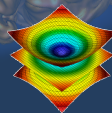


Visualization and Analysis of HPC Simulation Data using VisIt



LLNL Spring 2020 Tutorials

LLNL VisIt Team



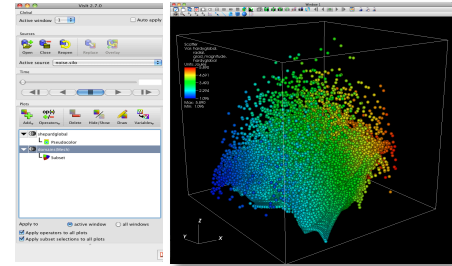
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

LLNL-PRES-784377

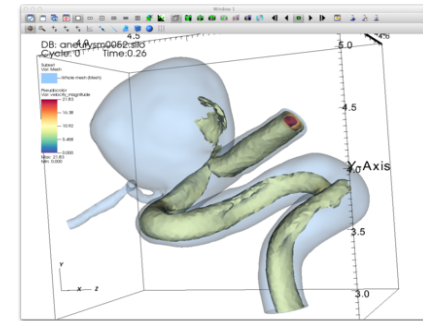


Outline

- Wed 4/15 (11:00am ~ 12:00pm)
 - VisIt Project Introduction
 - Tips for Remote Usage
 - Scripting VisIt with Python
 - Visualization of an Aneurysm (Blood Flow) Simulation
- Thurs 4/16 (10:00am ~ 11:00am)
 - Movie Making Tutorial



Intro to VisIt



Simulation Exploration



Tutorial Resources

- **Tutorial Materials:**

- http://visitusers.org/index.php?title=VisIt_Tutorial

- **How to get in touch:**

- **VisIt Hotline:** (925) 424-2847 [42-VIS onsite]

- **GitHub:** <https://github.com/visit-dav/visit>

- **Email List:** visitusers@ornl.gov



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/>)

Potential Flow Simulation Dataset

Simple tutorial simulation built using MFEM (<https://mfem.org/>)

Available thanks to:

- Aaron Fisher and Mark Miller, LLNL



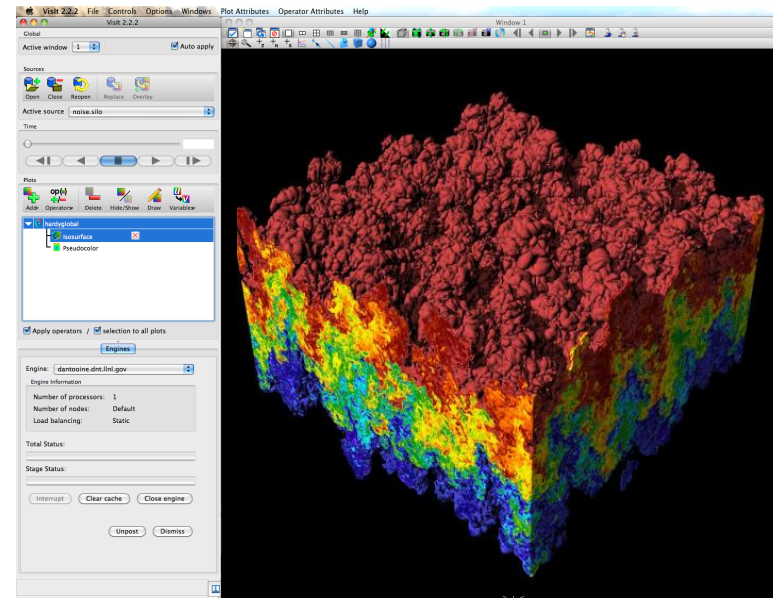
VisIt Project Introduction





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

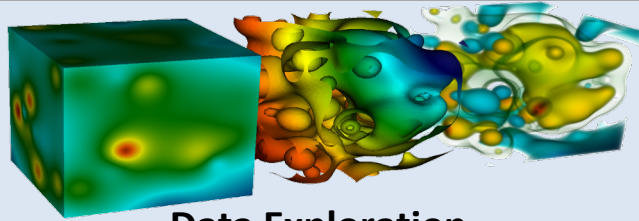
- Production end-user tool supporting scientific and engineering applications.
- Provides an infrastructure for parallel post-processing that scales from desktops to massive HPC clusters.
- Source released under a BSD style license.



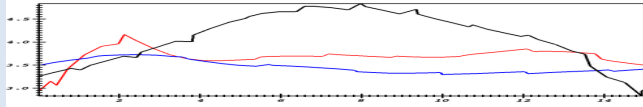
Pseudocolor plot of Density
(27 billion element dataset)



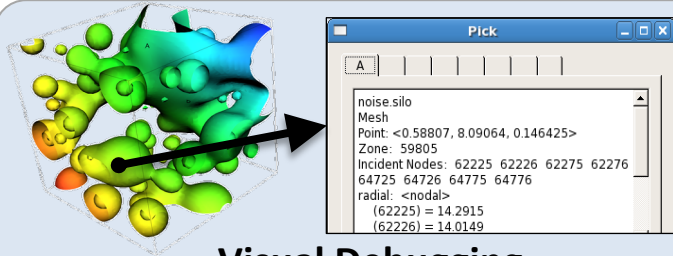
VisIt supports a wide range of use cases



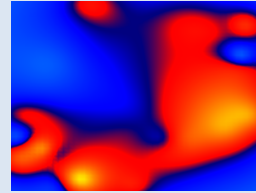
Data Exploration



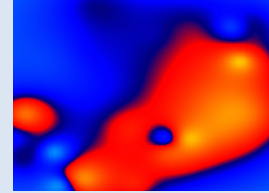
Quantitative Analysis



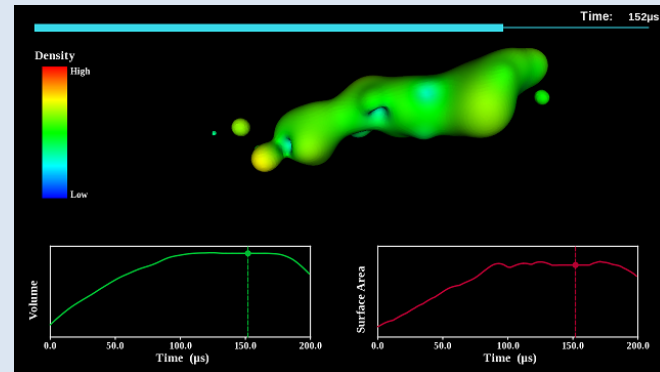
Visual Debugging



||-?



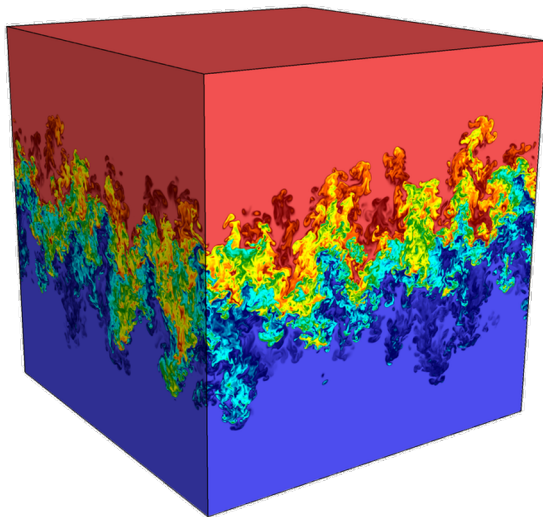
Comparative Analysis



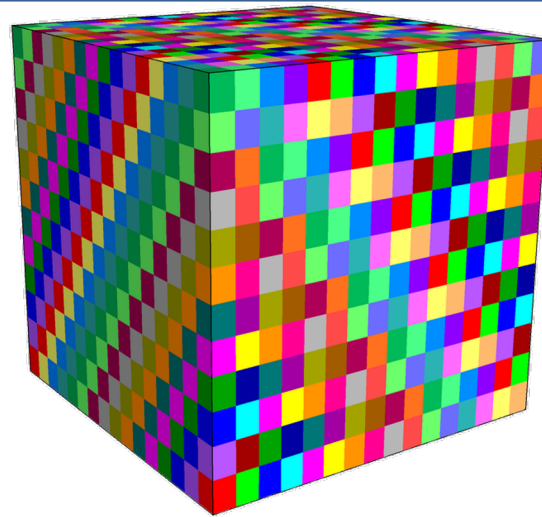
Presentation Graphics



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

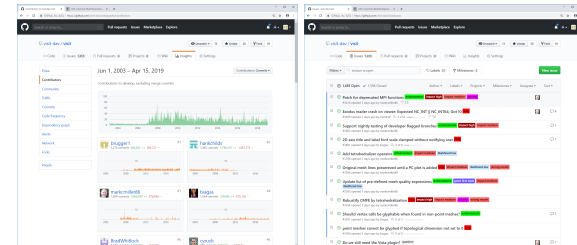


Visit 3.0 (April 2019) included major updates to our software development process

- We migrated our source repo from *svn* at NERSC to *git* on GitHub and our issue tracking from an ORNL Redmine instance to GitHub

— <https://github.com/visit-dav/visit>

GitHub



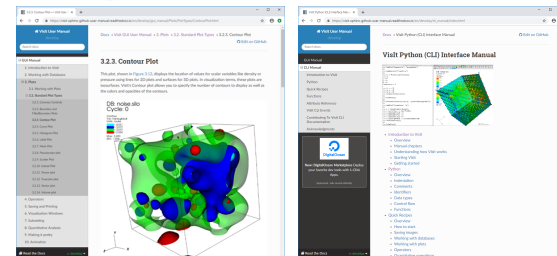
Visit source repo and issue tracking on GitHub

- We ported our legacy docs to Sphinx, now hosted on Read the Docs

— <https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/>



Read the Docs

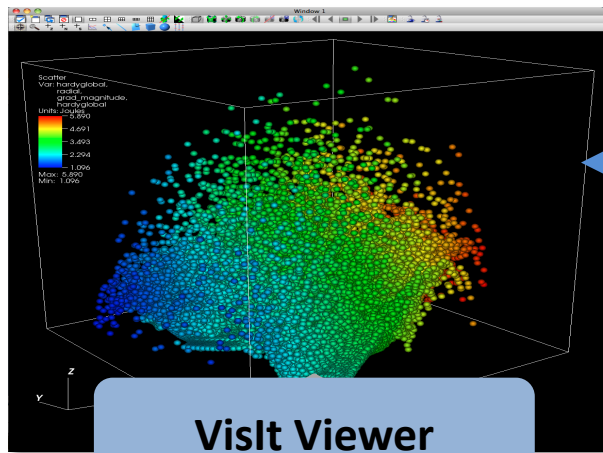


Visit manuals on Read the Docs



VisIt employs a parallelized client-server architecture

Client Computer



VisIt Viewer

VisIt GUI

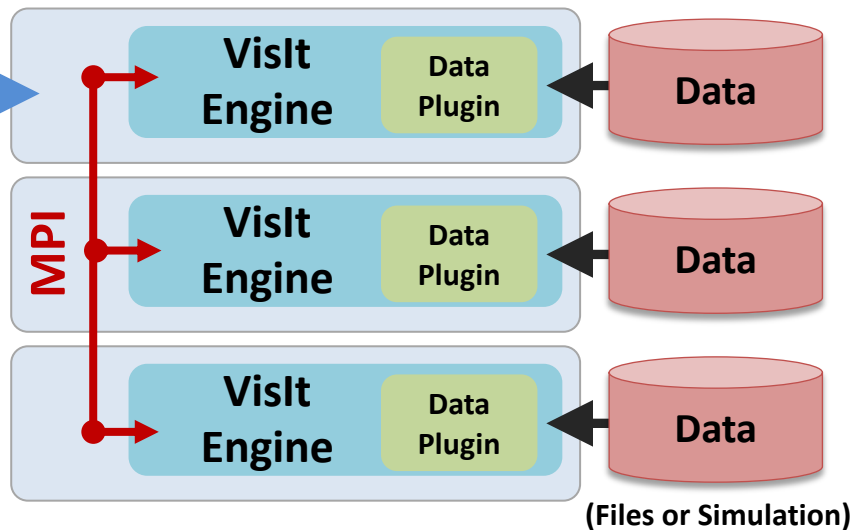
VisIt CLI

Python
Clients

Java
Clients

network
connection

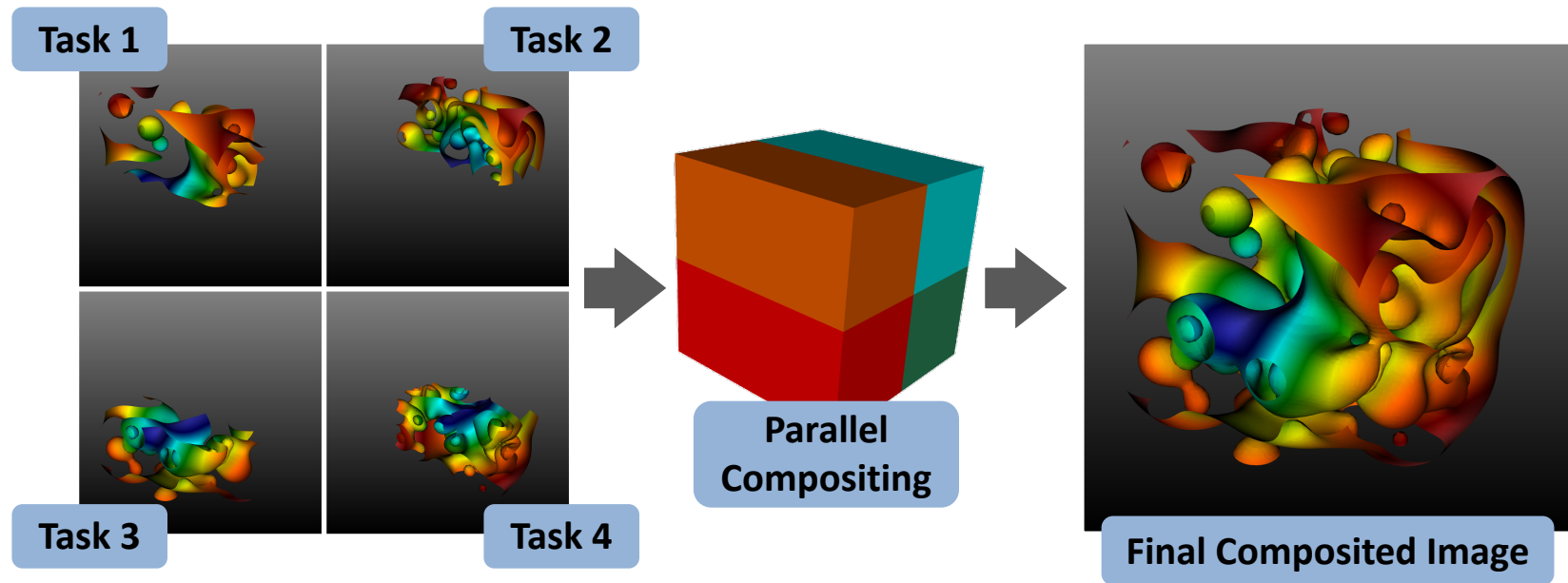
Parallel HPC Cluster



(Files or Simulation)



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



The VisIt team is investing to create a next generation in situ visualization infrastructure




**Flyweight in-situ visualization and
analysis for HPC simulations**

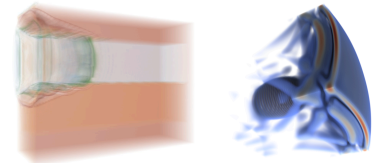
<http://ascent-dav.org>



Ascent is an easy to use flyweight in situ visualization and analysis library for HPC simulations

- **Easy to use in-memory visualization and analysis**
 - Use cases: ***Making Pictures***, ***Transforming Data***, and ***Capturing Data***
 - Young effort, yet already supports most common visualization operations
 - Provides a simple infrastructure to integrate custom analysis
 - Provides C++, C, Python, and Fortran APIs
- **Uses a flyweight design targeted at next-generation HPC platforms**
 - Efficient distributed-memory (MPI) and many-core (CUDA or OpenMP) execution
 - Demonstrated scaling: In situ filtering and ray tracing across **16,384 GPUs** on LLNL's Sierra Cluster
 - Has lower memory requirements than current tools
 - Requires less dependencies than current tools (ex: no OpenGL)
 - Builds with  Spack <https://spack.io/>

 **Ascent**



Visualizations created using Ascent



Extracts supported by Ascent

<http://ascent-dav.org>

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

Website and GitHub Repo



VisIt's Visualization Building Blocks



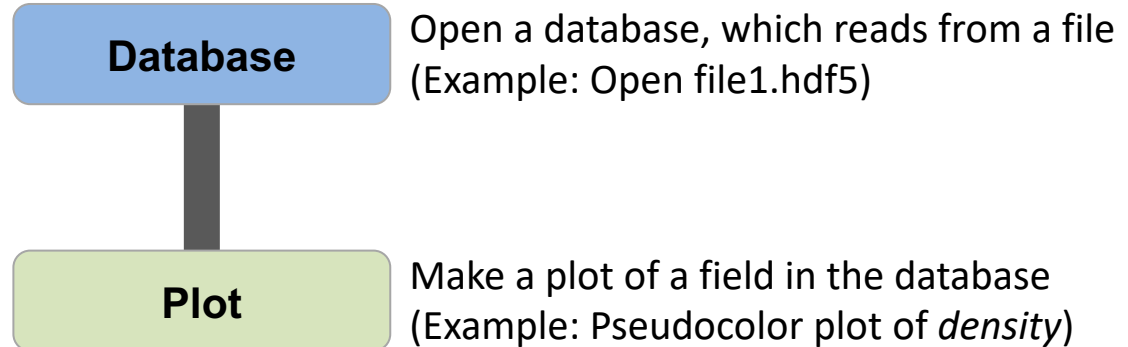
VisIt's interface is built around five core abstractions

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



Examples of VisIt Pipelines

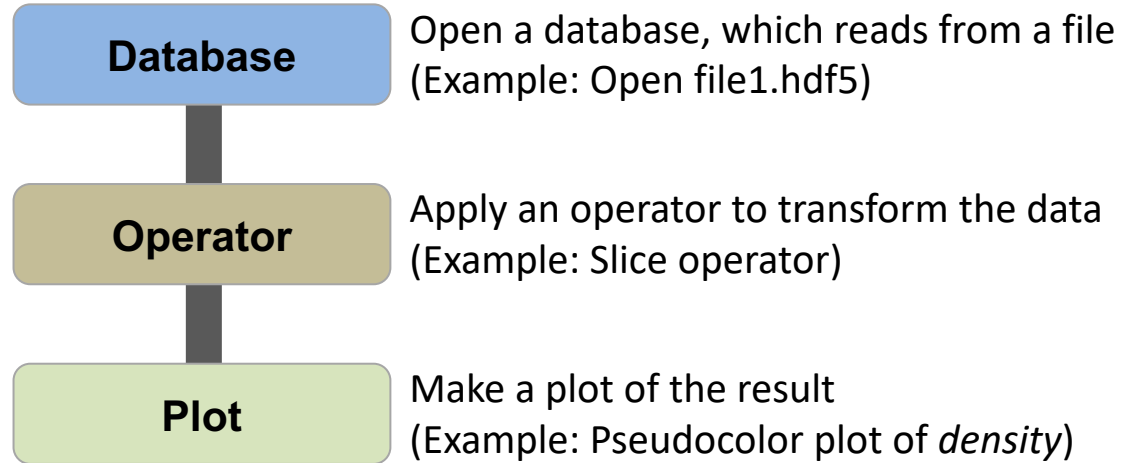
- **Databases:** Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data





Examples of VisIt Pipelines

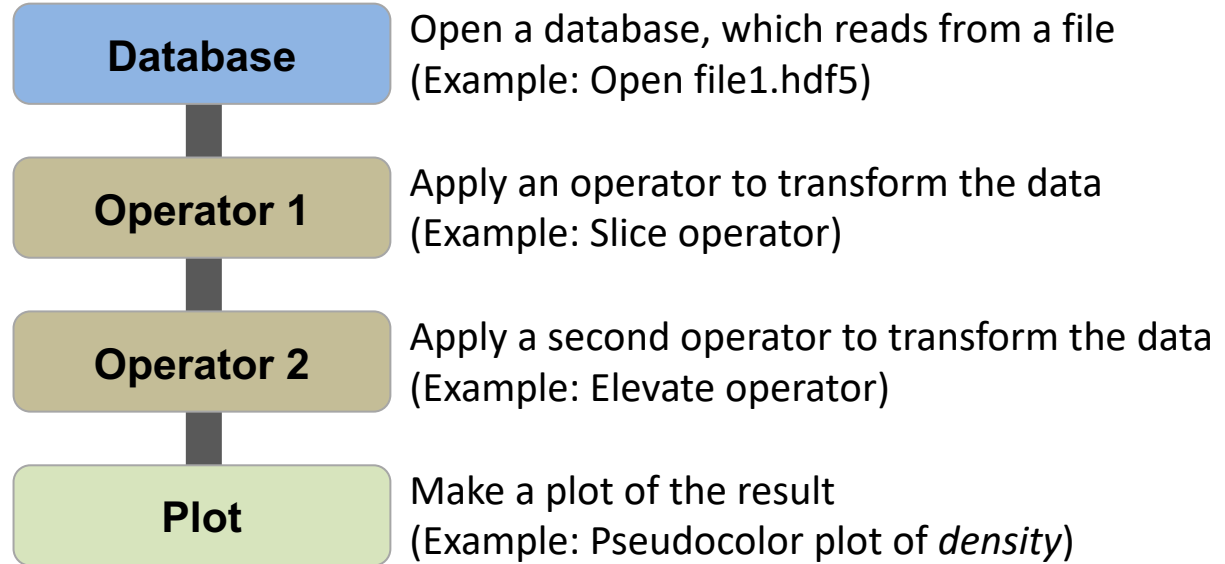
- **Databases:** Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data





Examples of VisIt Pipelines

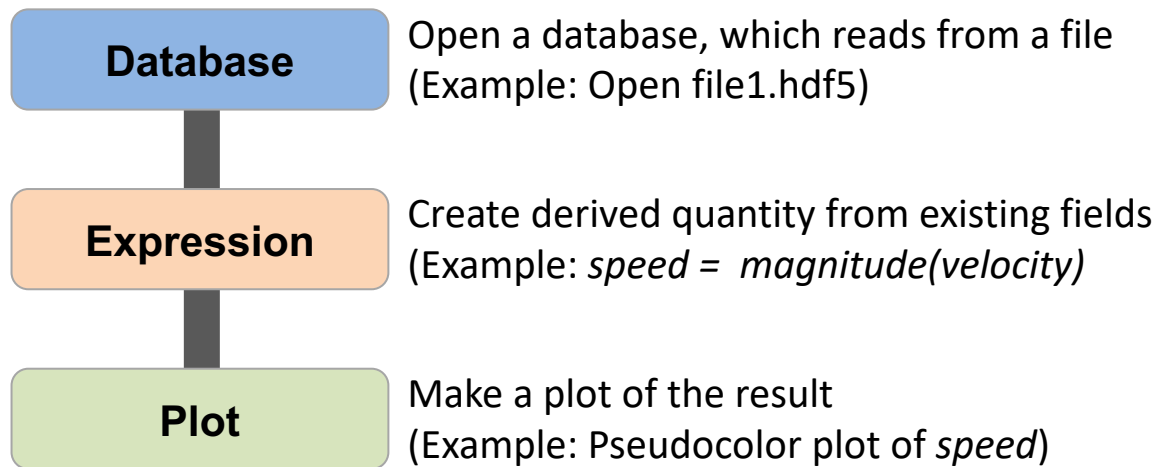
- **Databases:** Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data





Examples of VisIt Pipelines

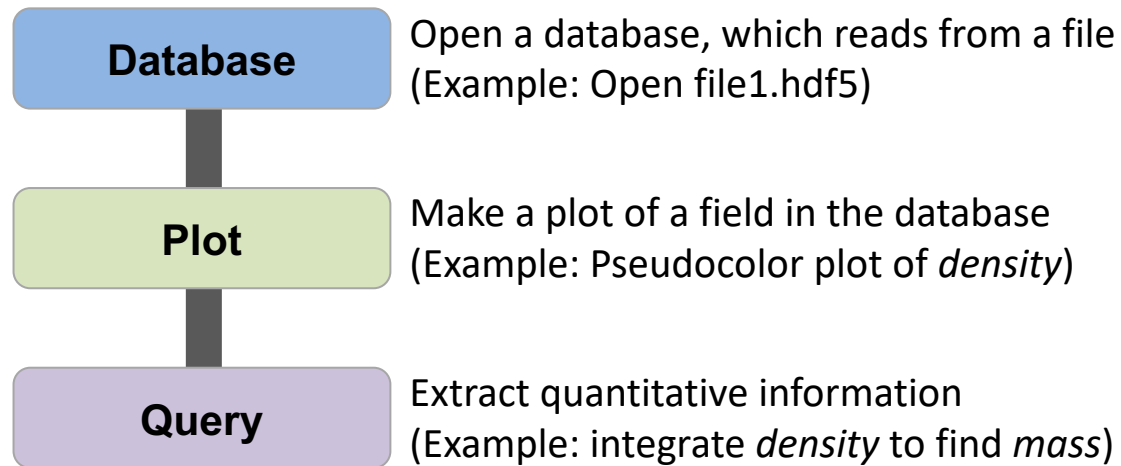
- **Databases:** Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data





Examples of VisIt Pipelines

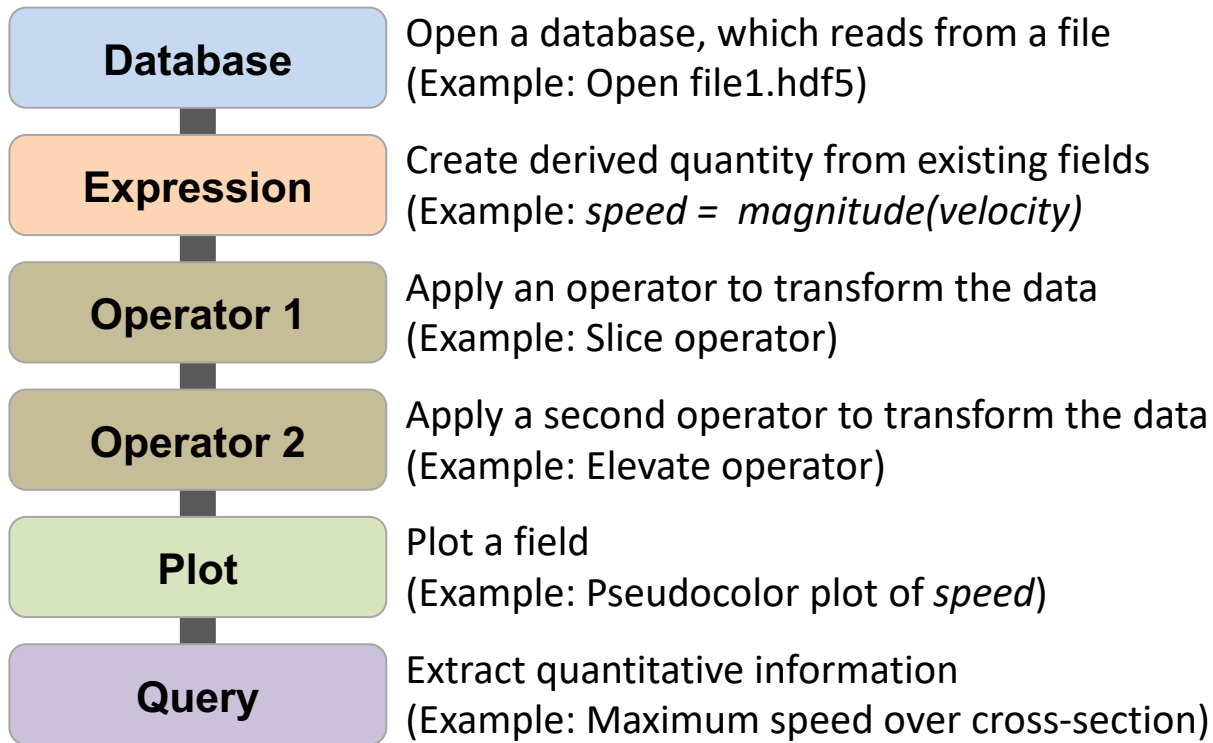
- **Databases:** Read data
- **Plots:** Render data
- **Operators:** Manipulate data
- **Expressions:** Generate derived quantities
- **Queries:** Summarize data





Examples of VisIt Pipelines

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





Resources

Presenter Contact Info:

- Eric Brugger: brugger1@llnl.gov
- Alister Maguire: maguire7@llnl.gov
- 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
- Github: <https://github.com/visit-dav/visit>



Remote Usage Tips

<https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/tutorials/Aneurysm.html>



Python Scripting Basics

<https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/tutorials/Scripting.html>





Aneurysm Simulation Exploration

<https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/tutorials/Aneurysm.html>



Movie Making Tutorial

<http://visitusers.org/index.php?title=LLNL-tutorial-movie-making>





Additional Hands-on Materials

- **Potential Flow Simulation Exploration**

- <https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/tutorials/PotentialFlow.html>

- **Water Flow Simulation Exploration**

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

- **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:

- Eric Brugger: brugger1@llnl.gov
- Alister Maguire: maguire7@llnl.gov
- 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
- Github: <https://github.com/visit-dav/visit>



Additional Slides





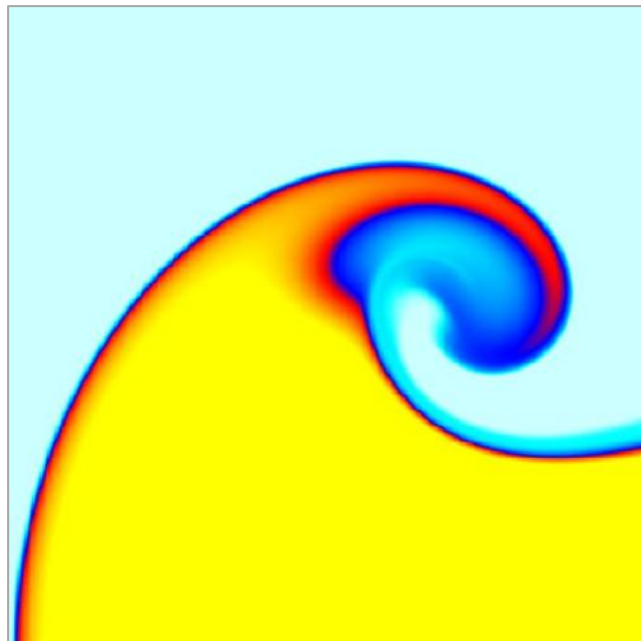
Visualization Techniques for Mesh-based Simulations



Pseudocolor rendering maps scalar fields to a range of colors



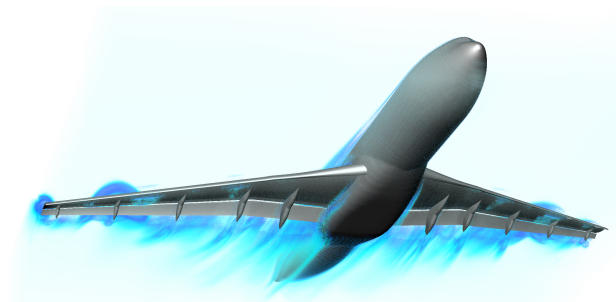
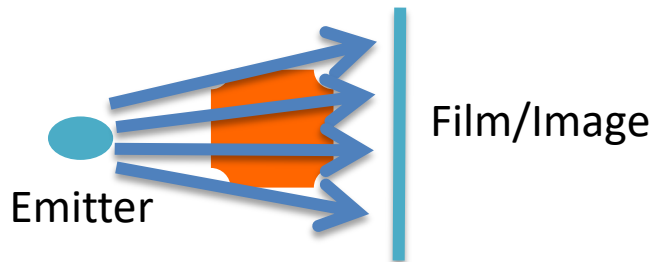
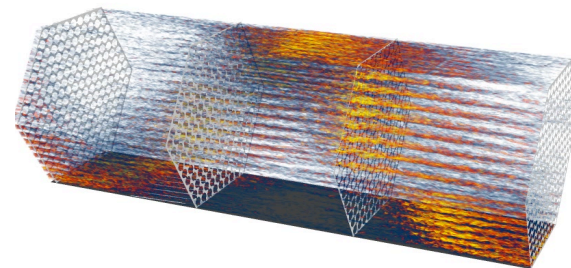
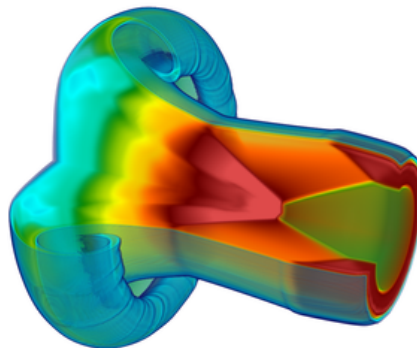
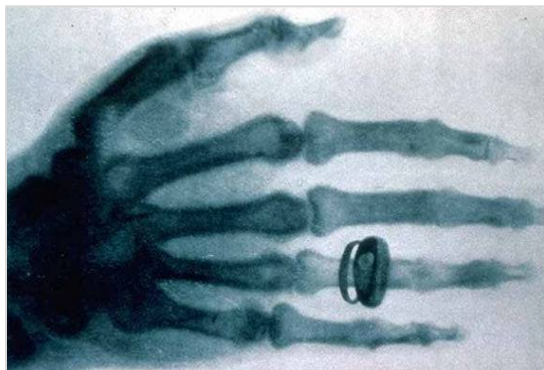
Pseudocolor rendering of Elevation



Pseudocolor rendering of Density

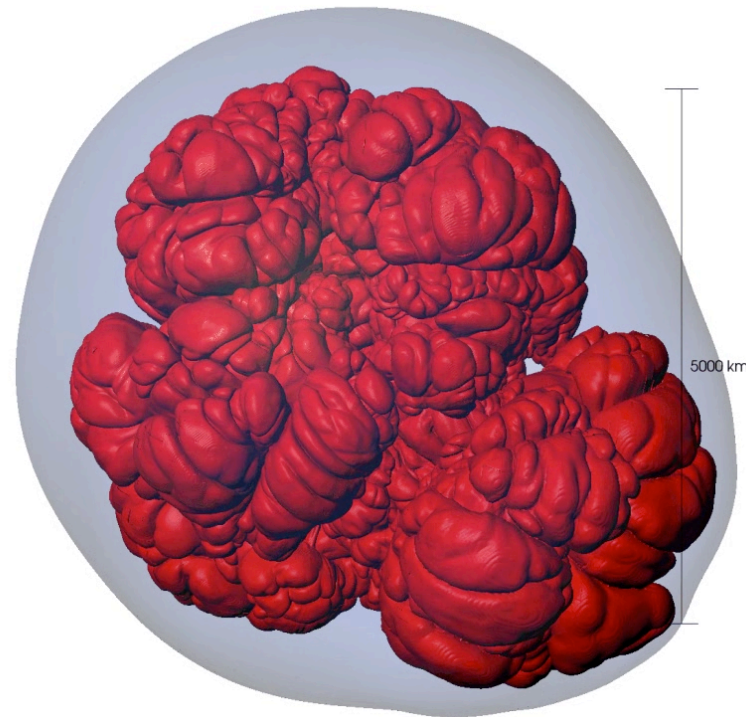


Volume Rendering cast rays through data and applies transfer functions to produce an image





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

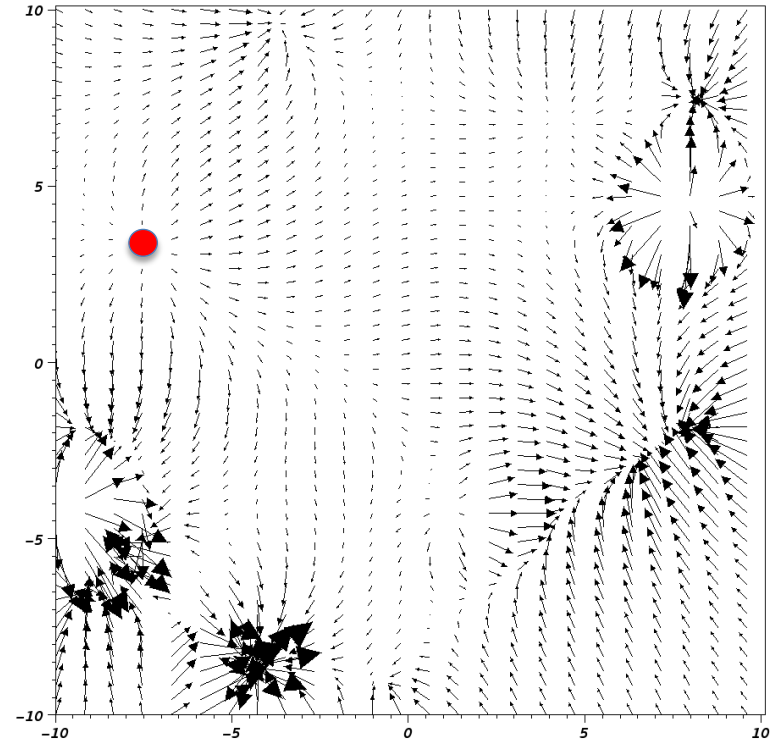




Particle advection is the foundation of several flow visualization techniques

- $S(t)$ = 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

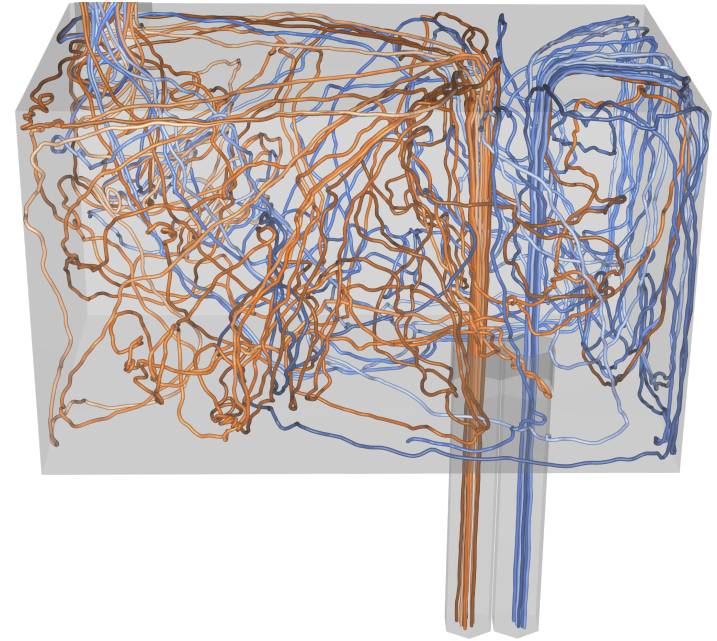
This is an ordinary differential equation.





Streamline and Pathline computation are built on particle advection

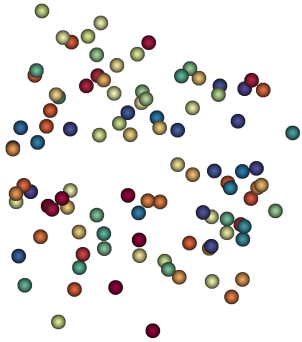
- **Streamlines** – Instantaneous paths
- **Pathlines** – Time dependent paths



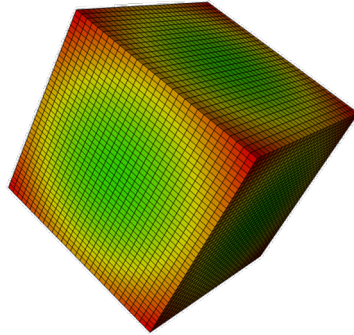


Meshes discretize continuous space

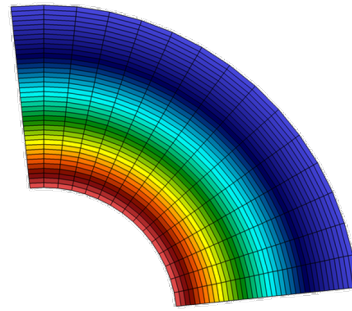
- **Simulations use a wide range of mesh types, defined in terms of:**
 - A set of coordinates (“nodes” / “points” / “vertices”)
 - A collection of “zones” / “cells” / “elements” on the coordinate set



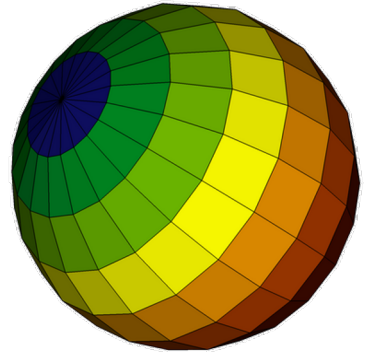
Points



Uniform



Curvilinear



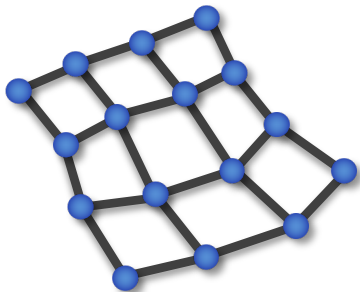
Unstructured

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

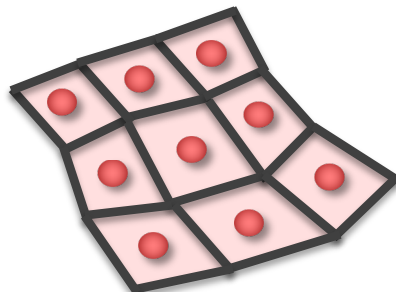


Mesh fields are variables associated with the mesh that hold simulation state

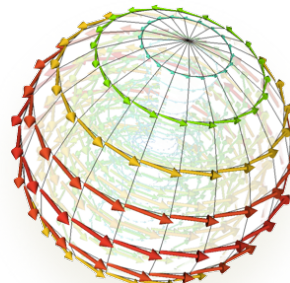
- Field values are associated with the zones or nodes of a mesh
 - Nodal: Linearly interpolated between the nodes of a zone
 - Zonal: Piecewise Constant across a zone
- Field values for each zone or node can be scalar, or multi-valued (vectors, tensors, etc.)



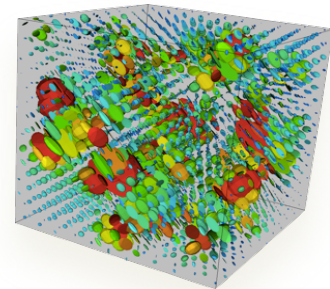
Nodal Association



Zonal Association



**Vector
Field**

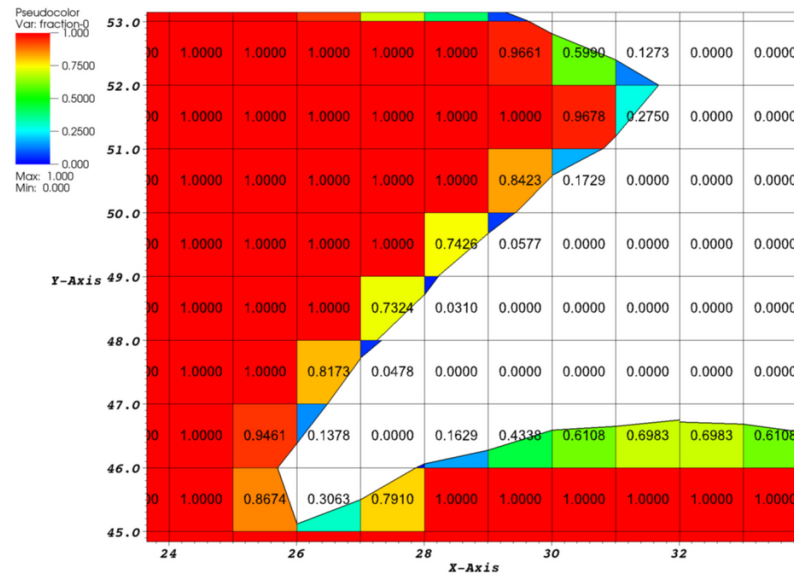


Tensor Field



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.





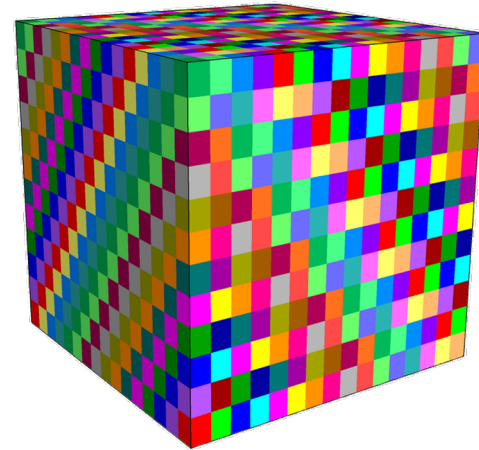
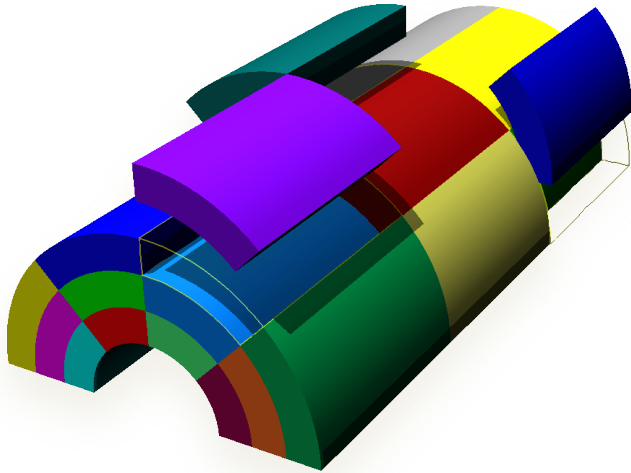
Species are used to capture sub-zonal weightings

- 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

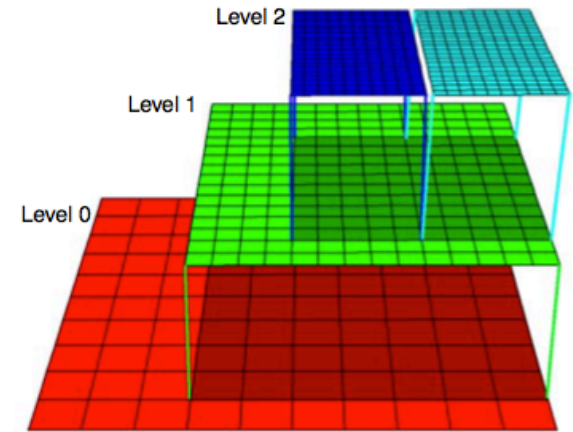
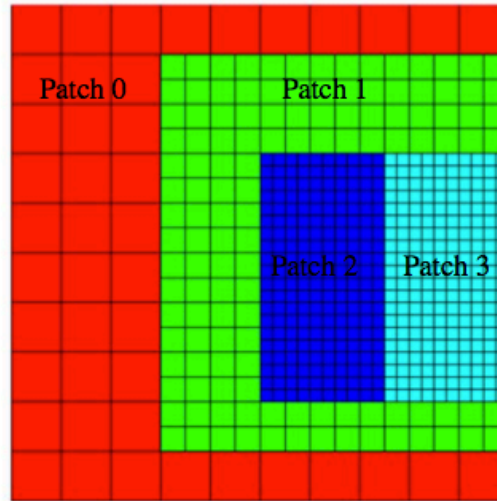
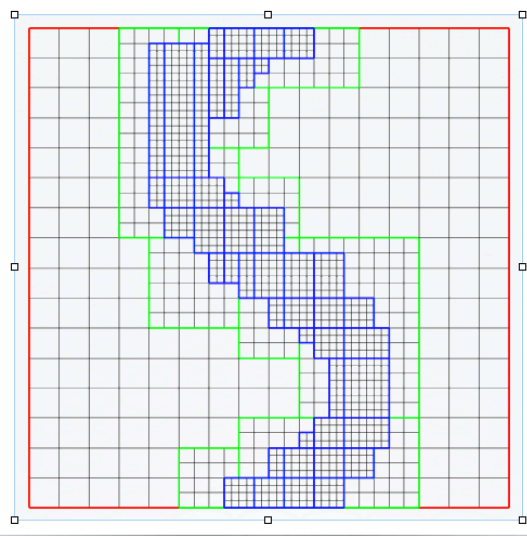
- Simulation meshes may be composed of smaller mesh “blocks” or “domains”.
- Domains are partitioned across MPI tasks for processing.





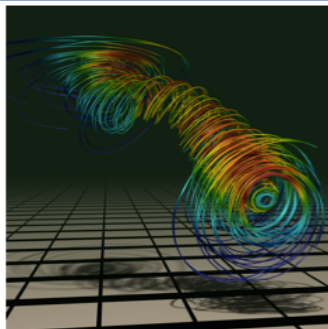
Adaptive Mesh Refinement (AMR) refines meshes into patches that capture details across length scales

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

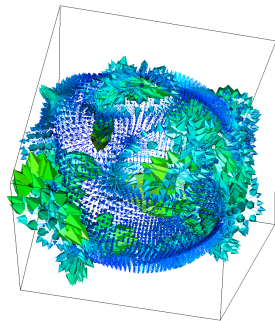




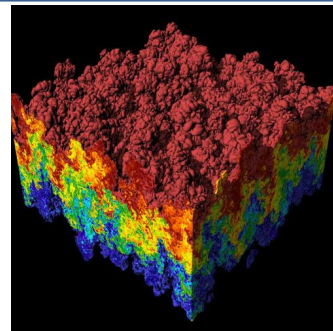
VisIt provides a wide range of plotting features for simulation data across many scientific domains



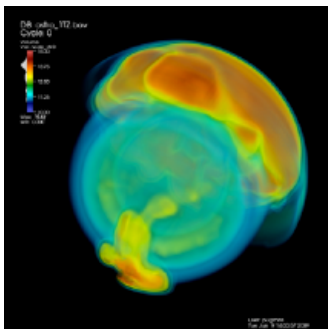
Streamlines / Pathlines



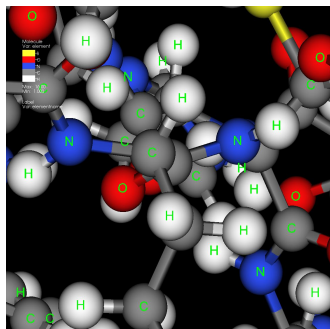
Vector / Tensor Glyphs



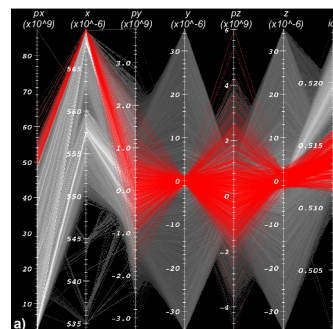
Pseudocolor Rendering



Volume Rendering



Molecular Visualization

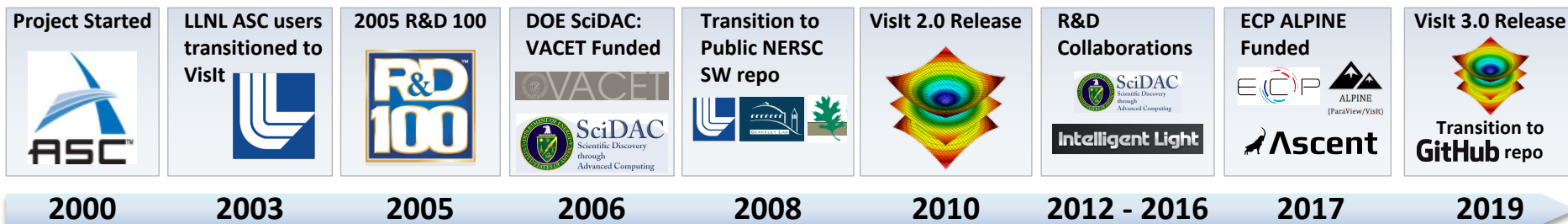


Parallel Coordinates



VisIt 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 100 person years of effort, 1.5+ million lines of code.





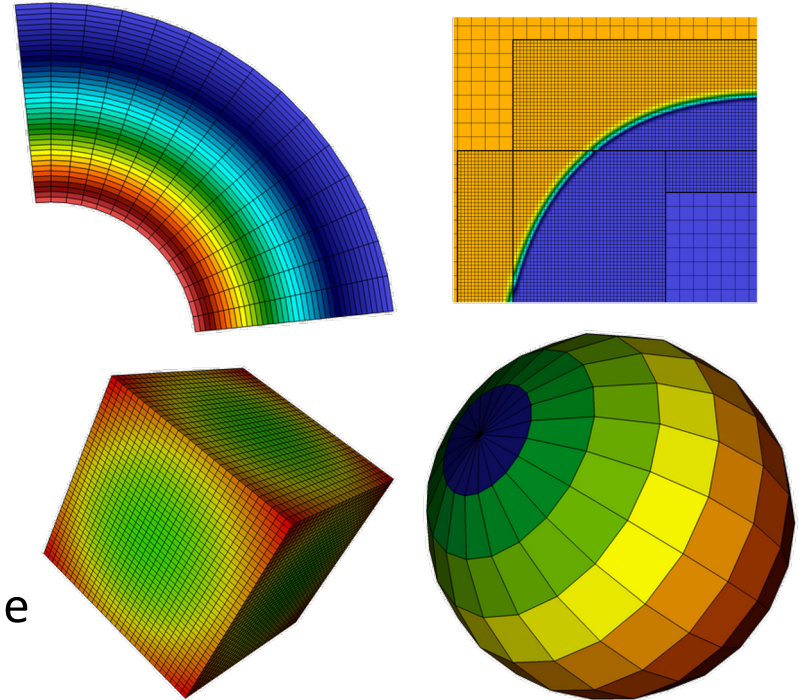
VisIt provides a flexible data model, suitable for many application domains

■ Mesh Types

- Point, Curve, 2D/3D Rectilinear, Curvilinear, Unstructured
- Domain Decomposed, AMR
- Time Varying
- Primarily linear element support, limited quadratic element support

■ Field Types

- Scalar, Vector, Tensor, Material Volume Fractions, Species





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>)
 - VisIt on ALCF’s Cooley:
 - <https://www.alcf.anl.gov/user-guides/visit-cooley>
 - Several other 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 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's infrastructure provides a flexible platform for custom workflows

■ C++ Plugin Architecture

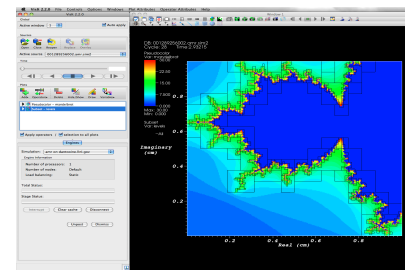
- Custom File formats, Plots, Operators
- Interface for custom GUIs in Python, C++ and Java

■ Python Interfaces

- Python scripting and batch processing
- Data analysis via Python Expressions and Queries

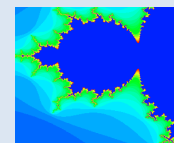
■ In-Situ Coupling

- VisIt's *Libsim* library allows simulation codes to link in VisIt's engine for in situ visualization



VisIt

Simulation



Libsim
Adaptor



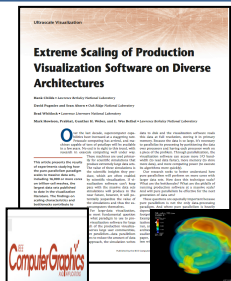
VisIt is used as a platform to deploy visualization research

■ DOE Research Collaborations



■ Research Focus Areas

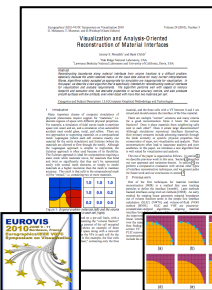
- Light weight In Situ Processing
- Node Level Parallelism
- Distributed Memory Parallel Algorithms



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



<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



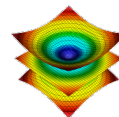
ALPINE

(ParaView/VisIt)

<http://alpine.dsscale.org>



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



VisIt LibSim

<https://visit.llnl.gov>



<http://www.sensei-insitu.org>



Ascent

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



The VisIt team is investing in Conduit and Ascent to create next generation in situ infrastructure



**Intuitive APIs for in-memory data
description and exchange**

<http://software.llnl.gov/conduit>



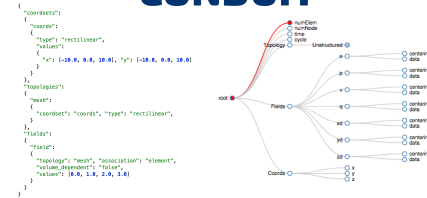
**Flyweight in-situ visualization and
analysis for HPC simulations**

<http://ascent-dav.org>

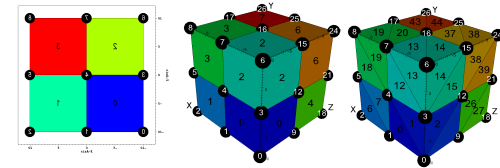


Conduit provides intuitive APIs for in-memory data description and exchange

- **Provides an intuitive API for in-memory data description**
 - Enables *human-friendly* hierarchical data organization
 - Can describe in-memory arrays without copying
 - Provides C++, C, Python, and Fortran APIs
- **Provides common conventions for exchanging complex data**
 - Shared conventions for passing complex data (eg: *Simulation Meshes*) enable modular interfaces across software libraries and simulation applications
- **Provides easy to use I/O interfaces for moving and storing data**
 - Enables use cases like binary checkpoint restart
 - Supports moving complex data with MPI (serialization)



Hierarchical in-memory data description



Conventions for sharing in-memory mesh data

<http://software.llnl.gov/conduit>
<http://github.com/llnl/conduit>

Website and GitHub Repo



Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.