ALPINE: Algorithms and Infrastructure for In-situ Visualization and Analysis

2018 IXPUG Software-Defined Visualization Workshop

Argonne National Lab, July 10-12, 2018

Cyrus Harrison, Matt Larsen, Eric Brugger (LLNL)

LLNL-PRES-754452
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

Lawrence Livermore National Laboratory

Outline

- ALPINE Project Overview
- ALPINE Algorithm Efforts
- ALPINE Infrastructure Efforts
  - Ascent Overview
  - Ascent Concepts
  - Ascent Architecture
  - Ascent Examples
  - Ascent Roadmap

Ascent

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

Outline

- **ALPINE Project Overview**
- ALPINE Algorithm Efforts
- ALPINE Infrastructure Efforts
  - Ascent Overview
  - Ascent Concepts
  - Ascent Architecture
  - Ascent Examples
  - Ascent Roadmap

DOE's visualization community is collaborating to create open source tools ready for Exascale simulation data

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

ALPINE is an ECP project focused on delivering new in-situ algorithms and infrastructure

- **Algorithms**
  - Feature-centric Analysis
  - Sampling-based Analysis
  - Lagrangian Analysis
  - Topological Analysis

- **Infrastructure**
  - Distributed-memory parallel infrastructure
  - A simplified in-situ interface
  - A flyweight in-situ runtime

EXASCALE COMPUTING PROJECT

ALPINE is joint development effort from LANL, LLNL, LBNL, Univ. of Oregon, and Kitware

- **Funding Sources**
  - ALPINE ECP Project
  - LLNL ASC+ATDM Funds (VisIt + Workflow)

- **Development Thrusts**
  - In-situ Infrastructure
  - Visualization and Analysis Algorithms
  - DOE Application Integration

ALPINE is joint development effort from LANL, LLNL, LBNL, Univ. of Oregon, and Kitware

ALPINE Team:

- **LANL:** James Ahrens (PI), David Rogers, Roxana Bujack, Ayan Biswas
- **University of Oregon:** Hank Childs (Deputy PI), Sudhanshu Sane, Nicole Marsaglia
- **LBL:** Gunther H. Weber (Site PI), Oliver Rübel
- **LLNL:** Eric Brugger (Site PI), Matt Larsen, Cyrus Harrison
- **Kitware:** Berk Geveci (Site PI), Utkarsh Ayachit, Andy Bauer

ALPINE is an ECP project focused on delivering new in-situ algorithms and infrastructure

- **Algorithms (LANL, LBL, Univ of Oregon)**
  - Feature-centric Analysis (LANL)
  - Sampling-based Analysis (LANL)
  - Lagrangian Analysis (Univ of Oregon)
  - Topological Analysis (LBL)

- **Infrastructure (LLNL, Kitware, Univ of Oregon)**
  - Distributed-memory parallel infrastructure
  - A simplified in-situ interface
  - A flyweight in-situ runtime

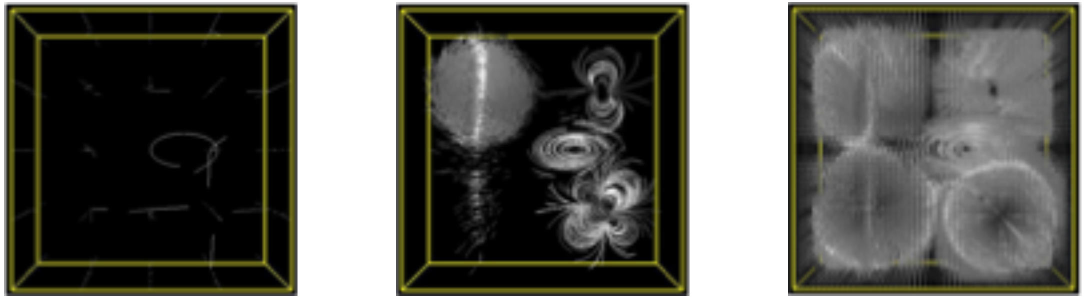
EXASCALE COMPUTING PROJECT

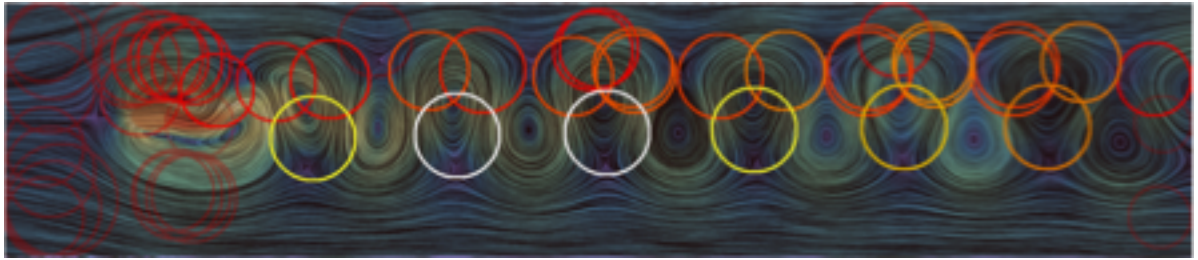
Outline

- ALPINE Project Overview
- **ALPINE Algorithm Efforts**
- ALPINE Infrastructure Efforts
  - Ascent Overview
  - Ascent Concepts
  - Ascent Architecture
  - Ascent Examples
  - Ascent Roadmap

ALPINE has four in-situ algorithm efforts focused on automated data reduction

Feature-centric Analysis (LANL)

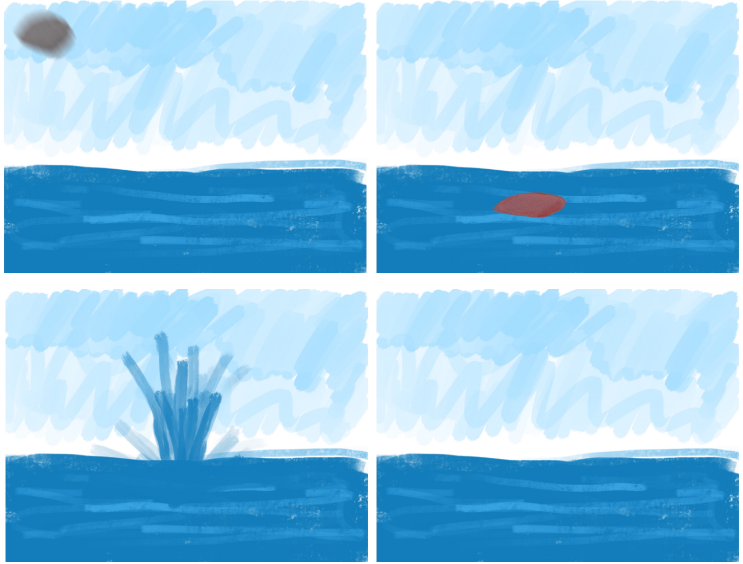


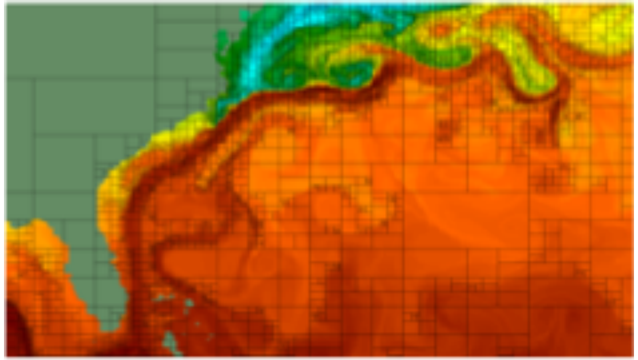


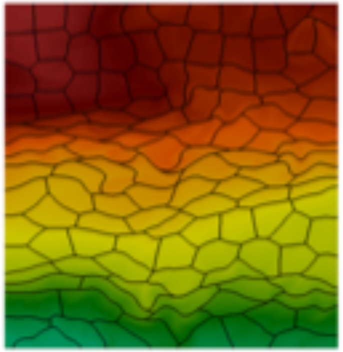
Detecting features with rotationally invariant moments or topological analysis

ALPINE has four in-situ algorithm efforts focused on automated data reduction

Sampling-based Analysis (LANL)



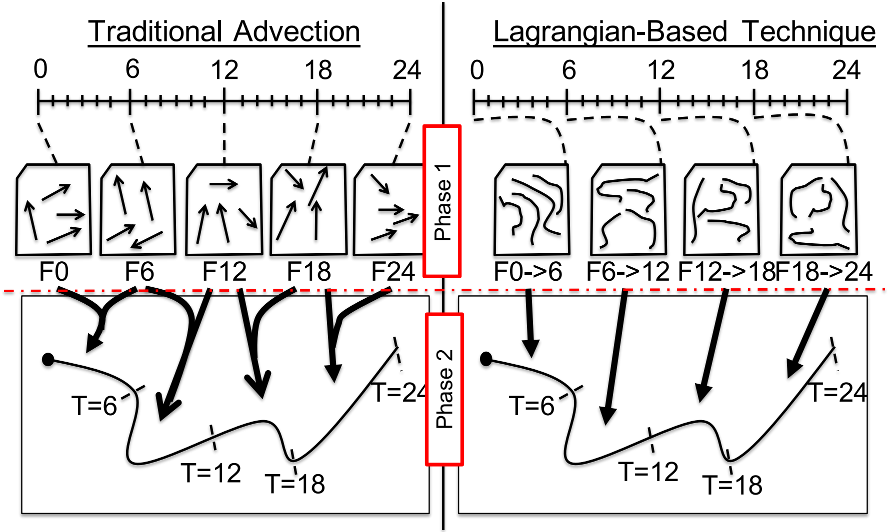




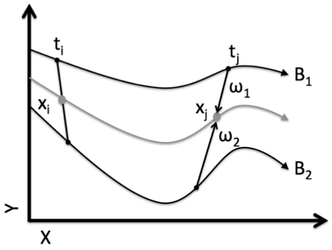
Importance sampling based on measurements in time, space, and correlations

ALPINE has four in-situ algorithm efforts focused on automated data reduction

Lagrangian Analysis (Univ. of Oregon)



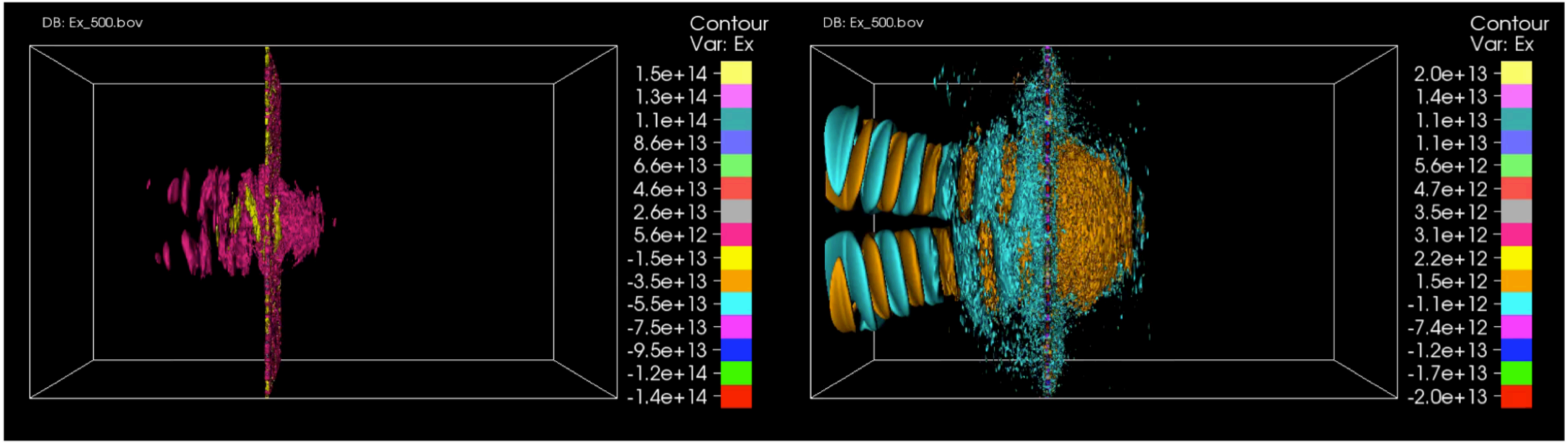
Notional example:
Interpolating a new trajectory from two extracted basis flows



In-situ extraction of lagrangian basis flows that can be interpolated post-hoc to explore new trajectories

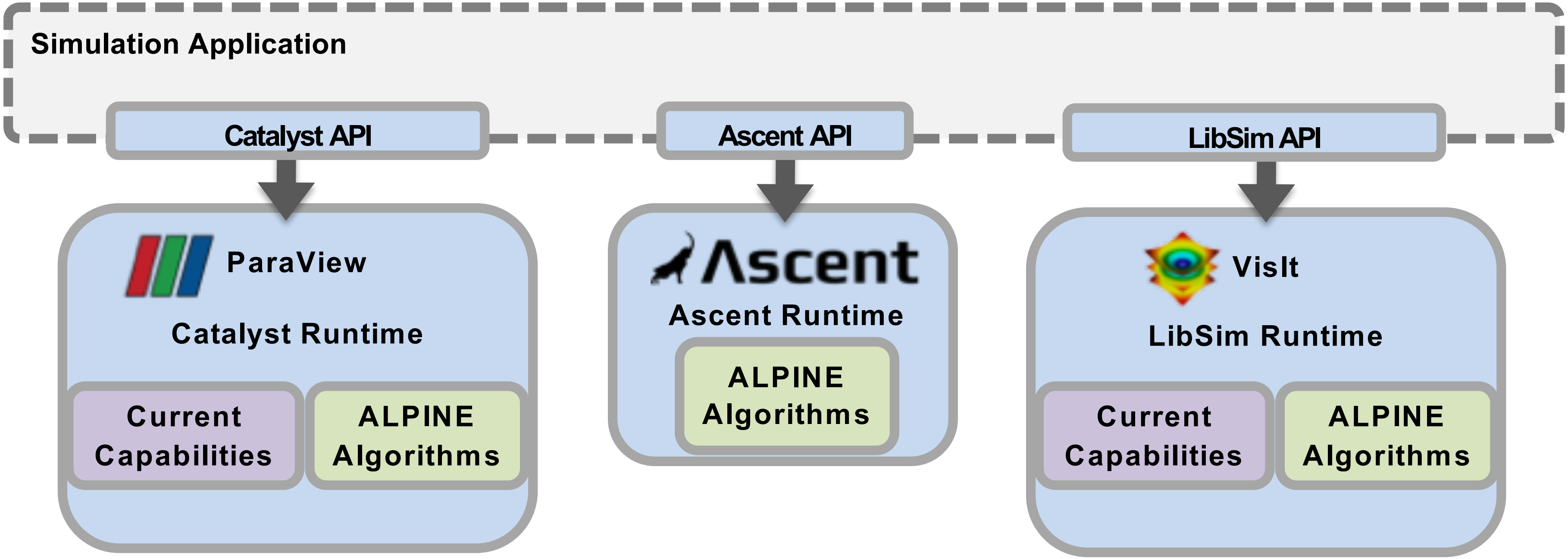
ALPINE has four in-situ algorithm efforts focused on automated data reduction

Topological Analysis (LBL)



Using contour trees to intelligently select iso-values for contouring

ALPINE algorithms will be deployed in several in-situ infrastructures



Outline

- ALPINE Project Overview
- ALPINE Algorithm Efforts
- **ALPINE Infrastructure Efforts**
  - Ascent Overview
  - Ascent Concepts
  - Ascent Architecture
  - Ascent Examples
  - Ascent Roadmap

The rest of this talk focuses on ALPINE infrastructure efforts

- Algorithms
  - Feature-centric Analysis
  - Sampling-based Analysis
  - Lagrangian Analysis
  - Topological Analysis

- **Infrastructure**
  - Distributed-memory parallel infrastructure
  - A simplified in-situ interface
  - A flyweight in-situ runtime

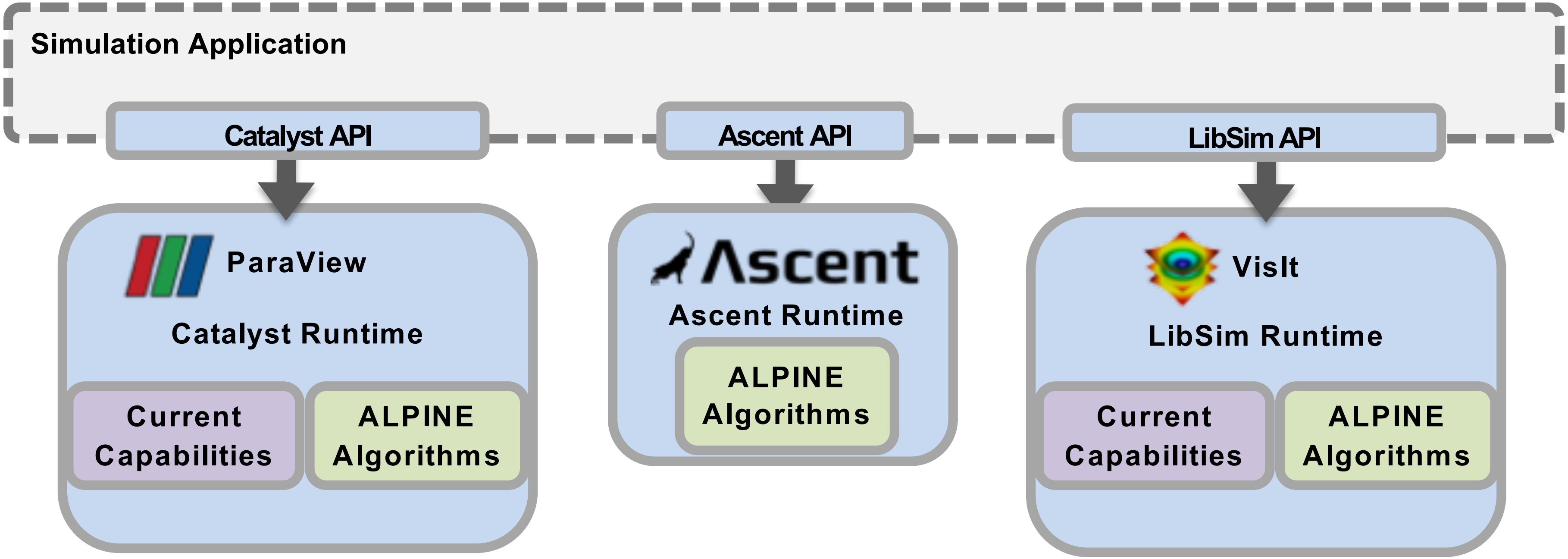
The overarching goal of ALPINE infrastructure efforts is to make it easy to ***develop*** and ***deploy*** in-situ algorithms to users of simulation applications

ALPINE infrastructure efforts focus on VTK-m and Ascent

- **VTK-m Distributed-Memory Support**
  - A new distributed-memory parallel layer for algorithms that use VTK-m for node-level parallelism
    - A Filter interface for domain-decomposed datasets
    - Compositing infrastructure using MPI and DIY

- **Ascent**
  - A new in-situ infrastructure that provides:
    - A simplified in-situ interface
    - A flyweight in-situ runtime

ALPINE's VTK-m development efforts will enable us to deploy algorithms in several in-situ infrastructures



Outline

- ALPINE Project Overview
- ALPINE Algorithm Efforts
- ALPINE Infrastructure Efforts
  - **Ascent Overview**
  - Ascent Concepts
  - Ascent Architecture
  - Ascent Examples
  - Ascent Roadmap

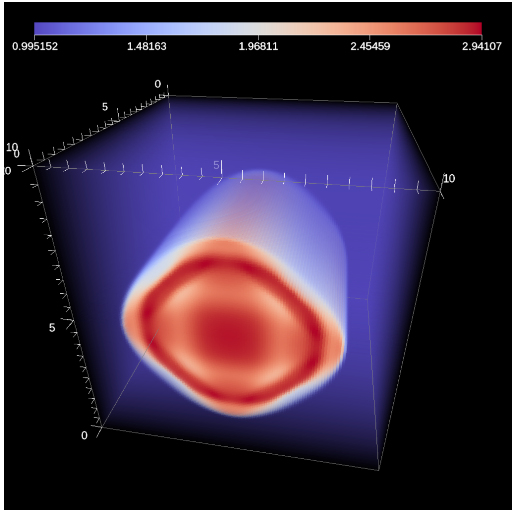
Ascent

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

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

Project Info

- GitHub Repo: https://github.com/Alpine-DAV/ascent
- Docs: https://alpine-dav.github.io/ascent
- Supported Languages: C++, Python, C, Fortran
- License: BSD Style
- Builds with Spack https://spack.io/



Example in-situ render created using Ascent

Ascent focuses on ease of use and efficient in-situ execution

Ascent Delivers

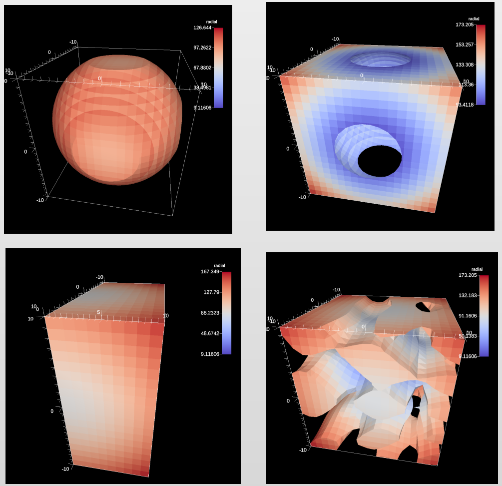
- An easy to use API
  - Designed to enable three use cases
    - Making Pictures
    - Transforming Data
    - Capturing Data
  - Leverages Conduit (http://software.llnl.gov/conduit)
    - Simplifies handoff of mesh-based simulation data
    - Underpins support for C, C++, Fortran, and Python

- A flyweight design
  - Efficient distributed-memory + many-core execution
    - Leverages MPI, VTK-m (http://m.vtk.org/)
  - Lower memory requirements then current tools
  - Less dependencies than current tools (ex: no OpenGL)

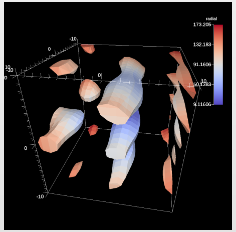
```
//
// Run Ascent
//

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

Arguments are Conduit *Node* Trees

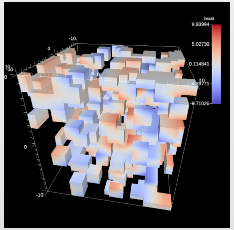
Ascent is ready for common visualization use cases



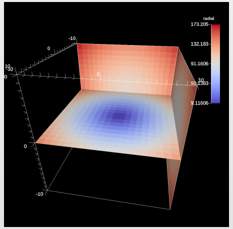
Clips



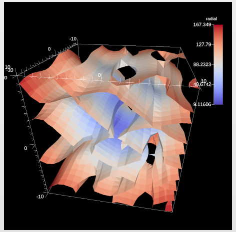
Iso-Volume



Threshold

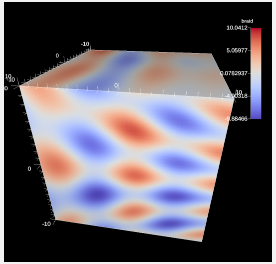


Slice

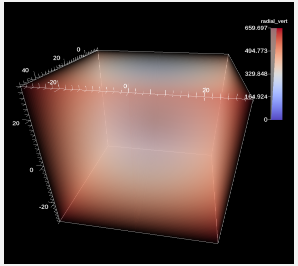


Contour

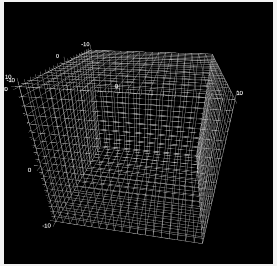
Rendering



Pseudocolor



Volume

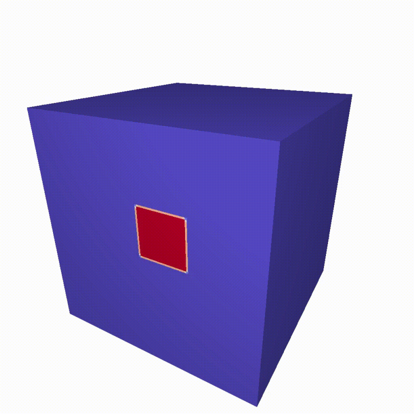


Mesh

Ascent supports multiple languages and output types

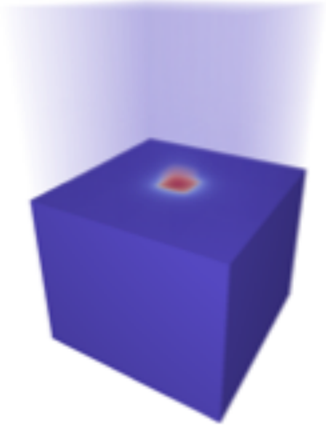
- **Language Bindings**
  - C/C++
  - python™
  - Fortran
- **Output Types**
  - HDF
  - ADIOS
  - CINEMA

Ascent provides example integrations that serve as built-in data sources

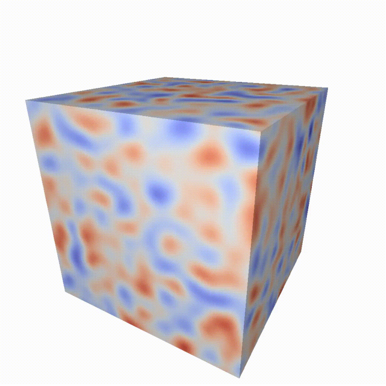


Cloverleaf3D

Lulesh



Kripke



Smooth Noise

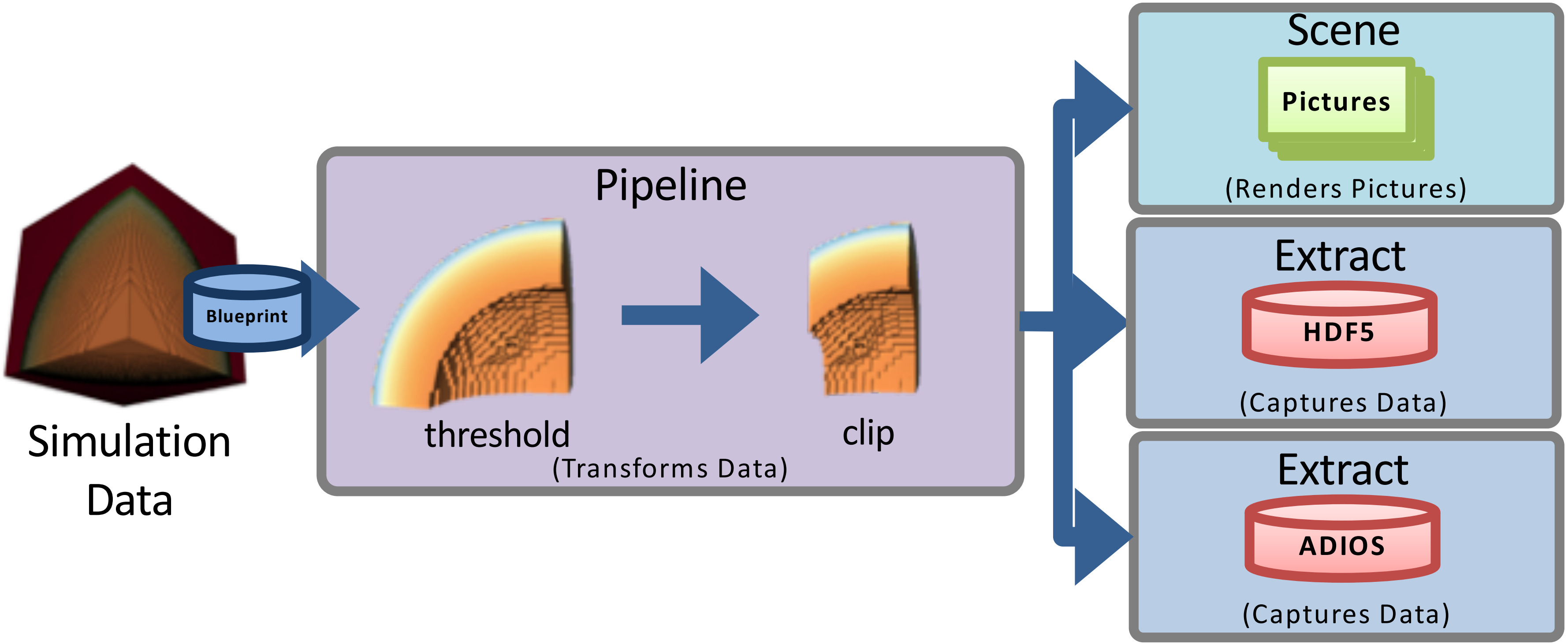
Outline

- ALPINE Project Overview
- ALPINE Algorithm Efforts
- ALPINE Infrastructure Efforts
  - Ascent Overview
  - **Ascent Concepts**
  - Ascent Architecture
  - Ascent Examples
  - Ascent Roadmap

Ascent's API provides three key building blocks

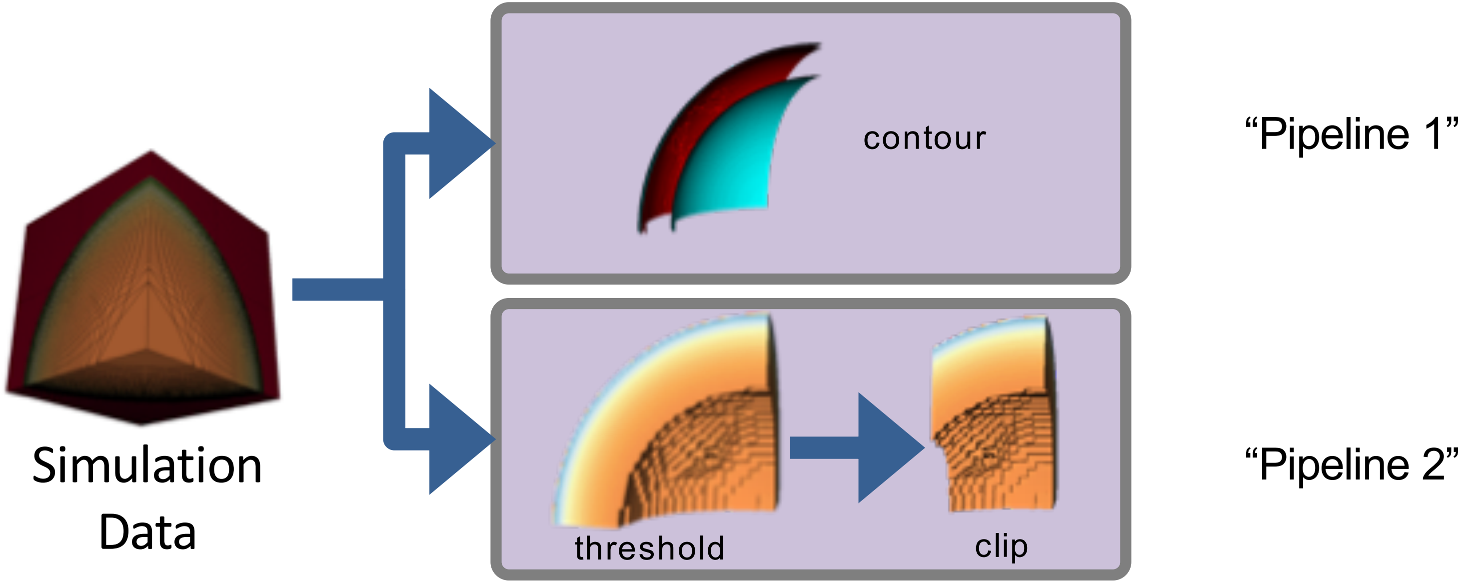
- Pipelines (transform data):
  - Allows users to describe how they want to transform their data
- Scenes(make pictures):
  - Allows users to describe the pictures they want to create
- Extracts (capture data):
  - Allows users to describe how they want capture data

Ascent end-to-end conceptual example



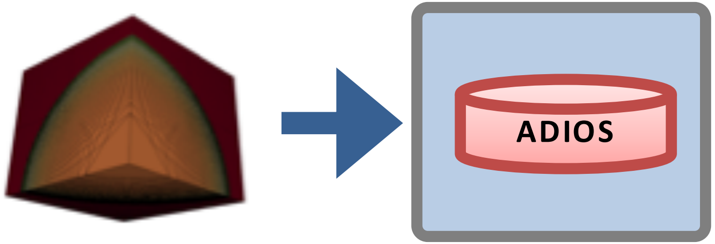
A pipeline is a series data transformations (i.e., filters)

- Ascent allows an arbitrary number of pipelines to be described

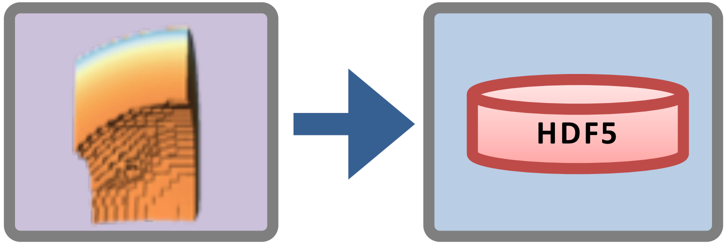


An extract is a way to capture data for use outside of Ascent

- Examples:
  - Export published simulation data to HDF5, ADIOS, etc



  - Export pipeline results to HDF5, ADIOS, etc.

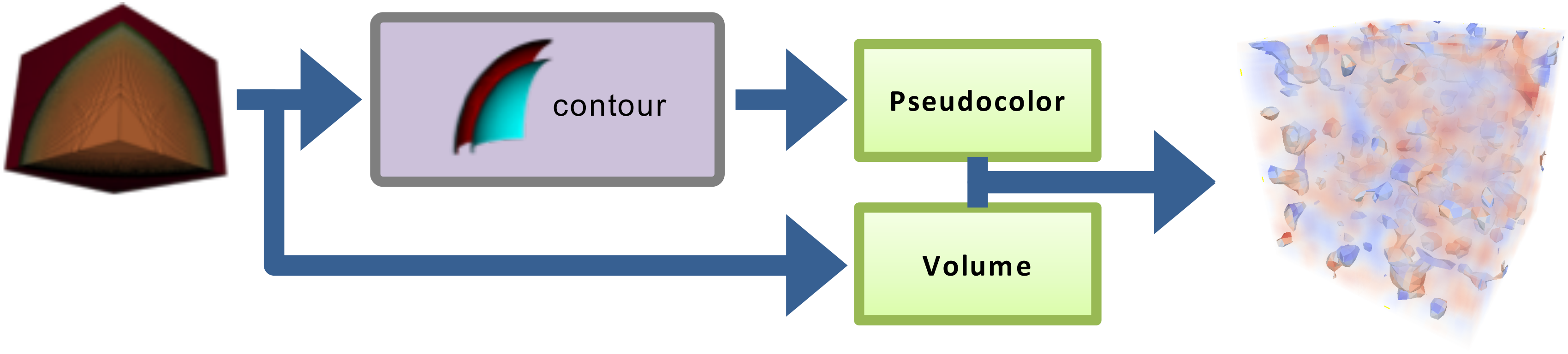


Currently supported extracts:

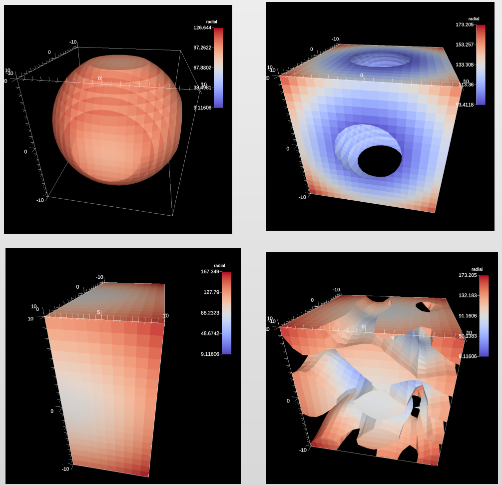
- Create Cinema databases
- Export to HDF5 files
- Publish to an embedded Python interpreter
- Publish to ADIOS (proof-of-concept)

A scene is a way to render pictures

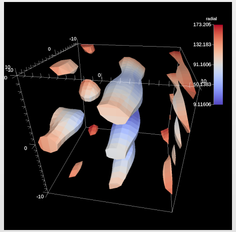
- Contains a list of plots
  - E.g., volume, pseudocolor, and mesh
- Contains a list of camera parameters



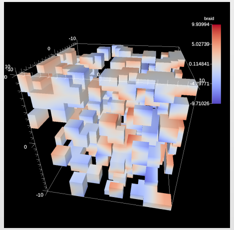
Ascent's filters and plots



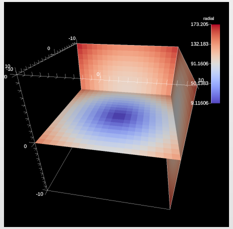
Clips



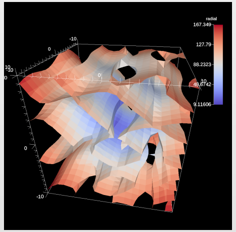
Iso-Volume



Threshold

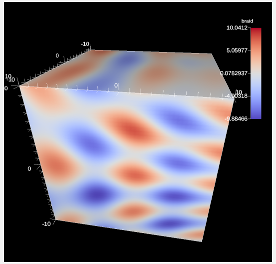


Slice

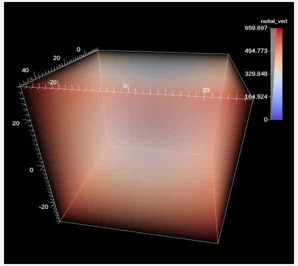


Contour

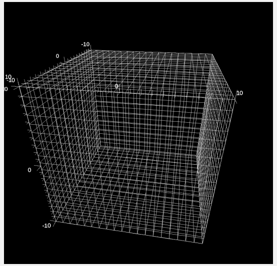
Rendering



Pseudocolor



Volume

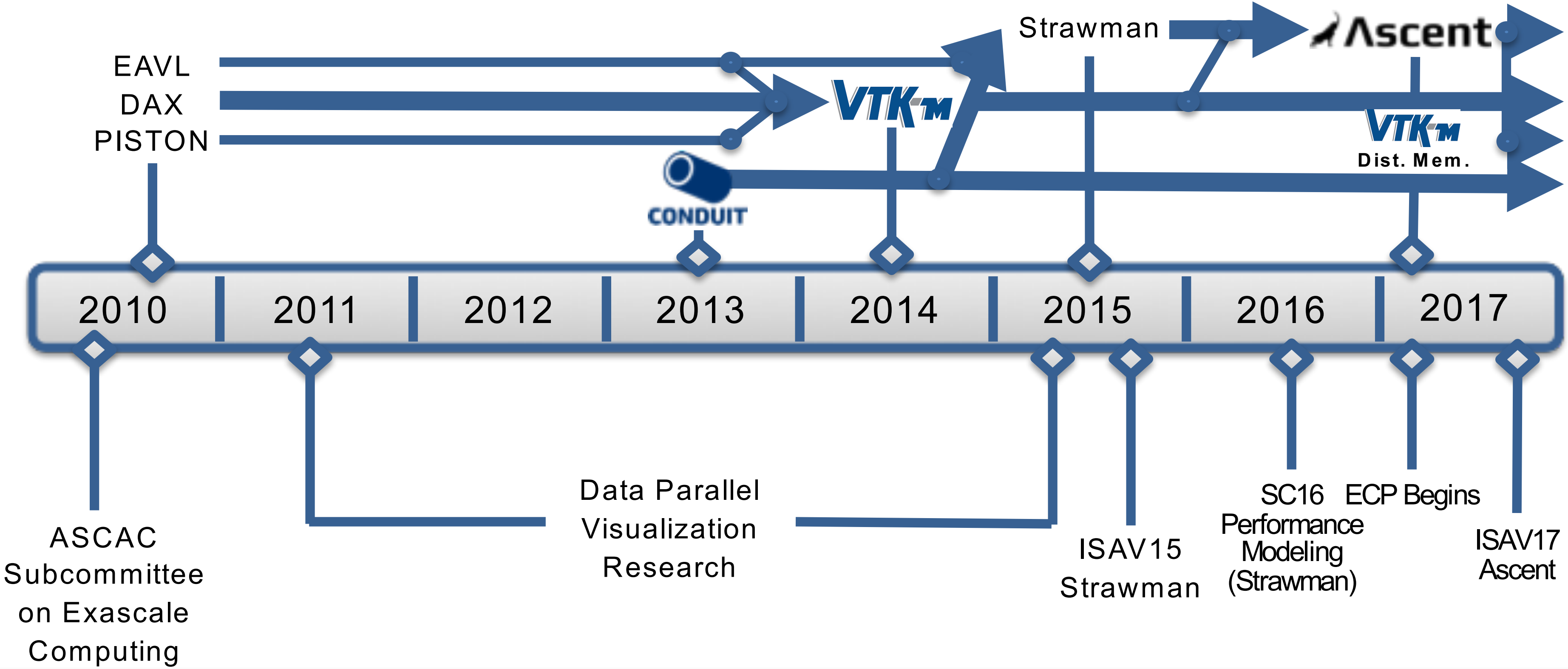


Mesh

Outline

- ALPINE Project Overview
- ALPINE Algorithm Efforts
- ALPINE Infrastructure Efforts
  - Ascent Overview
  - Ascent Concepts
  - **Ascent Architecture**
  - Ascent Examples
  - Ascent Roadmap

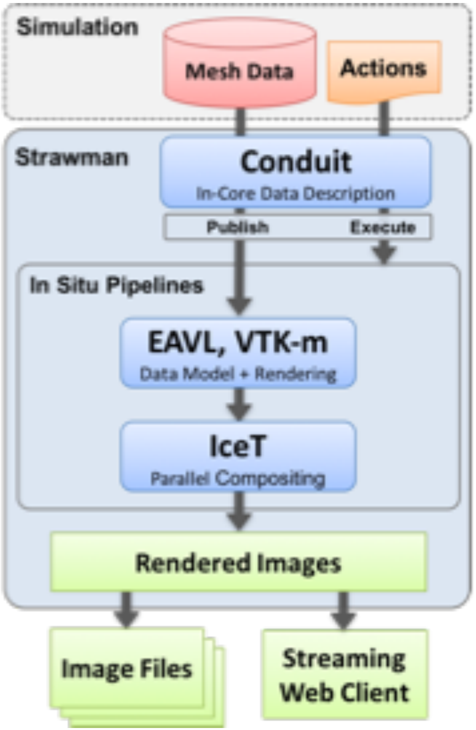
So, how did we get here?



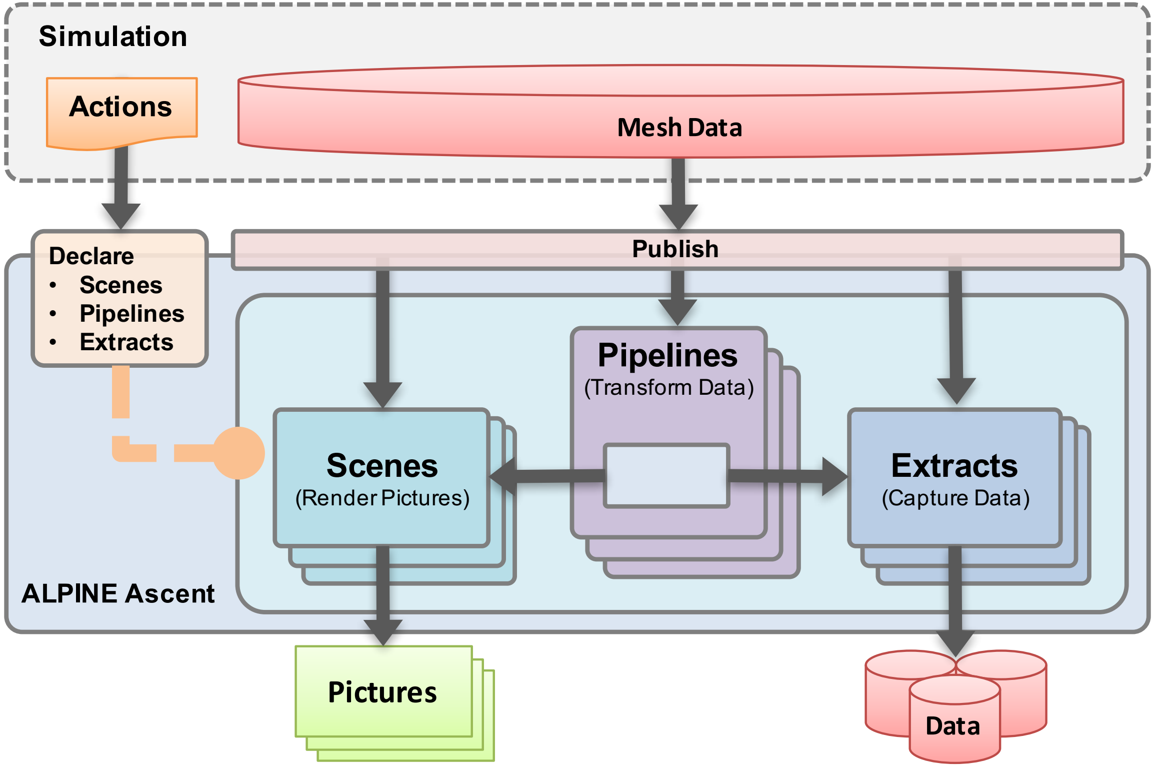
Ascent is an an evolution of the Strawman visualization proxy application

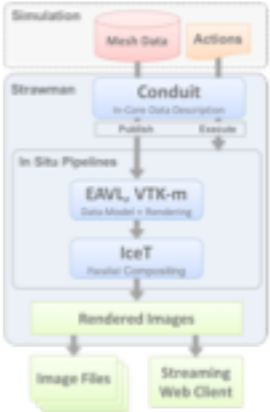
What did Strawman provide?

- Single-plot rendered images
  - No filters or data-flow
- Built-in integration of three proxy-apps
  - Lulesh
  - Cloverleaf3D
  - Kripke



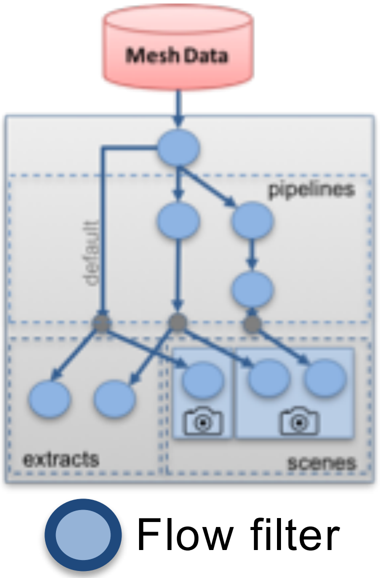
The Ascent architecture has matured to support a variety of in-situ use cases





Ascent uses Flow, simple data flow network library, to compose and execute VTK-m filters

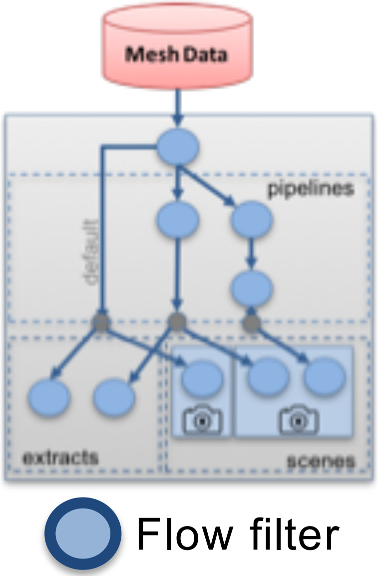
- VTK-m does not provide an execution model
- ParaView and VisIt have their own rich execution models
- VTK-m features in those tools will be exposed through their existing execution models
- Ascent needs a basic execution model to support complex user requests



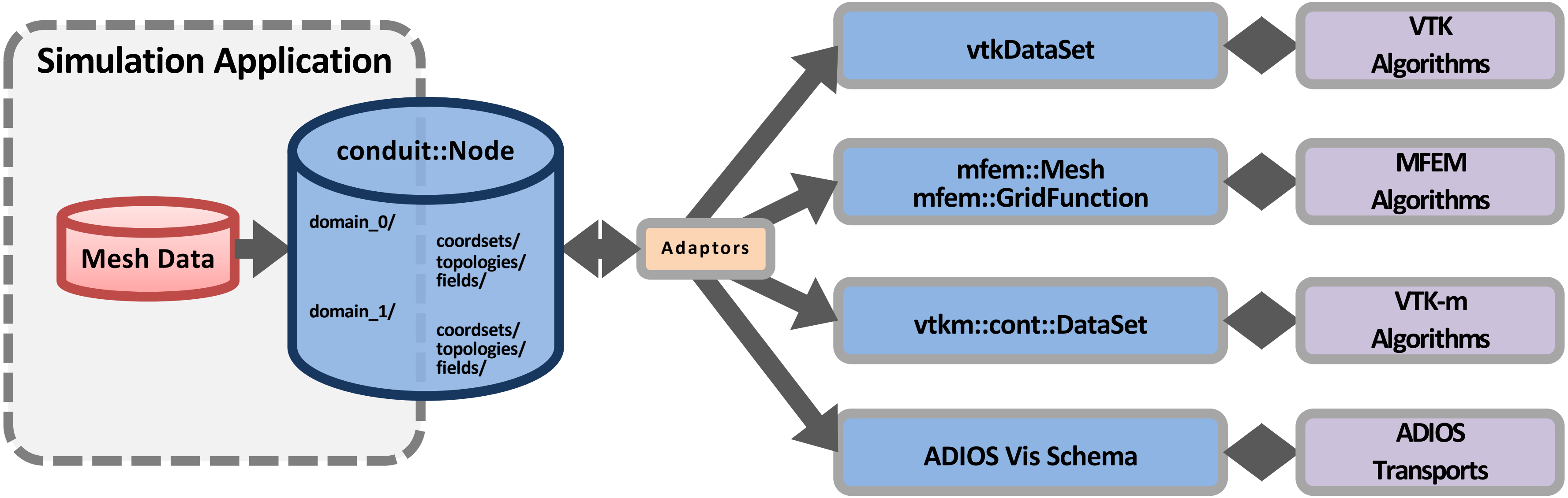
Flow provides the execution model for ALPINE Ascent

Flow is generic and allows for interoperability between components

- Flow filters can be anything
  - Each filter specifies:
    - What parameters it expects
    - What type of input it expects
- Ascent Runtime manages Flow
  - Data set conversions
  - Filter execution
- Flow is accessible inside Ascent



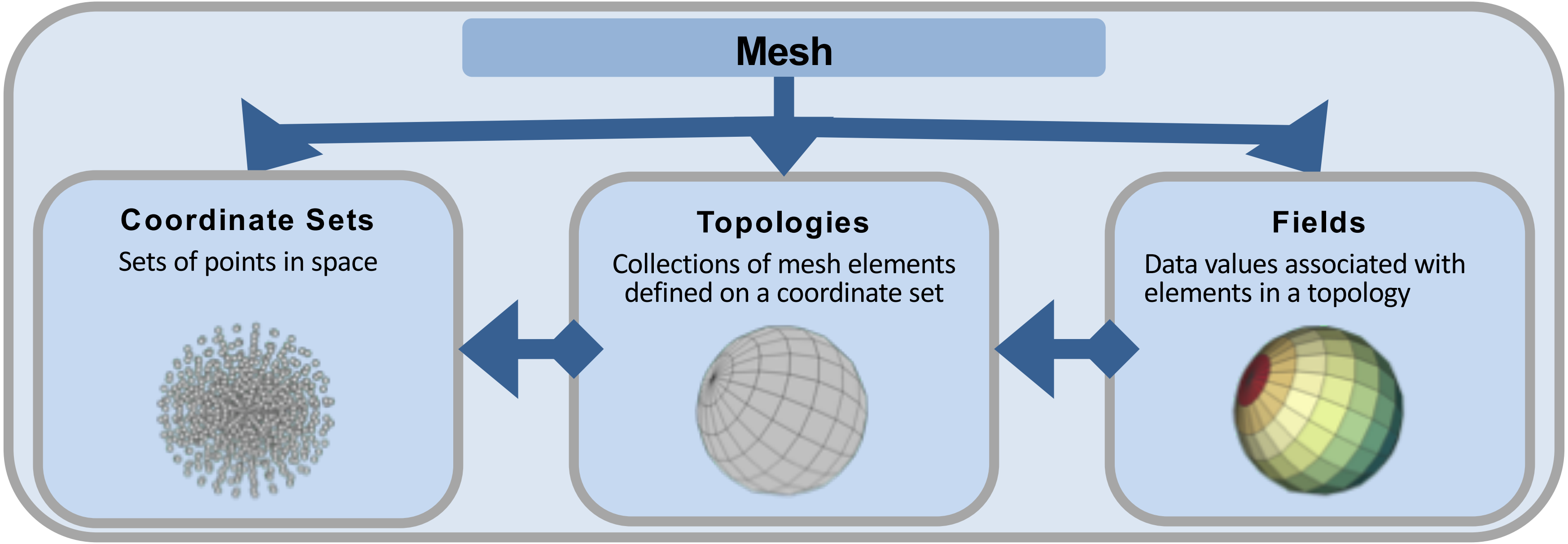
Mesh data is published to Ascent via Conduit Mesh Blueprint



The mesh blueprint provides conventions for describing and organizing simulation mesh data so that it can be used (often zero-copy) via multiple full featured data APIs

http://software.llnl.gov/conduit/blueprint.html

The Mesh Blueprint supports mesh representation concepts common in several full featured mesh data models



Ideas were shaped by surveying projects including: ADIOS, BoxLib, Chombo, Damaris, EAVL, Exodus, ITAPS, MFEM, SAF, SAMRAI, Silo, VisIt's AVT, VTK, VTK-m, Xdmf.

Example: Conduit Blueprint Rectilinear Mesh

```
node["coordsets/coords/type"] = "uniform";

node["coordsets/coords/dims/i"] = m_point_dims[0];
node["coordsets/coords/dims/j"] = m_point_dims[1];
node["coordsets/coords/dims/k"] = m_point_dims[2];

node["coordsets/coords/origin/x"] = m_origin[0];
node["coordsets/coords/origin/y"] = m_origin[1];
node["coordsets/coords/origin/z"] = m_origin[2];

node["coordsets/coords/spacing/dx"] = m_spacing[0];
node["coordsets/coords/spacing/dy"] = m_spacing[1];
node["coordsets/coords/spacing/dz"] = m_spacing[2];

node["topologies/mesh/type"]     = "uniform";
node["topologies/mesh/coordset"] = "coords";

node["fields/nodal_noise/association"] = "vertex";
node["fields/nodal_noise/type"]        = "scalar";
node["fields/nodal_noise/topology"]    = "mesh";
node["fields/nodal_noise/values"].set_external(m_nodal_scalars);
```

Outline

- ALPINE Project Overview
- ALPINE Algorithm Efforts
- ALPINE Infrastructure Efforts
  - Ascent Overview
  - Ascent Concepts
  - Ascent Architecture
  - **Ascent Examples**
  - Ascent Roadmap

Design Goal: Support custom analysis as a first class citizen

- Mainstream visualization only gets you so far
  - Scientists often want something other than a contour

- In-situ visualization frameworks need to be
  - Flexible
  - Easy to use
  - Easy to connect with other "things"

Proof-of-concept: In-situ machine learning

- Python has a massive menu of data science tools
- Goal: Use Ascent to connect Python data science tools to HPC simulations
- Demonstrate ease of use:
  - Ascent provides curated simulation data that is easy to digest in python
  - Conduit Blueprint data published in Fortran, C, or C++ codes can be accessed as numpy arrays

What does using Python in Ascent look like?

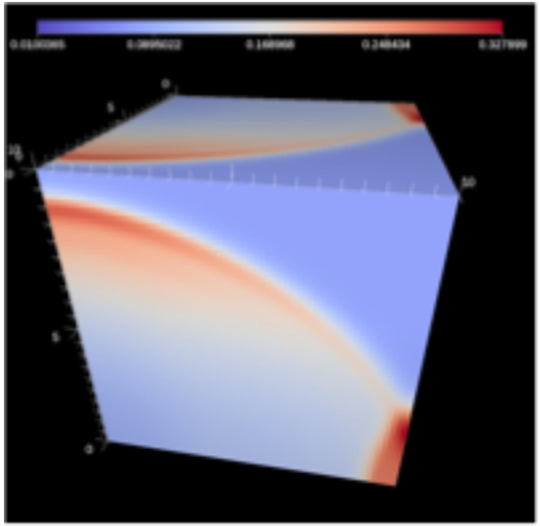
```
import numpy as np
from mpi4py import MPI

# obtain a mpi4py mpi comm object
comm = MPI.Comm.f2py(ascent_mpi_comm_id())

# get this MPI task's published blueprint data
mesh_data = ascent_data()

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

```
# find the data extents of the energy field using mpi

# first get local extents
e_min, e_max = e_vals.min(), e_vals.max()

# declare vars for reduce results
e_min_all = np.zeros(1)
e_max_all = np.zeros(1)

# reduce to get global extents
comm.Allreduce(e_min, e_min_all, op=MPI.MIN)
comm.Allreduce(e_max, e_max_all, op=MPI.MAX)
```

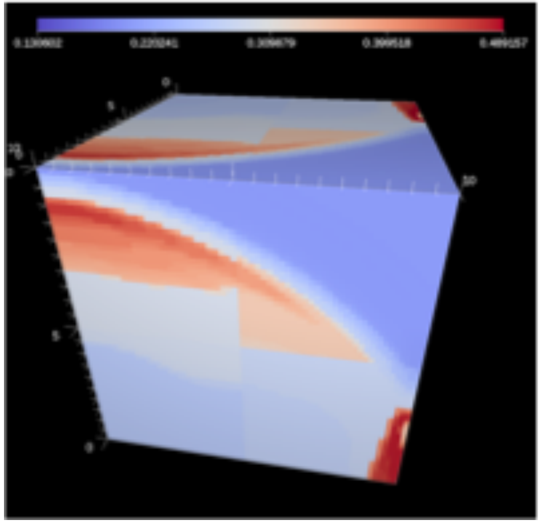
Custom Python Example: Distributed machine learning

- Harvey Mudd Clinic
  - 2 semester senior project
  - 4 team members
- Investigate distributed machine learning
  - Naïve bayes
  - Random forest
  - Mondrian forest
- Demonstrate proof-of-concept in-situ
  - Ascent + Cloverleaf3D + python extract

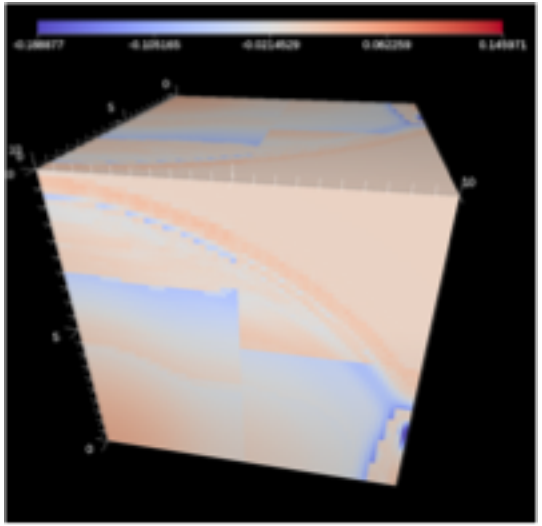
Proof-of-concept: In-situ distributed machine learning results



Actual Pressure



Predicted Pressure

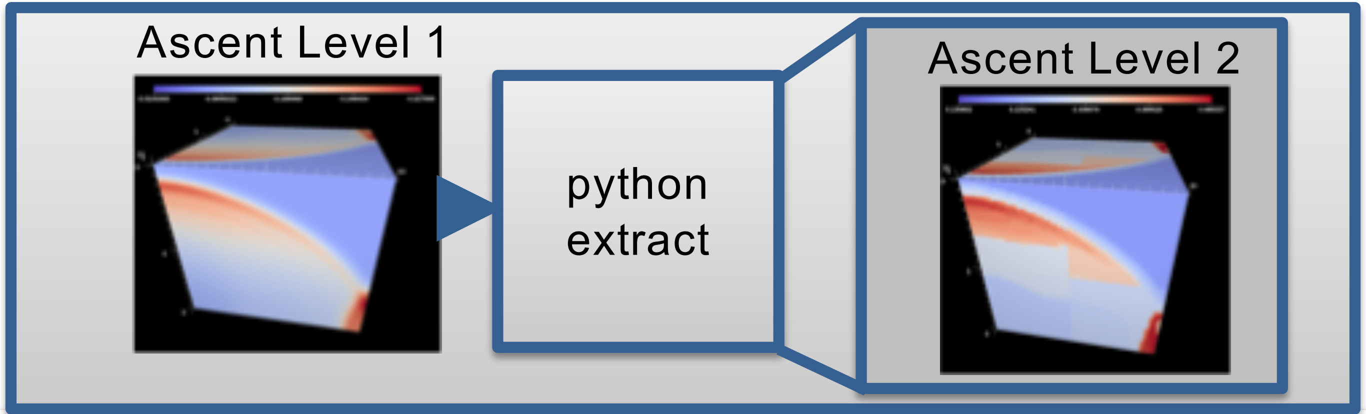


Difference

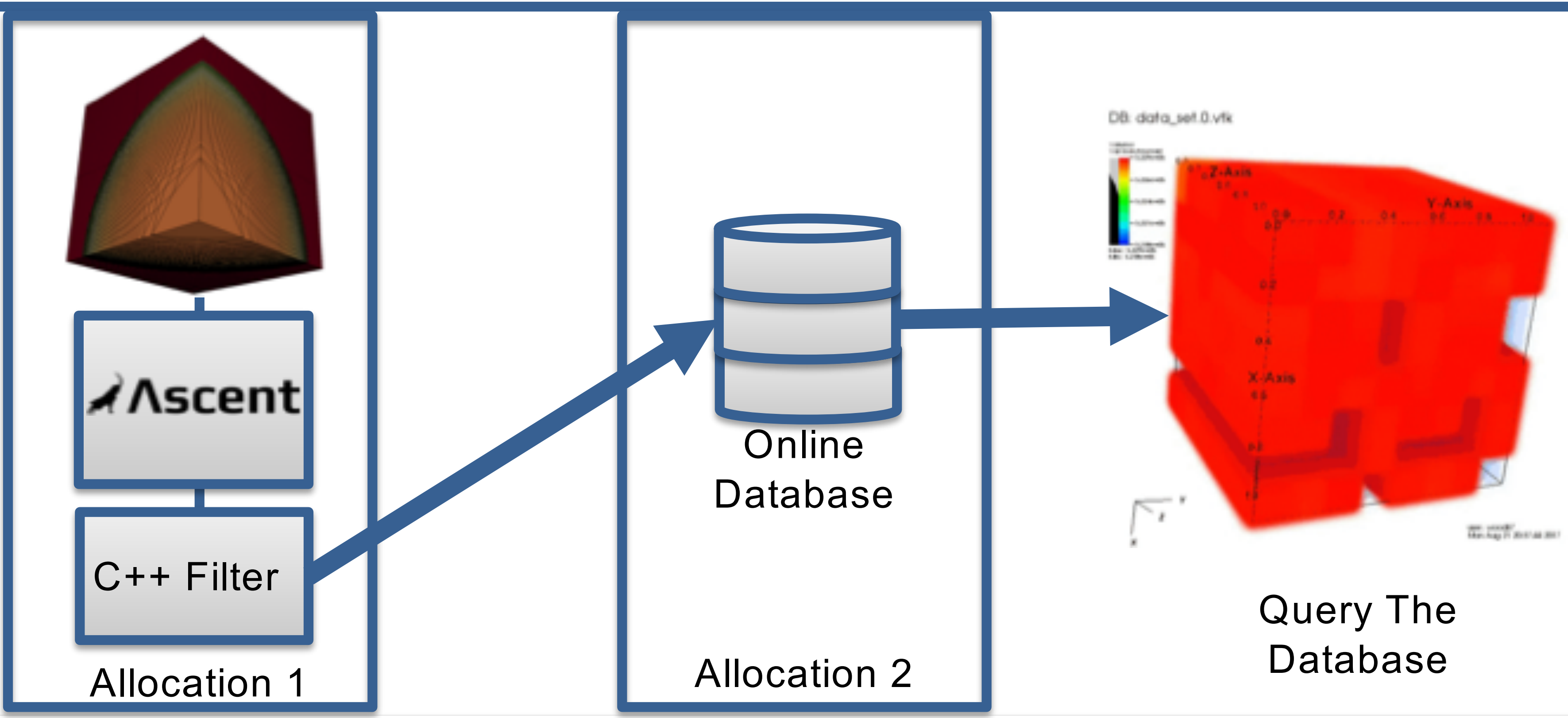
Inception: Using Ascent from a python extract

- We support python filters
  - Must edit the Ascent runtime to execute
  - How do I pass my data back to Ascent?

- Inside of a python extract
  - Create another instance of Ascent
  - Publish the new data set and actions



Custom C++ Filter: Gathering performance + mesh data

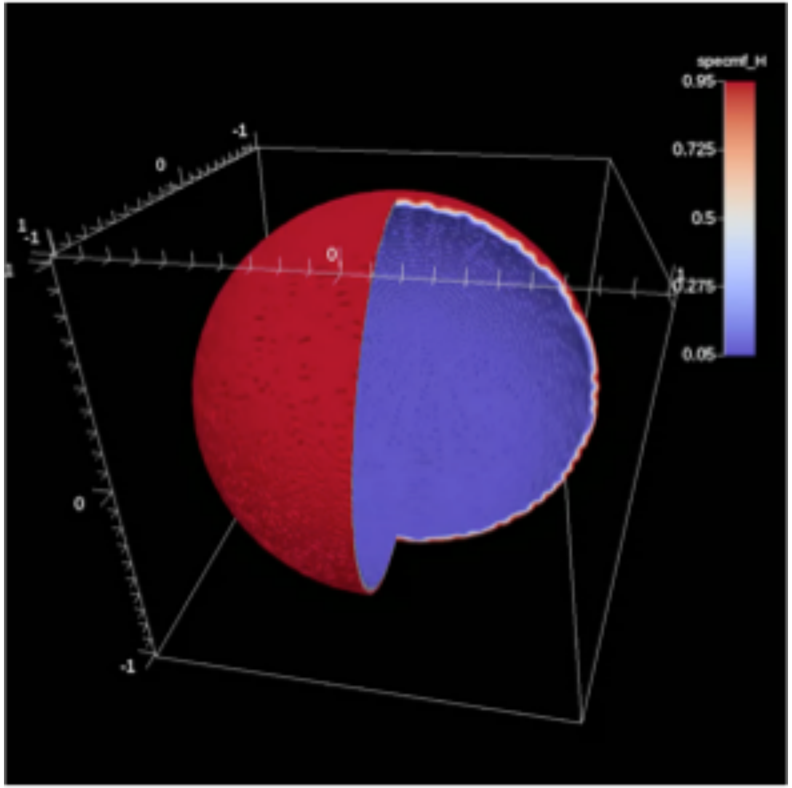


Integrating with LLNL codes: Ares on Sierra

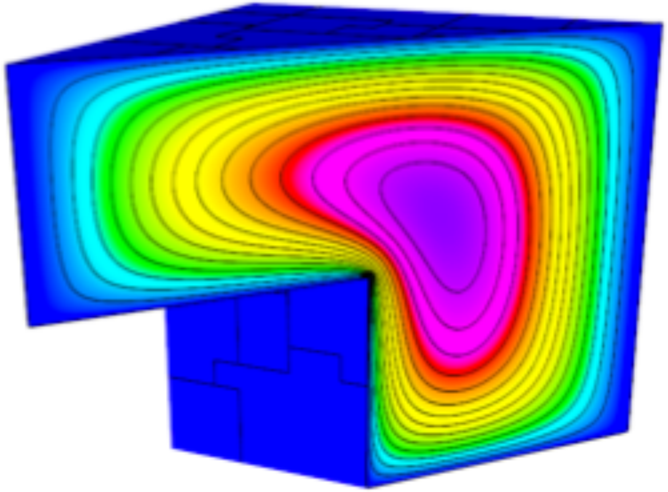
Resources	# of Nodes	Runtime (min)
256 V100 GPUs	64	213

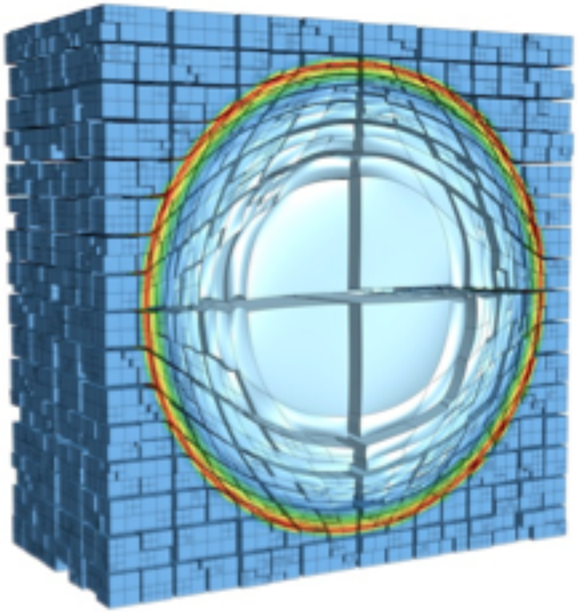
RT Mixing Layer in a Convergent Geometry

- 4π, 1.52B zones
- 29,375 cycles
- ALE Hydrodynamics
- Dynamic Species



Roadmap: Native MFEM rendering support (Summer 2018)



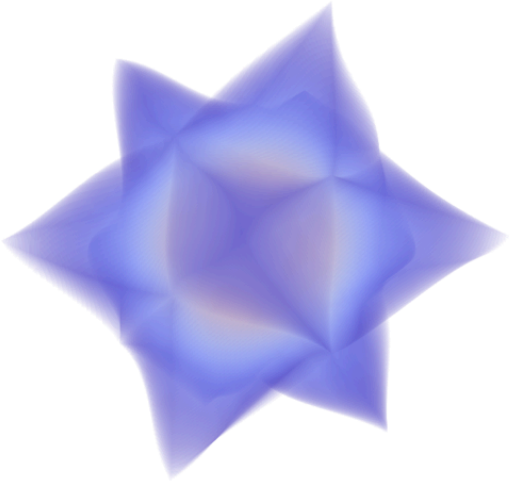


LLNL is developing Devil Ray to support native MFEM rendering

- Devil Ray is a library for direct ray tracing and volume rendering of high-order MFEM meshes

- Many-core capable, built using:
  – MFEM (http://mfem.org/)
  – RAJA (https://github.com/LLNL/RAJA)
  – Umpire (https://github.com/LLNL/Umpire)

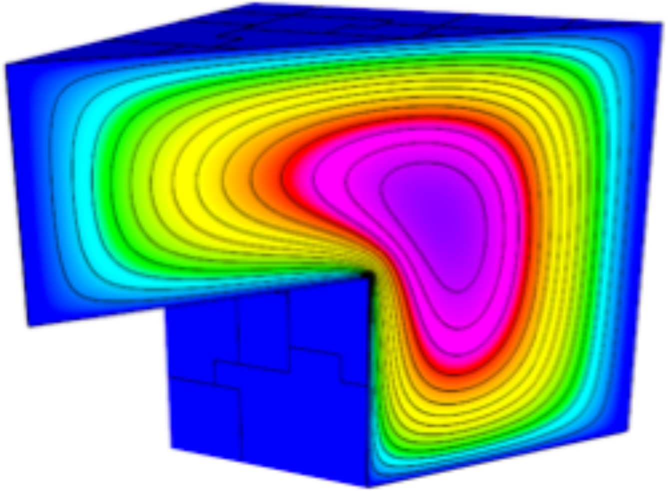
- Devil Ray will be integrated as a component of Ascent



Example render of a high-order mesh using Devil Ray

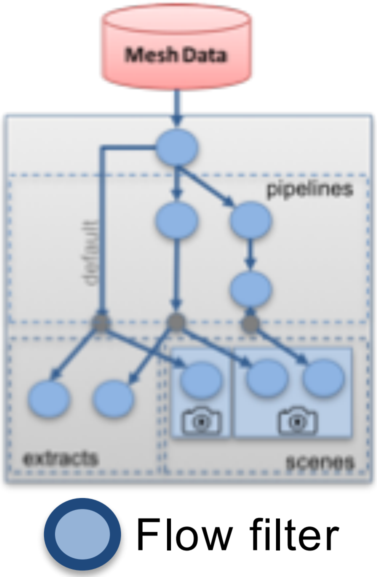
Why do we need direct MFEM support?

- Traditional visualization refines high-order meshes into meshes with linear elements
- Increases memory usage
  - Might not have the memory in-situ
- Low-order refine can miss important features
  - Ex, high-order contours

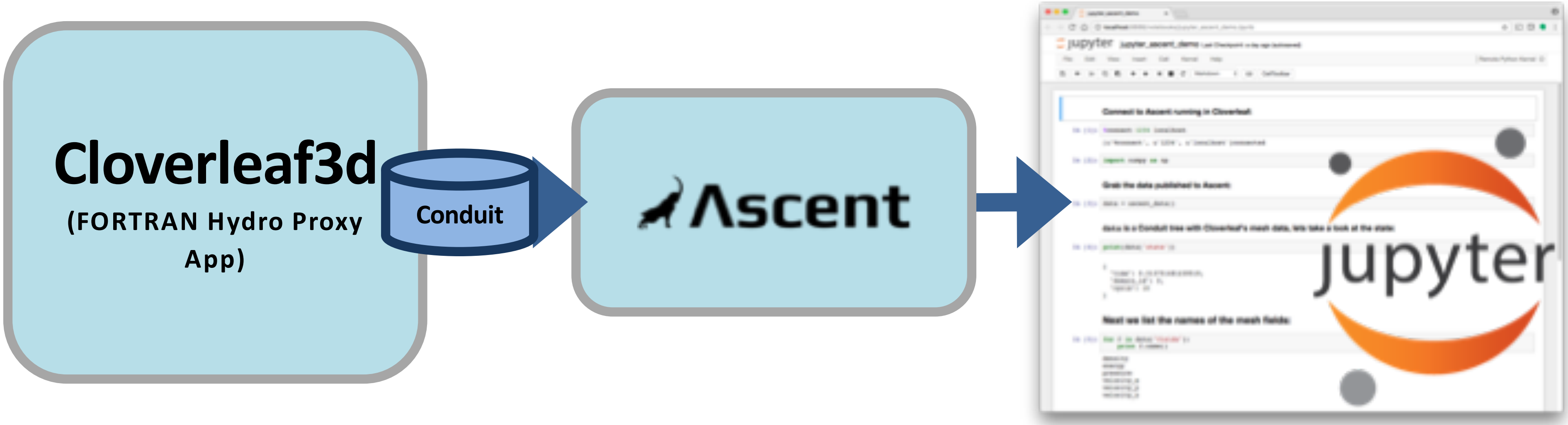


Roadmap: Triggers (Summer 2018 ?)

- Performing visualization every cycle takes time and resources away from the simulation
- We plan to add support for "Triggers":
  - When X happens do Y
- Examples:
  - Entropy in energy reaches some threshold
    - Save data or render
  - Not enough node memory
    - Examine data flow network and make adjustments
    - Resample data to fit within constraints

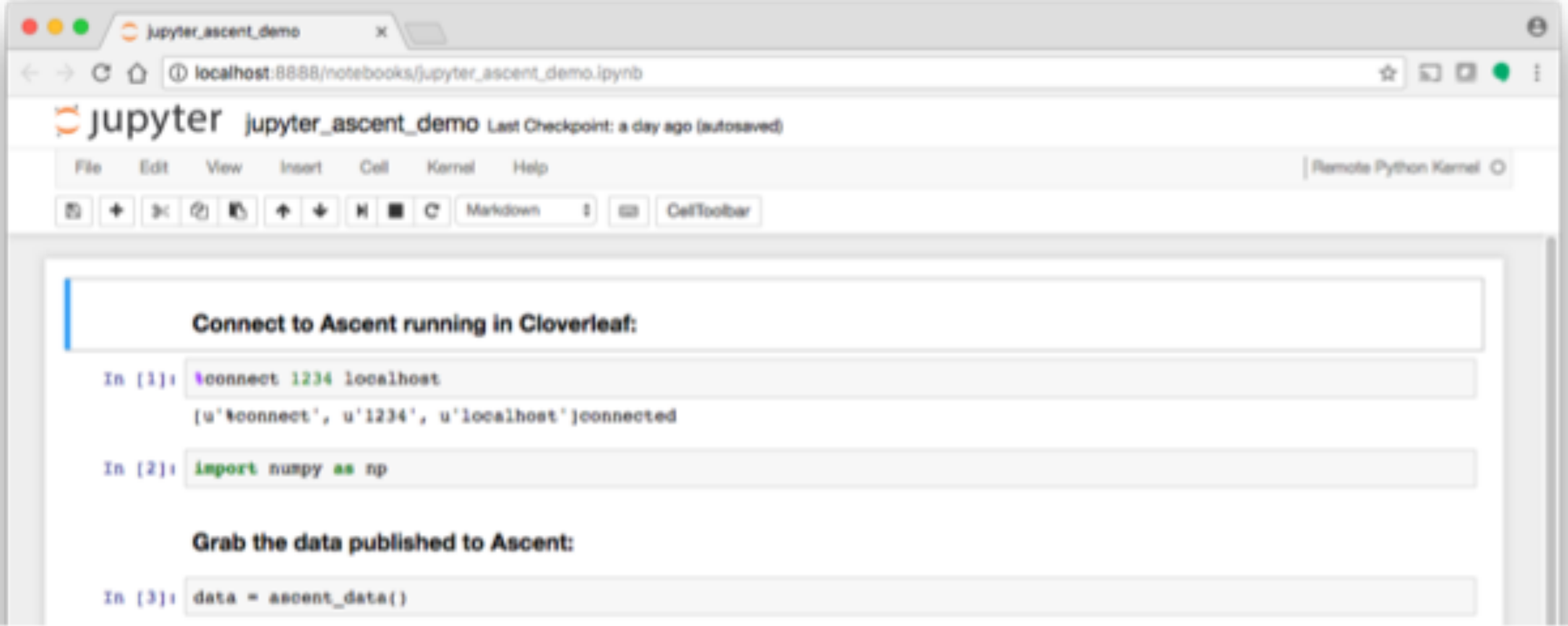


Roadmap: Jupyter Notebook Support (Fall 2018)



Ascent's Jupyter support will allow you to connect to a running simulation, access published data, run scripts, and yield back to the simulation.

Jupyter Notebook Demonstration



Jupyter Notebook Demonstration

data is a Conduit tree with Cloverleaf's mesh data, lets take a look at the state:

In [4]:
```
print(data['state'])
```

```
{
  "time": 0.313751481239519,
  "domain_id": 0,
  "cycle": 10
}
```

Next we list the names of the mesh fields:

In [5]:
```
for f in data["fields"]:
    print f.name()
```

```
density
energy
pressure
velocity_x
velocity_y
velocity_z
```

Why is Ascent Important?

- Designed for batch-focused in-situ analysis
- Helps connect your data with other ecosystems
- Light weight
  - Streamlined API
  - Low dependency count
- Targeting unique capabilities
  - Rendering of high-order meshes
- Easy to use and extend
  - Lowers barriers to custom analysis

Ascent is ready for common visualization use cases

- GitHub Repo: https://github.com/Alpine-DAV/ascent
- Docs: https://alpine-dav.github.io/ascent
- Try it out using Docker:
  - `docker pull alpinedav/ascent`
  - More info: http://ascent.readthedocs.io/en/latest/Tutorial.html
- Primary Contacts:
  - **Matt Larsen** larsen30@llnl.gov
  - **Cyrus Harrison** cyrush@llnl.gov

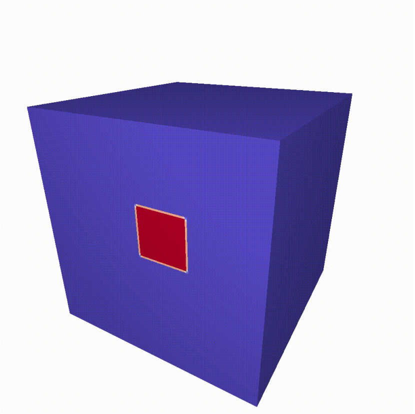
Acknowledgements

- This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy, Office of Science and the National Nuclear Security Administration.

- 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-CONF-737832)

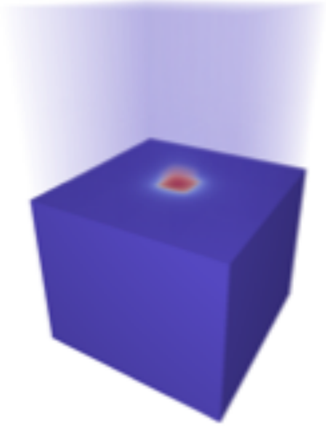
Questions?

Proxy-applications included with Ascent

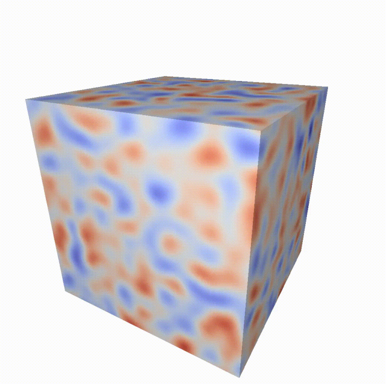


Cloverleaf3D

Lulesh



Kripke



Smooth Noise

Lawrence Livermore National Laboratory