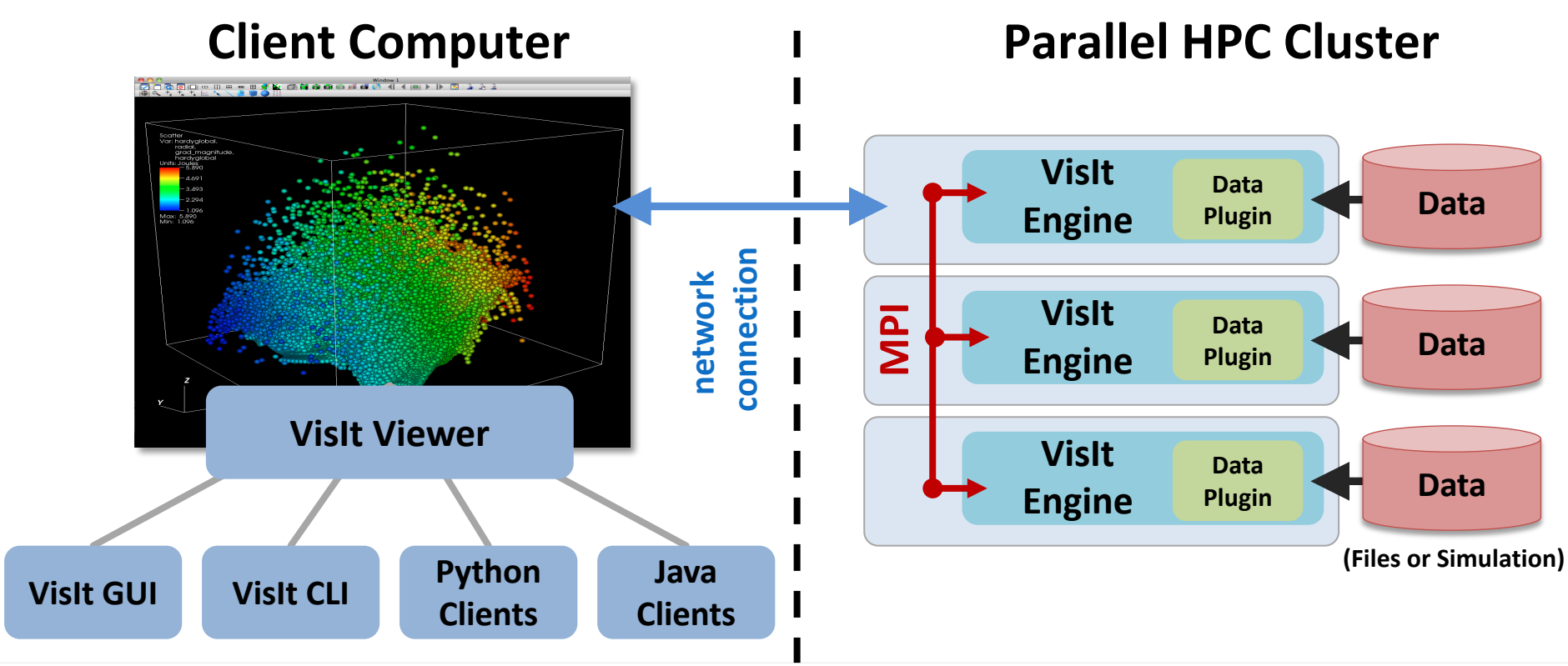


VisIt supports more than 110 file formats

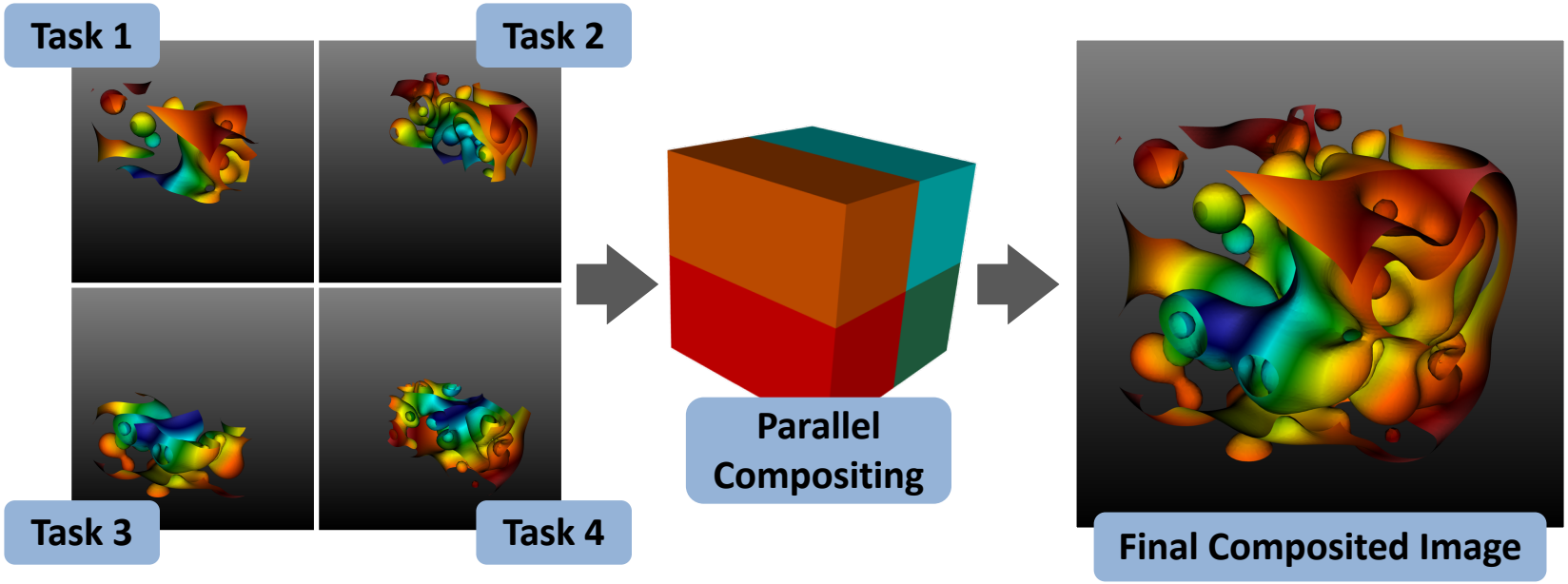
“How do I get my data into VisIt?”

- The *PlainText* database reader can read simple text files (CSV, etc)
  - http://visitusers.org/index.php?title=Using_the_PlainText_reader
- Experiment with the *visit_writer* utility:
  - http://visitusers.org/index.php?title=VisItWriter
- Write to a commonly used format:
  - *VTK, Silo, Xdmf, PVTK*
- We are ramping up support for Mesh-based data in Conduit Blueprint:
  - http://llnl-conduit.readthedocs.io/en/latest/blueprint_mesh.html
- Consult the Getting Data Into VisIt Manual and its associated source code examples.

VisIt employs a parallelized client-server architecture



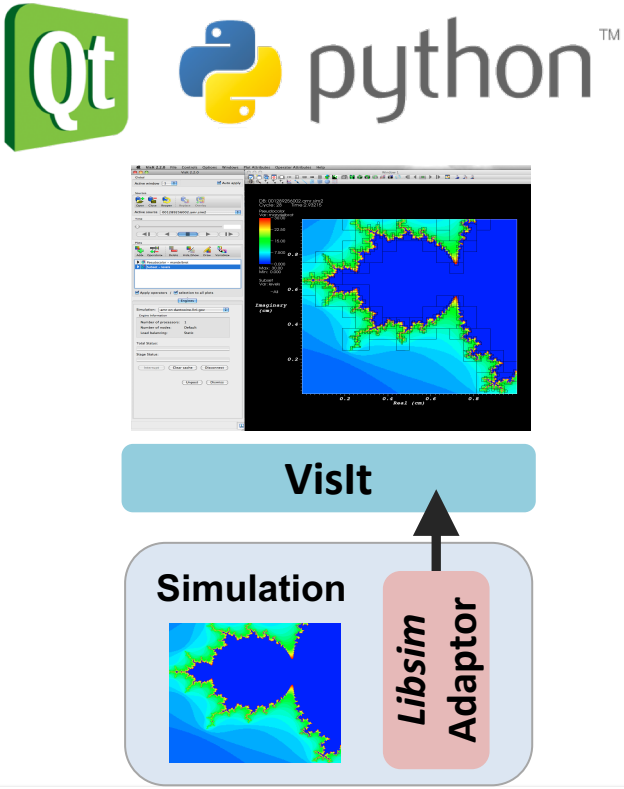
VisIt automatically switches to a scalable rendering mode when plotting large data sets on HPC clusters



In addition to scalable surface rendering, 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
- **In-Situ Coupling**
  - VisIt's *Libsim* library allows simulation codes to link in VisIt's engine for in situ visualization



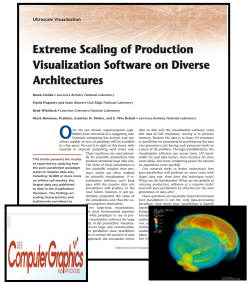
VisIt is used as a platform to deploy visualization research

- **DOE Research Collaborations**

SciDAC
Scientific Discovery through Advanced Computing

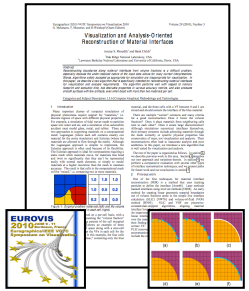
ECP

- **Research Focus Ares**
  - Light weight In Situ Processing
  - Node Level Parallelism
  - Distributed Memory Parallel Algorithms



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

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



Algorithms research:
Reconstructing material interfaces for visualization

Methods research:
How to incorporate statistics into visualization.

DOE's visualization community is collaborating to create open source tools ready for Exascale simulations

Addressing node-level parallelism

- VTK-m is an effort to provide a toolkit of visualization algorithms that leverage emerging node-level HPC architectures
- We are also exploring using VTK-m and DIY to share more distributed-memory infrastructure across projects

VTK-m

http://m.vtk.org

DIY

https://github.com/diatomic/diy

Addressing I/O gaps with in-situ

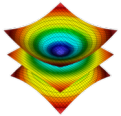
- There are several efforts focused on in-situ infrastructure and algorithms

ALPINE

(ParaView/VisIt)

http://alpine.dsscale.org

http://www.paraview.org/in-situ



VisIt LibSim

https://visit.llnl.gov

http://www.sensei-insitu.org

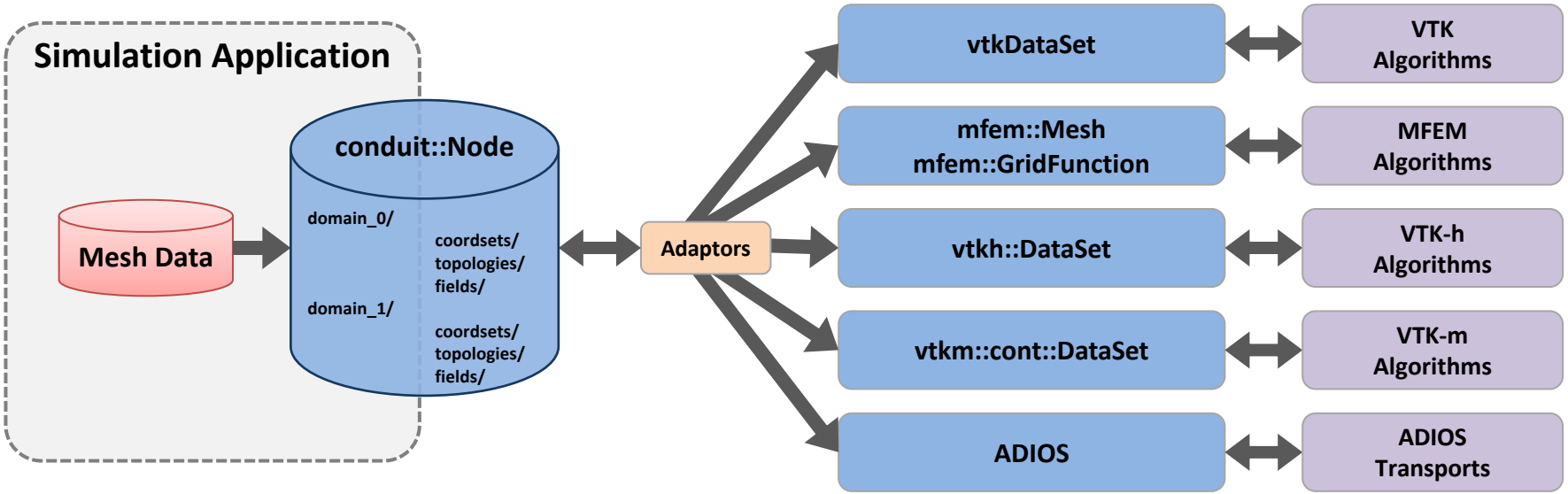
https://github.com/Alpine-DAV/ascent

We plan to release VisIt 3.0 this fall

Highlights:

- Rendering Improvements:
  - Upgrade from VTK6 to VTK8
    - Faster rendering times
    - Less memory usage
    - Improvements in transparency and volume rendering
  - OpenSWR Support
    - OpenSWR is a software rendering library highly optimized for Intel CPUs
- Enhancements to Conduit Blueprint data support
- VisIt 3.0 will include ALPINE Ascent in addition to LibSim for in-situ processing
- We also updating our development processes for VisIt 3.0
  - We are moving our issue tracking and source code repository to git and github
    - (We are refactoring our ~70gb SVN repo to be suitable for git)
  - We are porting our manuals to Sphinx and Read The Docs

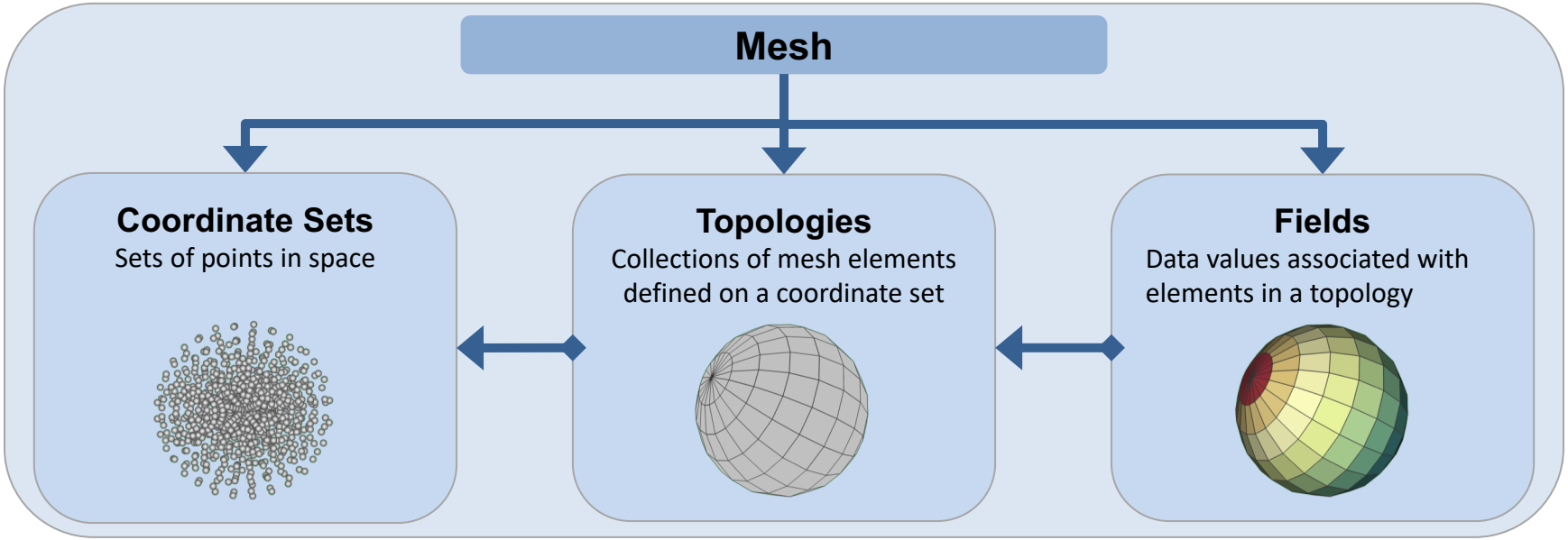
The Conduit “Mesh” Blueprint is a set of conventions for sharing mesh data in-memory and using files



The mesh blueprint provides conventions for describing and organizing simulation mesh data so that it can be used (often zero-copy) via multiple full featured data APIs

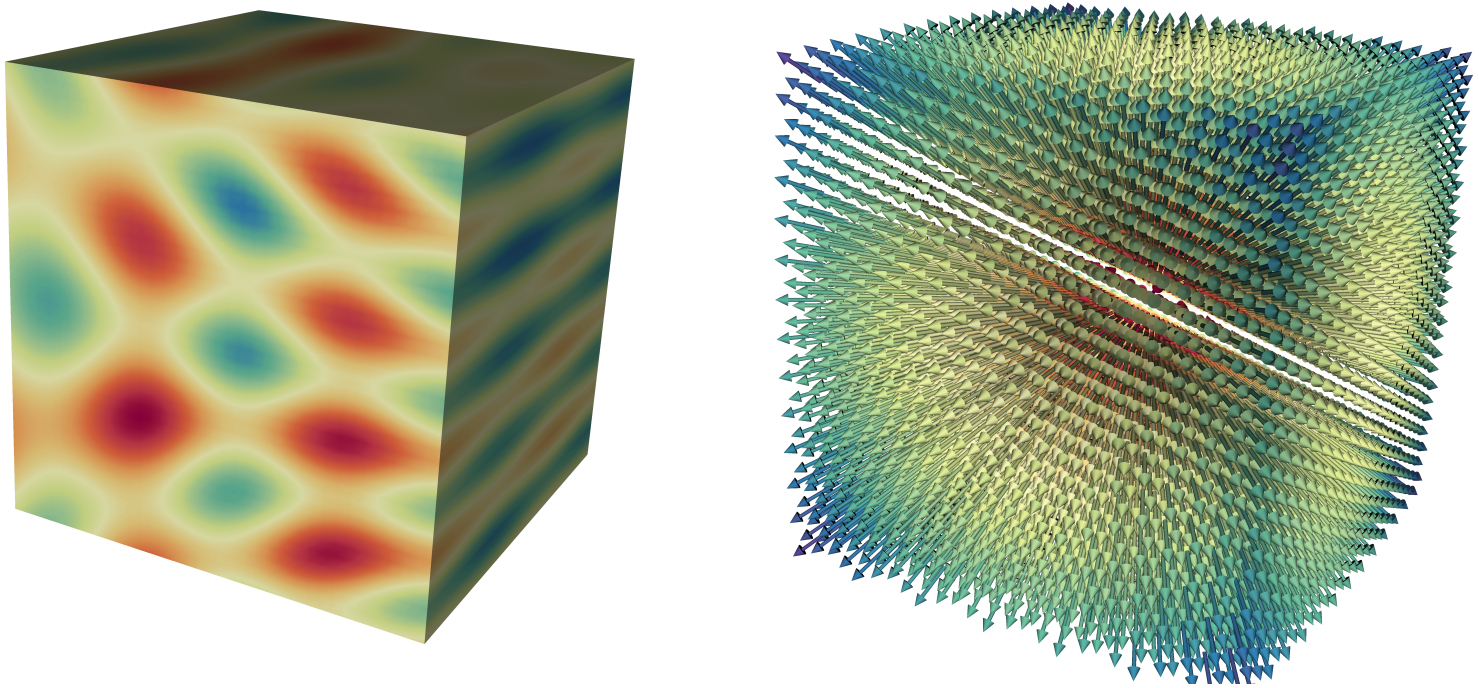
http://software.llnl.gov/conduit/blueprint.html

The Mesh Blueprint supports mesh constructs common in several full featured mesh data models



Ideas were shaped by surveying projects including: ADIOS, BoxLib, Chombo, Damaris, EAVL, Exodus, ITAPS, MFEM, SAF, SAMRAI, Silo, VisIt's AVT, VTK, VTK-m, Xdmf

VisIt 2.13 already reads blueprint-based files



Example blueprint mesh read via HDF5 and rendered in VisIt

VisIt is a robust, usable tool, that provides a broad set of visualization capabilities for HPC simulation data

- **Provides Features that span the "power of visualization"**
  - Data Exploration
  - Confirmation
  - Communication
- **Provides Features for different kinds of users**
  - Visualization Experts
  - Code Developers
  - Code Consumers

VisIt is actively developed and has vibrant developer and user communities

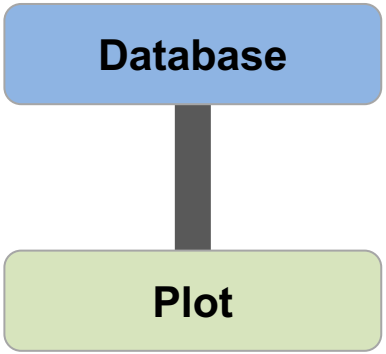
VisIt's Visualization Building Blocks

VisIt's interface is built around five core abstractions

- **Databases**: Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data

Examples of VisIt Pipelines

- **Databases**: Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data

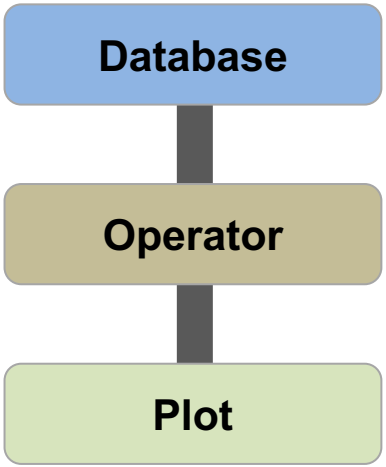


Open a database, which reads from a file
(Example: Open file1.hdf5)

Make a plot of a field in the database
(Example: Pseudocolor plot of *density*)

Examples of VisIt Pipelines

- **Databases**: Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data



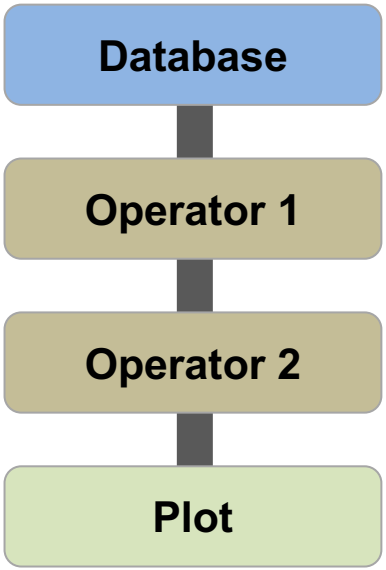
Open a database, which reads from a file
(Example: Open file1.hdf5)

Apply an operator to transform the data
(Example: Slice operator)

Make a plot of the result
(Example: Pseudocolor plot of *density*)

Examples of VisIt Pipelines

- **Databases**: Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data



Open a database, which reads from a file
(Example: Open file1.hdf5)

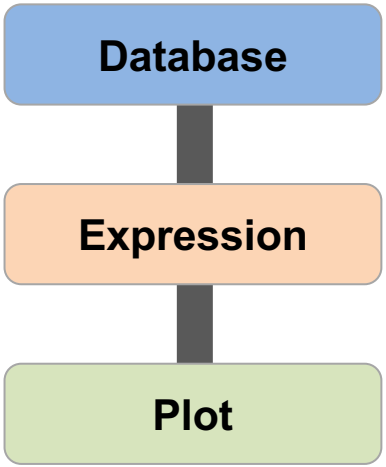
Apply an operator to transform the data
(Example: Slice operator)

Apply a second operator to transform the data
(Example: Elevate operator)

Make a plot of the result
(Example: Pseudocolor plot of *density*)

Examples of VisIt Pipelines

- **Databases**: Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data



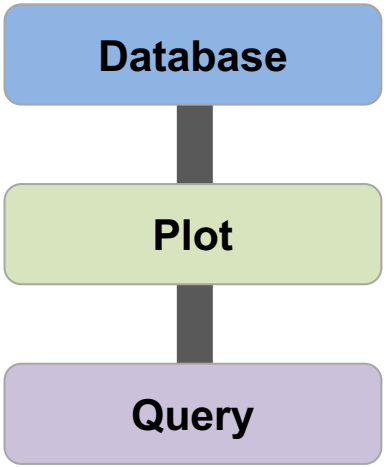
Open a database, which reads from a file
(Example: Open file1.hdf5)

Create derived quantity from existing fields
(Example: *speed = magnitude(velocity)*

Make a plot of the result
(Example: Pseudocolor plot of *speed*)

Examples of VisIt Pipelines

- **Databases**: Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data



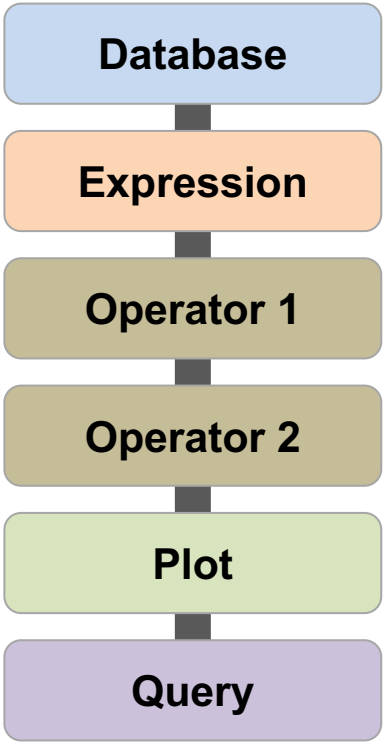
Open a database, which reads from a file
(Example: Open file1.hdf5)

Make a plot of a field in the database
(Example: Pseudocolor plot of *density*)

Extract quantitative information
(Example: integrate *density* to find *mass*)

Examples of VisIt Pipelines

- **Databases**: Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data



Open a database, which reads from a file
(Example: Open file1.hdf5)

Create derived quantity from existing fields
(Example: *speed = magnitude(velocity)*

Apply an operator to transform the data
(Example: Slice operator)

Apply a second operator to transform the data
(Example: Elevate operator)

Plot a field
(Example: Pseudocolor plot of *speed*)

Extract quantitative information
(Example: Maximum speed over cross-section)

Resources

Presenter Contact Info:

- Cyrus Harrison: cyrush@llnl.gov

User Resources:

- Main website: http://www.llnl.gov/visit
- Wiki: http://www.visitusers.org
- Email: visitusers@ornl.gov

Developer Resources:

- Email: visit-developers@ornl.gov
- SVN: http://visit.ilight.com/svn/visit/

Hands-on Session

Guided Tour of VisIt

- **Materials from:**
  - http://visitusers.org/index.php?title=VisIt-tutorial-basics
  - http://visitusers.org/index.php?title=VisIt-tutorial-data-analysis
  - http://visitusers.org/index.php?title=VisIt-tutorial-Python-scripting

Aneurysm Simulation Exploration

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

Water Flow Simulation Exploration

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

Additional Hands-on Materials

- **Volume Rendering**
  - http://visitusers.org/index.php?title-Visit-tutorial-Volume-Rendering
- **Advanced Movie Making**
  - http://visitusers.org/index.php?title=Visit-tutorial-Advanced-movie-making

Resources

Presenter Contact Info:

- Cyrus Harrison: cyrush@llnl.gov

User Resources:

- Main website: http://www.llnl.gov/visit
- Wiki: http://www.visitusers.org
- Email: visitusers@ornl.gov

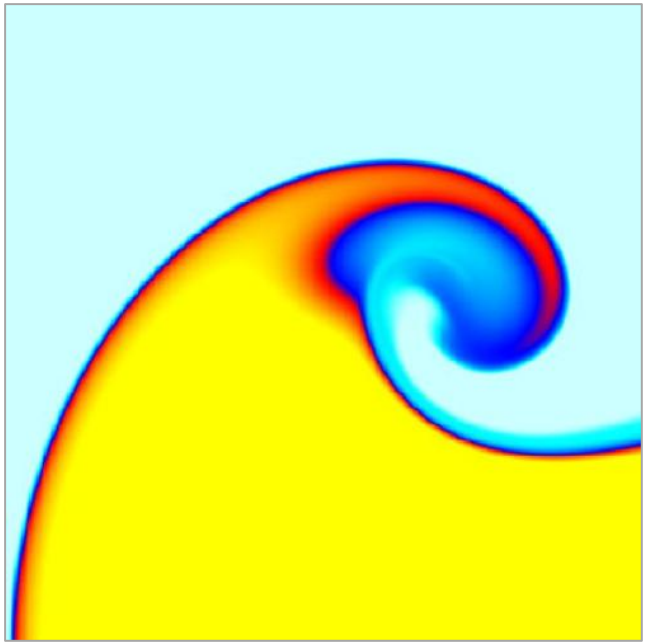
Developer Resources:

- Email: visit-developers@ornl.gov
- SVN: http://visit.ilight.com/svn/visit/

Visualization Techniques for Mesh-based Simulations

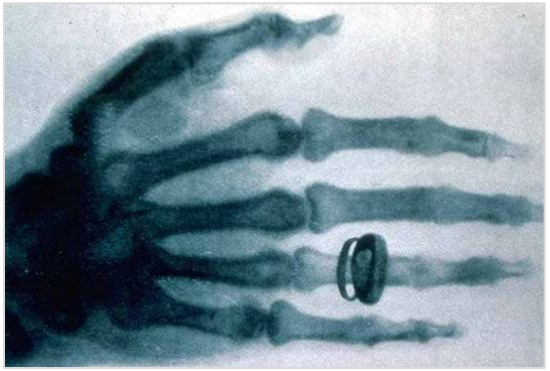
Pseudocolor rendering maps scalar fields to a range of colors

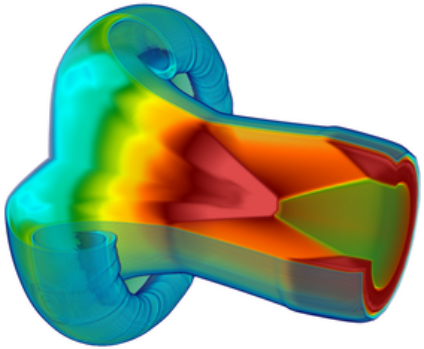
Pseudocolor rendering of Elevation

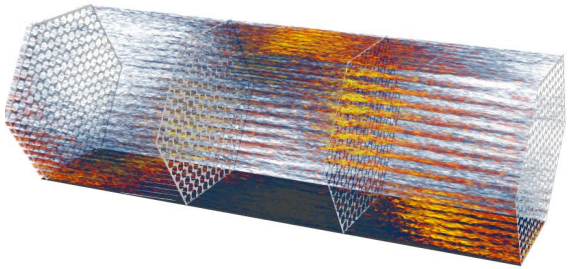


Pseudocolor rendering of Density

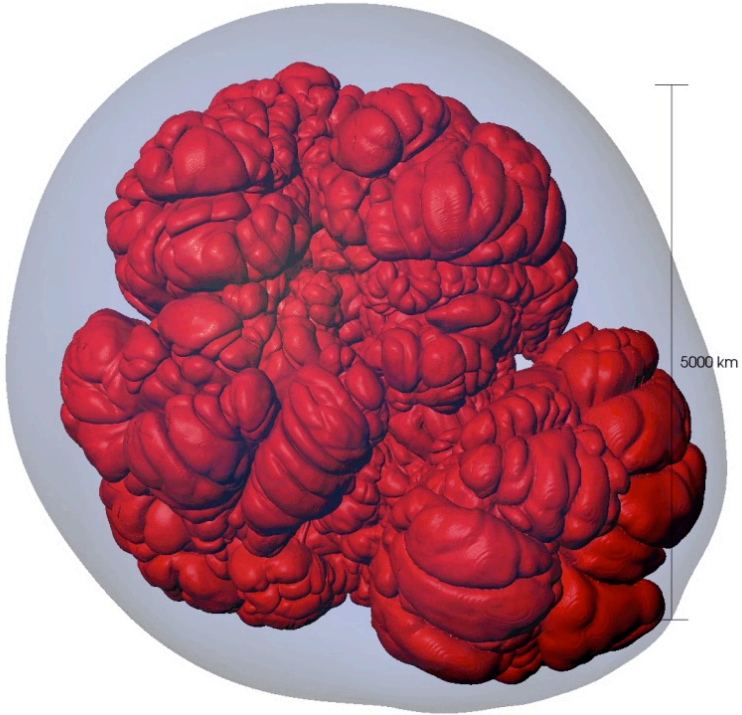
Volume Rendering cast rays though data and applies transfer functions to produce an image







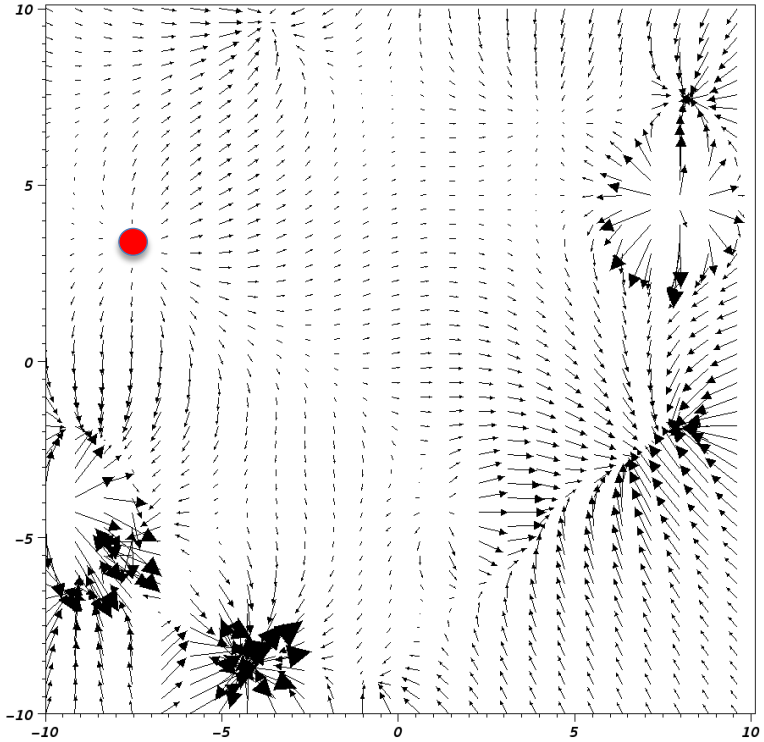
Isosurfacing (Contouring) extracts surfaces of that represent level sets of field values



Particle advection is the foundation of several flow visualization techniques

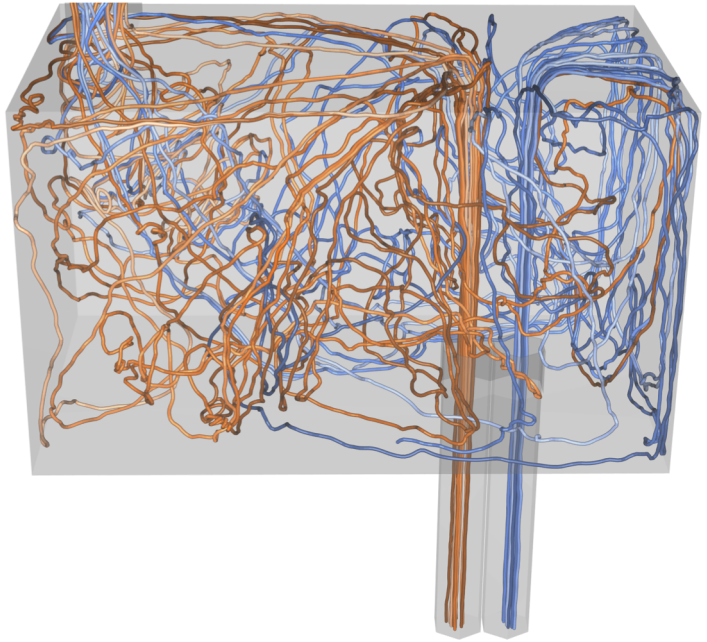
- S(t) = position of particle at time t
- $S(t_0) = p_0$
  - t_0: initial time
  - p_0: initial position
- $S'(t) = v(t, S(t))$
  - v(t, p): velocity at time t and position p
  - S'(t): derivative of the integral curve at time t

This is an ordinary differential equation.



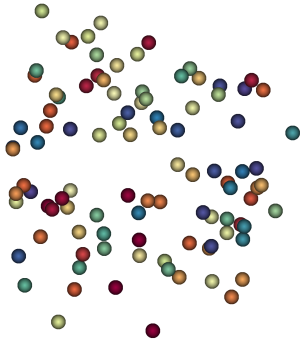
Streamline and Pathline computation are built on particle advection

- **Streamlines** – Instantaneous paths
- **Pathlines** – Time dependent paths

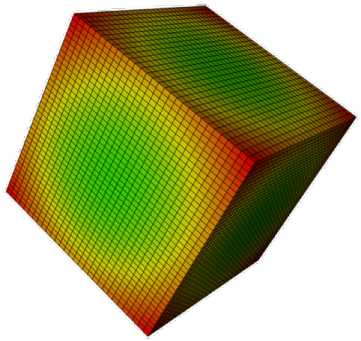


Meshes discretize continuous space

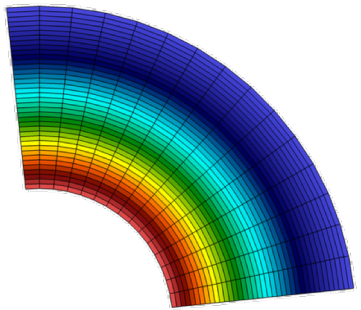
- **Simulations use a wide range of mesh types, defined in terms of:**
  - A set of coordinates (“nodes” / “points” / “vertices”)
  - A collection of “zones” / “cells” / “elements” on the coordinate set



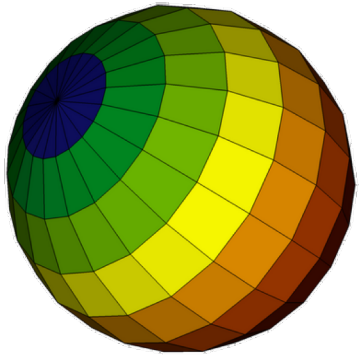
Points



Uniform



Curvilinear



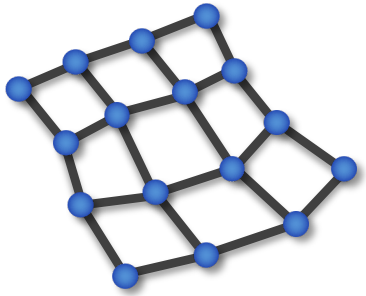
Unstructured

VisIt uses the “Zone” and “Node” nomenclature throughout its interface.

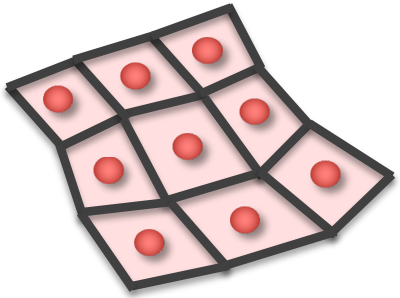
Mesh fields are variables associated with the mesh that hold simulation state

- Field values are associated with the zones or nodes of a mesh
  - Nodal: Linearly interpolated between the nodes of a zone
  - Zonal: Piecewise Constant across a zone

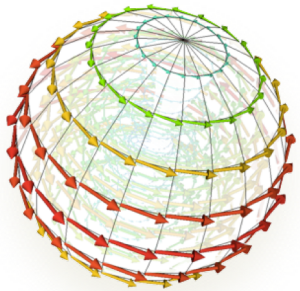
- Field values for each zone or node can be scalar, or multi-valued (vectors, tensors, etc.)



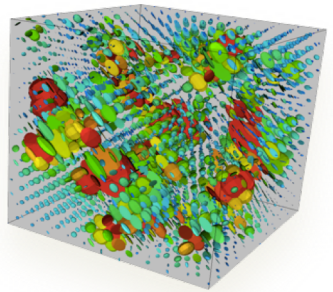
Nodal Association



Zonal Association



Vector Field

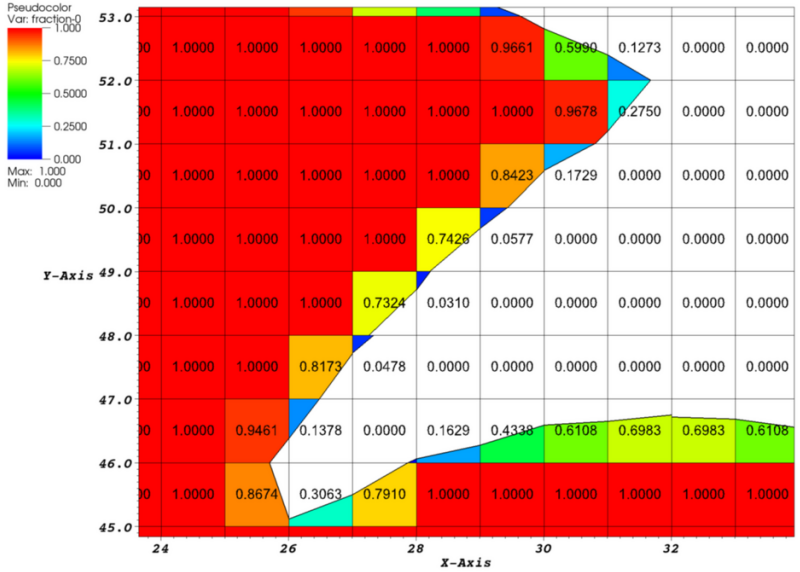


Tensor Field

Material volume fractions are used to capture sub-zonal interfaces

- Multi-material simulations use volume/area fractions to capture disjoint spatial regions at a sub-grid level.

- These fractions can be used as input to high-quality sub-grid material interface reconstruction algorithms.

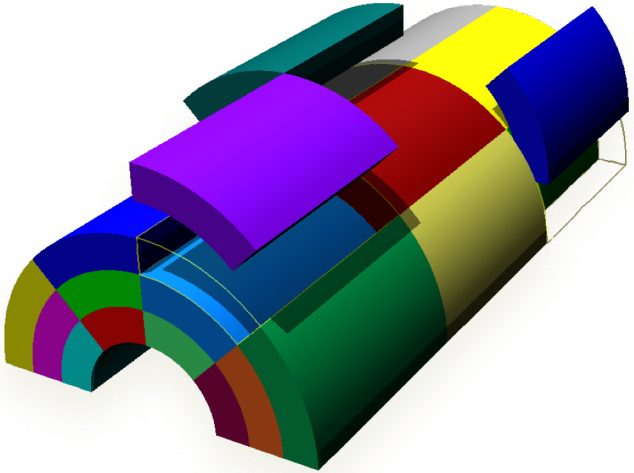


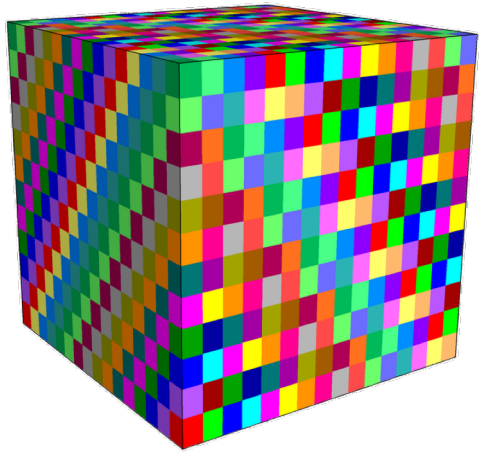
Species are used to capture sub-zonal weightings

- Species describe sub-grid variable composition
  - Example: Material “Air” is made of species “N2” ,“O2”, “Ar”, “CO2”, etc.
- Species are used for weighting, not to indicate sub-zonal interfaces.
  - They are typically used to capture fractions of “atomically mixed” values.

Domain decomposed meshes enable scalable parallel visualization and analysis algorithms

- Simulation meshes may be composed of smaller mesh “blocks” or “domains”.
- Domains are partitioned across MPI tasks for processing.





Adaptive Mesh Refinement (AMR) refines meshes into patches that capture details across length scales

- Mesh domains are associated with patches and levels
- Patches are nested to form a AMR hierarchy