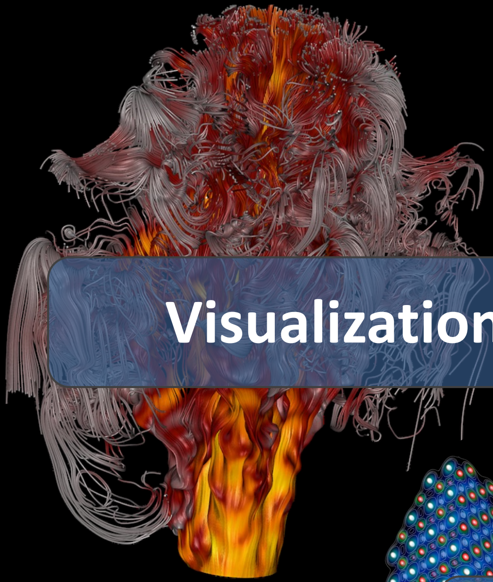
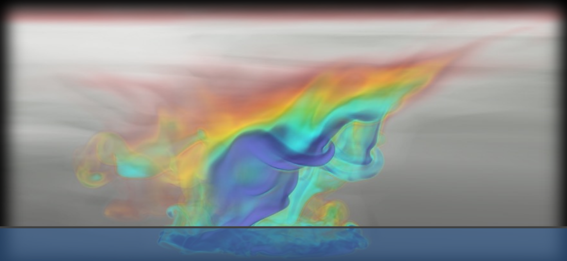
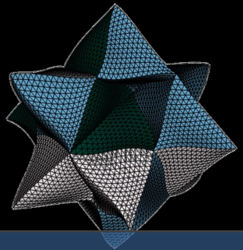
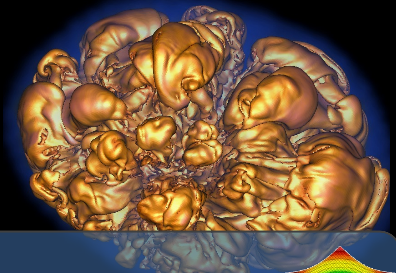
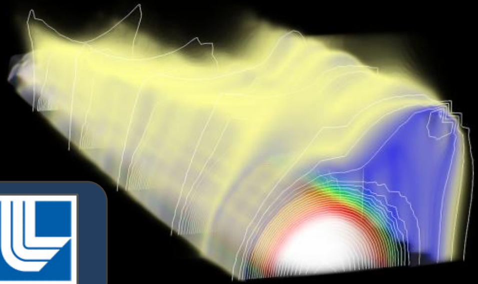
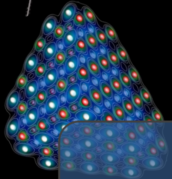
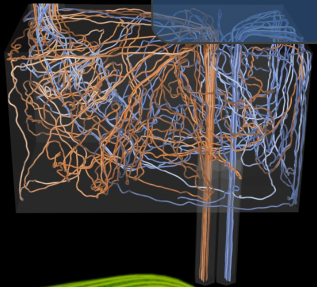





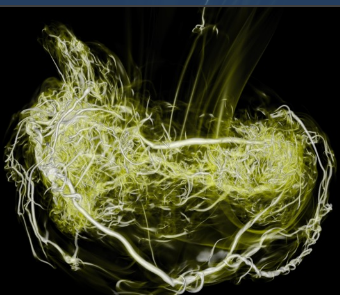
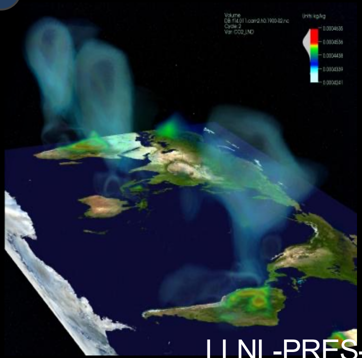
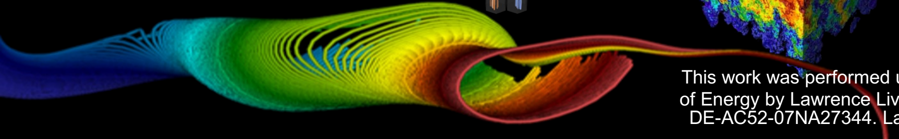
Visualization and Analysis of HPC Simulation Data with VisIt

ATPESC 2023
Monday August 7th, 2023

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-852365

Acknowledgements

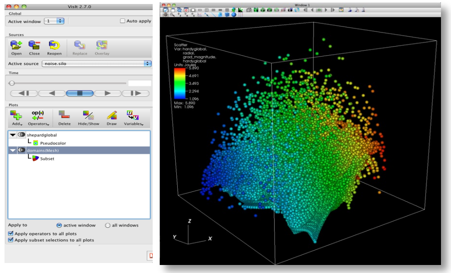
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344.
Lawrence Livermore National Security, LLC

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

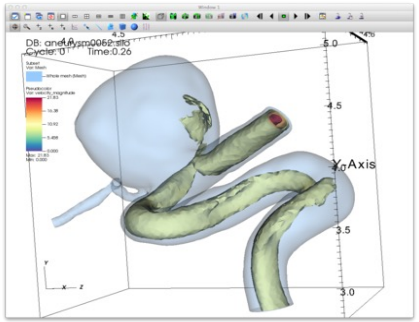
Outline

- VisIt Project Introduction (~30 min)

- Hands-on: (~60 min)
  - Guided tour of VisIt
  - Visualization of an Aneurysm (Blood Flow) Simulation



Intro to VisIt



Simulation Exploration

Tutorial Resources

- **VisIt 3.3.3**
  - https://github.com/visit-dav/visit/releases

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

- **How to get in touch**
  - **GitHub:** https://github.com/visit-dav/visit
  - **GitHub Discussions:** https://github.com/visit-dav/visit/discussions

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/)

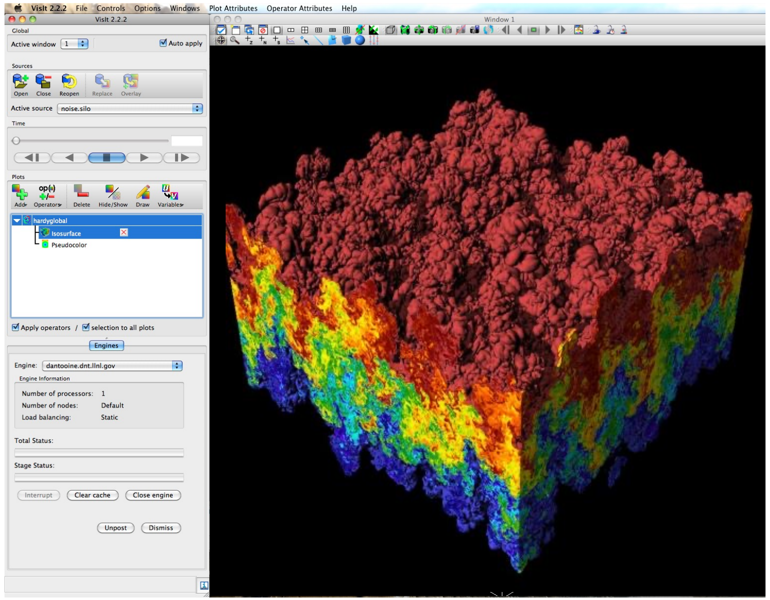
Available thanks to:

- 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.



Active window: 1
☑ Auto apply
Active source: noise.silo

hardyglobal
- Isosurface
- Pseudocolor

☑ Apply operators / ☑ selection to all plots

Engine: dantooine.dnt.llnl.gov

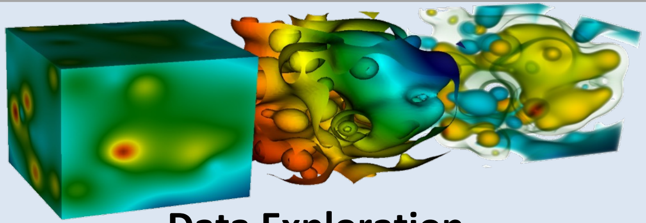
Engine Information

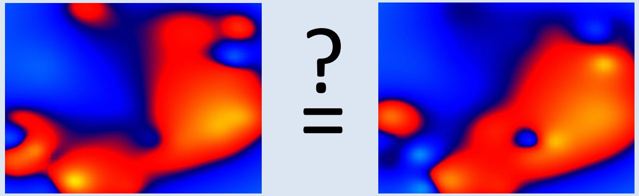
Number of processors: 1
Number of nodes: Default
Load balancing: Static

Total Status:
Stage Status:

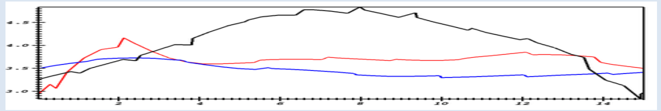
Pseudocolor plot of Density
(27 billion element dataset)

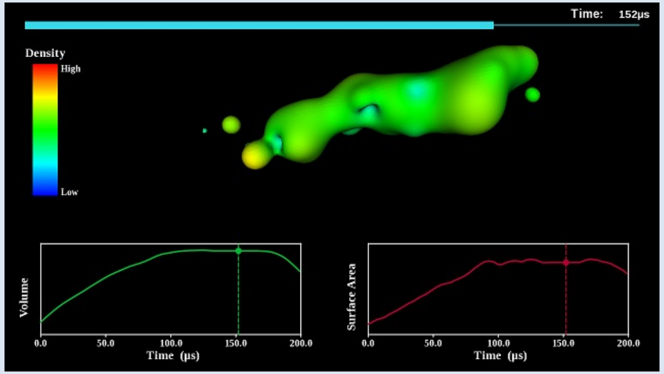
VisIt supports a wide range of use cases


Data Exploration

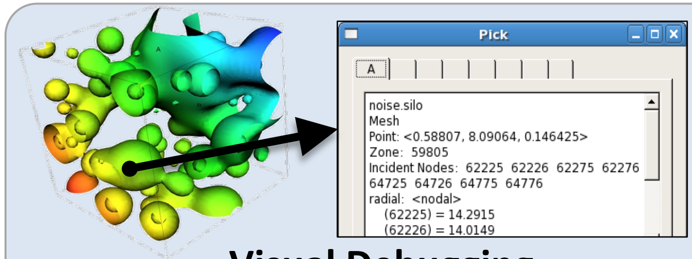


Comparative Analysis


Quantitative Analysis



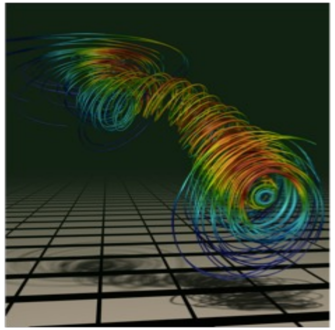
Presentation Graphics

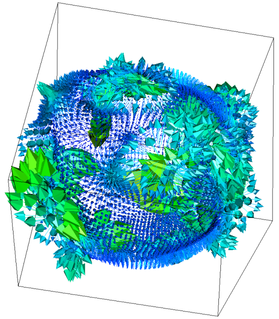


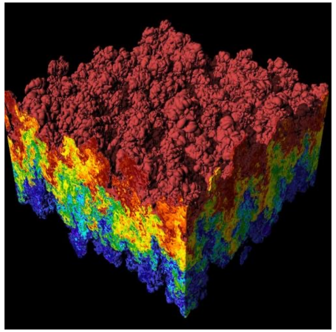
noise.silo
Mesh
Point: <0.58807, 8.09064, 0.146425>
Zone: 59805
Incident Nodes: 62225 62226 62275 62276
64725 64726 64775 64776
radial: <nodal>
(62225) = 14.2915
(62226) = 14.0149

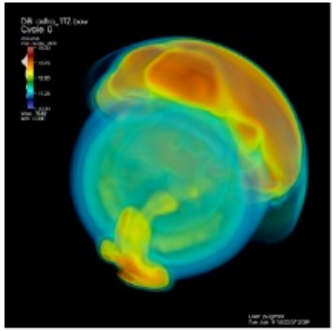
Visual Debugging

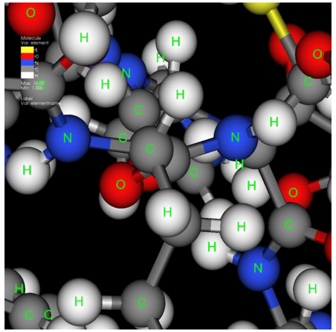
VisIt provides a wide range of plotting features for simulation data across many scientific domains

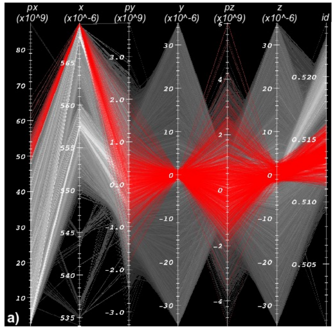

Streamlines / Pathlines


Vector / Tensor Glyphs


Pseudocolor Rendering

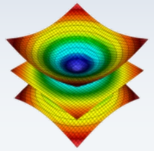

Volume Rendering


Molecular Visualization


Parallel Coordinates

VisIt is a vibrant project with many participants

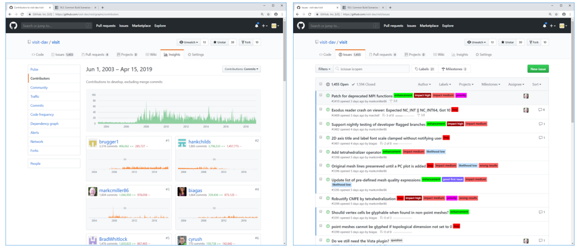
- The VisIt project started in 2000 to support LLNL's large scale ASC physics codes.
- The project grew beyond LLNL and ASC with development from DOE SciDAC and other efforts.
- VisIt is now supported by multiple organizations:
  - LLNL, LBNL, ORNL, Univ of Oregon, Univ of Utah, Intelligent Light, …
- Over 100 person years of effort, 1.5+ million lines of code.

2000	2003	2005	2006	2008	2010	2012 - 2016	2017	2019
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
ASC		R&D 100	VACET SciDAC Scientific Discovery through Advanced Computing			SciDAC Scientific Discovery through Advanced Computing Intelligent Light	ECP ALPINE (ParaView/VisIt) Ascent	GitHub repo

VisIt is hosted and developed using GitHub

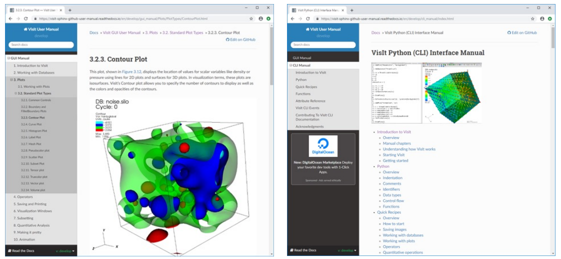
- Our Source Code, Issue tracking, and Discussions are in the `visit-dav` GitHub organization:
  - https://github.com/visit-dav/
- Our Docs are hosted on Read the Docs
  - https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/

GitHub



VisIt source repo and issue tracking on GitHub

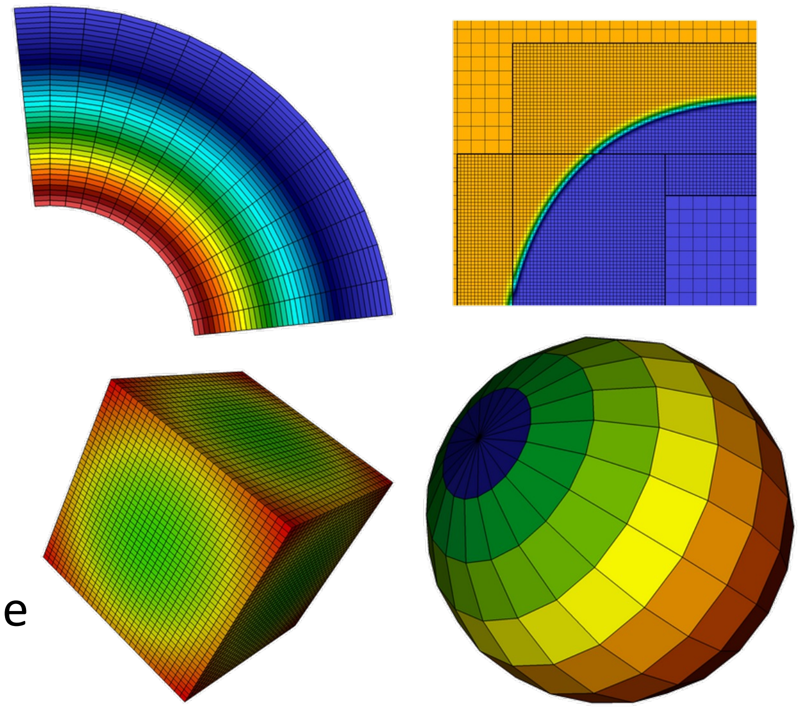
Read the Docs



VisIt manuals on Read the Docs

VisIt 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, and many other flavors): (https://github.com/visit-dav/visit/releases/)
  - 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://github.com/visit-dav/visit/releases/))
  - 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?"

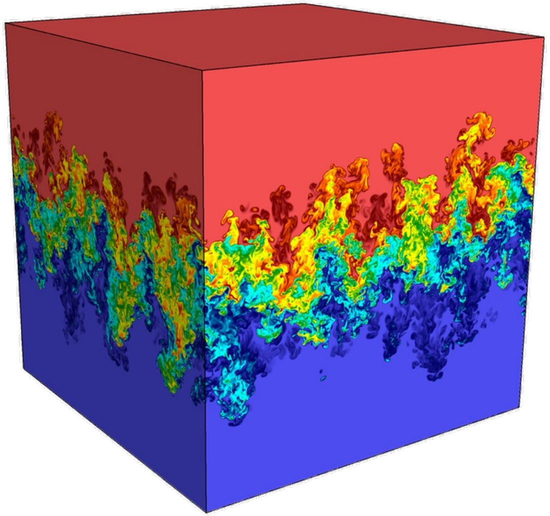
- The *PlainText* database reader can read simple text files (CSV, etc)
  - http://visitusers.org/index.php?title=Using_the_PlainText_reader
- Write to a commonly used format:
  - *VTK, Silo, Xdmf, PVTK, Conduit Blueprint (JSON/YAML, or HDF5 files)*
- We are investing heavily in Conduit Blueprint Support
  - http://llnl-conduit.readthedocs.io/en/latest/blueprint_mesh.html
- Experiment with the *visit_writer* utility.
- Consult the Getting Data Into VisIt Manual.

We have continued to update our infrastructure and add new features in recent releases

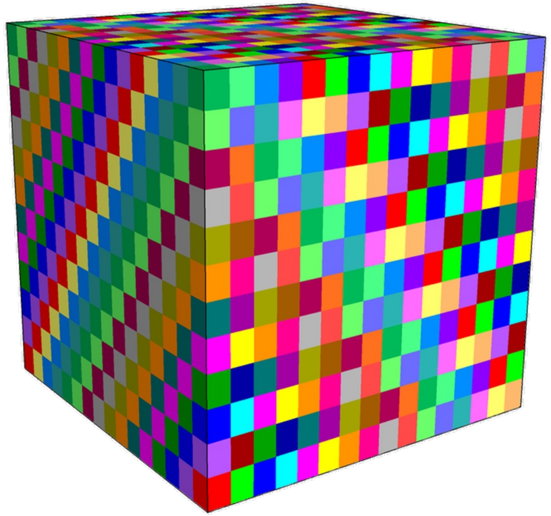
- VisIt 3.3 Series -- Database Plugin, Query, and GPU Execution Updates
  - 3.3.1 (April 2021)
    - Color table tagging improvements, 20 bugfixes, 11 enhancements, 9 developer changes
    - https://visit-dav.github.io/visit-website/releases/release-notes-3.3.1/
  - 3.3.2 (July 2021)
    - X Ray Image query blueprint output, 13 bug fixes, 15 enhancements
    - https://visit-dav.github.io/visit-website/releases/release-notes-3.3.2/
  - 3.3.3 (January 2022)
    - AMD GPU support, 5 bugfixes, 10 enhancements
    - https://visit-dav.github.io/visit-website/releases/release-notes-3.3.3/

- VisIt 3.4.0 (Fall/Winter 2023)
  - Infrastructure upgrades
    - Update to VTK-9
    - Update to Qt 6
    - Update to using modern CMake design paradigms based on targets
    - Using the practices described in Effective CMake

VisIt uses MPI for distributed-memory parallelism on HPC clusters

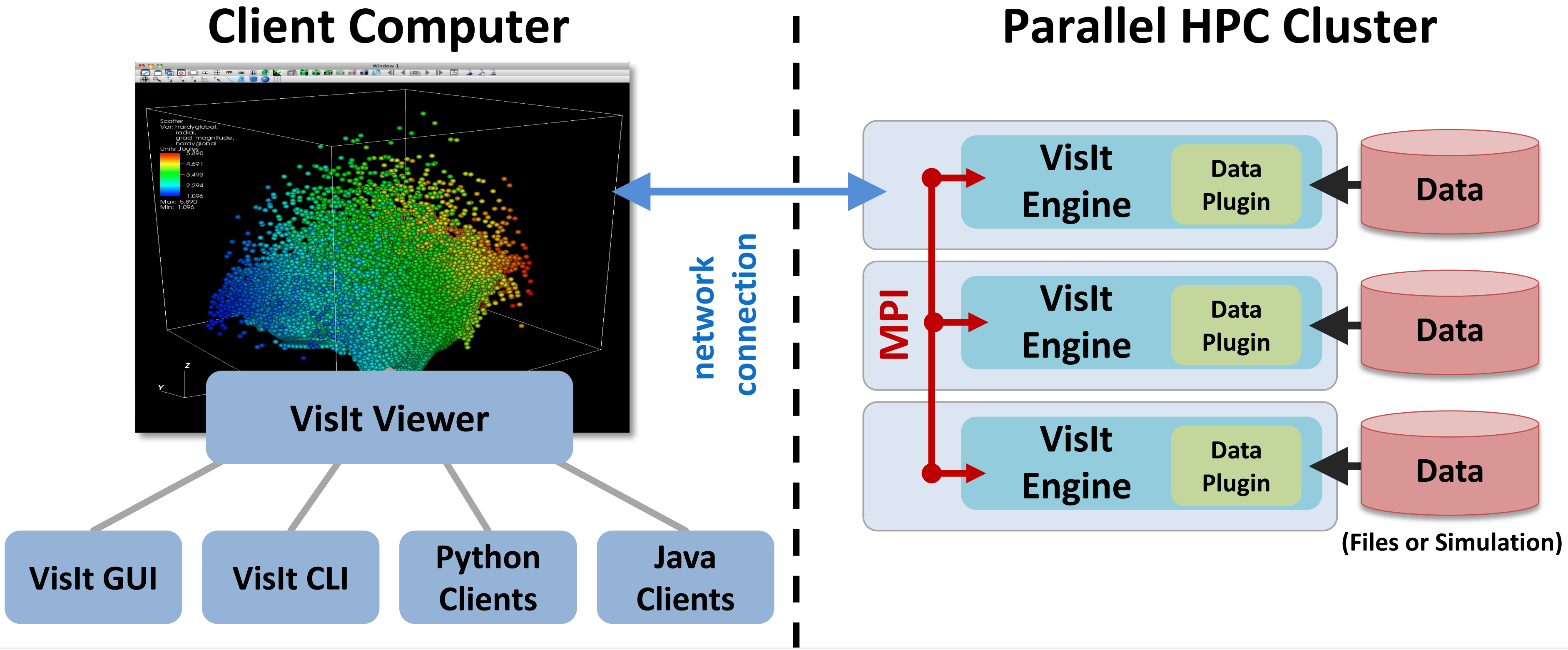


Full Dataset
(27 billion total elements)

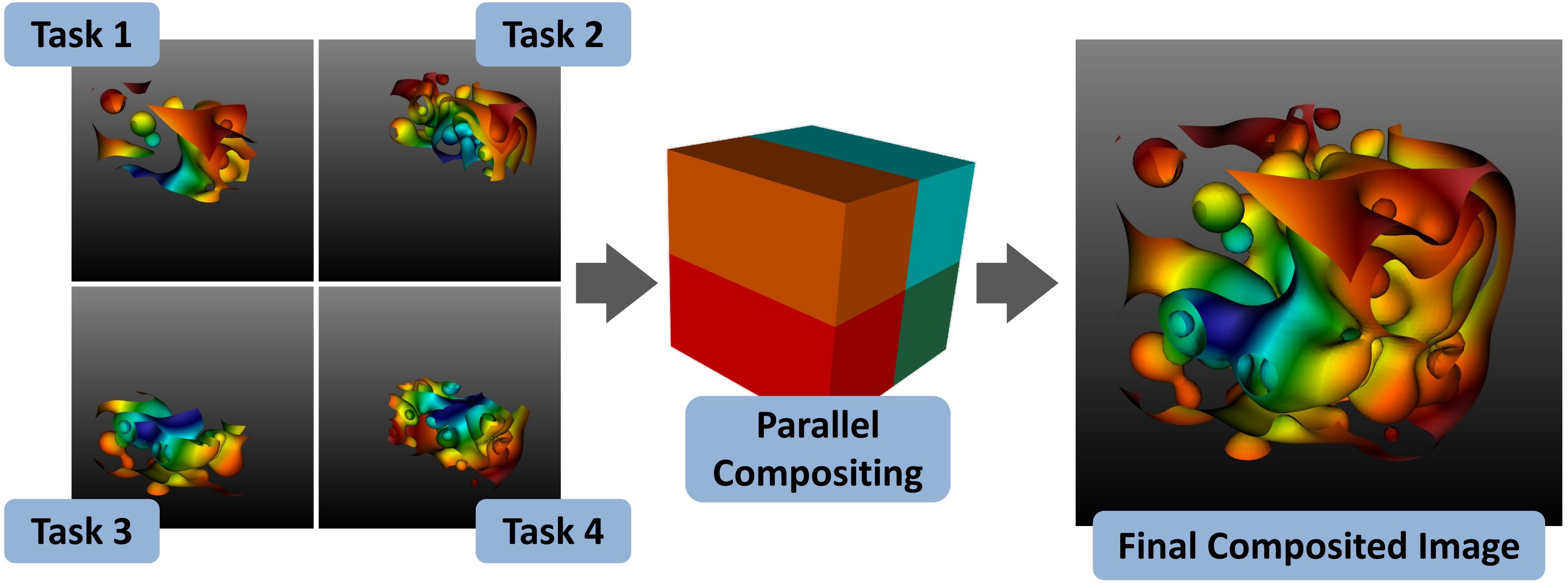


3072 sub-grids
(each 192x129x256 cells)

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

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 from NVIDIA, AMD, Intel.

VTK-m

http://m.vtk.org

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

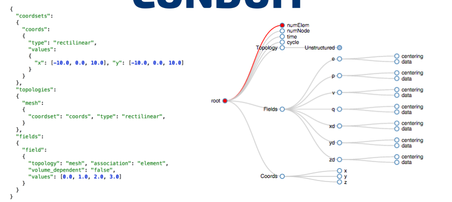
Ascent

Flyweight in-situ visualization and analysis for HPC simulations

http://ascent-dav.org

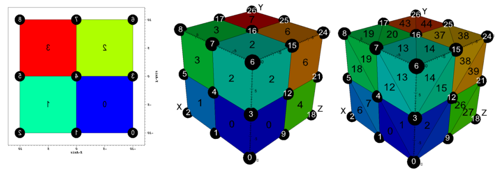
Conduit provides intuitive APIs for in-memory data description and exchange

- **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)



```
{
  "coordsets":
  {
    "coords":
    {
      "type": "rectilinear",
      "values":
      {
        "x": [-10.0, 0.0, 10.0], "y": [-10.0, 0.0, 10.0]
      }
    }
  },
  "topologies":
  {
    "mesh":
    {
      "coordset": "coords", "type": "rectilinear",
    }
  },
  "fields":
  {
    "field":
    {
      "topology": "mesh", "association": "element",
      "volume_dependent": "false",
      "values": [0.0, 1.0, 2.0, 3.0]
    }
  }
}
```

Hierarchical in-memory data description



Conventions for sharing in-memory mesh data

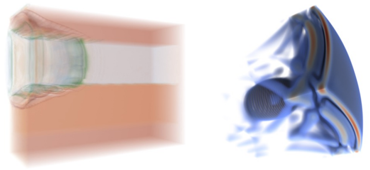
http://software.llnl.gov/conduit
http://github.com/llnl/conduit

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***
  - Supports 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, HIP, 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

python™ HDF CINEMA

Extracts supported by Ascent

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

Website and GitHub Repo

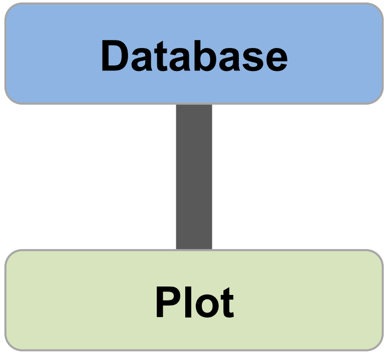
VisIt's Visualization Building Blocks

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

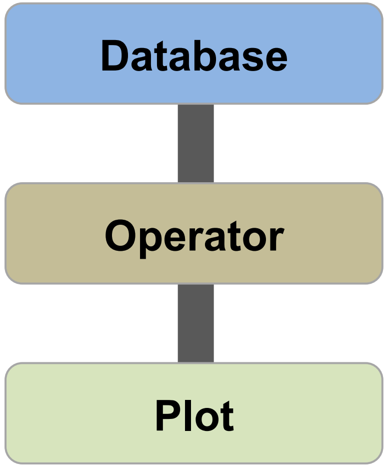


Open a database, which reads from a file (Example: Open file1.hdf5)

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



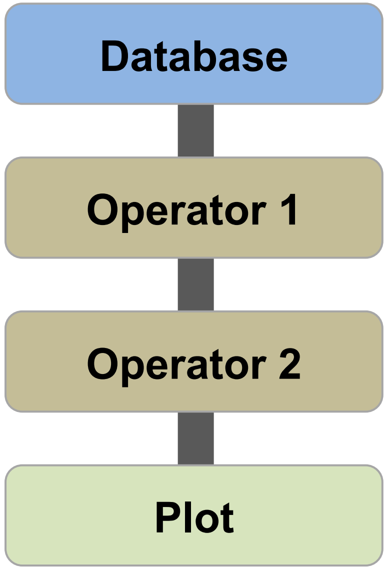
Open a database, which reads from a file (Example: Open file1.hdf5)

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)

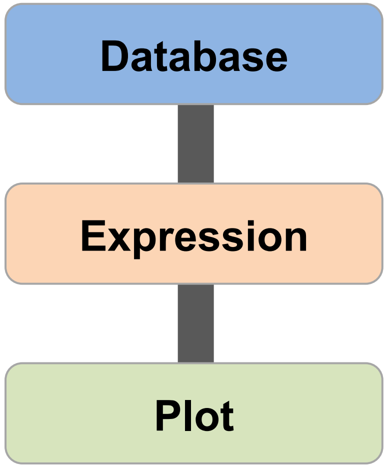
Apply an operator to transform the data (Example: Slice operator)

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



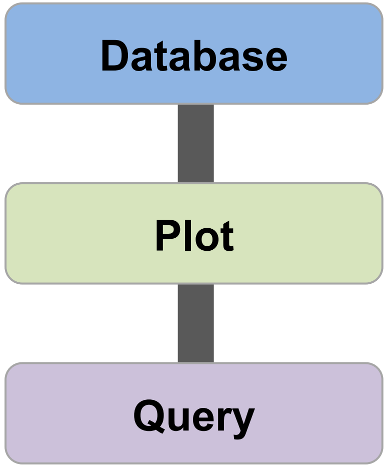
Open a database, which reads from a file
(Example: Open file1.hdf5)

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



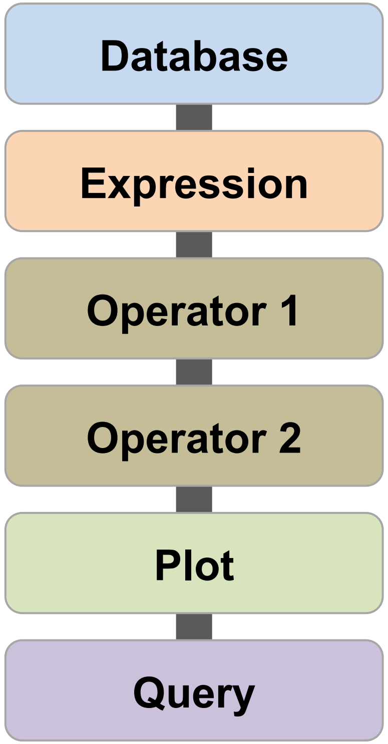
Open a database, which reads from a file (Example: Open file1.hdf5)

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



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

Resources:

- Main website: http://www.llnl.gov/visit
- Github: https://github.com/visit-dav/visit
- GitHub Discussions: https://github.com/visit-dav/visit/discussions
- Wiki: http://www.visitusers.org

Aneurysm Simulation Exploration

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

Remote Usage Tips

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

Python Scripting Basics

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

Connected Components

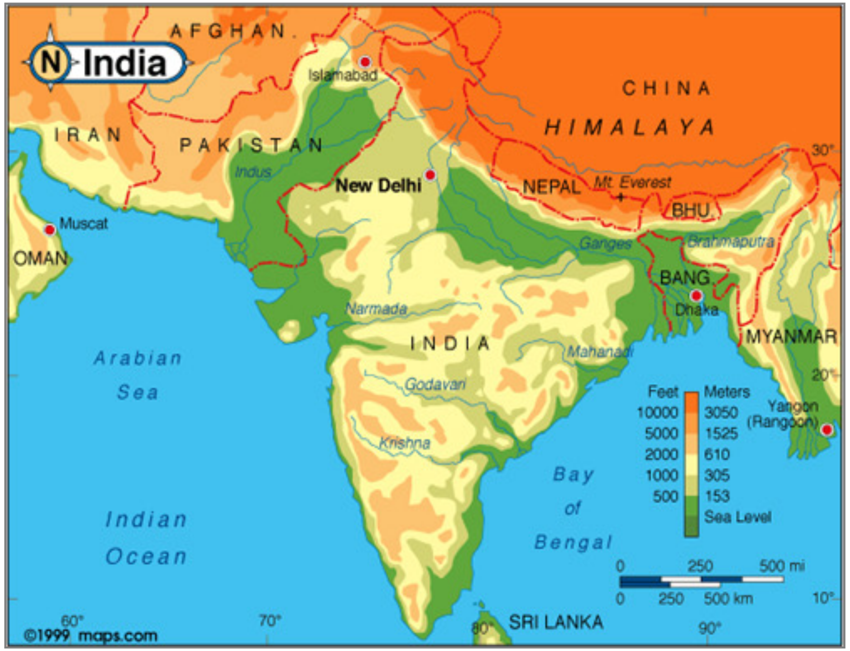
https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/tutorials/CCL.html

Additional Hands-on Materials

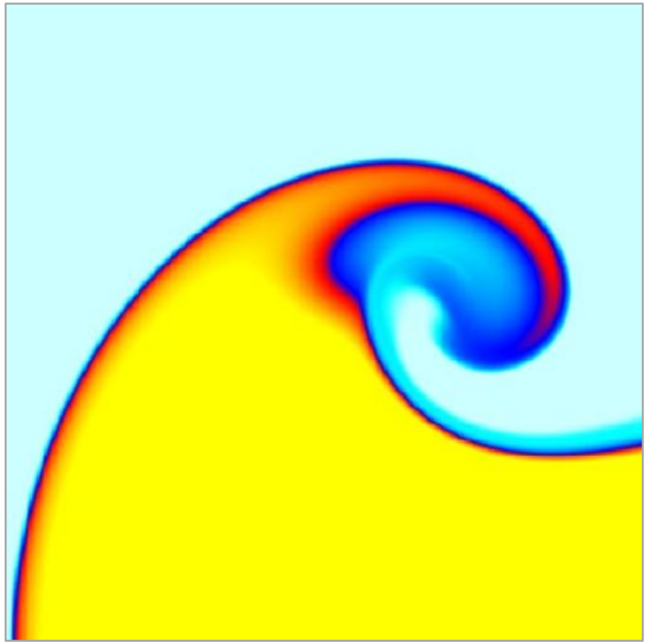
- **Potential Flow Simulation Exploration**
  - https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/tutorials/PotentialFlow.html
- **Water Flow Simulation Exploration**
  - http://visitusers.org/index.php?title=Water_Flow_Tutorial
- **Volume Rendering**
  - http://visitusers.org/index.php?title-Visit-tutorial-Volume-Rendering
- **Movie Making**
  - https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/tutorials/MakingMovies.html
- **Advanced Movie Making**
  - http://visitusers.org/index.php?title=Visit-tutorial-Advanced-movie-making

Visualization Techniques for Mesh-based Simulations

Pseudocolor rendering maps scalar fields to a range of colors

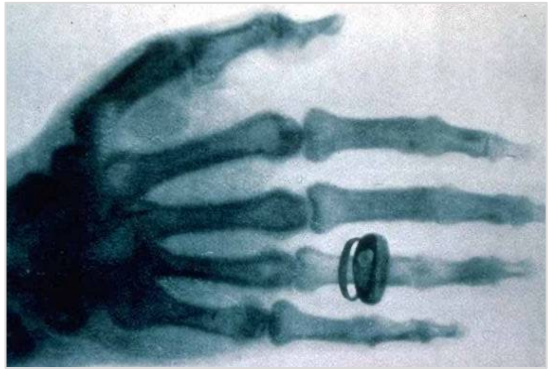


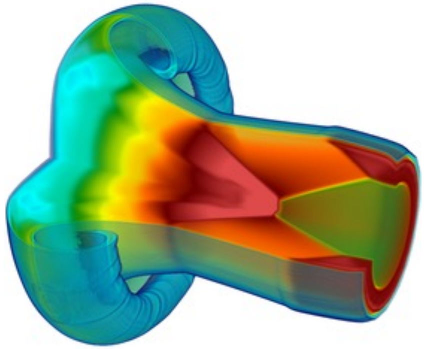
Pseudocolor rendering of Elevation

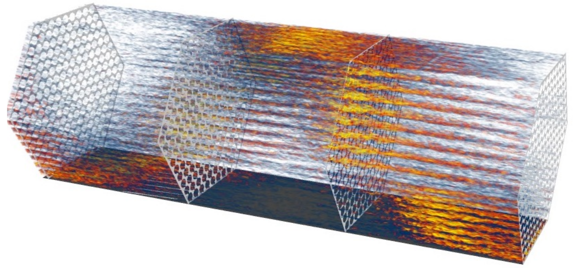


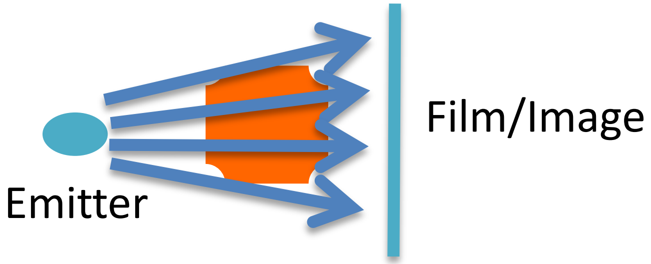
Pseudocolor rendering of Density

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

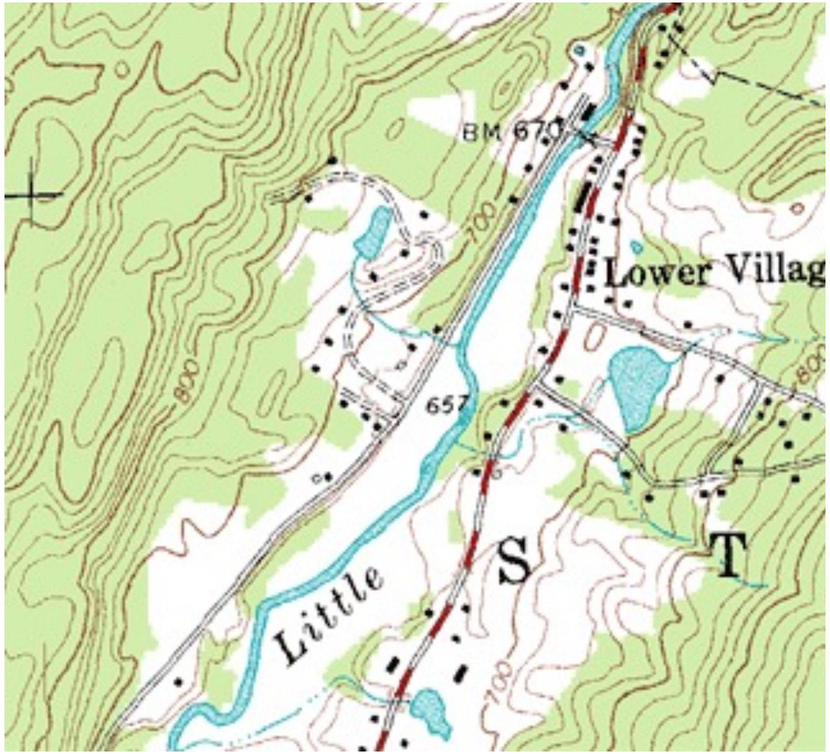


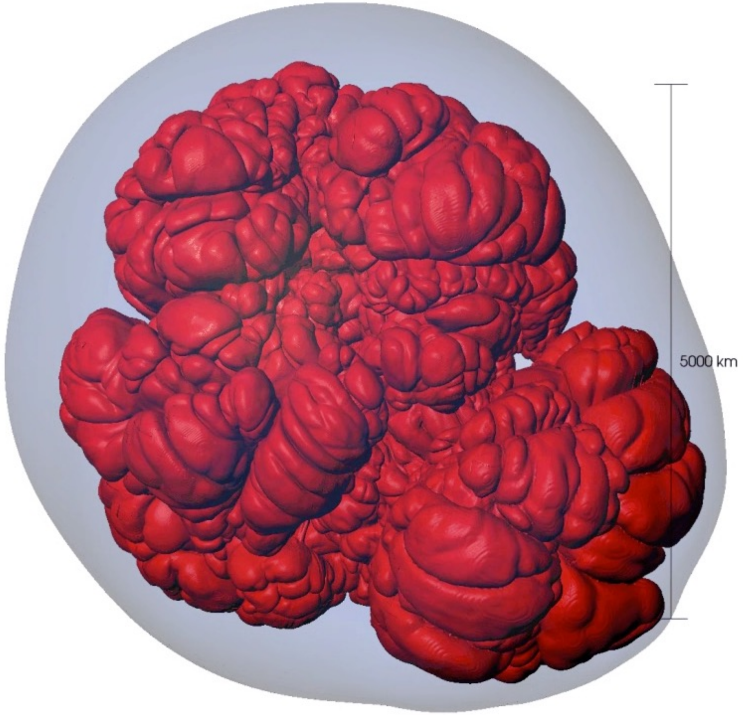






Isosurfacing (Contouring) extracts surfaces of that represent level sets of field values

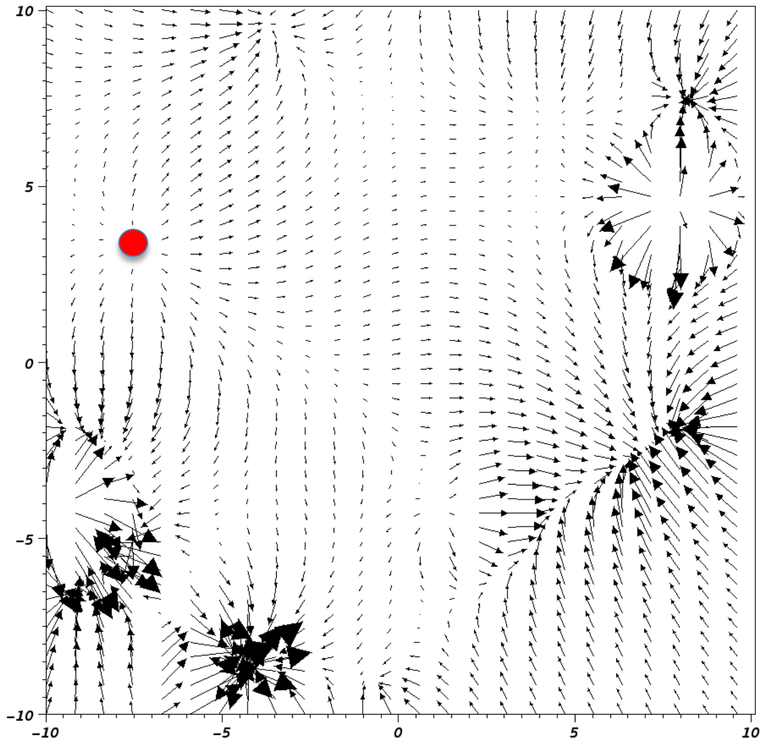




Particle advection is the foundation of several flow visualization techniques

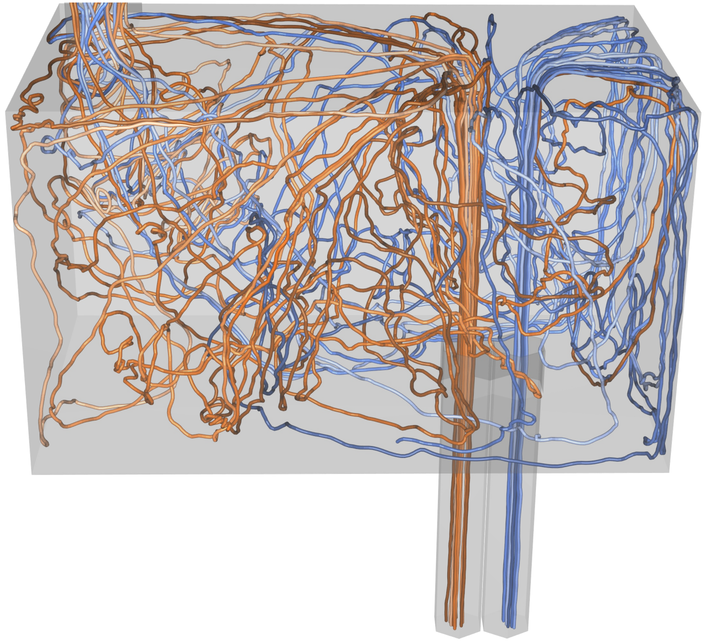
- $S(t)$ = position of particle at time t
- $S(t_0) = p_0$
  - t_0: initial time
  - p_0: initial position
- $S'(t) = v(t, S(t))$
  - $v(t, p)$: velocity at time t and position p
  - $S'(t)$: derivative of the integral curve at time t

This is an ordinary differential equation.



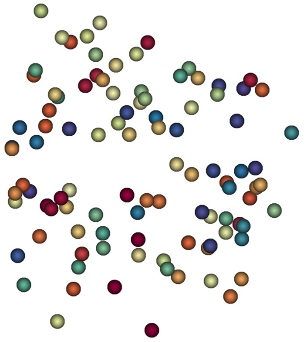
Streamline and Pathline computation are built on particle advection

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

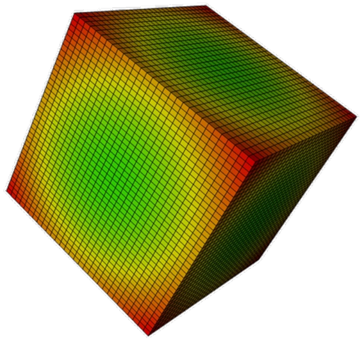


Meshes discretize continuous space

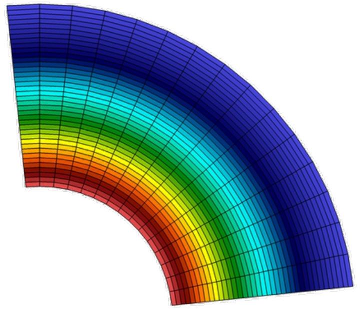
- **Simulations use a wide range of mesh types, defined in terms of:**
  - A set of coordinates (“nodes” / “points” / “vertices”)
  - A collection of “zones” / “cells” / “elements” on the coordinate set



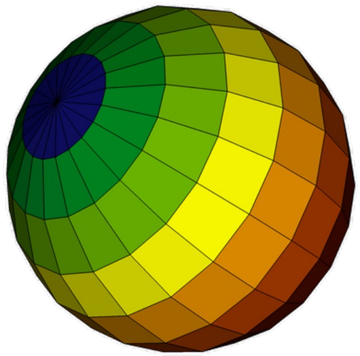
Points



Uniform



Curvilinear

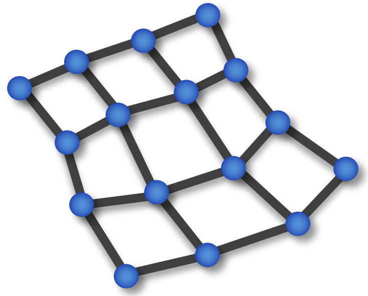


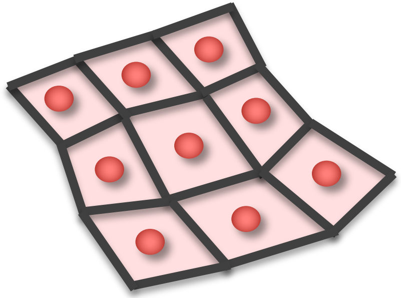
Unstructured

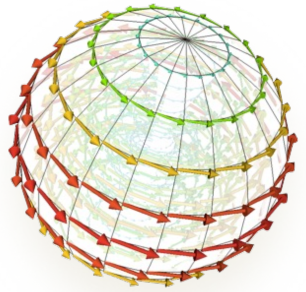
VisIt uses the “Zone” and “Node” nomenclature throughout its interface.

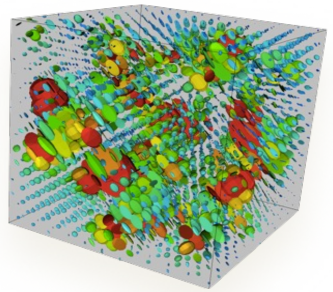
Mesh fields are variables associated with the mesh that hold simulation state

- 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

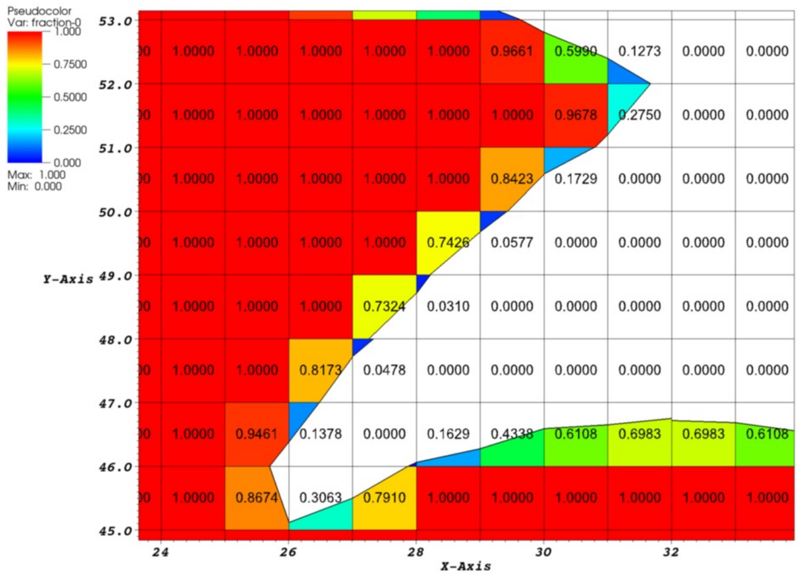

Vector Field


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.



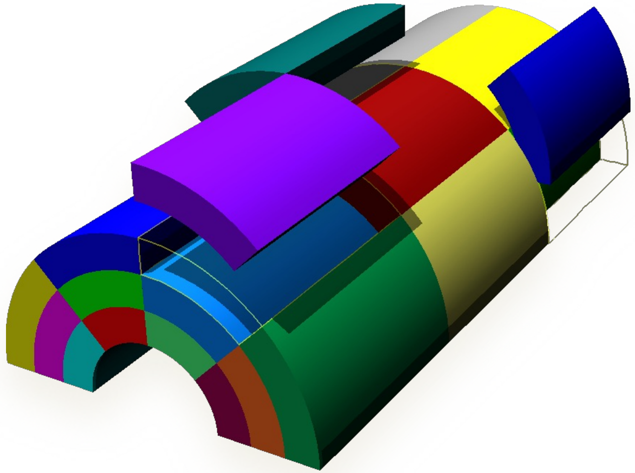
Species are used to capture sub-zonal weightings

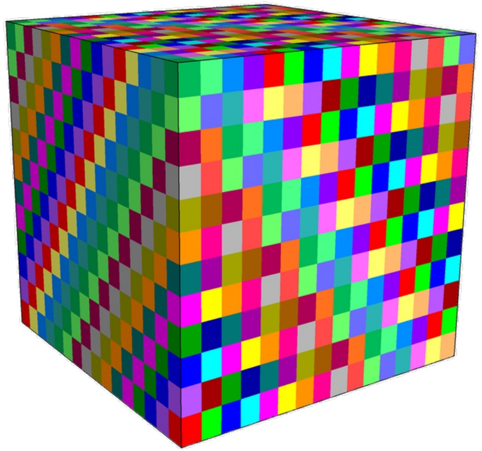
- Species describe sub-grid variable composition
  - Example: Material “Air” is made of species “N2” ,“O2”, “Ar”, “CO2”, etc.

- 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

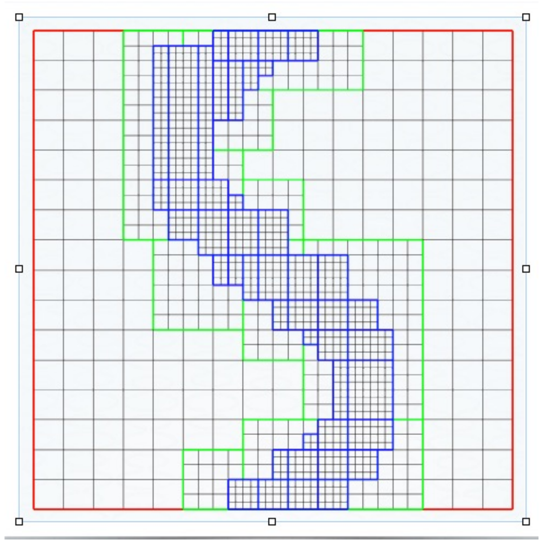
- Simulation meshes may be composed of smaller mesh “blocks” or “domains”.
- Domains are partitioned across MPI tasks for processing.

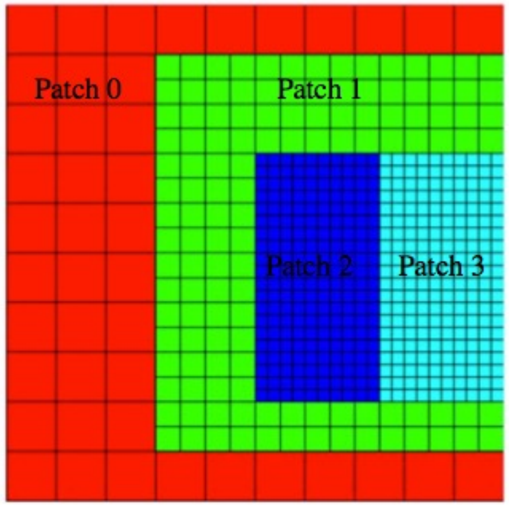


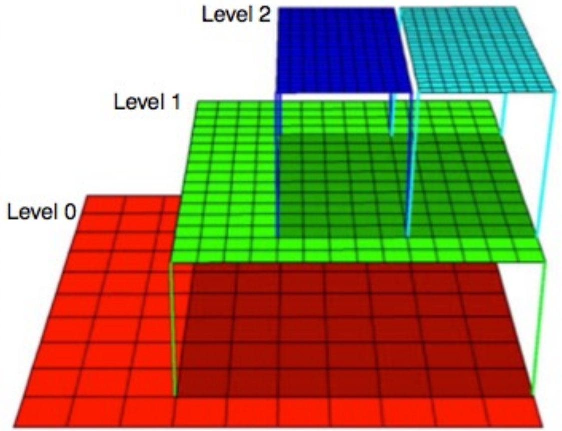


Adaptive Mesh Refinement (AMR) refines meshes into patches that capture details across length scales

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







Resources

Presenter Contact Info:

- Cyrus Harrison: cyrush@llnl.gov

Resources:

- Main website: http://www.llnl.gov/visit
- Github: https://github.com/visit-dav/visit
- GitHub Discussions: https://github.com/visit-dav/visit/discussions
- Wiki: http://www.visitusers.org

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

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

Lawrence Livermore National Laboratory