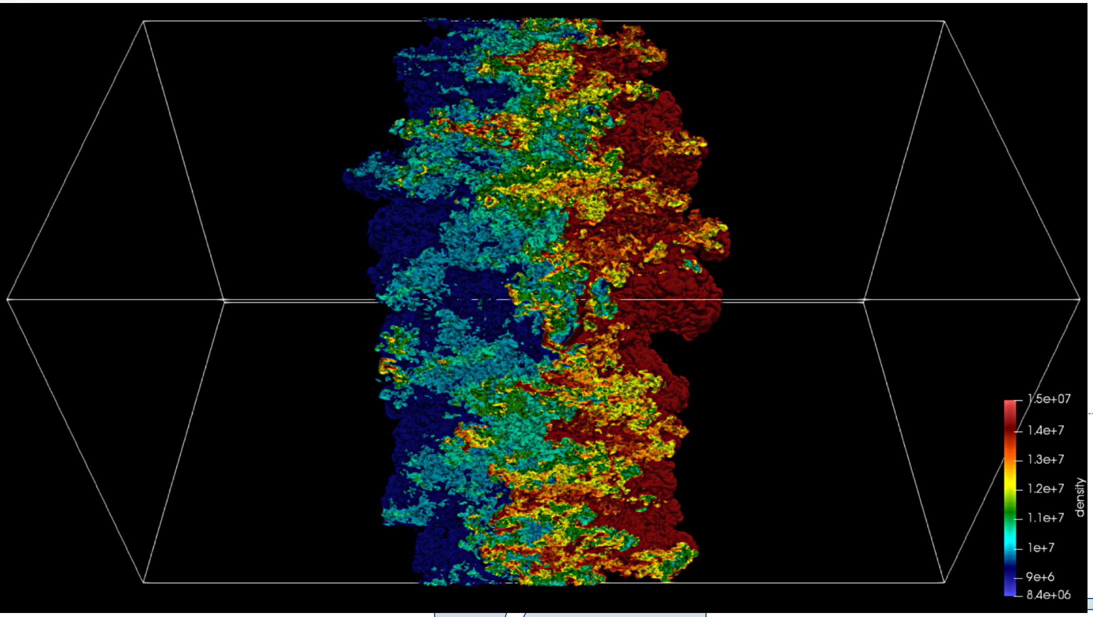
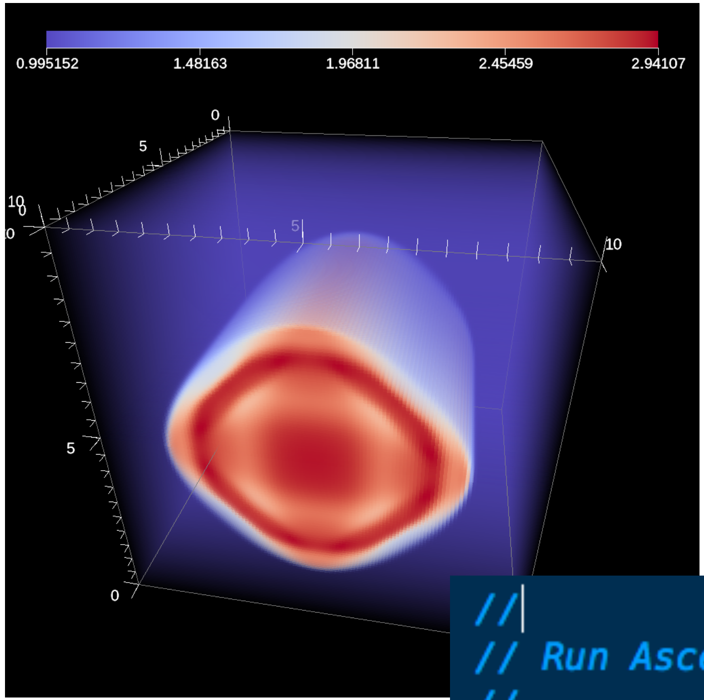
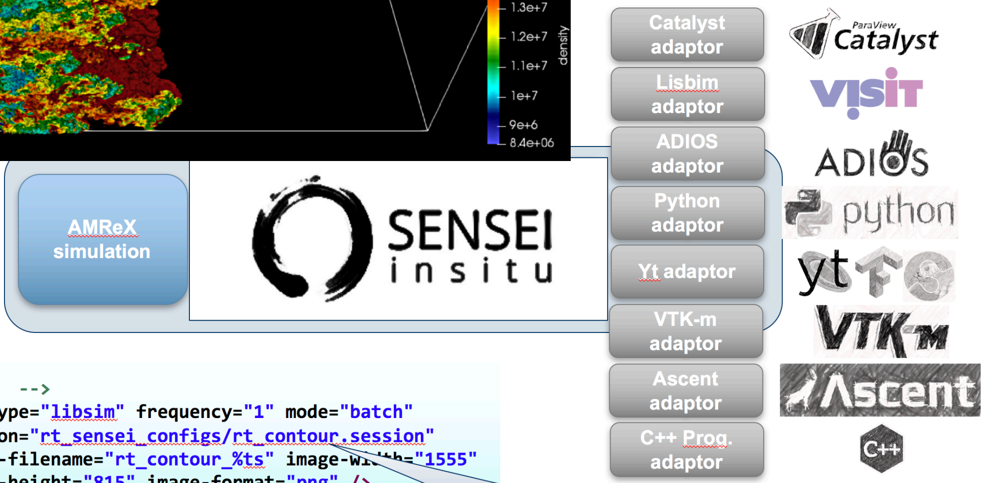
In Situ Analysis and Visualization with SENSEI and Ascent








```
<sensei>
  <!-- libsim  -->
  <analysis type="libsim" frequency="1" mode="batch"
        session="rt_sensei_configs/rt_contour.session"
        image-filename="rt_contour_%ts" image-width="1555"
        image-height="815" image-format="png" />
</sensei>
```

Session file created in VisIt GUI configures VisIt

```
//
// Run Ascent
//

Ascent ascent;
ascent.open();
ascent.publish(data);
ascent.execute(actions);
ascent.close();
```

Tutorial Schedule

30m	Introduction
30m	Software install
30m	SENSEI concepts
30m	Break
75m	SENSEI demonstrations
15m	Ascent overview
----	Lunch
40m	Ascent concepts
50m	Ascent demonstrations
30m	Break
20m	Cinema
20m	VTK-m
20m	Automatic visualization
20m	Customizable vis with Python
10m	Wrap up / discussion

LLNL-PRES-796577

Introduction to:

Ascent

Flyweight in-situ visualization and analysis for HPC simulations

Matt Larsen(LLNL), Cyrus Harrison (LLNL), Hank Childs (Univ of Oregon)

http://ascent-dav.org

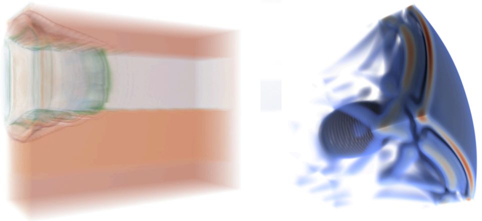
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

LLNL-PRES-796577

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 rendering 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)

Ascent



Visualizations created using Ascent

python HDF CINEMA

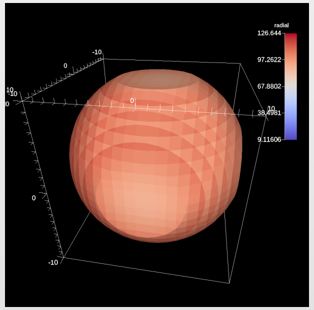
Extracts supported by Ascent

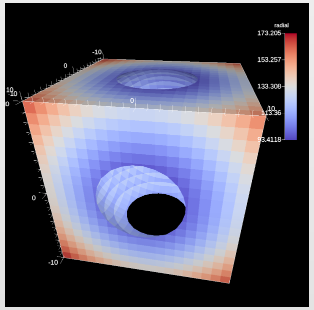
http://ascent-dav.org
https://github.com/Alpine-DAV/ascent

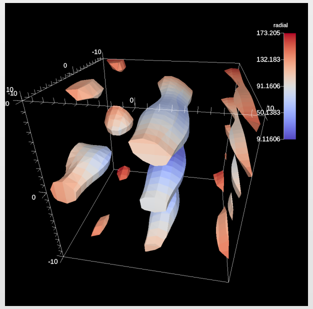
Website and GitHub Repo

SC19

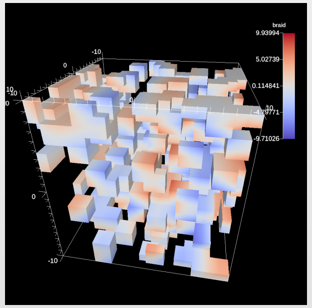
Ascent is ready for common visualization use cases



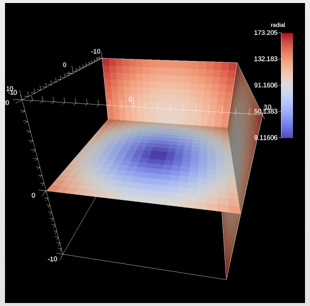




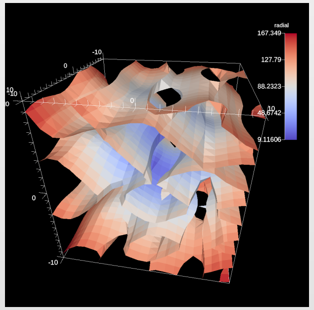
Iso-Volume



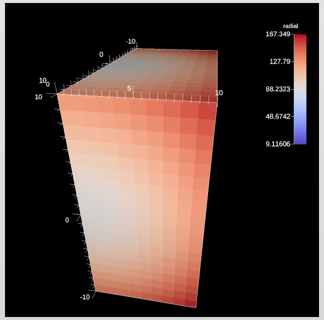
Threshold

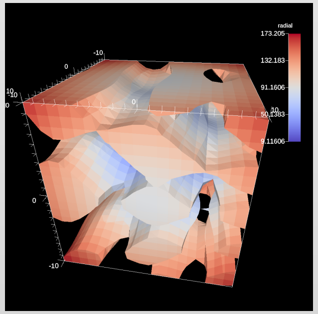


Slice



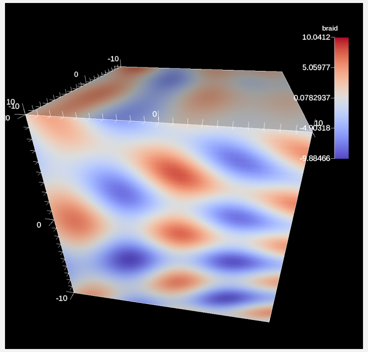
Contour



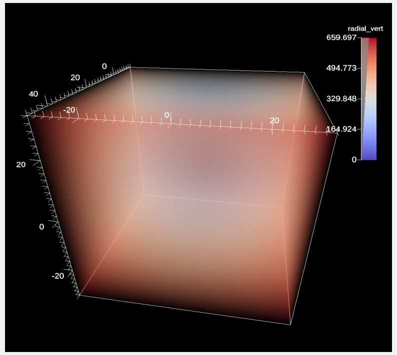


Clips

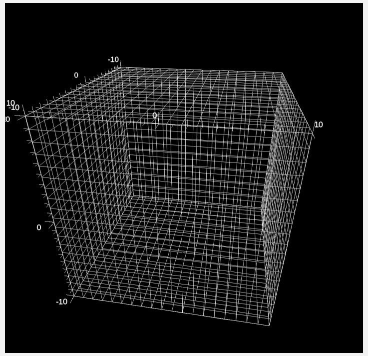
Rendering



Pseudocolor



Volume



Mesh

Ascent is developed as part of the ECP ALPINE (2.3.4.12) Software Technology Project

Scope & Intent	R&D Themes	Delivery Process	Target ECP Users	Support Model
Deliver in situ visualization and analysis algorithms and infrastructure.	1) Automated in situ massive data reduction **algorithms** 2) Portable, scalable, performant **infrastructure**	Regular releases of software and documentation, open access to production software from GitHub	All ECP applications. Focused delivery for co-design centers applications.	Ongoing developer support. Dedicated email, issue tracking portals, comprehensive web-based documentation, regular tutorials.

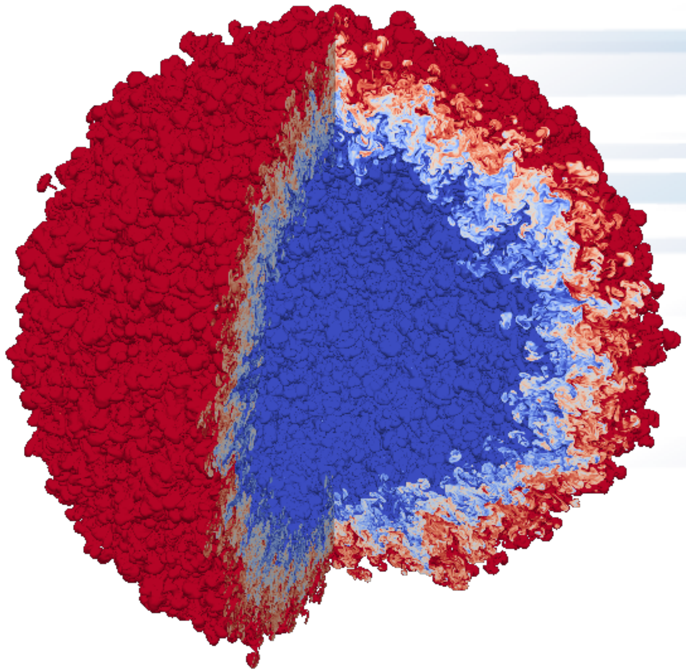
LLNL-PRES-796577

Ascent in situ success stories: Example 1

In October 2018, LLNL used Ascent in a massive turbulent fluid mixing simulation run on Sierra using over 16,000 GPUs

Highlights:

- The ***97.8 billion*** element simulation ran across ***16,384 GPUs*** on ***4,096 Nodes***
- The simulation application used **CUDA** via **RAJA** to run on the GPUs
- Time-varying evolution of the mixing was visualized in-situ using **Ascent**, also leveraging 16,384 GPUs
- Ascent leveraged **VTK-m** to run visualization algorithms on the GPUs
- The last time step was exported to the parallel file system for detailed post-hoc visualization using **VisIt**

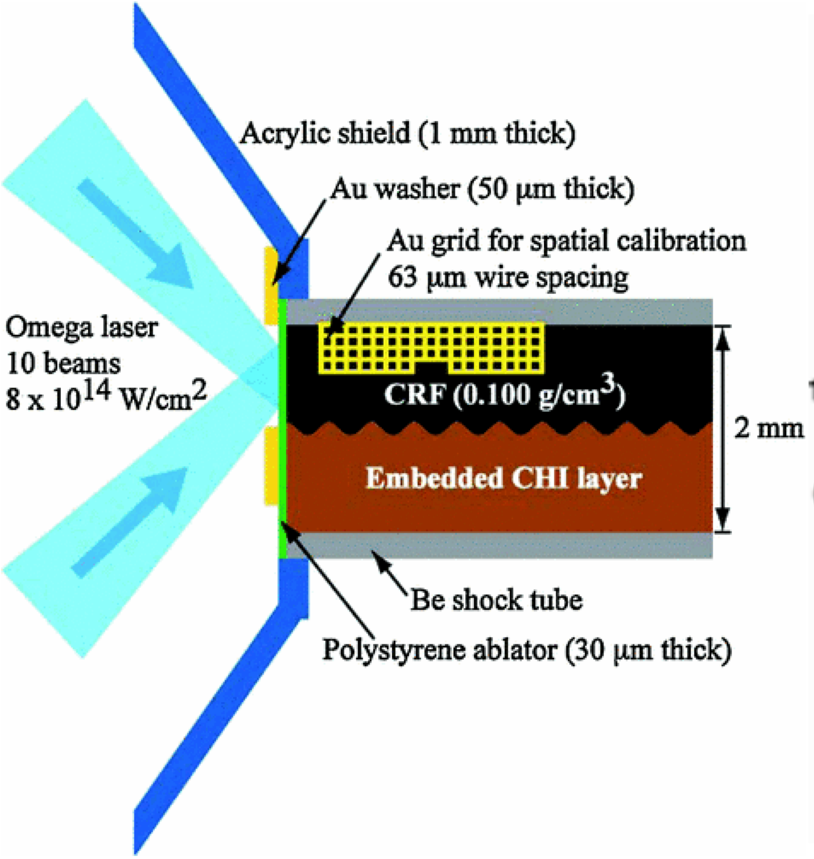


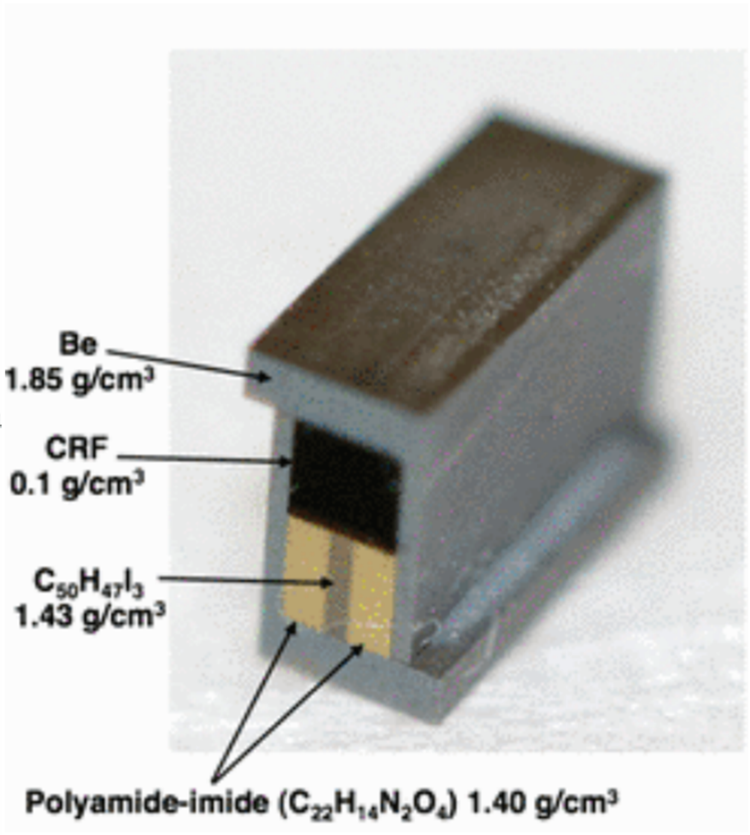
Ascent in situ rendering at Scale:
Visualization of an idealized Inertial Confinement Fusion (ICF) simulation of Rayleigh-Taylor instability with two fluids mixing in a spherical geometry.

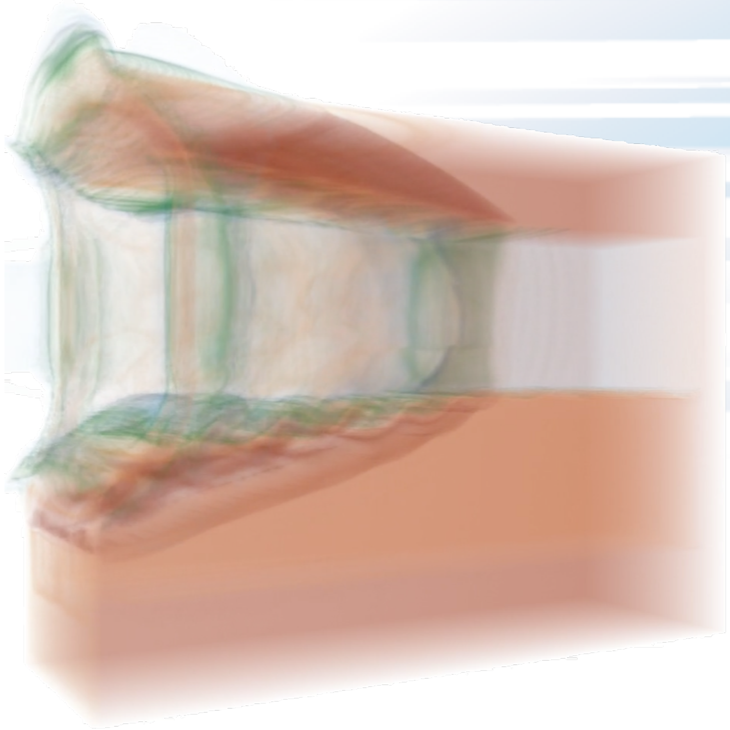
LLNL-PRES-796577

Ascent in situ success stories: Example 2

Ascent visualized and exported data from a blast-wave driven Kelvin-Helmholtz simulation (big laser, tiny box simulation)







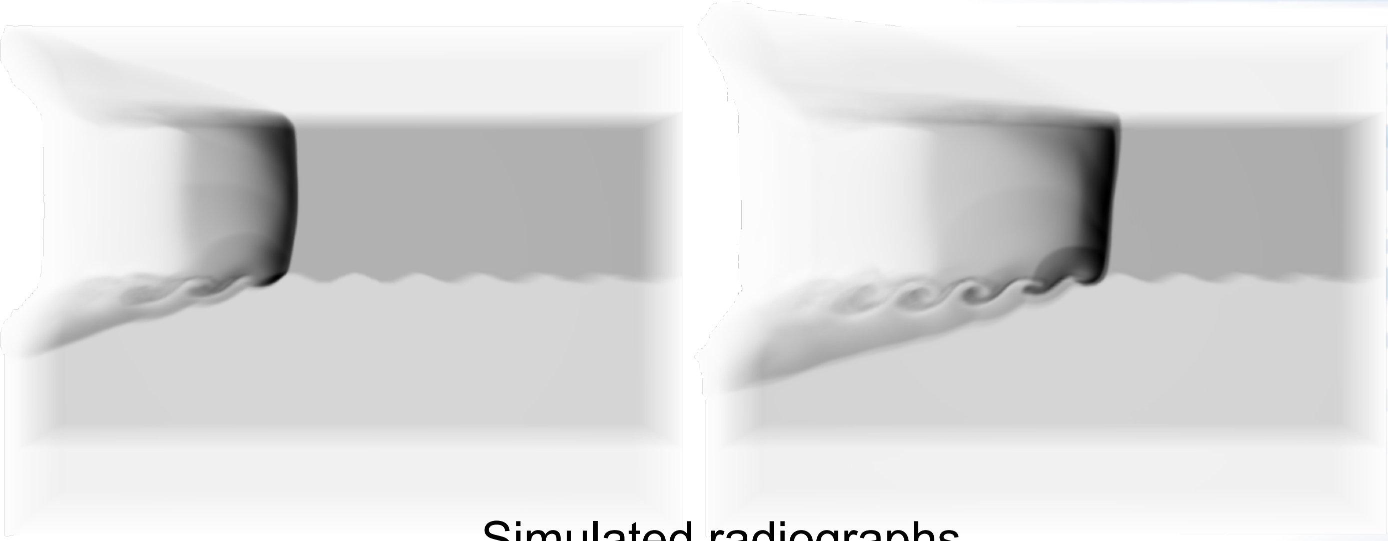
*Hurricane, O. A., et al. "Blast-wave driven Kelvin-Helmholtz shear layers in a laser driven high-energy-density plasma." *Astrophysics and Space Science* 336.1 (2011): 139-143.

Ascent created simulated radiographs to compare the simulation against experimental results

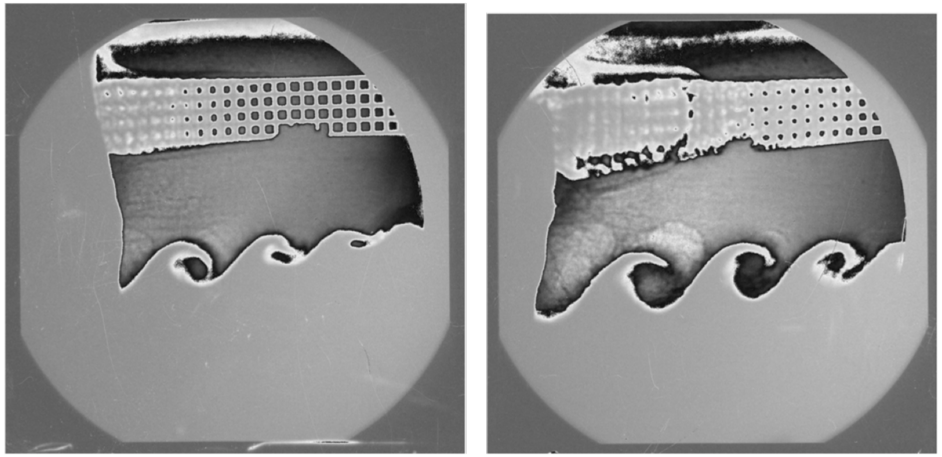
In-situ radiography of a laser driven Kelvin-Helmholtz instability using ROVER radiography via Ascent

Simulation Details:

- *2.3K MPI tasks*
- *120 hours wall time*
- *3.5M 3D Q2 elements, 100M quad points*
- *~20K RK2 timesteps*



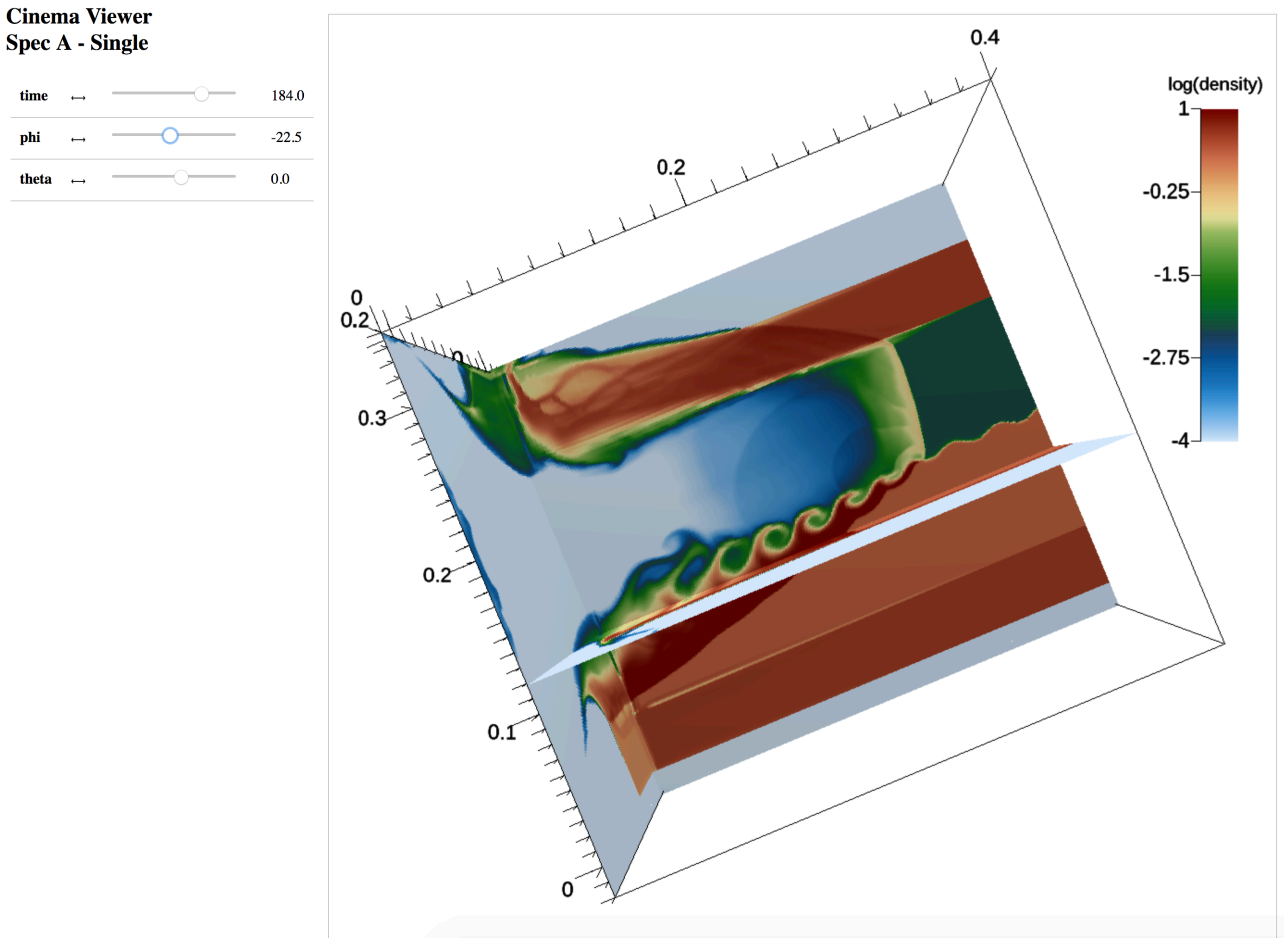
Simulated radiographs



Experimental radiographs

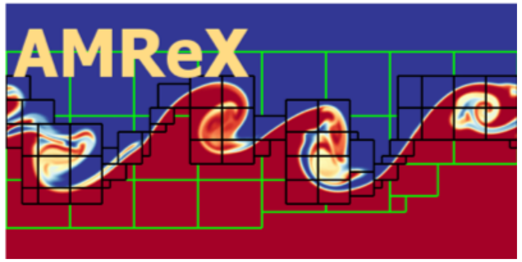
Ascent in situ success stories: Example 2 (cont.)

We created a Cinema Image Database from this simulation that you can view



- http://portal.nersc.gov/project/visit/larsen/cinema/rad_kh/cinema.html
- https://bit.ly/2VUOyYE

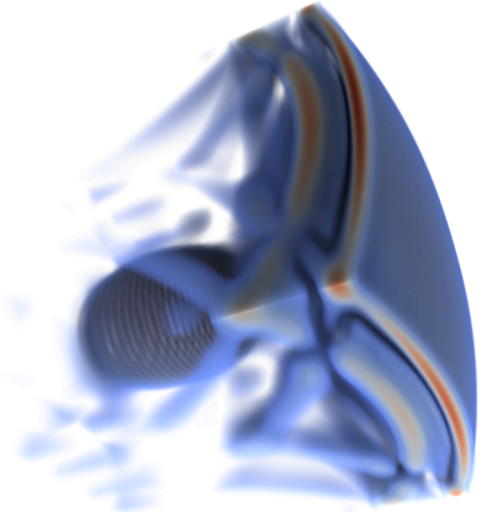
We are working to provide ECP Co-Design Centers easy paths to publish simulation mesh data to Ascent

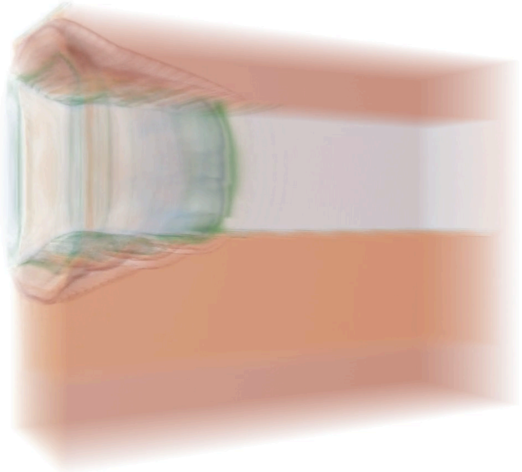


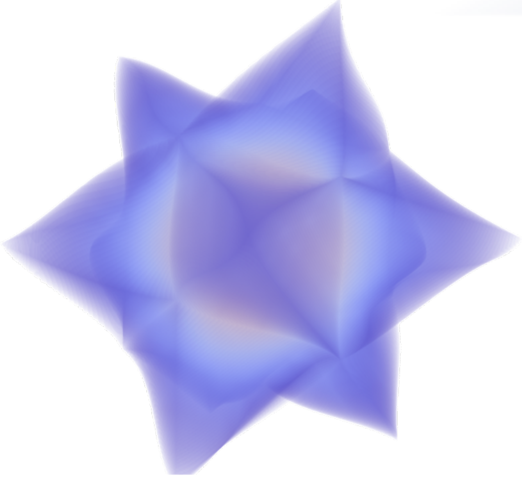
CEED
EXASCALE DISCRETIZATIONS

We are developing AMReX functions to wrap AMR Grids and Particle Containers for use in Ascent

MFEM includes Conduit support which wraps MFEM High-order meshes for use in Ascent



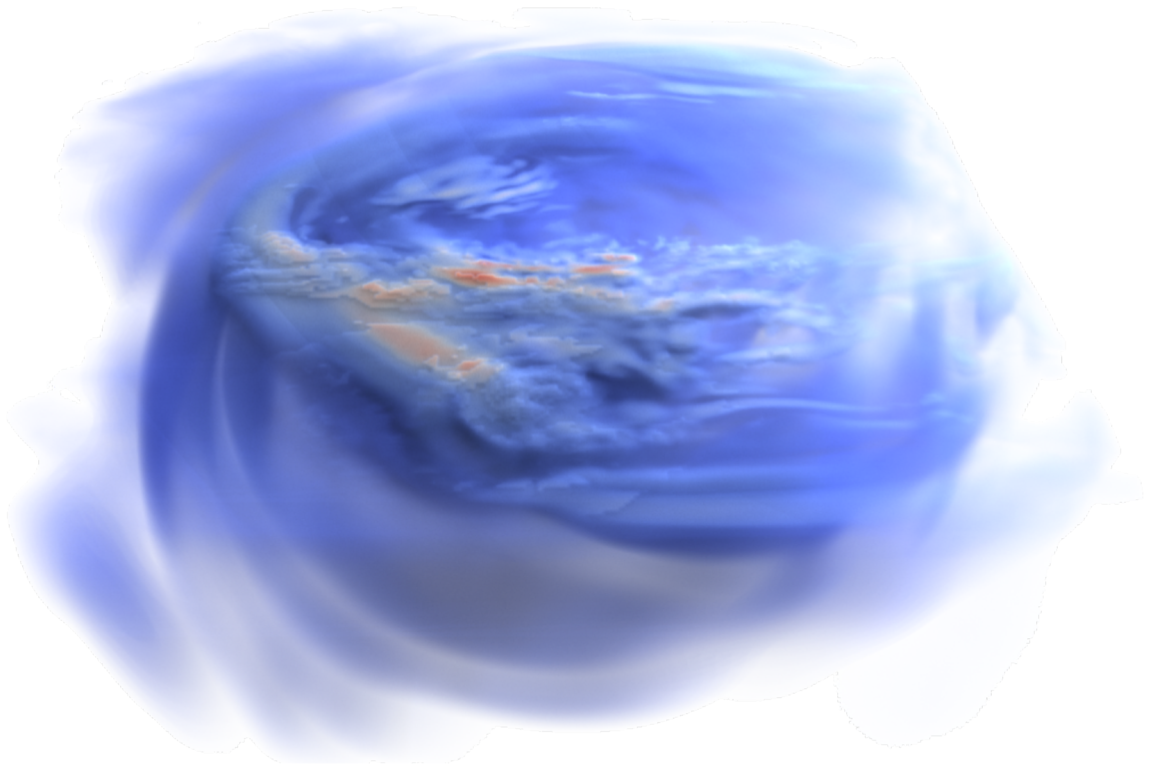


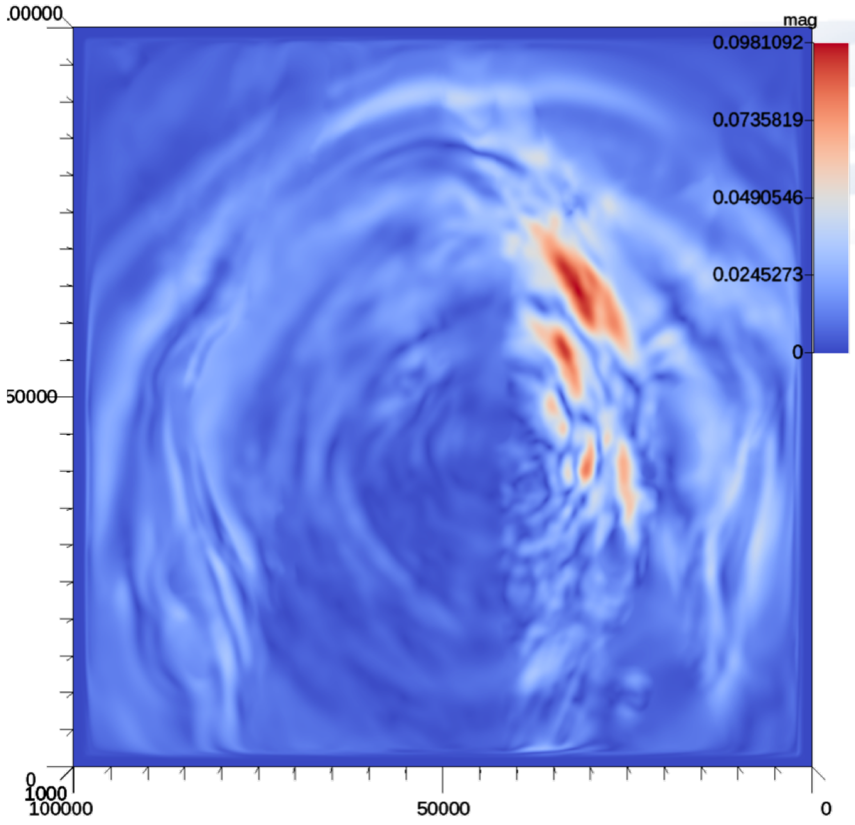


AMReX App Integrations: ***Nyx, WarpX, PeleC***

App Integrations: ***MARBL***

We are also working with the EQSIM:SW4 ECP seismology application





https://github.com/geodynamics/sw4

SC19

Ascent supports multiple languages and output types

Language Bindings:

C/C++

python™

Fortran

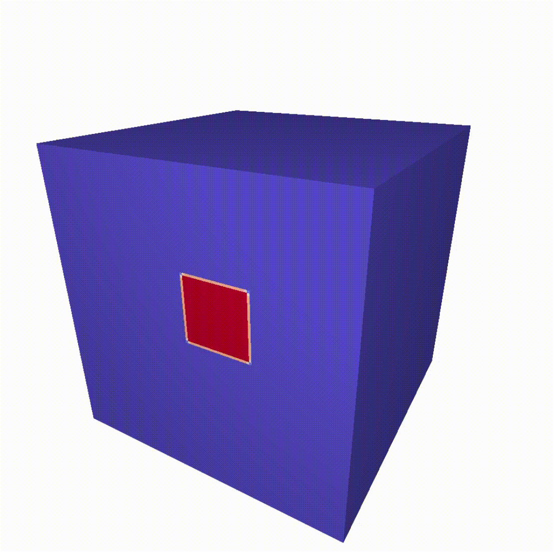
Output Types:

HDF

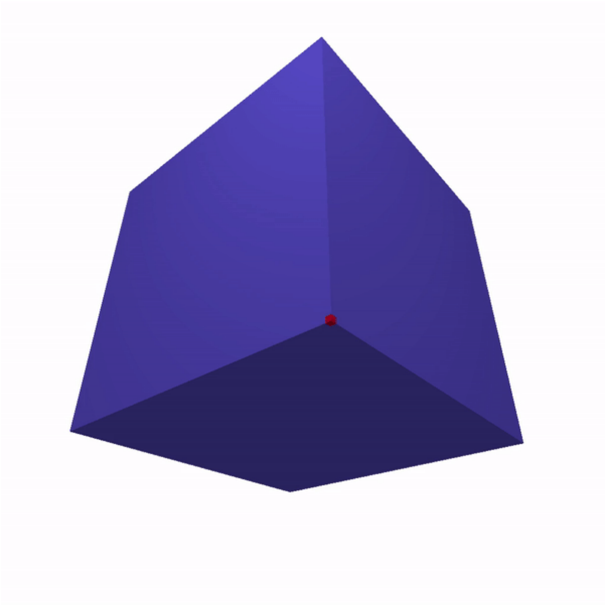
ADIOS

CINEMA

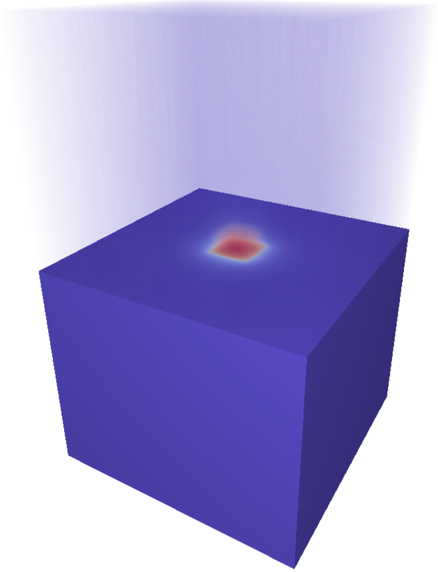
Ascent provides example integrations with built-in proxy simulations that also serve as data sources



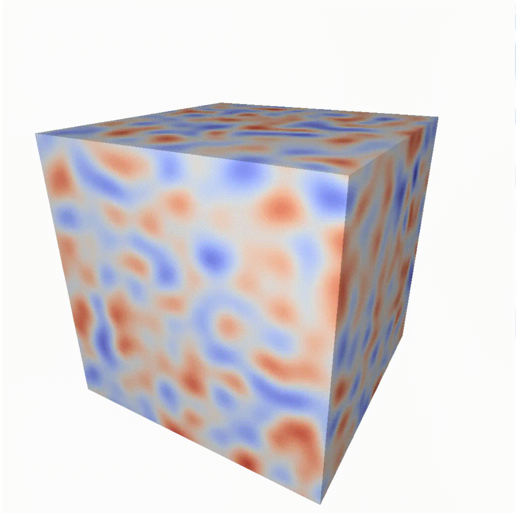
Cloverleaf3D



Lulesh

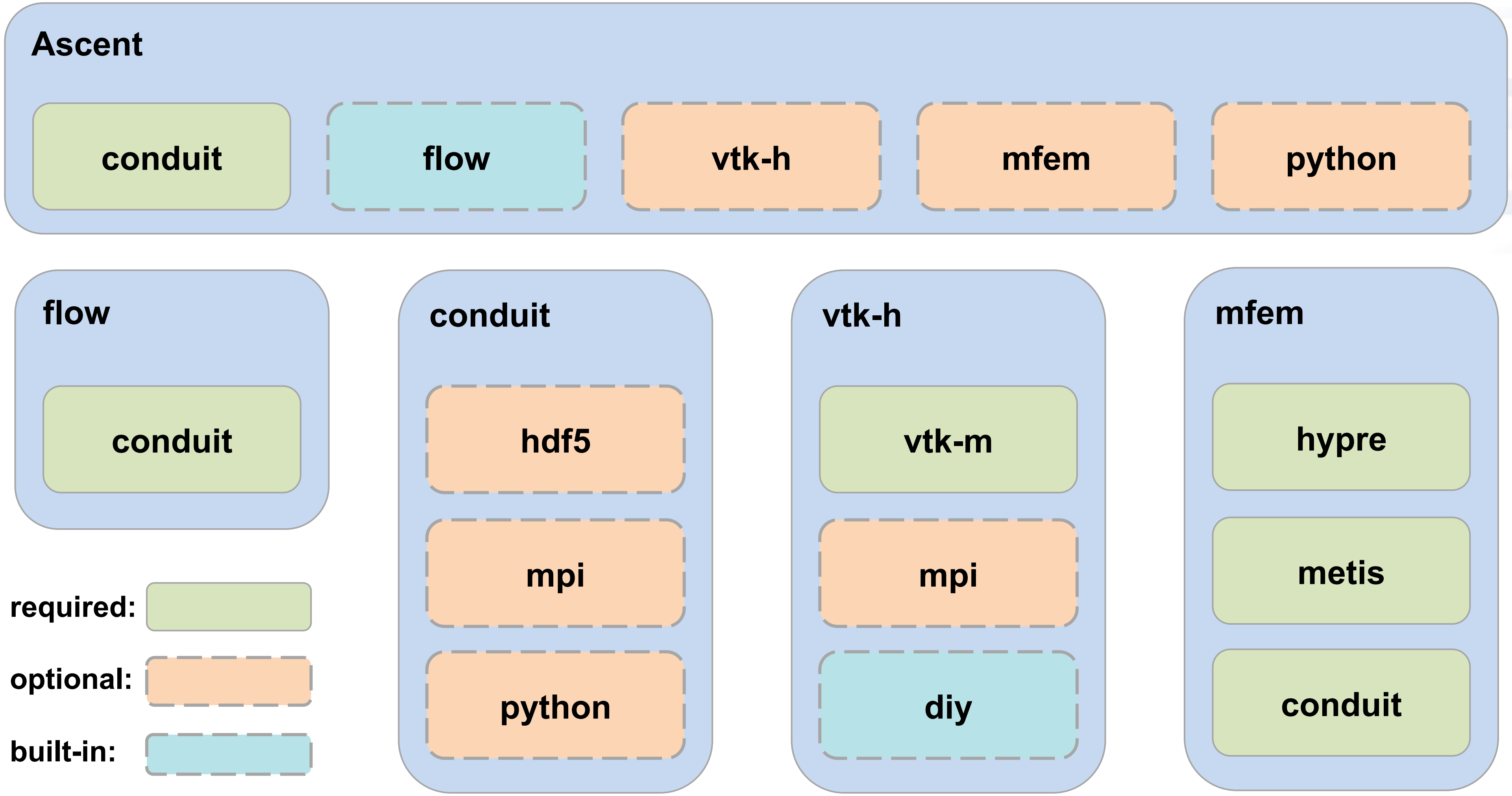


Kripke



Smooth Noise

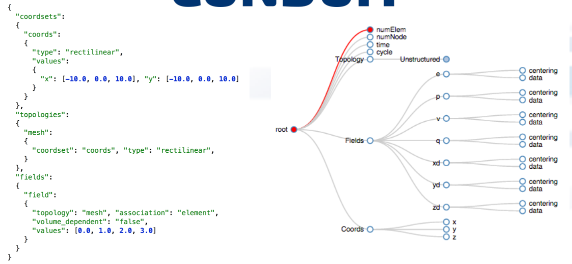
Ascent software stack



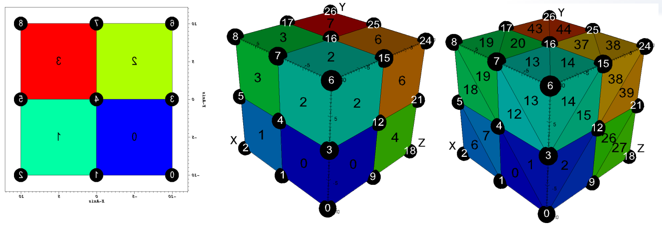
Ascent heavily leverages Conduit, which provides intuitive APIs for in-memory data description and exchange

- **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

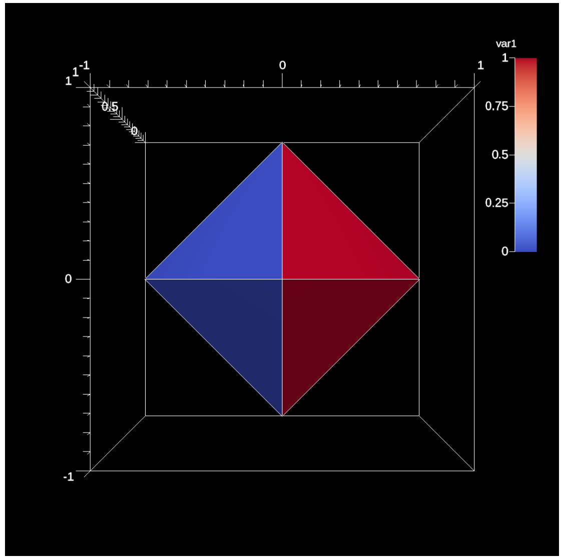
LLNL-PRES-796577

Ascent's API uses Conduit to provide a flexible interface

SC19

Visualization actions are specified using Conduit Trees (C,C++,Python, Fortran) or equivalent YAML files

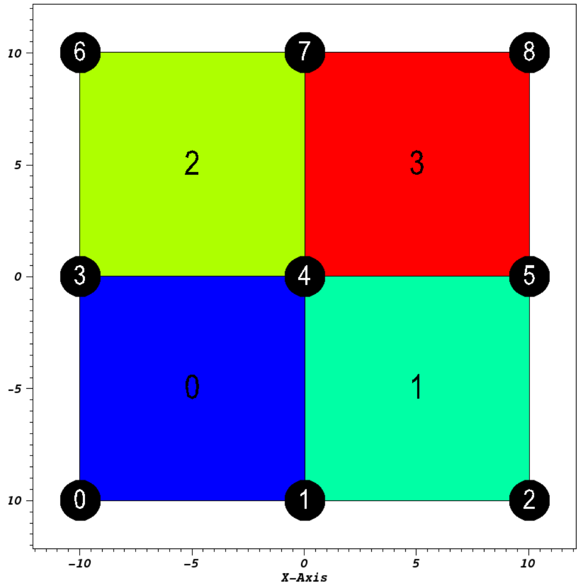
```cpp
// Use Ascent to render multiple plots to a single image

Ascent a;

// open ascent
a.open();

// publish mesh to ascent
a.publish(mesh);

// setup actions
Node actions;
Node &add_act = actions.append();
add_act["action"] = "add_scenes";

// declare a scene to render the dataset
Node & scenes = add_act["scenes"];
// add a pseudocolor plot (`p1`)
scenes["s1/plots/p1/type"] = "pseudocolor";
scenes["s1/plots/p1/field"] = "var1";
// add a mesh plot (`p1`)
// (experiment with commenting this out)
scenes["s1/plots/p2/type"] = "mesh";
scenes["s1/image_prefix"] = "out_scene_ex2_render_two_plots";

// print our full actions tree
std::cout << actions.to_yaml() << std::endl;

// execute the actions
a.execute(actions);

a.close();
```



C++ Rendering Actions Example and Result

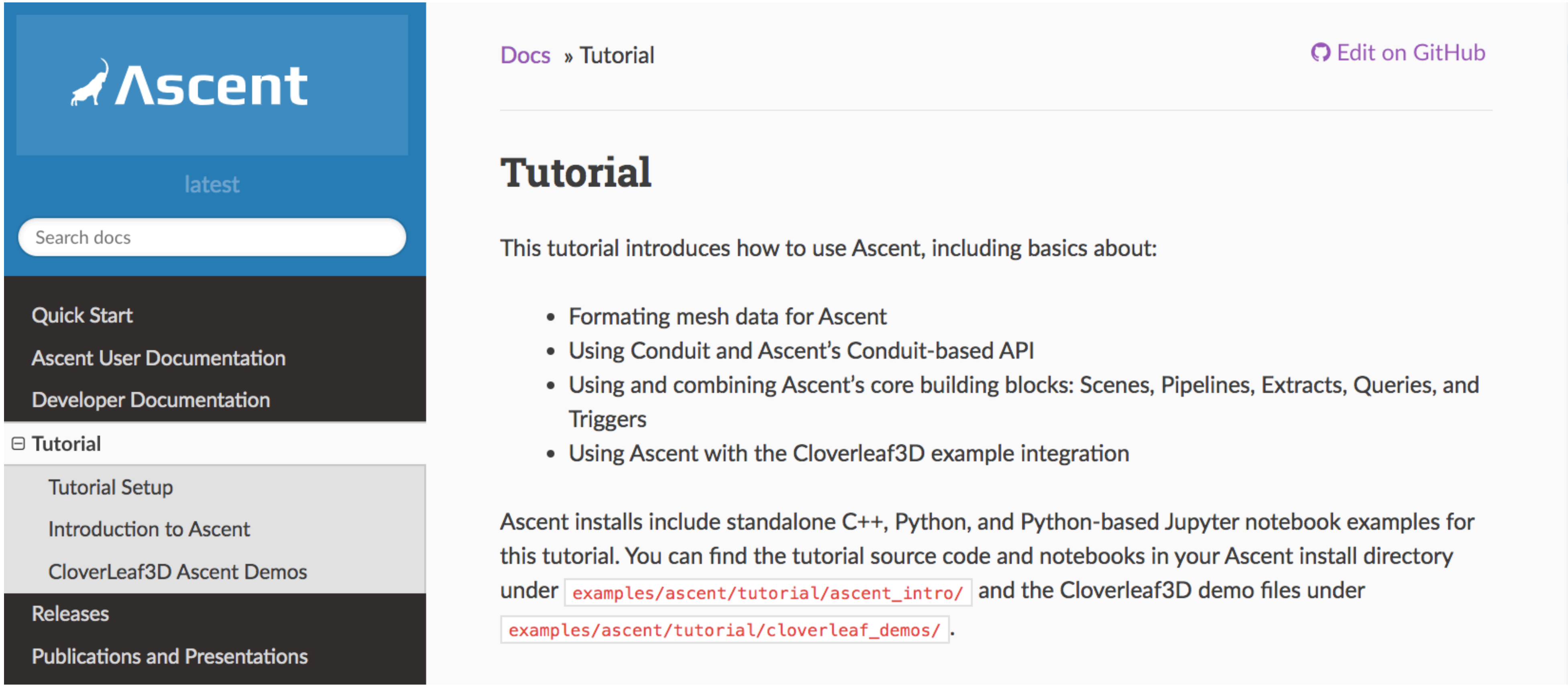
Mesh data is published to Ascent using Conduit's Mesh Blueprint conventions

```json
{
  "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]
    }
  }
}
```



Example Rectilinear Mesh Blueprint Tree and Visualization

Ascent tutorial examples are outlined in our documentation and included ready to run in Ascent installs



Ascent

latest

Search docs

- Quick Start
- Ascent User Documentation
- Developer Documentation
- ⊟ Tutorial
  - Tutorial Setup
  - Introduction to Ascent
  - CloverLeaf3D Ascent Demos
- Releases
- Publications and Presentations

Docs » Tutorial

Edit on GitHub

Tutorial

This tutorial introduces how to use Ascent, including basics about:

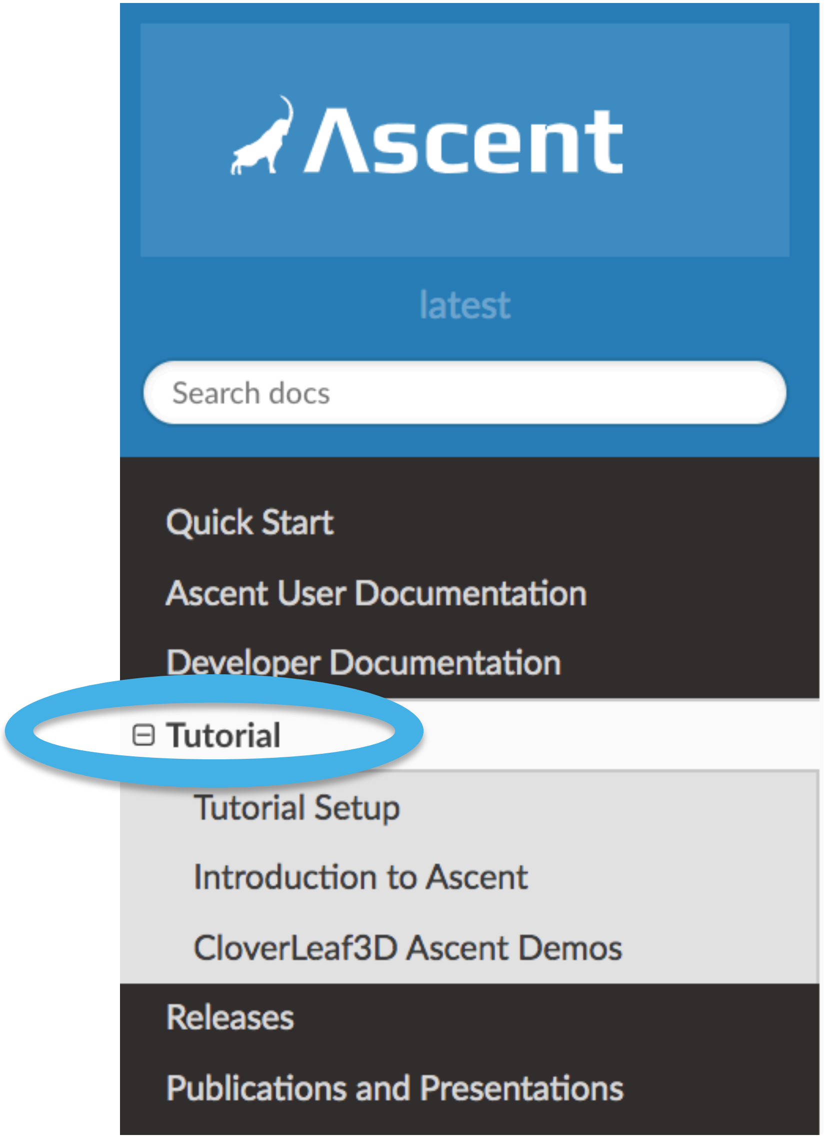
- Formating mesh data for Ascent
- Using Conduit and Ascent's Conduit-based API
- Using and combining Ascent's core building blocks: Scenes, Pipelines, Extracts, Queries, and Triggers
- Using Ascent with the Cloverleaf3D example integration

Ascent installs include standalone C++, Python, and Python-based Jupyter notebook examples for this tutorial. You can find the tutorial source code and notebooks in your Ascent install directory under `examples/ascent/tutorial/ascent_intro/` and the Cloverleaf3D demo files under `examples/ascent/tutorial/cloverleaf_demos/`.

http://ascent-dav.org

Ascent tutorial examples are outlined in our documentation and included ready to run in Ascent installs

- http://ascent-dav.org
- Click on “Tutorial”



Tutorial Schedule

30m	• Introduction
30m	• Software install
30m	• SENSEI concepts
30m	• Break
75m	• SENSEI demonstrations
15m	• Ascent overview
----	• Lunch

40m	• Ascent concepts
50m	• Ascent demonstrations
30m	• Break
20m	• Cinema
20m	• VTK-m
20m	• Automatic visualization
20m	• Customizable vis with Python
10m	• Wrap up / discussion

LLNL-PRES-796577

Tutorial Schedule

30m	• Introduction
30m	• Software install
30m	• SENSEI concepts
30m	• Break
75m	• SENSEI demonstrations
15m	• Ascent overview
----	• Lunch

40m	• Ascent concepts
50m	• Ascent demonstrations
30m	• Break
20m	• Cinema
20m	• VTK-m
20m	• Automatic visualization
20m	• Customizable vis with Python
10m	• Wrap up / discussion

LLNL-PRES-796577

Ascent's interface provides five composable building blocks

Scenes
(Render Pictures)

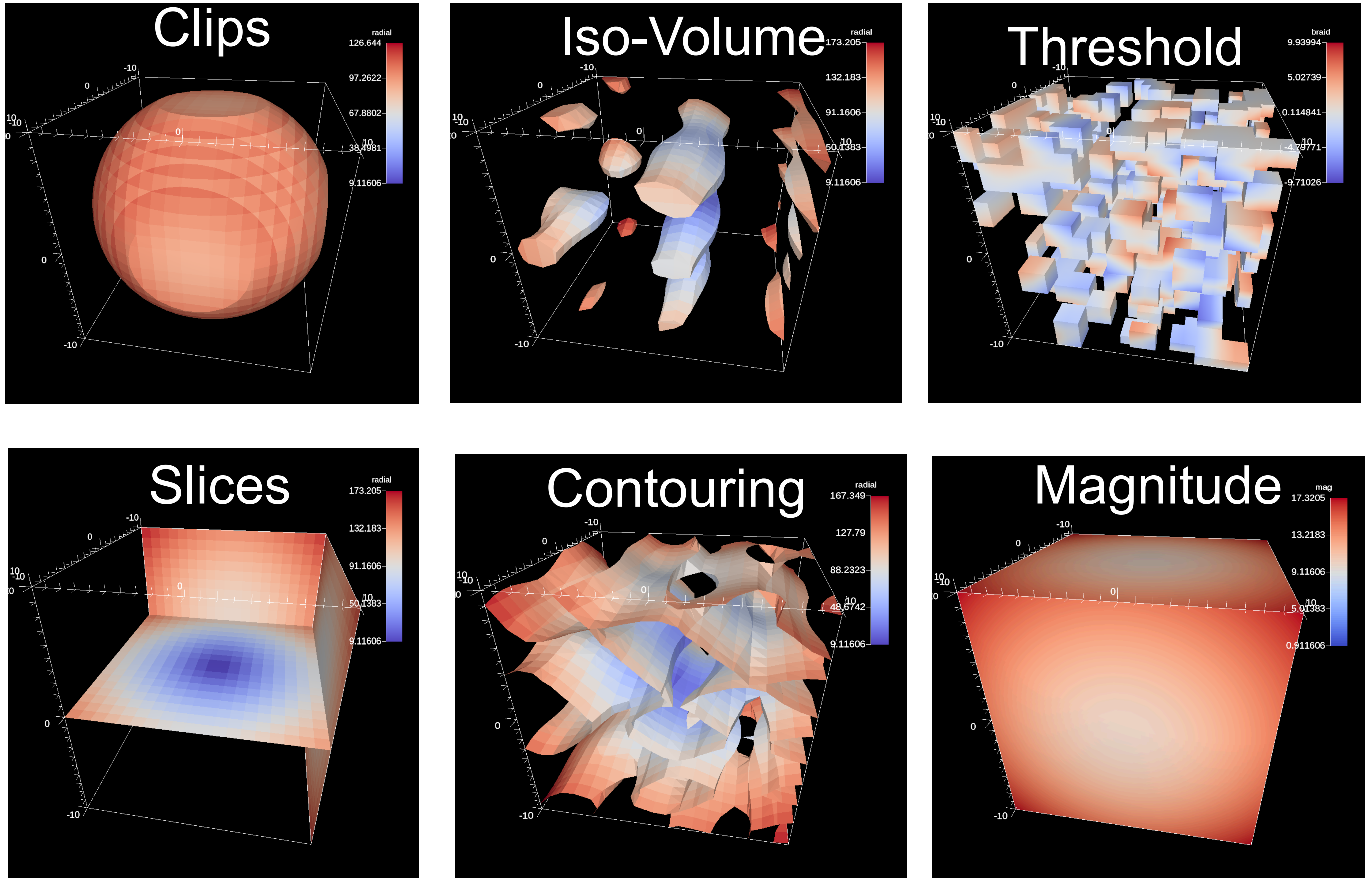
Pipelines
(Transform Data)

Extracts
(Capture Data)

Queries
(Ask Questions)

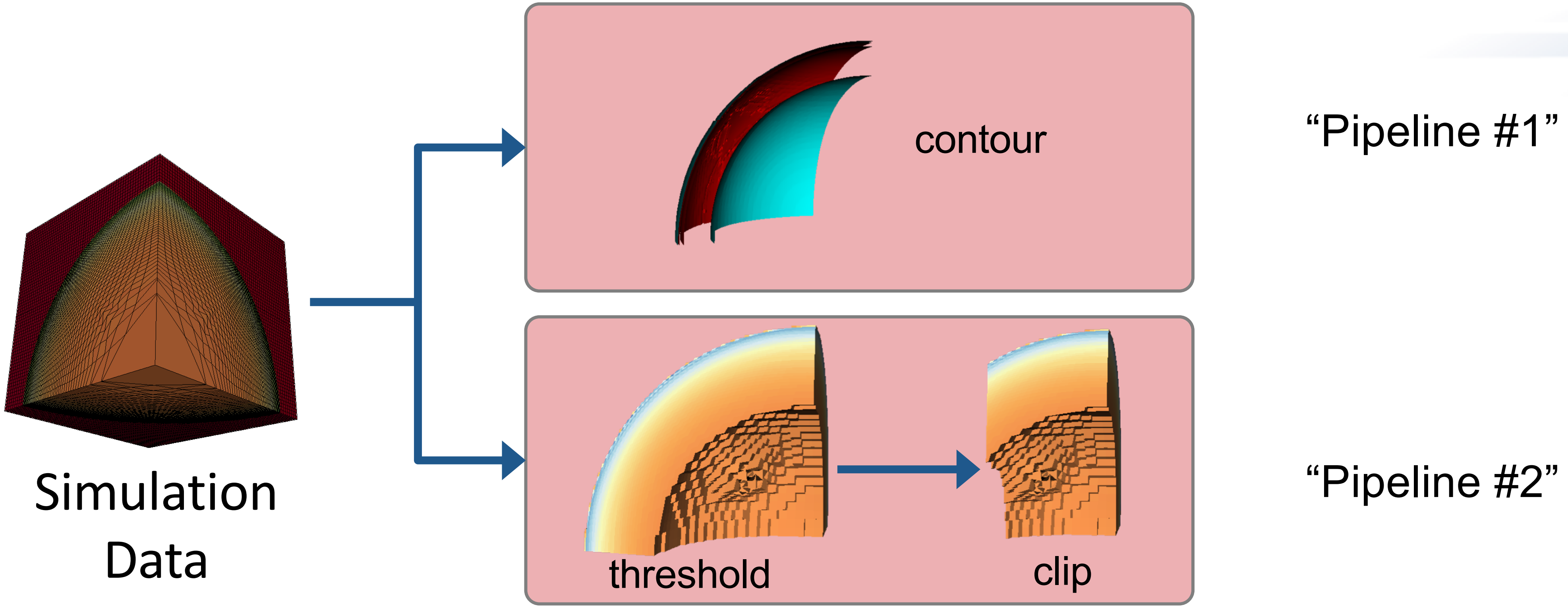
Triggers
(Adapt Actions)

Pipelines: transform data



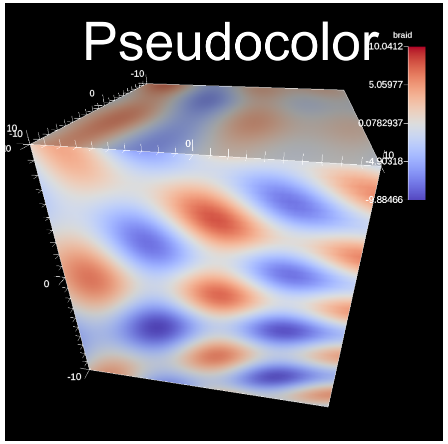
A pipeline is a series data transformations (i.e., filters)

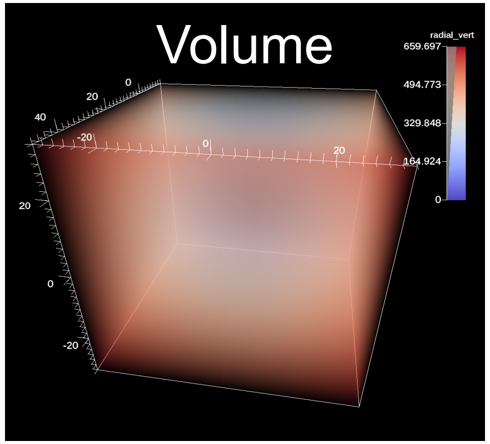
- Ascent allows an arbitrary number of pipelines to be described

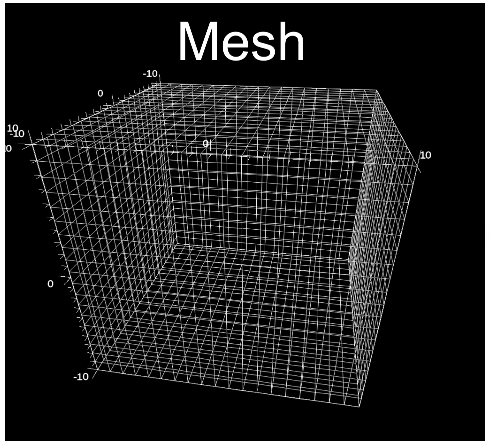


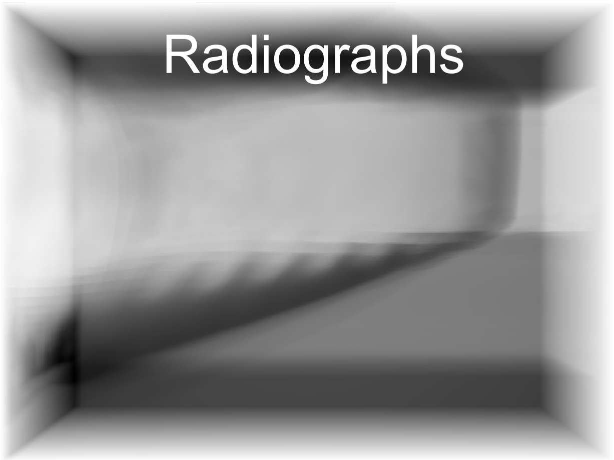
LLNL-PRES-796577

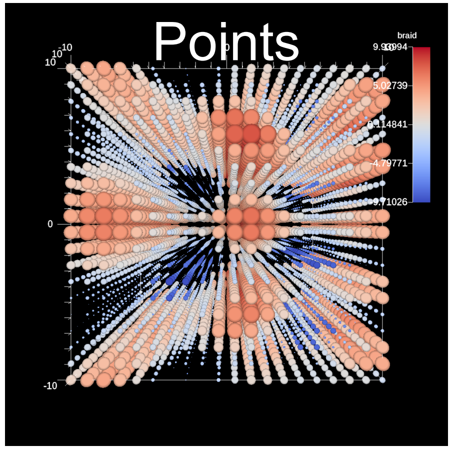
Scenes: render pictures





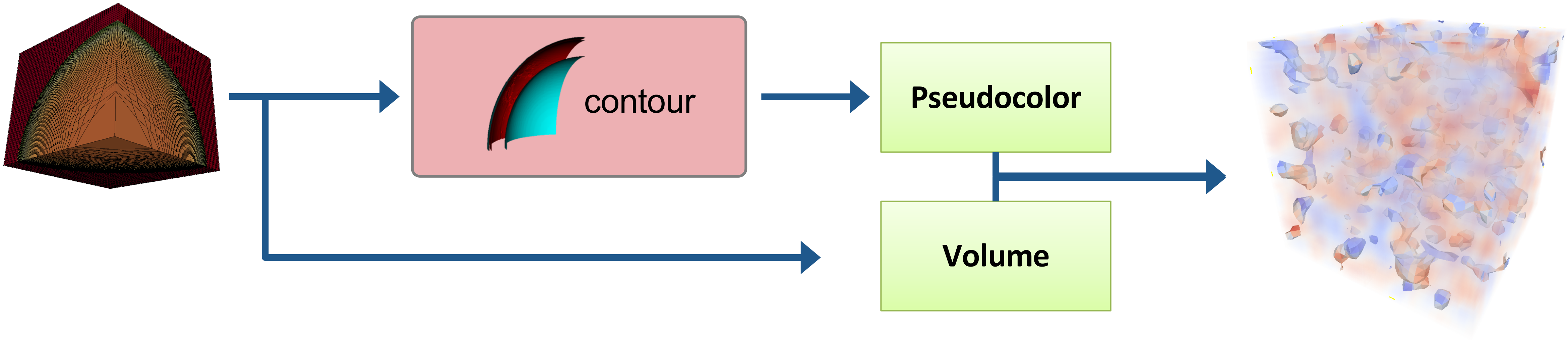






A scene is a way to render pictures

- Contains a list of plots
  - E.g., volume, pseudocolor, and mesh
- Contains a list of camera parameters



Extracts: capture data

CINEMA

HDF

python™

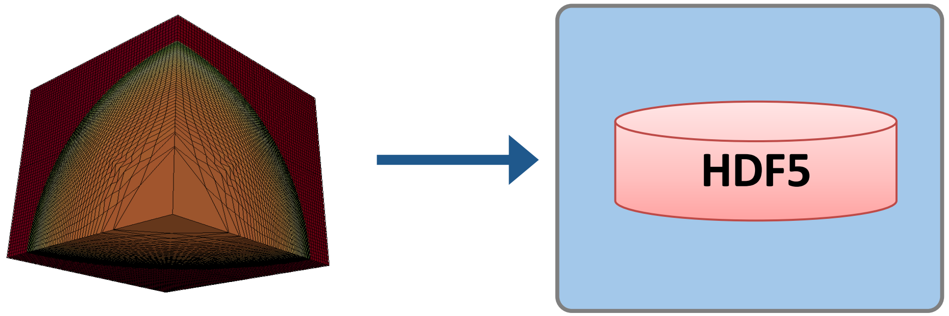
ADIOS

jupyter

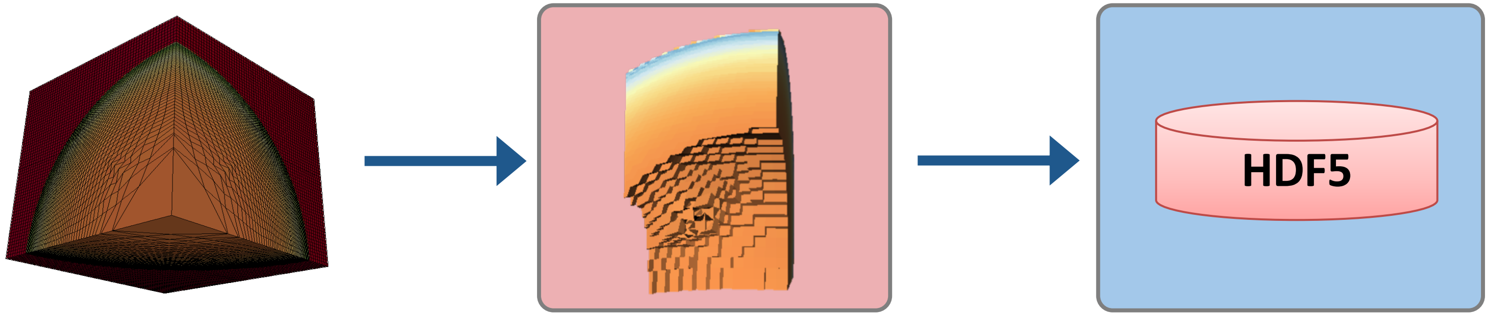
ParaView Catalyst

An extract captures data for use outside of Ascent

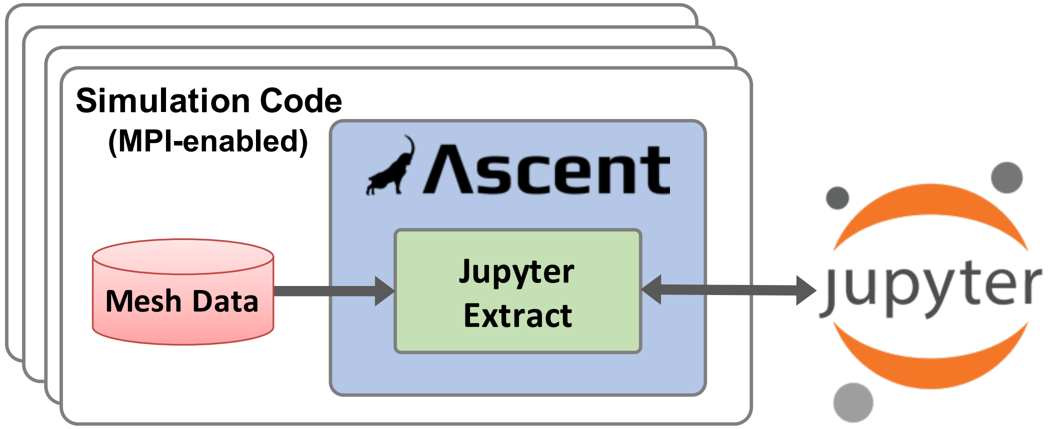
- Examples:
  - Export published simulation data to HDF5, Python environment, etc



  - Export pipeline *results* to HDF5, Python environment, etc.



With Ascent's Jupyter Extract, users of any simulation code with Ascent integrated can run Jupyter Notebooks and use Python to interact with in-memory data



File Edit View Insert Cell Kernel Widgets Help Trusted Bridge Kernel

In [42]:
```
# get this MPI task's published blueprint data
mesh_data = ascent_data().child(0)

# fetch the numpy array for the energy field values
e_vals = mesh_data["fields/energy/values"]
```

Get Sim Data from Ascent

In [43]:
```
compute_local_data()
```

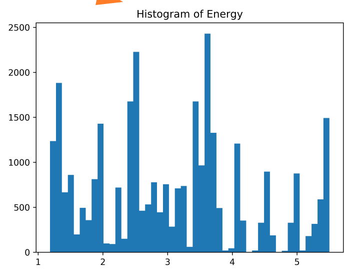
In [44]:
```
# obtain a mpi4py mpi comm object
comm = MPI.Comm.f2py(ascent_mpi_comm_id())

# sum histogram counts with MPI to get final histogram
comm.Allreduce(hist, hist_all, op=MPI.SUM)
```

Use MPI to analyze in parallel

In [45]:
```
# print result on mpi task 0
if comm.Get_rank() == 0:
    print_results()
```

Print Results and Plot Graphs

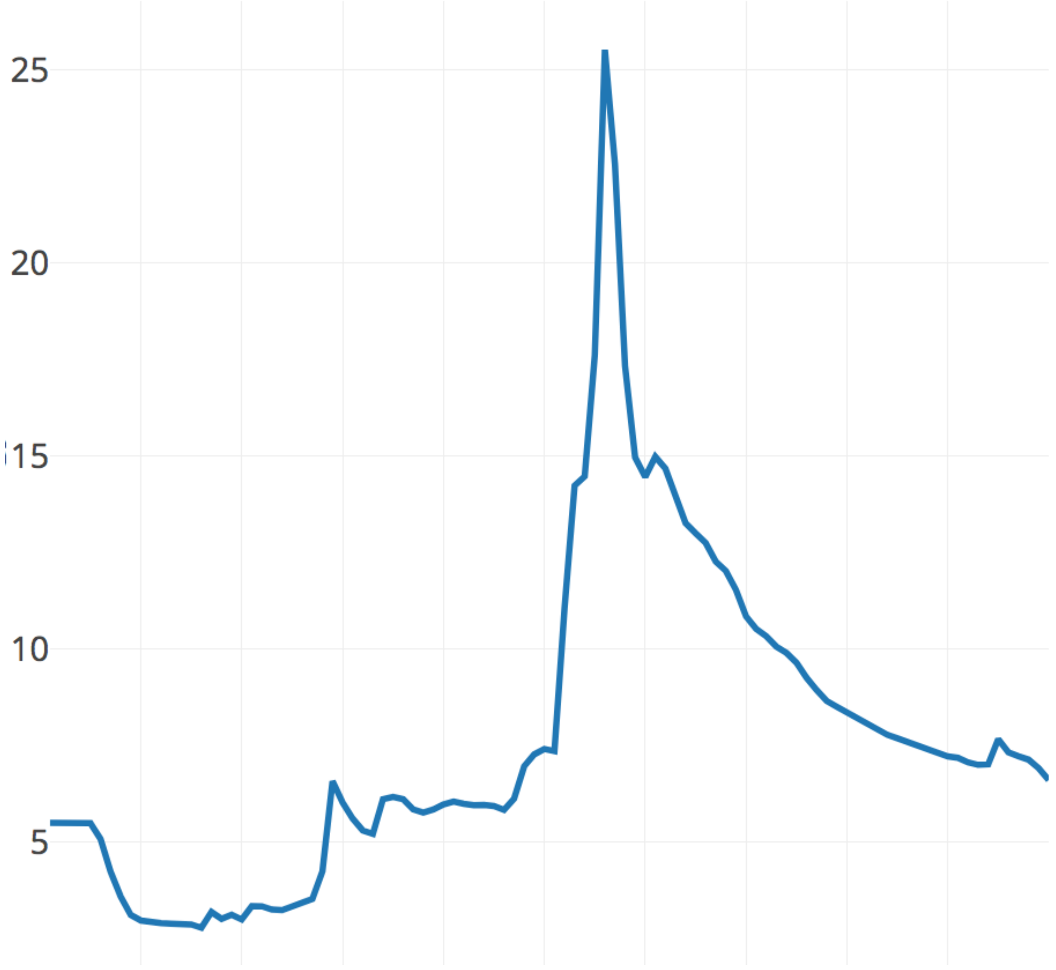
```
Histogram of Energy:

Counts:
[281645   1357   1236   1882    666    860    198    494    357    813
   1429     98     91    719    150   1677   2228    462    532    778
    444    756    284    710    737     60   1677    966   2430   1328
    492     20     44   1208    352      4     20    328    896    188
      0     16    328    876     20    180    316    588   1492]

Bin Edges:
[1.         1.09183673 1.18367347 1.2755102  1.36734694 1.45918367
 1.55102041 1.64285714 1.73469388 1.82653061 1.91836735 2.01020408
 2.10204082 2.19387755 2.28571429 2.37755102 2.46938776 2.56122449
 2.65306122 2.74489796 2.83673469 2.92857143 3.02040816 3.1122449
 3.20408163 3.29591837 3.3877551  3.47959184 3.57142857 3.66326531
 3.75510204 3.84693878 3.93877551 4.03061224 4.12244898 4.21428571
 4.30612245 4.39795918 4.48979592 4.58163265 4.67346939 4.76530612
 4.85714286 4.94897959 5.04081633 5.13265306 5.2244898  5.31632653
 5.40816327 5.5       ]
```



LLNL-PRES-796577

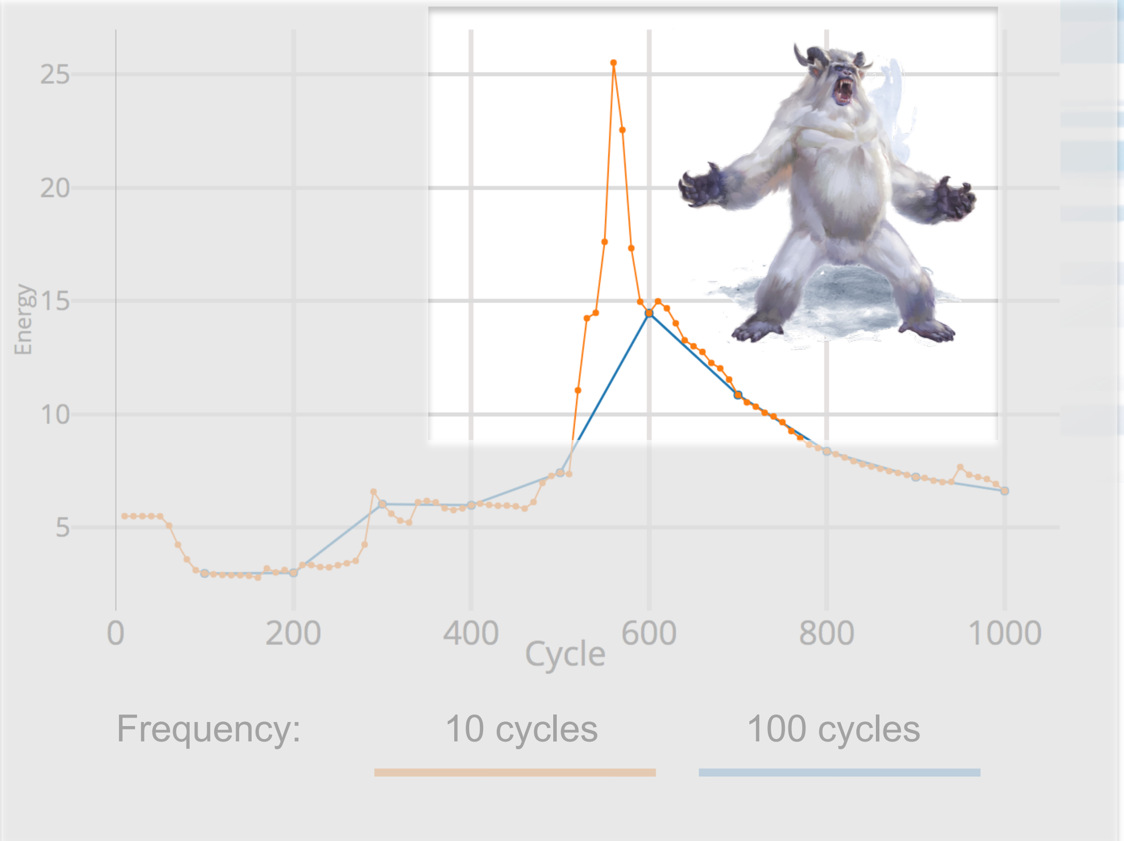
Queries: ask questions, get answers

- Composable expressions that reduce or summarize mesh data
- Query Examples:
  - max(field('pressure'))
  - location(min(field('energy')))
  - histogram(field('viscosity'), num_bins=100)
- Getting answers
  - Query results are named and available to the simulation
  - Named queries can be used to build more complex expressions

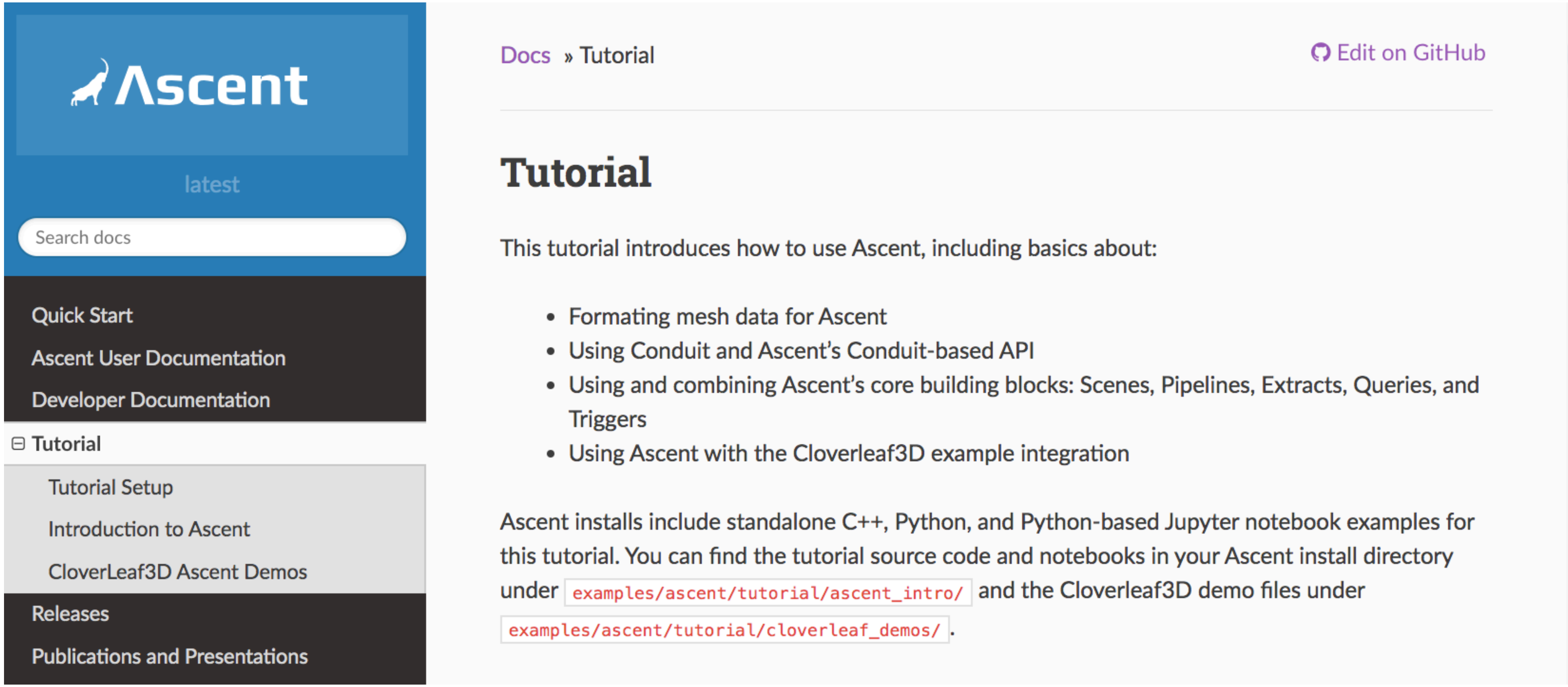


Triggers: adapt actions

- Provide in situ control to use vis and analysis features when most useful
- If X (*condition*) then do Y (*actions)*
- Trigger Examples:
  - max(field('pressure')) > 100
  - magnitude(location(min(field('energy'))) < 3.14
  - cycle() > 10 && cycle() < 20
- Can use named query results
- Actions are anything you can do in Ascent



Ascent tutorial examples are outlined in our documentation and are included ready to run in Ascent installs



Ascent

latest

Search docs

Quick Start
Ascent User Documentation
Developer Documentation
⊟ Tutorial
Tutorial Setup
Introduction to Ascent
CloverLeaf3D Ascent Demos
Releases
Publications and Presentations

Docs » Tutorial

Edit on GitHub

Tutorial

This tutorial introduces how to use Ascent, including basics about:

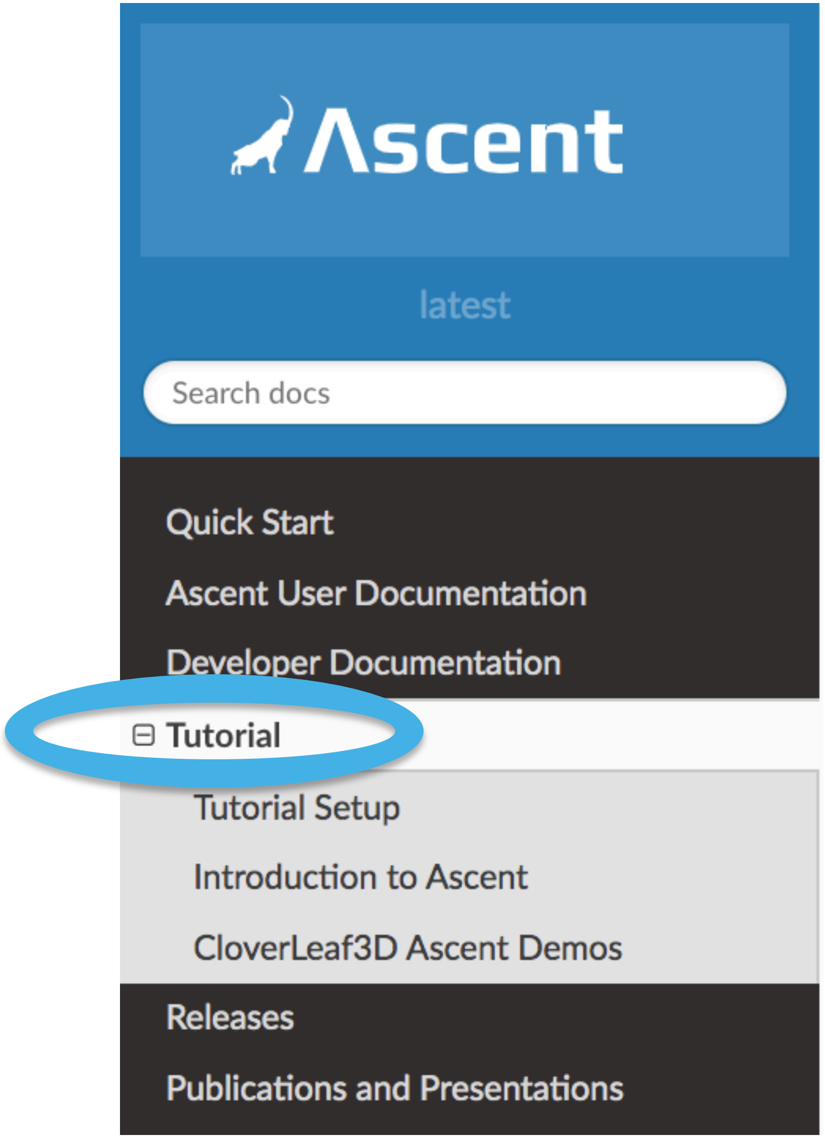
- Formating mesh data for Ascent
- Using Conduit and Ascent's Conduit-based API
- Using and combining Ascent's core building blocks: Scenes, Pipelines, Extracts, Queries, and Triggers
- Using Ascent with the Cloverleaf3D example integration

Ascent installs include standalone C++, Python, and Python-based Jupyter notebook examples for this tutorial. You can find the tutorial source code and notebooks in your Ascent install directory under `examples/ascent/tutorial/ascent_intro/` and the Cloverleaf3D demo files under `examples/ascent/tutorial/cloverleaf_demos/`.

http://ascent-dav.org

Ascent tutorial examples are outlined in our documentation and are included ready to run in Ascent installs

- http://ascent-dav.org
- Click on “Tutorial”



Lawrence Livermore National Laboratory