

Ascent: Flyweight In Situ Visualization and Analysis for HPC Simulations

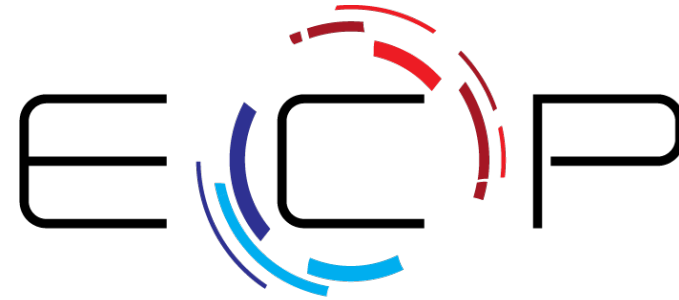
ATPESC 2022

Monday August 8th, 2022

Cyrus Harrison (LLNL),
Nicole Marsaglia (LLNL)



Acknowledgements



EXASCALE COMPUTING PROJECT

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.

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.

ATPESC 2022: Exploring Visualization with Jupyter Notebooks

Tutorial Plan

- Short Ascent Overview (~10 min)
- Follow along Ascent Jupyter Tutorial using cloud hosted Jupyter Notebooks (~30min)

ATPESC 2022: Exploring Visualization with Jupyter Notebooks

- Ascent is an in situ visualization and analysis library for HPC Codes
- Instead of a traditional UI, we use Jupyter Notebooks for some workflows and our Ascent tutorials
- Ascent's tutorial materials provide examples of using Jupyter for scientific visualization
- For those interested in more details about how we use Jupyter:
 - Source for our Jupyter Widgets:
 - https://github.com/Alpine-DAV/ascent/blob/develop/src/ascent/python/ascent_module/py_src/jupyter.py
 - Dockerfile for the Jupyter Container we use for the tutorial:
 - <https://github.com/Alpine-DAV/ascent/blob/develop/src/examples/docker/ubuntu/Dockerfile>
 - Research about connecting simulation codes to Jupyter using Ascent (not demonstrated today)
 - *Interactive in situ visualization and analysis using Ascent and Jupyter*
 - <https://dl.acm.org/doi/10.1145/3364228.3364232>

Important links and contact info:

Ascent Resources:


- Github: <https://github.com/alpine-dav/ascent>
- Docs: <http://ascent-dav.org/>
- Tutorial Landing Page: <https://www.ascent-dav.org/tutorial/>

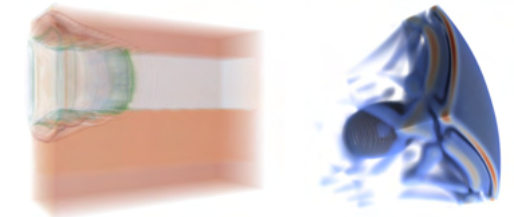
Contact Info:

Cyrus Harrison: cyrush@llnl.gov

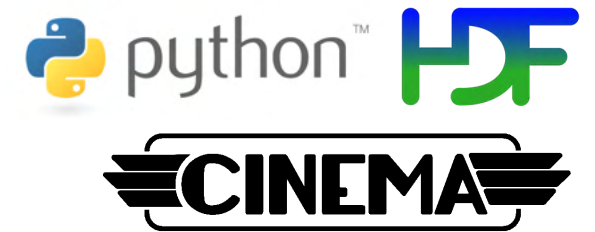
Nicole Marsaglia: marsaglia1@llnl.gov

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/>



Visualizations created using Ascent



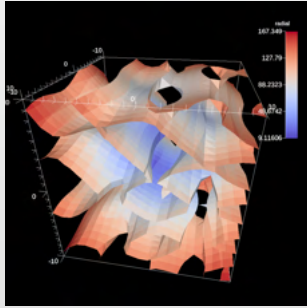
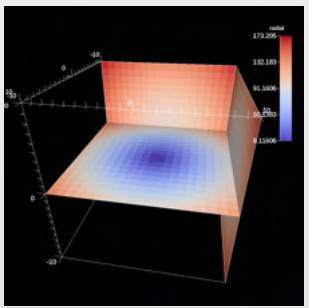
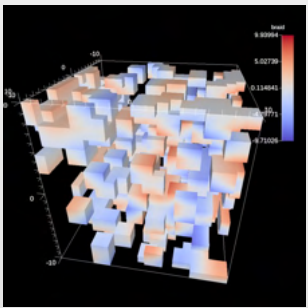
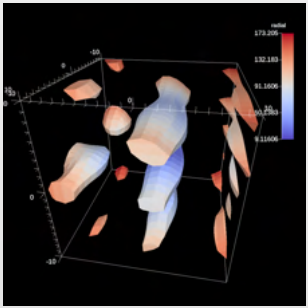
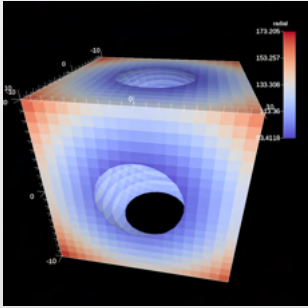
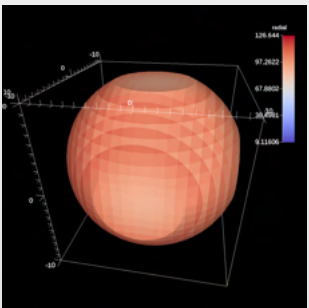
Extracts supported by Ascent

<http://ascent-dav.org>

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

Website and GitHub Repo

Ascent is ready for common visualization use cases

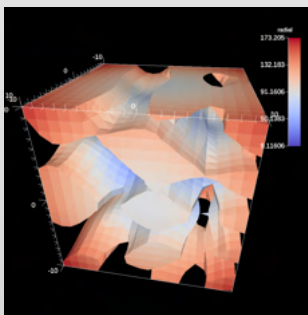
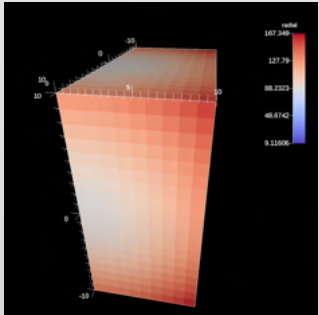


Iso-Volume

Threshold


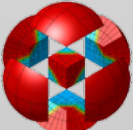

Slice

Contour



Clips

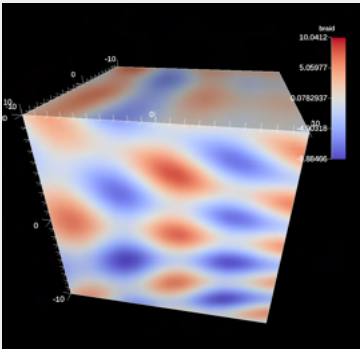
[powered by]



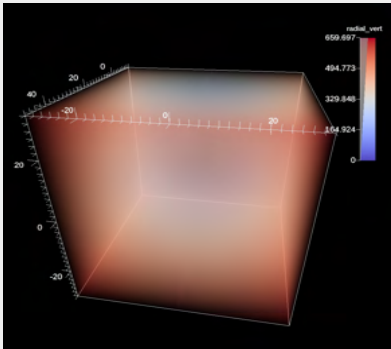
mfem

Devil Ray

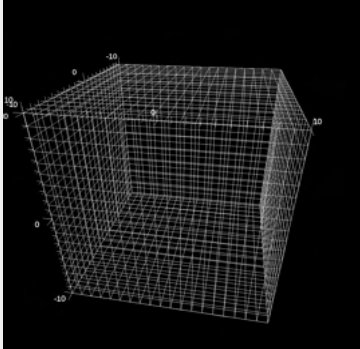
Rendering



Pseudocolor



Volume

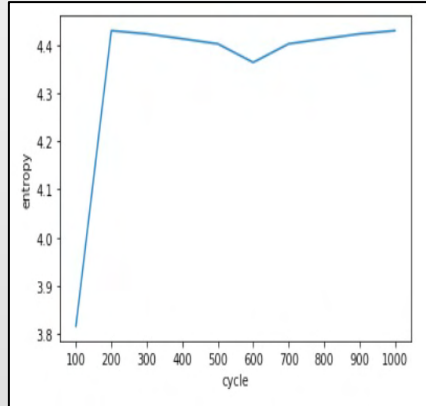


Mesh

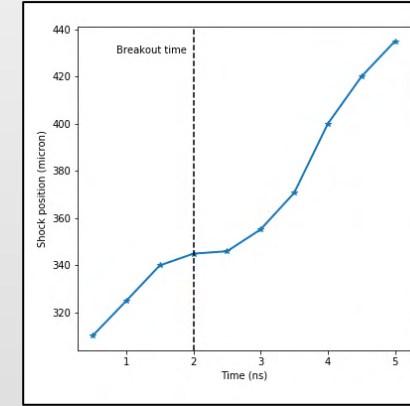
Ascent is ready for common analysis use cases

```
expression: |
  du = gradient(field('velocity','u'))
  dv = gradient(field('velocity','v'))
  dw = gradient(field('velocity','w'))
  w_x = dw.y - dv.z
  w_y = dw.z - dv.x
  w_z = dw.x - dv.y
  vector(w_x,w_y,w_z)
name: vorticity
```

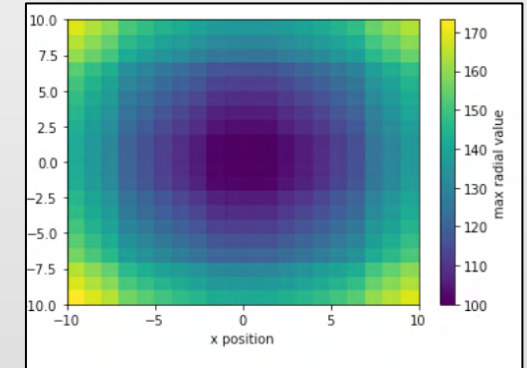
Derived Fields



Time Histories



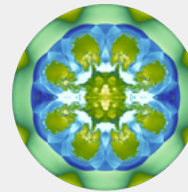
Lineouts and Spatial Binning



```
condition:
  entropy - history(entropy,
    relative_index = 1) > 0.5
```

Triggers

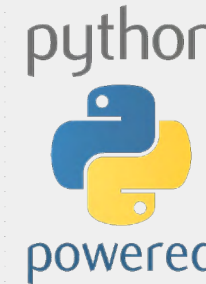
Extracts



Scalar Images



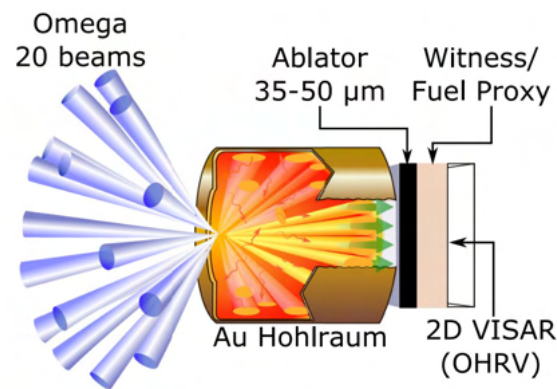
HDF5 Files



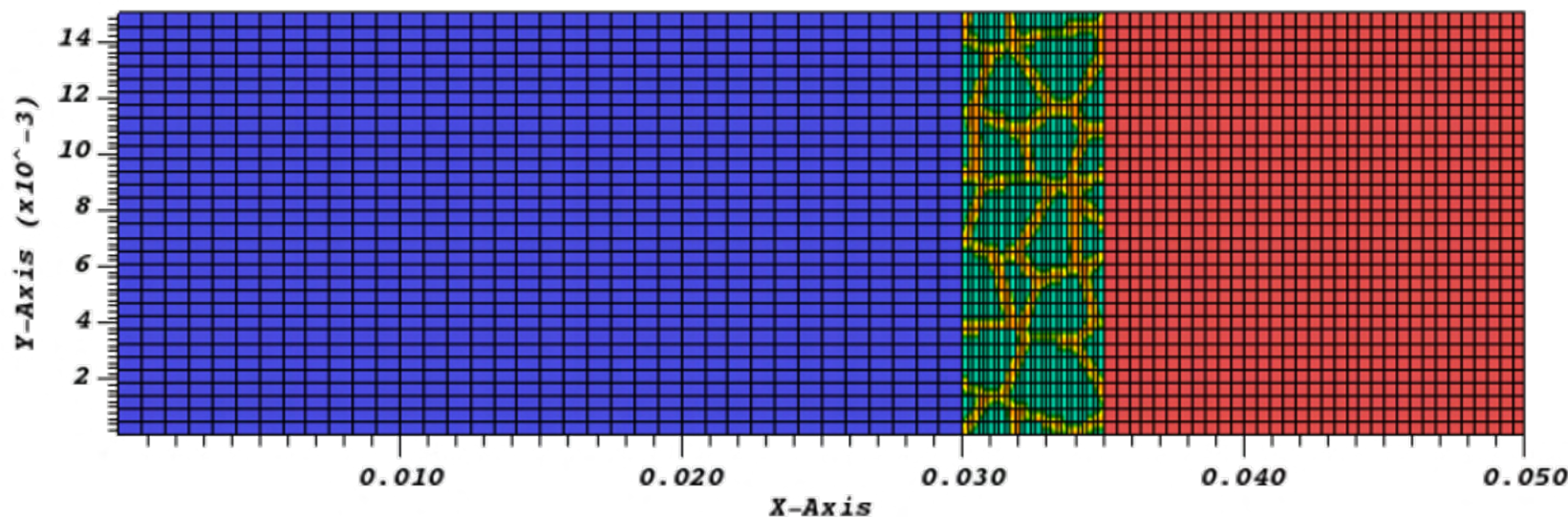
powered



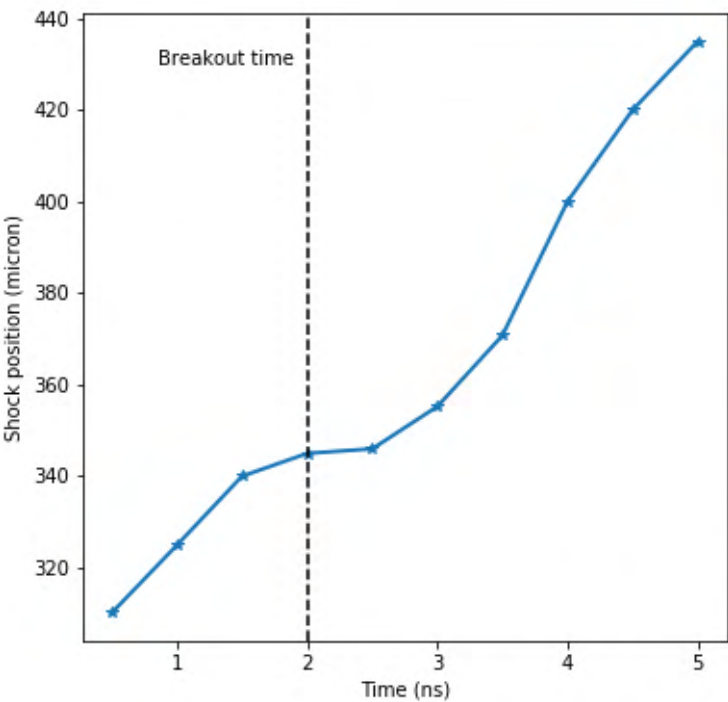
Science Enabling Results: Shock Front Tracking (VISAR)



Velocity interferometer system for any reflector (VISAR)

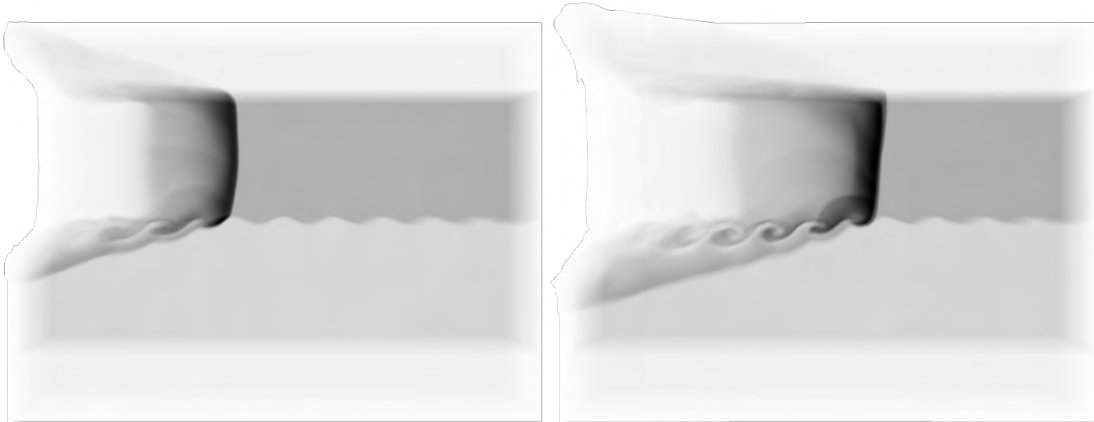


Shock position tracked in Ascent

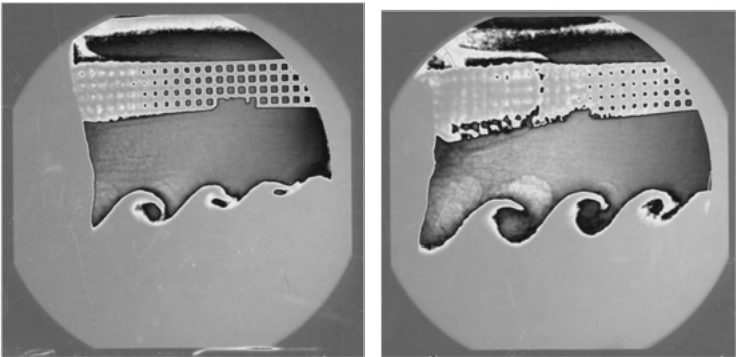


Science Enabling Results: Simulation Validation

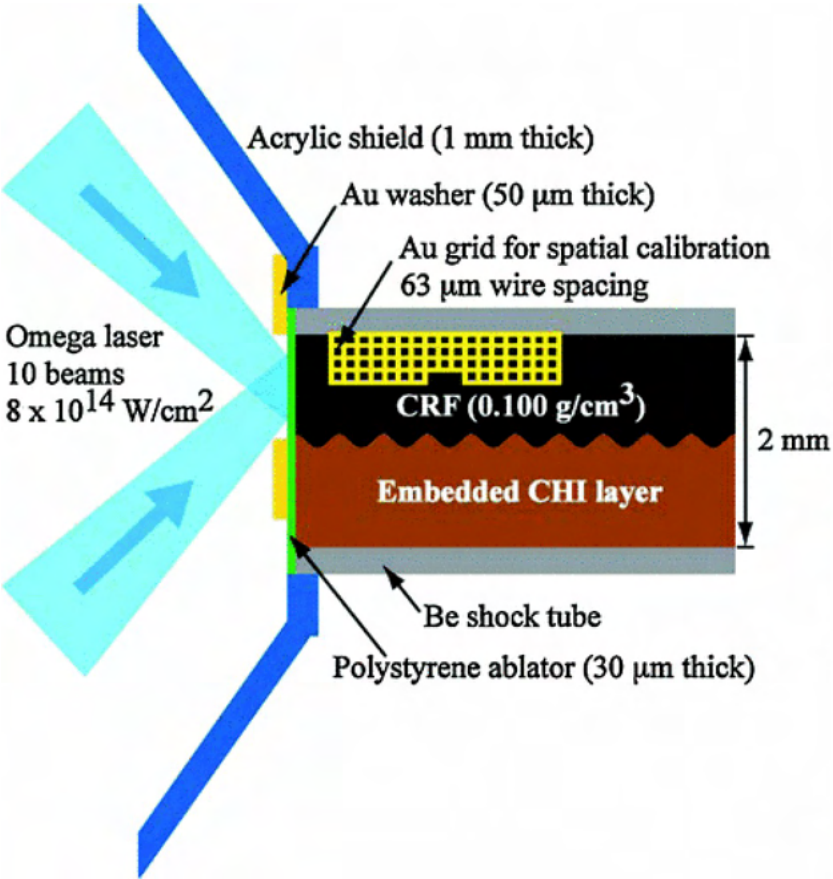
Radiographs



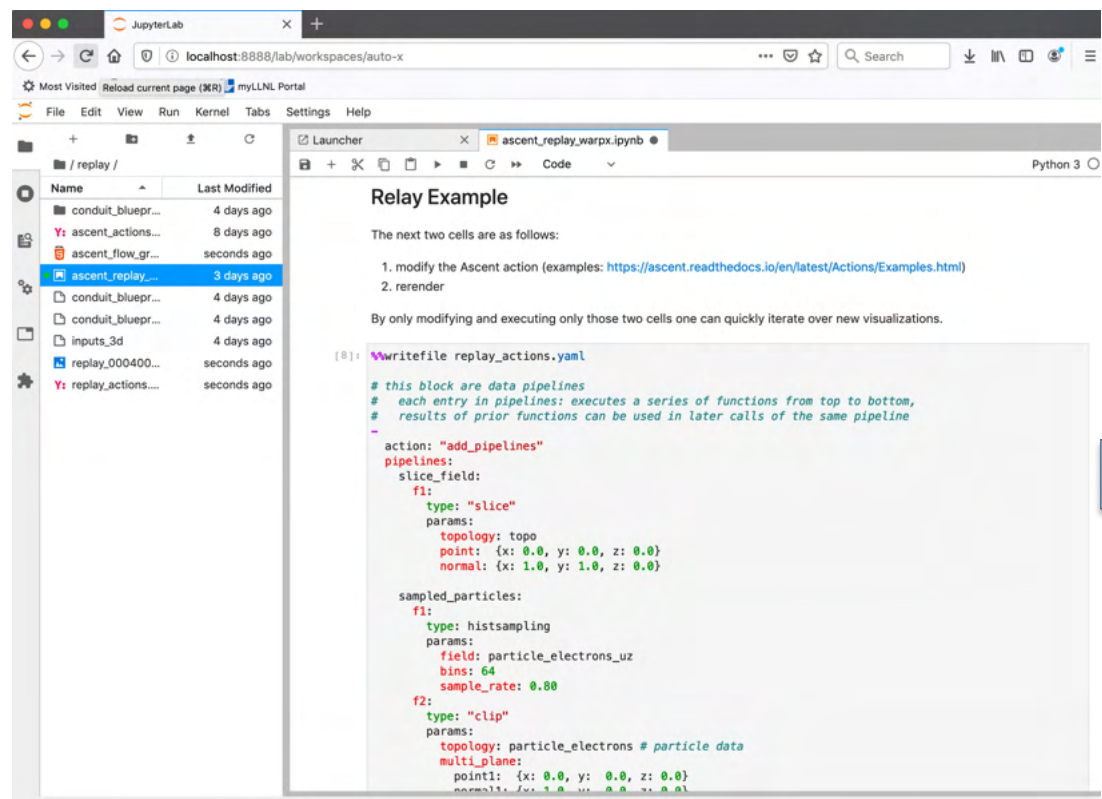
Simulated



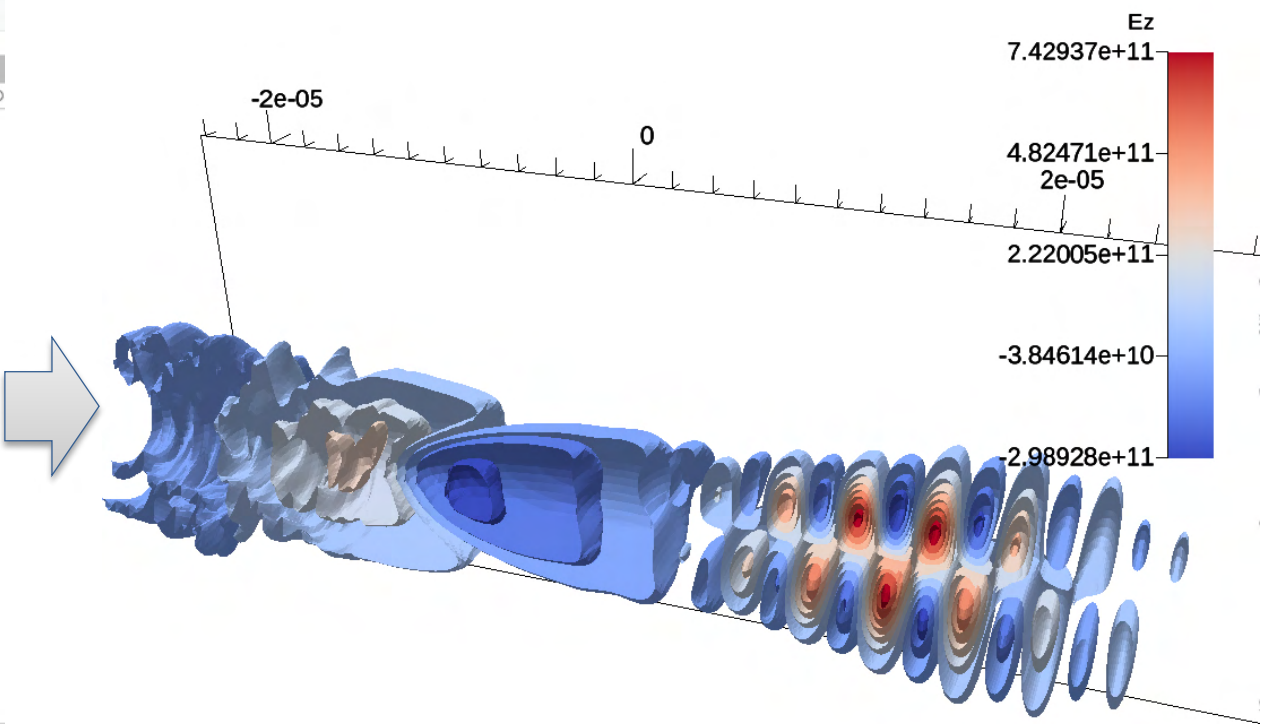
Experimental



Science Enabling Results: WarpX Workflow Tools (Jupyter Labs)



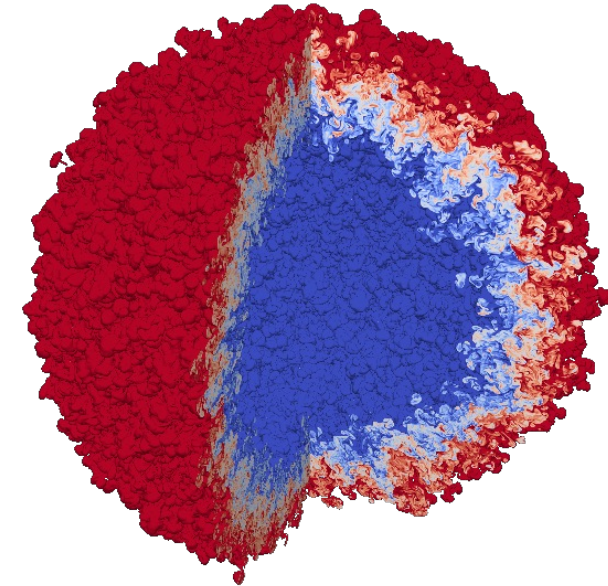
Jupyter Labs Interface



Resulting Image

Science Enabling Results: Rendering At Scale (2018)

- 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



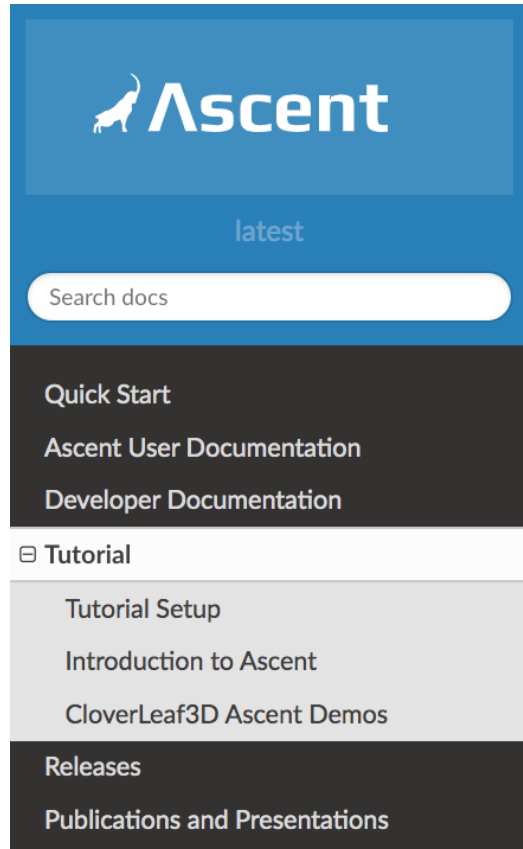
Visualization of an idealized Inertial Confinement Fusion (ICF) simulation of Rayleigh-Taylor instability with two fluids mixing in a spherical geometry.

Today we will teach you about Ascent's API and capabilities

You will learn:

- How to use Conduit, the foundation of Ascent's API
- How to get your simulation data into Ascent
- How to tell Ascent what pictures to render and what analysis to execute

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



[Docs](#) » [Tutorial](#)

[Edit on GitHub](#)

Tutorial

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

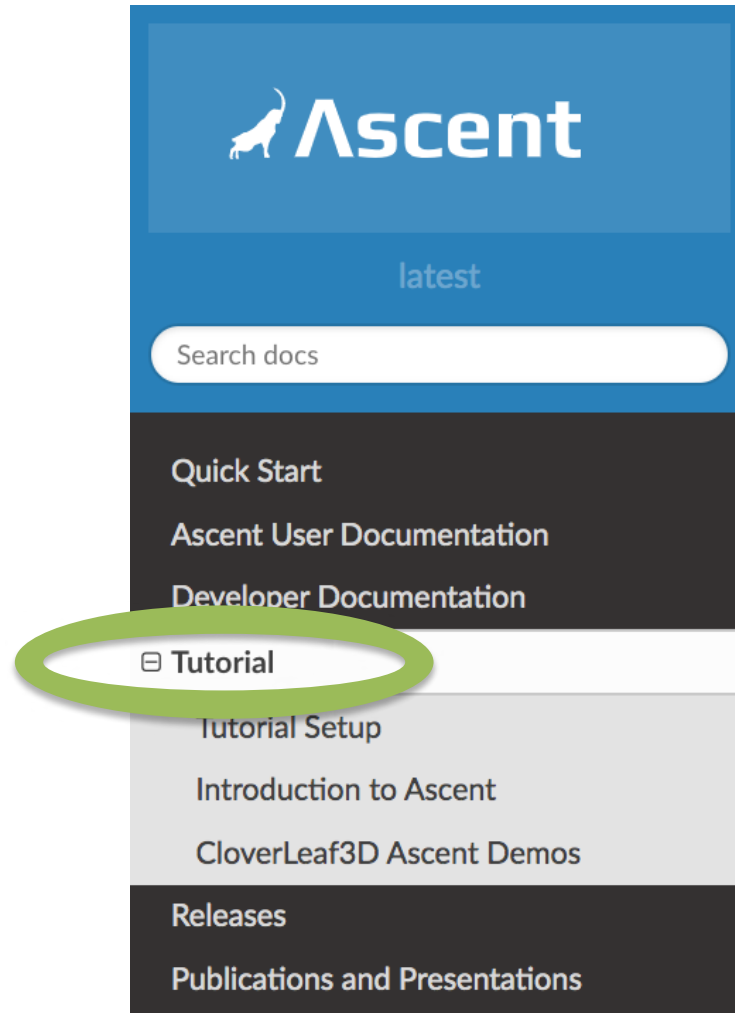
- Formating mesh data for Ascent
- Using Conduit and Ascent's Conduit-based API
- Using and combining Ascent's core building blocks: Scenes, Pipelines, Extracts, Queries, and Triggers
- Using Ascent with the Cloverleaf3D example integration

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”



Ascent's interface provides five top-level functions

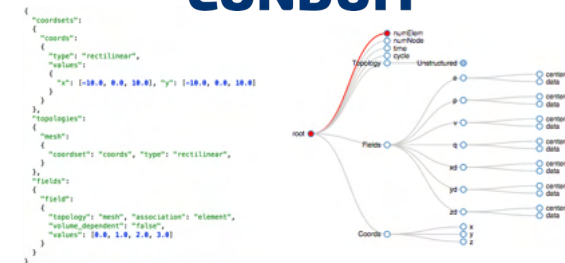
- **open() / close()**
 - Initialize and finalize an Ascent instance
- **publish()**
 - Pass your simulation data to Ascent
- **execute()**
 - Tell Ascent what to do
- **info()**
 - Ask for details about Ascent's last operation

```
//  
// Run Ascent  
//  
  
Ascent ascent;  
ascent.open();  
  
ascent.publish(data);  
ascent.execute(actions);  
ascent.info(details);  
  
ascent.close();
```

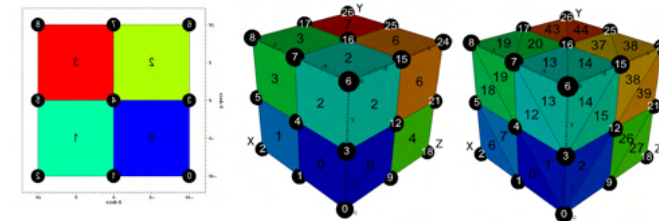
The *publish()*, *execute()*, and *info()* methods take a Conduit tree as an argument.
What is a Conduit tree?

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 (e.g. *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>

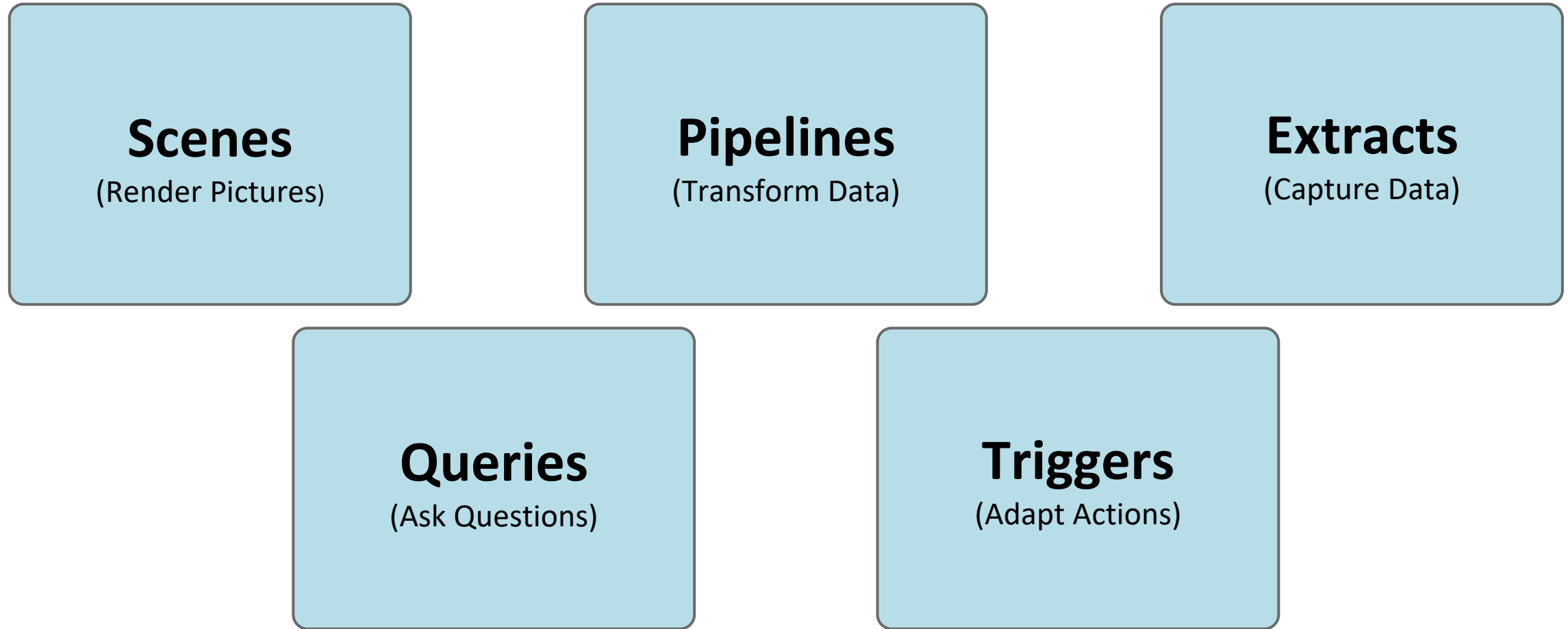
Website and GitHub Repo

Ascent uses Conduit to provide a flexible and extendable API

- Conduit underpins Ascent's support for C++, C, Python, and Fortran interfaces
- Conduit also enables using YAML to specify Ascent actions
- Conduit's zero-copy features help couple existing simulation data structures
- Conduit Blueprint provides a standard for how to present simulation meshes

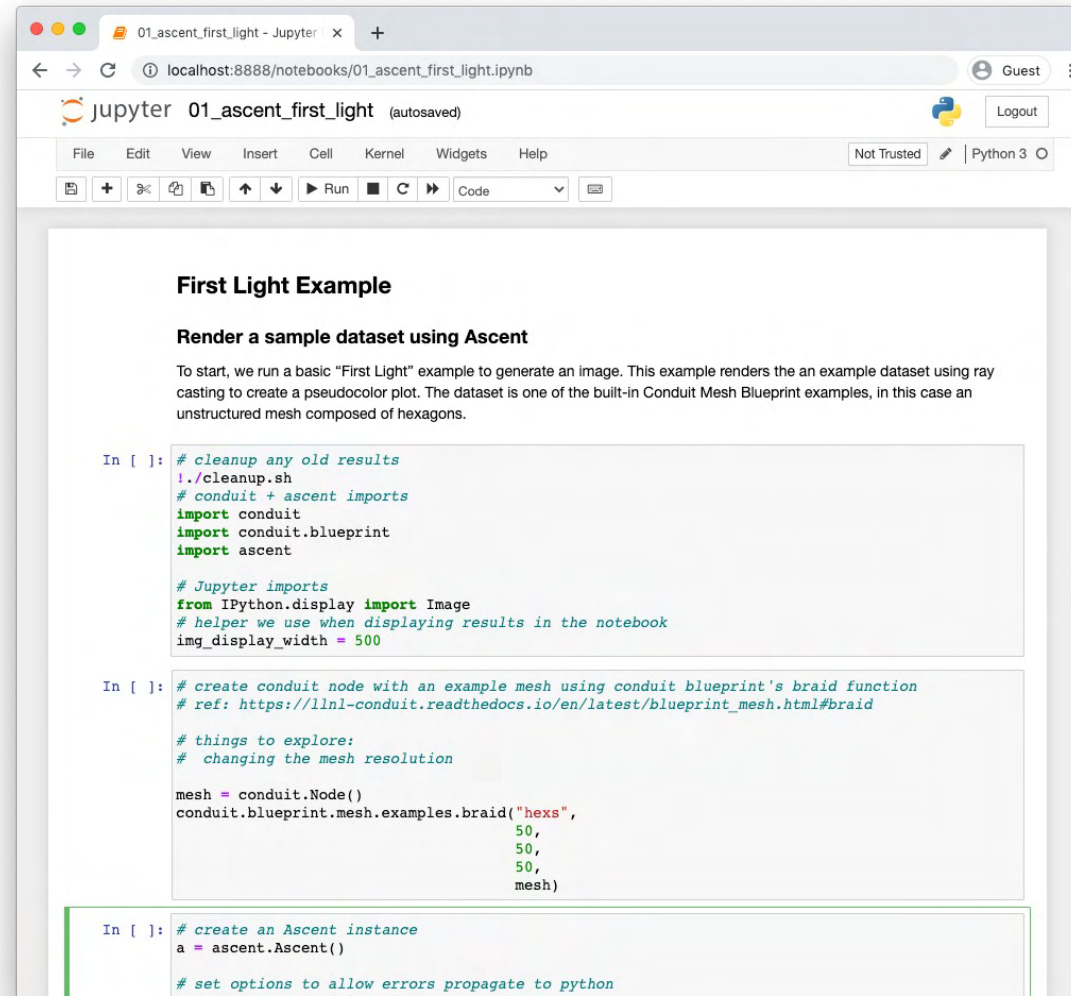
Learning Ascent equates to learning how to construct and pass Conduit trees that encode your data and your expectations.

Ascent's interface provides five composable building blocks



The tutorial provides examples for all of these.

For the remainder of the tutorial, we will run the Ascent Tutorial examples using Jupyter Notebooks



The screenshot shows a Jupyter Notebook interface in a web browser. The browser tab is labeled '01_ascent_first_light - Jupyter'. The address bar shows 'localhost:8888/notebooks/01_ascent_first_light.ipynb'. The Jupyter interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations and execution. The notebook title is '01_ascent_first_light (autosaved)' and the user is logged in as 'Guest'. The main content area is titled 'First Light Example' and contains the following text and code:

First Light Example

Render a sample dataset using Ascent

To start, we run a basic "First Light" example to generate an image. This example renders the an example dataset using ray casting to create a pseudocolor plot. The dataset is one of the built-in Conduit Mesh Blueprint examples, in this case an unstructured mesh composed of hexagons.

```
In [ ]: # cleanup any old results
        !./cleanup.sh
        # conduit + ascent imports
        import conduit
        import conduit.blueprint
        import ascent

        # Jupyter imports
        from IPython.display import Image
        # helper we use when displaying results in the notebook
        img_display_width = 500

In [ ]: # create conduit node with an example mesh using conduit blueprint's braid function
        # ref: https://llnl-conduit.readthedocs.io/en/latest/blueprint_mesh.html#braid

        # things to explore:
        # changing the mesh resolution

        mesh = conduit.Node()
        conduit.blueprint.mesh.examples.braid("hexs",
                                              50,
                                              50,
                                              50,
                                              mesh)

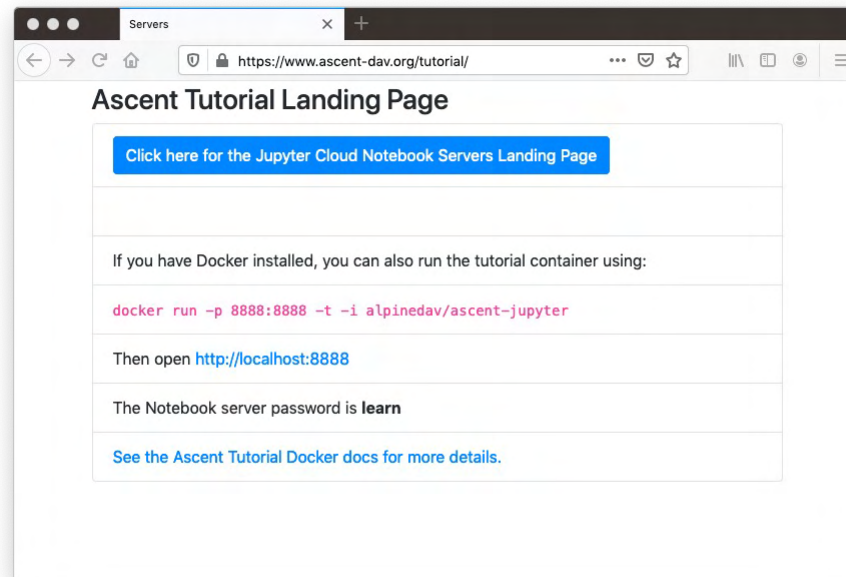
In [ ]: # create an Ascent instance
        a = ascent.Ascent()

        # set options to allow errors propagate to python
        ascent.opts = conduit.Node()
```

You can run our tutorial examples using cloud hosted Jupyter Lab servers

Start here:

<https://www.ascent-dav.org/tutorial/>



Thanks!

Ascent Resources:

- Github: <https://github.com/alpine-dav/ascent>
- Docs: <http://ascent-dav.org/>
- Tutorial Landing Page: <https://www.ascent-dav.org/tutorial/>

Contact Info:

Cyrus Harrison: cyrush@llnl.gov

Nicole Marsaglia: marsaglia1@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

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.



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.