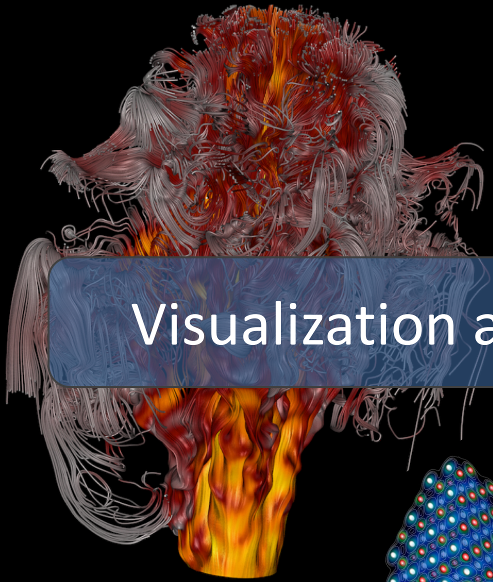
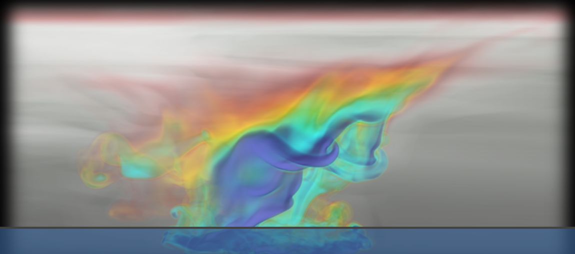
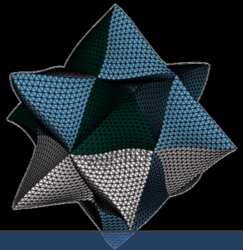
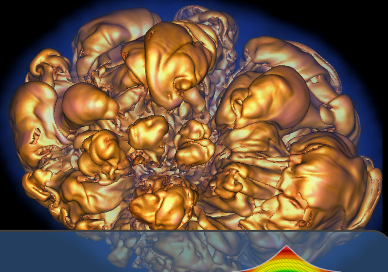
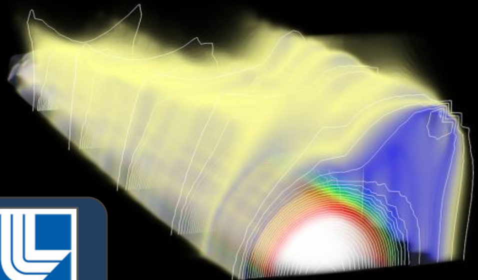
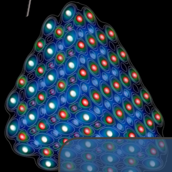





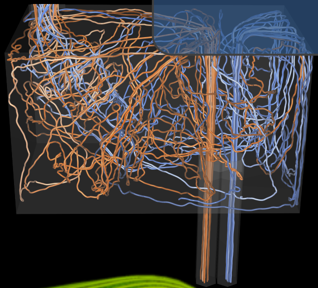
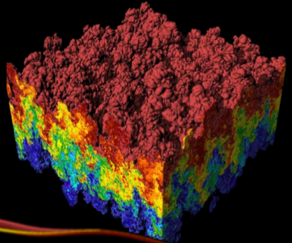
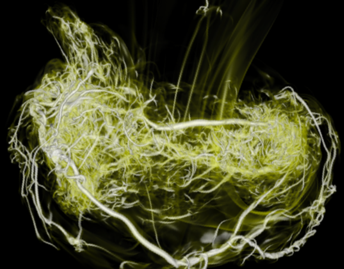
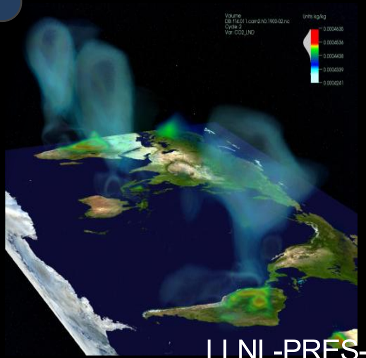
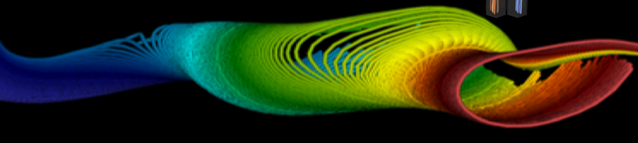
Visualization and Analysis of HPC Simulation Data using **VisIt**

ATPESC 2019

Monday August 5th, 2019

Cyrus Harrison (cyrush@llnl.gov)

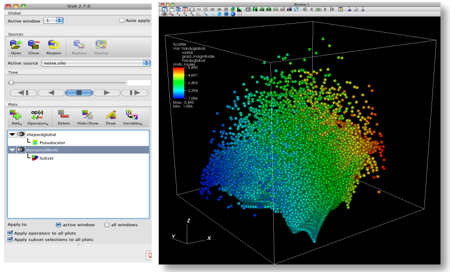







This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

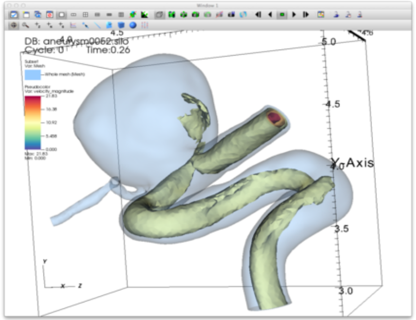
LLNL-PRES-784377

ATPESC 2019 Outline

- VisIt Project Introduction (~20 min)
- Hands-on: (~1.25 hours)
  - — Guided tour of VisIt
  - — Visualization of Potential flow
  - — 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

Potential Flow Simulation Dataset

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

Available thanks to:

– Aaron Fisher and Mark Miller, LLNL

Aneurysm Simulation Dataset

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

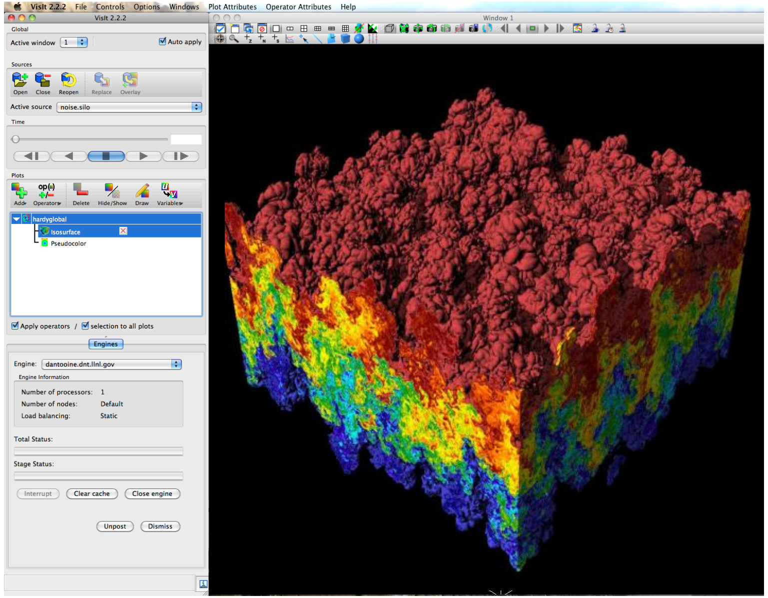
Available thanks to:

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

VisIt Project Introduction

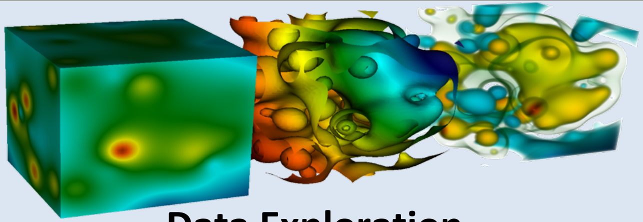
VisIt is an open source, turnkey application for data analysis and visualization of mesh-based data

- 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.

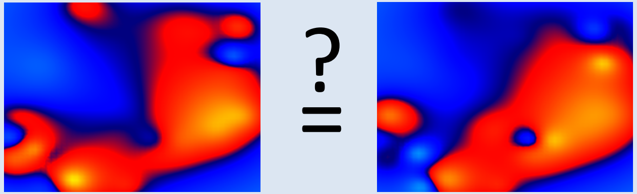


Pseudocolor plot of Density
(27 billion element dataset)

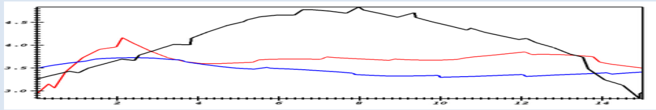
VisIt supports a wide range of use cases



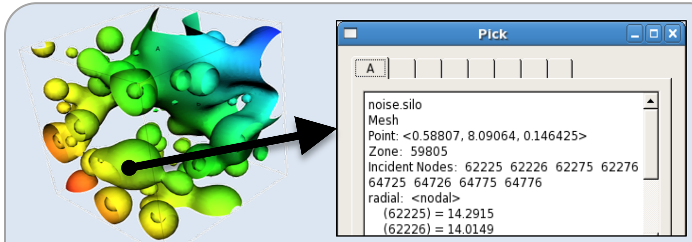
Data Exploration



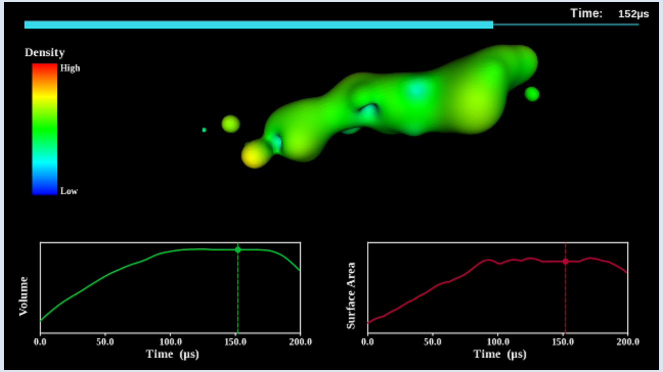
Comparative Analysis



Quantitative Analysis

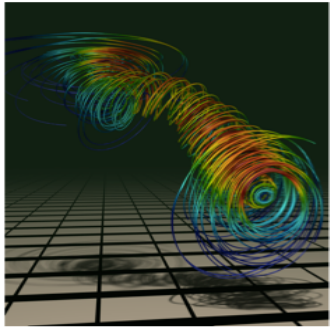


Visual Debugging

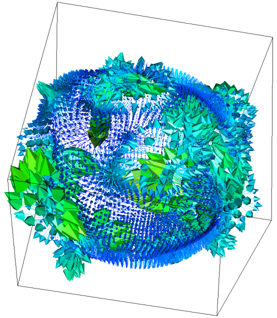


Presentation Graphics

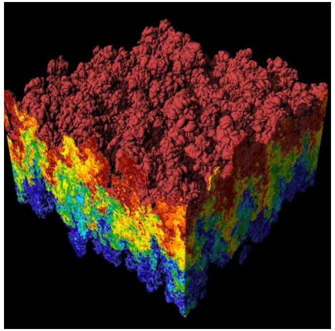
VisIt provides a wide range of plotting features for simulation data across many scientific domains



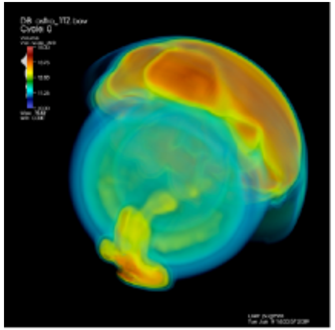
Streamlines / Pathlines



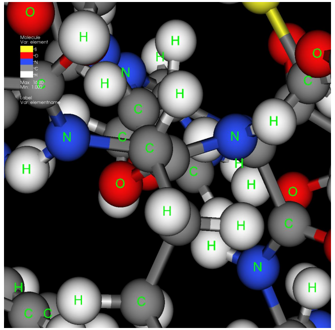
Vector / Tensor Glyphs



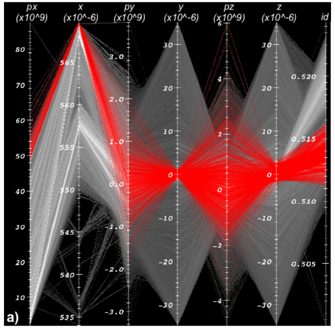
Pseudocolor Rendering



Volume Rendering



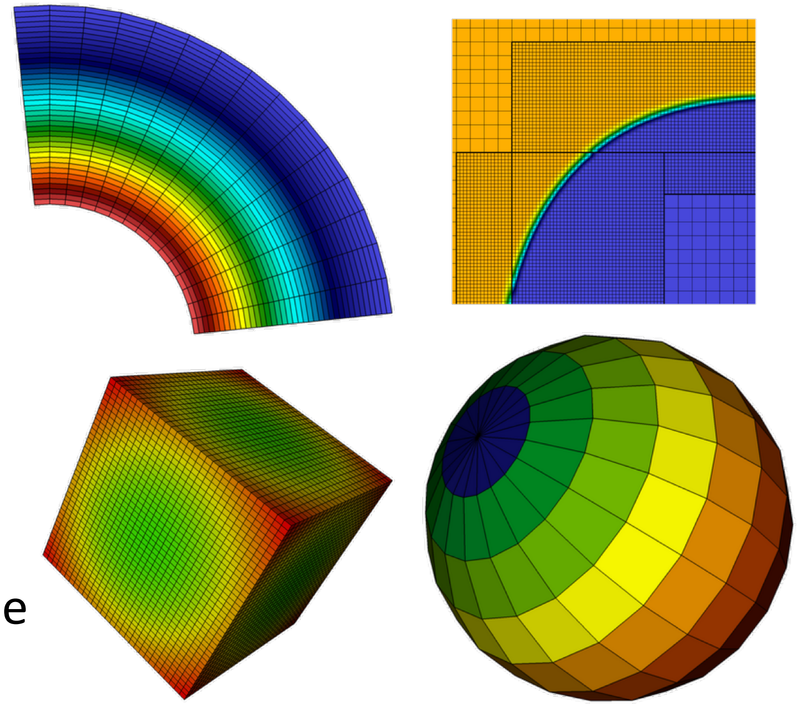
Molecular Visualization



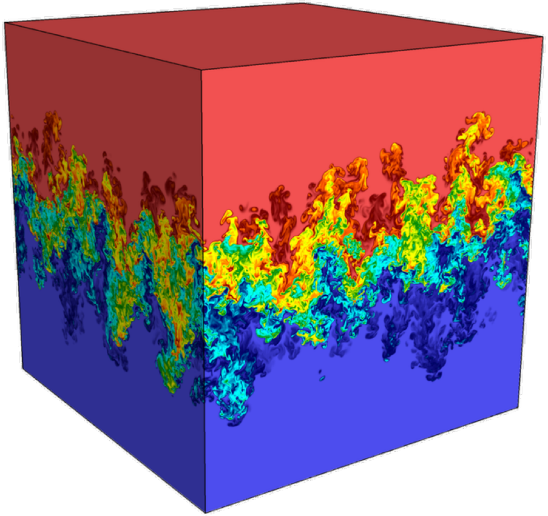
Parallel Coordinates

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

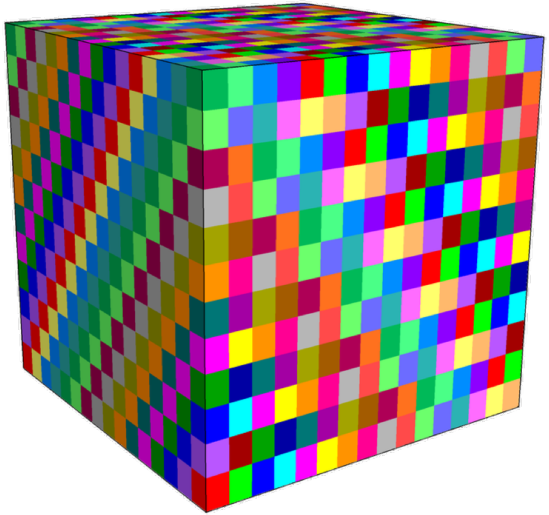
- **Mesh Types**
  - Point, Curve, 2D/3D Rectilinear, Curvilinear, Unstructured
  - Domain Decomposed, AMR
  - Time Varying
  - Primarily linear element support, limited quadratic element support
- **Field Types**
  - Scalar, Vector, Tensor, Material Volume Fractions, Species



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

Project Started	LLNL ASC users transitioned to VisIt	2005 R&D 100	DOE SciDAC: VACET Funded	Transition to Public NERSC SW repo	VisIt 2.0 Release	R&D Collaborations	ECP ALPINE Funded	VisIt 3.0 Release Transition to GitHub repo
2000	2003	2005	2006	2008	2010	2012 - 2016	2017	2019

We released VisIt 3.0 in April 2019

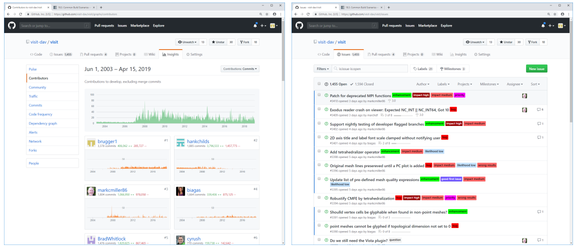
- Includes major feature updates:
  - Ported to VTK 8, major rendering improvements
  - Moved fully to Qt5
  - Added OSPRay Ray Tracing support
  - Added Cinema Database support
  - Additional support for Conduit Mesh Blueprint
  - Added Limited support for VTK-m filters

VisIt 3.0 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

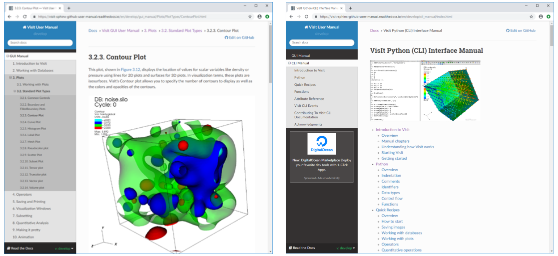
- We ported our legacy docs to Sphinx, now hosted on Read the Docs
  - https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/

GitHub



VisIt source repo and issue tracking on GitHub

Read the Docs



VisIt manuals on Read the Docs

VisIt 3.0 included major updates to our software development process

Better Scientific Software (BSSw) Blog Post written by Mark Miller about our team's efforts:

Continuous Technology Refreshment: An Introduction Using Recent Tech Refresh Experiences on VisIt

https://bssw.io/blog_posts/continuous-technology-refreshment-an-introduction-using-recent-tech-refresh-experiences-on-visit

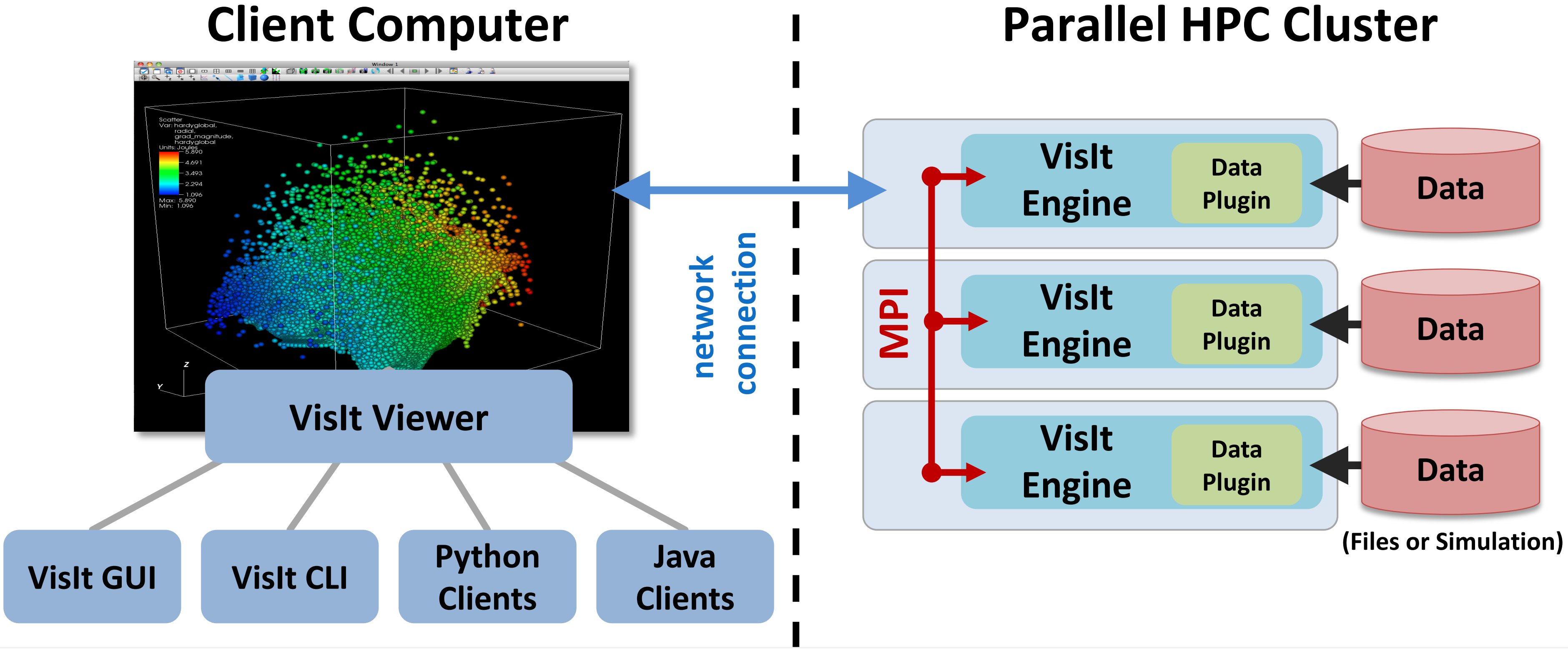
The VisIt team releases binaries for several platforms and a script that automates the build process

“How do I obtain VisIt?”

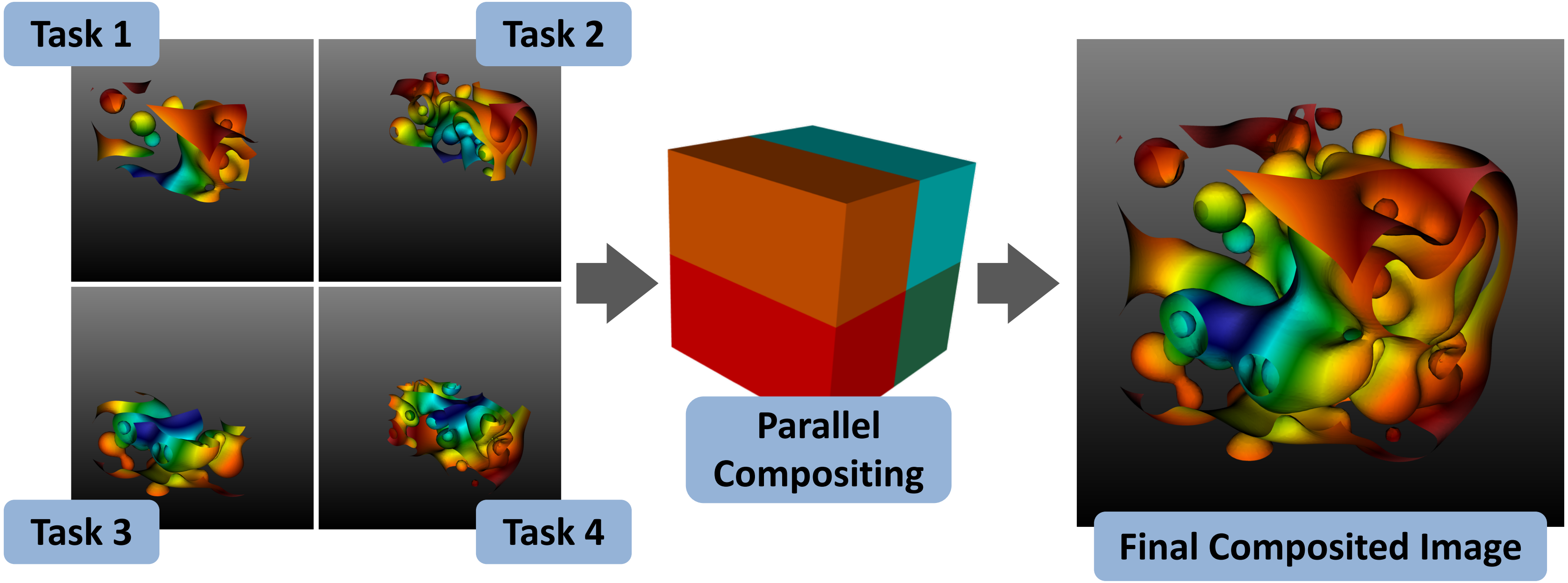
- Use an existing build:
  - For your Laptop or Workstation:
    - Binaries for Windows, OSX, and Linux (RHEL + Ubuntu): (https://wci.llnl.gov/simulation/computer-codes/visit/executables)
  - VisIt on ALCF’s Cooley:
    - https://www.alcf.anl.gov/user-guides/visit-cooley
  - Several other HPC centers 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 employs a parallelized client-server architecture



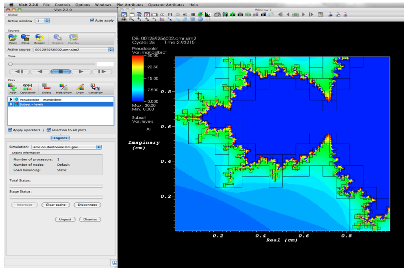
VisIt automatically switches to a scalable rendering mode when plotting large data sets on HPC clusters

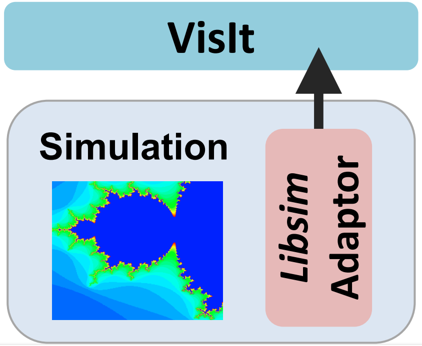


In addition to scalable surface rendering, VisIt also provides scalable volume rendering

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

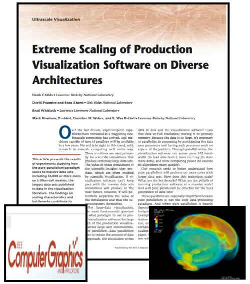




VisIt is used as a platform to deploy visualization research

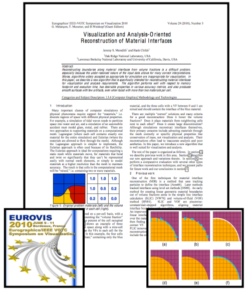
- **DOE Research Collaborations**

- **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

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

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

VisIt LibSim
https://visit.llnl.gov

SENSEI in situ
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

CONDUIT

Intuitive APIs for in-memory data description and exchange

http://software.llnl.gov/conduit

Ascent

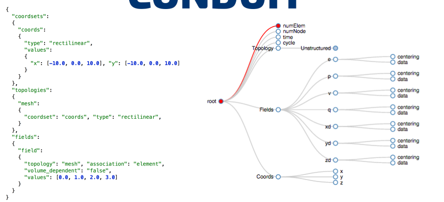
Flyweight in-situ visualization and analysis for HPC simulations

http://ascent-dav.org

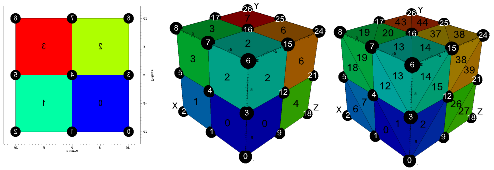
Conduit provides intuitive APIs for in-memory data description and exchange

- **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)

CONDUIT



Hierarchical in-memory data description



Conventions for sharing in-memory mesh data

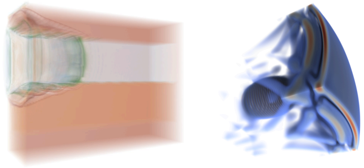
http://software.llnl.gov/conduit
http://github.com/llnl/conduit

Website and GitHub Repo

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
  - Has lower memory requirements then current tools
  - Requires less dependencies than current tools (ex: no OpenGL)

Ascent



Visualizations created using Ascent

python™ HDF CINEMA

Extracts supported by Ascent

http://ascent-dav.org
https://github.com/Alpine-DAV/ascent

Website and GitHub Repo

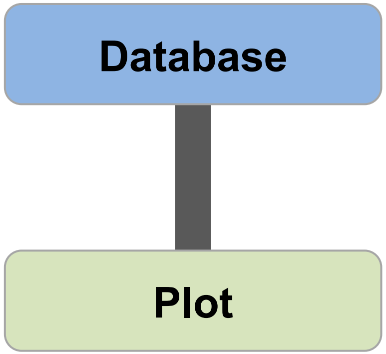
VisIt's Visualization Building Blocks

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

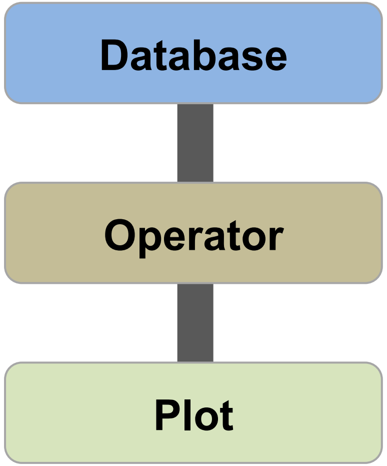


Open a database, which reads from a file
(Example: Open file1.hdf5)

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



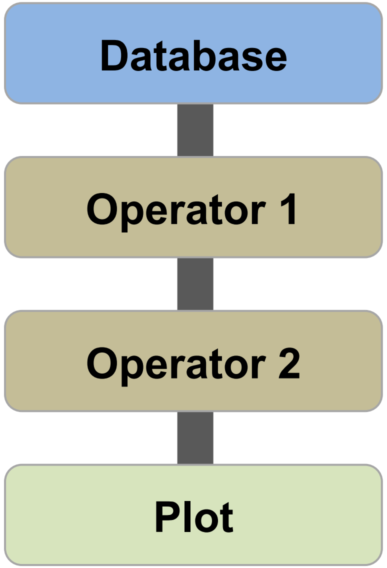
Open a database, which reads from a file
(Example: Open file1.hdf5)

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)

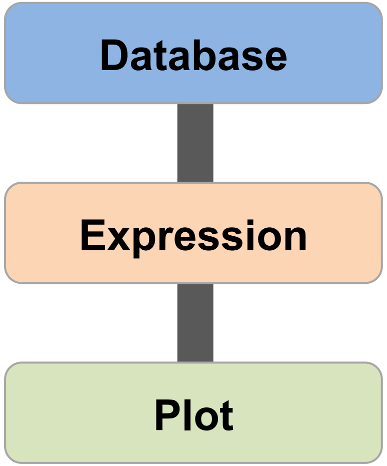
Apply an operator to transform the data
(Example: Slice operator)

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



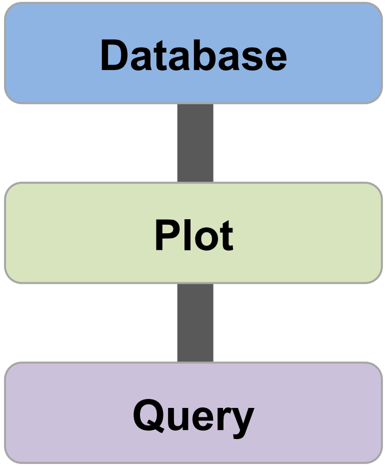
Open a database, which reads from a file
(Example: Open file1.hdf5)

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

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

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

Operator 1 — Apply an operator to transform the data (Example: Slice operator)

Operator 2 — Apply a second operator to transform the data (Example: Elevate operator)

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

Query — 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
- Github: https://github.com/visit-dav/visit

Hands-on Session

Guided Tour of VisIt

- **Materials from:**
  - https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/tutorials/VisIt_Basics.html
  - https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/tutorials/Data_Analysis.html
  - https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/tutorials/Scripting.html

Potential Flow Simulation Exploration

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

Aneurysm Simulation Exploration

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

Additional Hands-on Materials

- **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:

- Cyrus Harrison: cyrush@llnl.gov

User Resources:

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

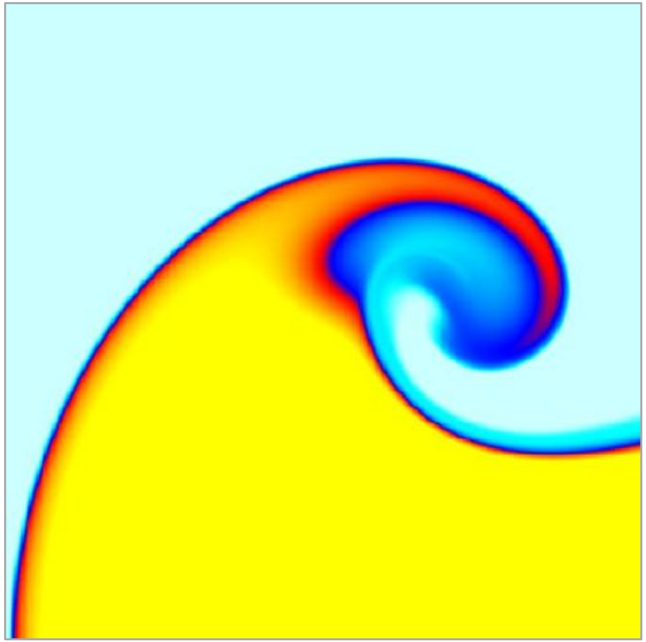
Developer Resources:

- Email: visit-developers@ornl.gov
- Github: https://github.com/visit-dav/visit

Visualization Techniques for Mesh-based Simulations

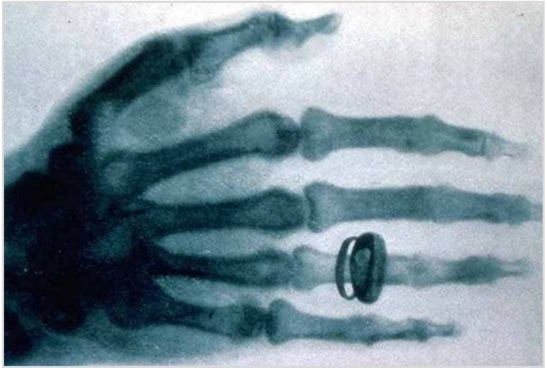
Pseudocolor rendering maps scalar fields to a range of colors

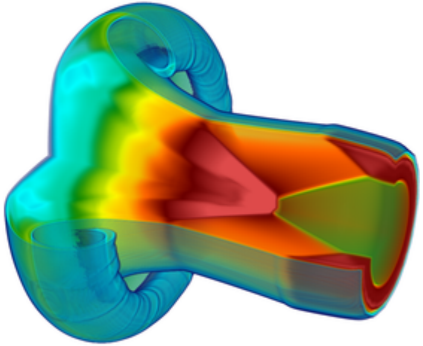
Pseudocolor rendering of Elevation

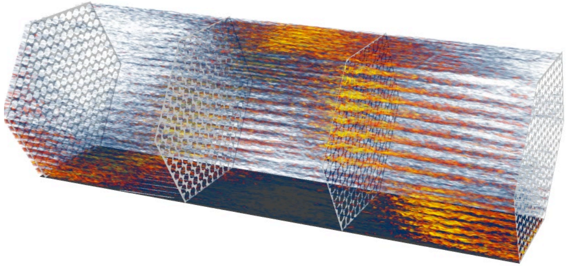


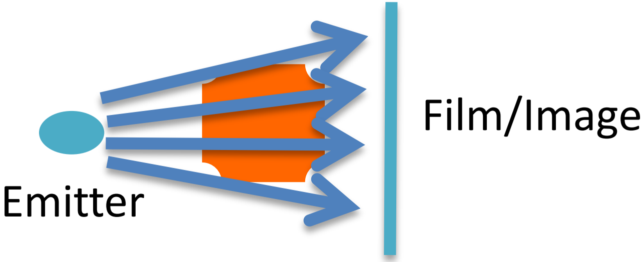
Pseudocolor rendering of Density

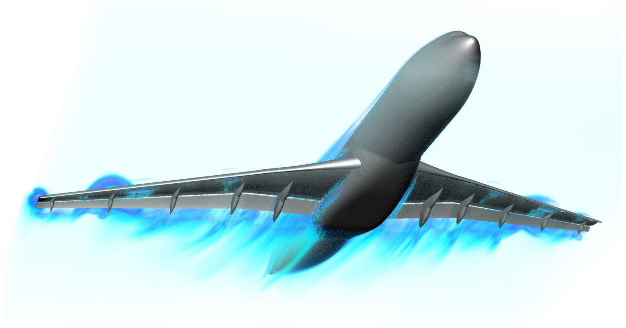
Volume Rendering cast rays though data and applies transfer functions to produce an image



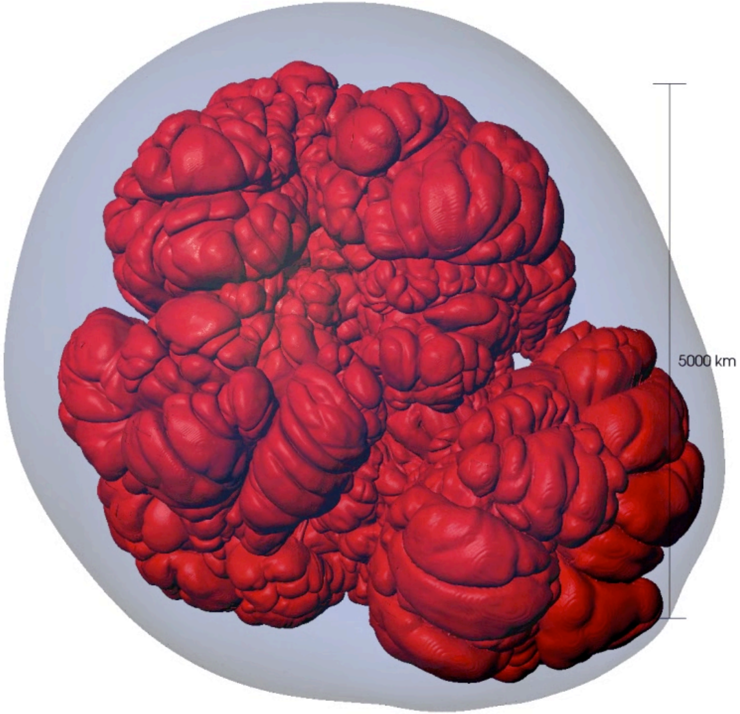








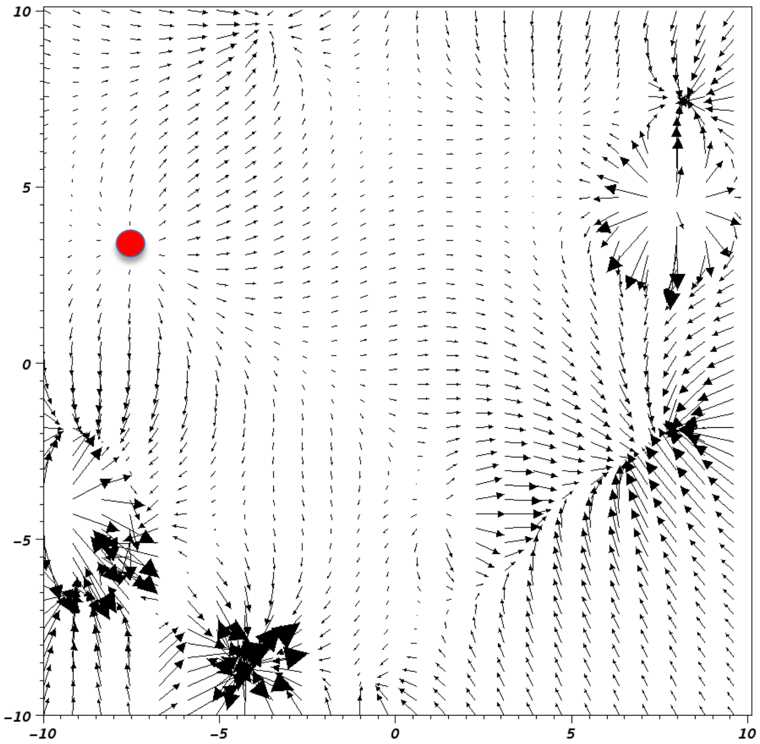
Isosurfacing (Contouring) extracts surfaces of that represent level sets of field values



Particle advection is the foundation of several flow visualization techniques

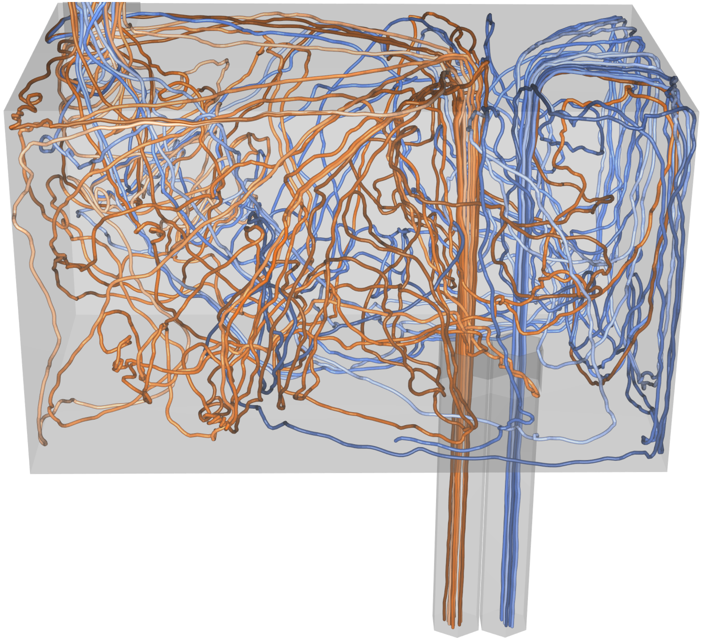
- $S(t)$ = position of particle at time t
- $S(t_0) = p_0$
  - t_0: initial time
  - p_0: initial position
- $S'(t) = v(t, S(t))$
  - $v(t, p)$: velocity at time t and position p
  - $S'(t)$: derivative of the integral curve at time t

This is an ordinary differential equation.



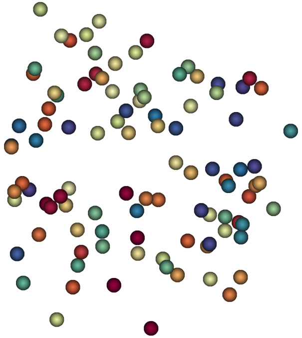
Streamline and Pathline computation are built on particle advection

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

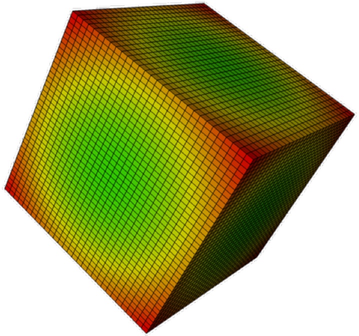


Meshes discretize continuous space

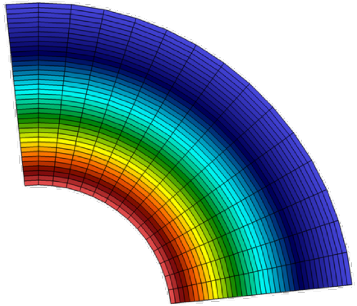
- **Simulations use a wide range of mesh types, defined in terms of:**
  - A set of coordinates ("nodes" / "points" / "vertices")
  - A collection of "zones" / "cells" / "elements" on the coordinate set



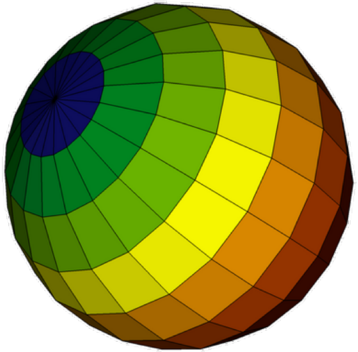
Points



Uniform



Curvilinear



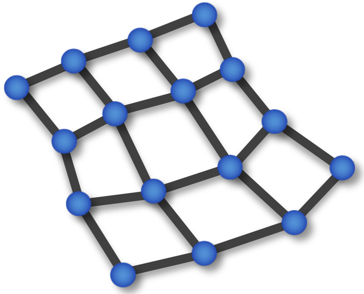
Unstructured

VisIt uses the "Zone" and "Node" nomenclature throughout its interface.

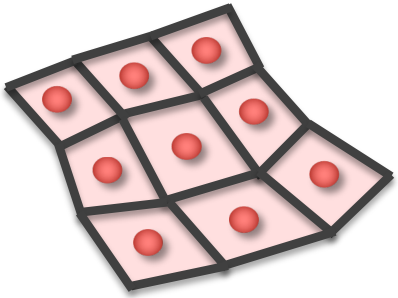
Mesh fields are variables associated with the mesh that hold simulation state

- 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

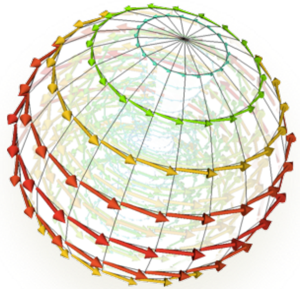
- Field values for each zone or node can be scalar, or multi-valued (vectors, tensors, etc.)



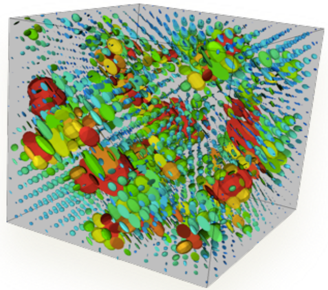
Nodal Association



Zonal Association



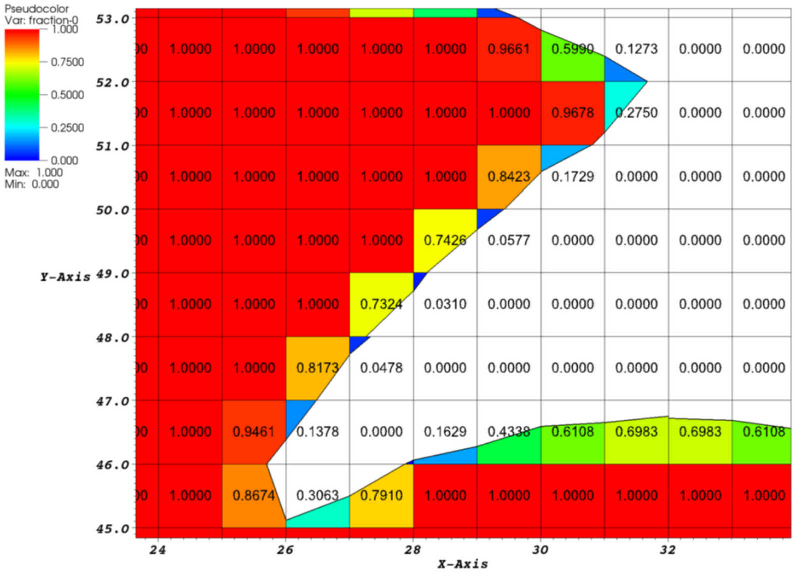
Vector Field



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

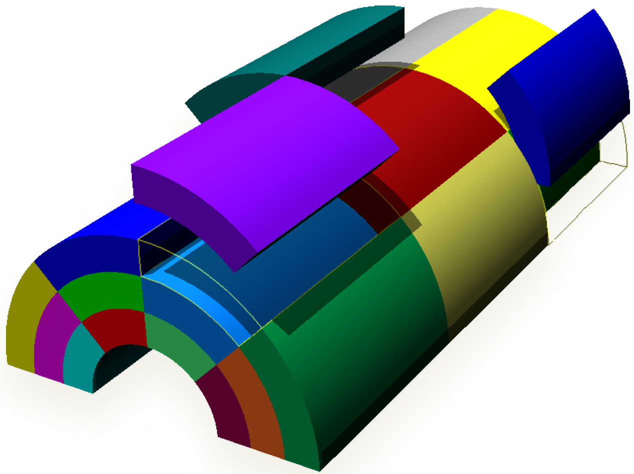
- Species describe sub-grid variable composition
  — Example: Material “Air” is made of species “N2” ,“O2”, “Ar”, “CO2”, etc.

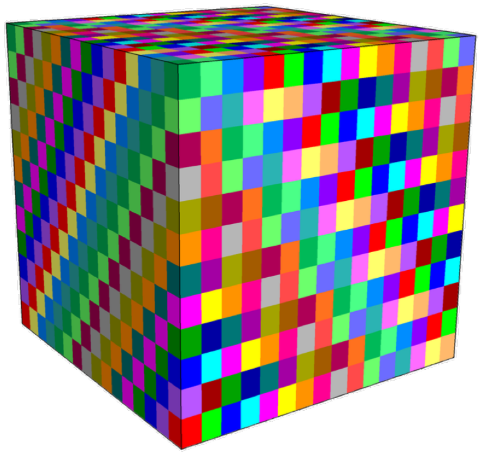
- Species are used for weighting, not to indicate sub-zonal interfaces.
  — They are typically used to capture fractions of “atomically mixed” values.

Domain decomposed meshes enable scalable parallel visualization and analysis algorithms

- Simulation meshes may be composed of smaller mesh “blocks” or “domains”.

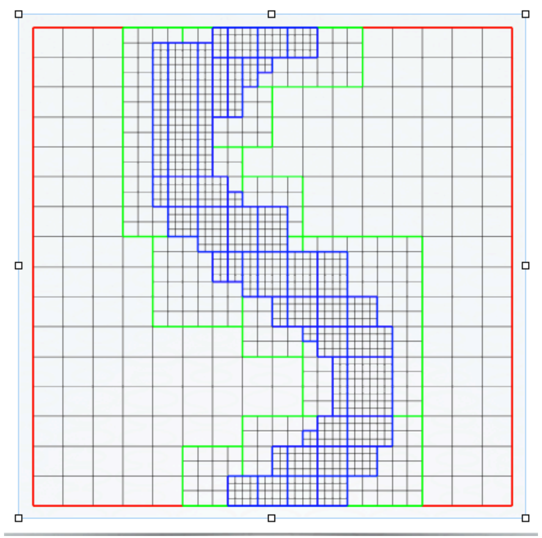
- Domains are partitioned across MPI tasks for processing.

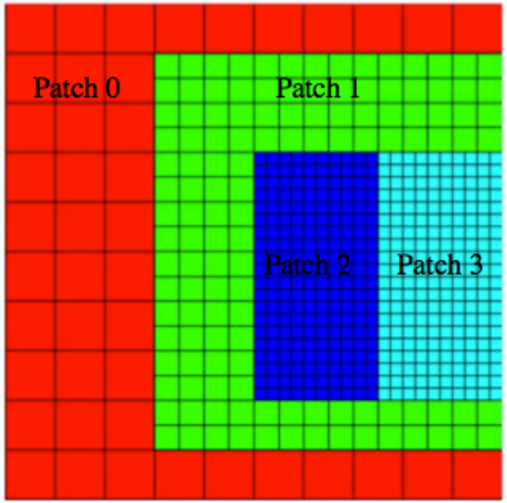


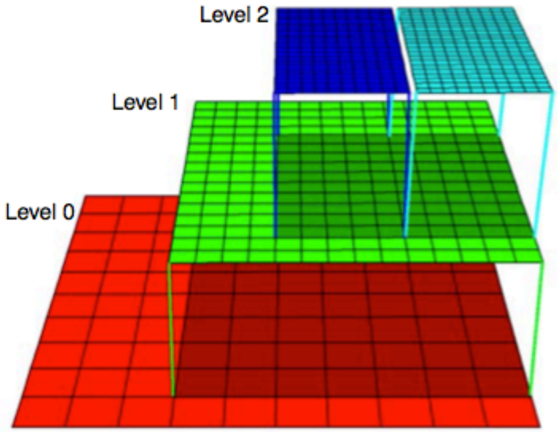


Adaptive Mesh Refinement (AMR) refines meshes into patches that capture details across length scales

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







Lawrence Livermore National Laboratory