The ALPINE In Situ Infrastructure: Ascending from the Ashes of Strawman

Matt Larsen, James Ahrens, Utkash Ayachit, Eric Brugger,
Hank Childs, Berk Geveci, and Cyrus Harrison

LLNL-PRES-741488

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

Overview

- **ALPINE Project Overview**
- VTK-h Overview
- Ascent Concepts and Examples

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

The ALPINE project will enable in-situ visualization and analysis for exascale simulation applications

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

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

This talk focuses on ALPINE infrastructure

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

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

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 builds on VTK-m and the Strawman proxy application

- **VTK-h**
  - A distributed-memory parallel layer for algorithms that use VTK-m for node-level parallelism

- **Ascent**
  - A simplified in-situ interface: **Ascent API**
    - An evolution of the Strawman Interface
  - A flyweight in-situ runtime: **Ascent Runtime**
    - An evolution of the Strawman Runtime

Ascent's current capabilities

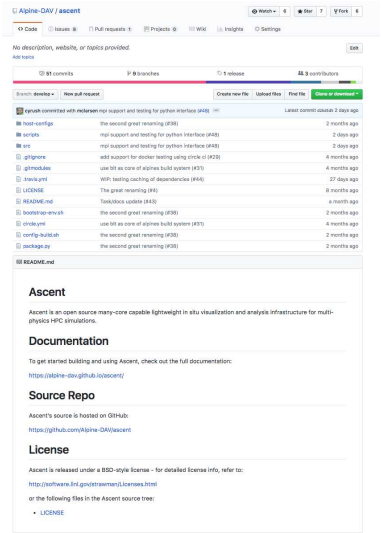
- Rendering
  - Pseudocolor plot
  - Volume plot
  - Mesh plot
- Output
  - Blueprint hdf5
  - ADIOS (currently only rectilinear meshes)
- Transforms / Filters
  - Contour
  - Clip
  - Threshold

- Behind the API is a data flow network
  - Manages pipelines of filters
  - Composes multiple plots together
  - Saves results
- Built-in Proxy Simulations
  - Lulesh, Cloverleaf, Kripke, Simplex Noise
- Language bindings

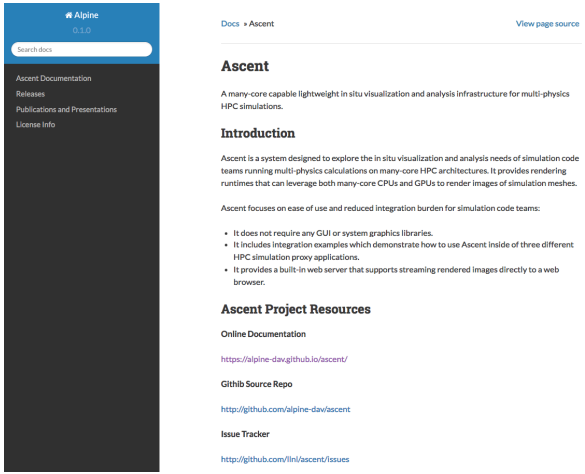
C/C++ python™ Fortran

How to get Ascent

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



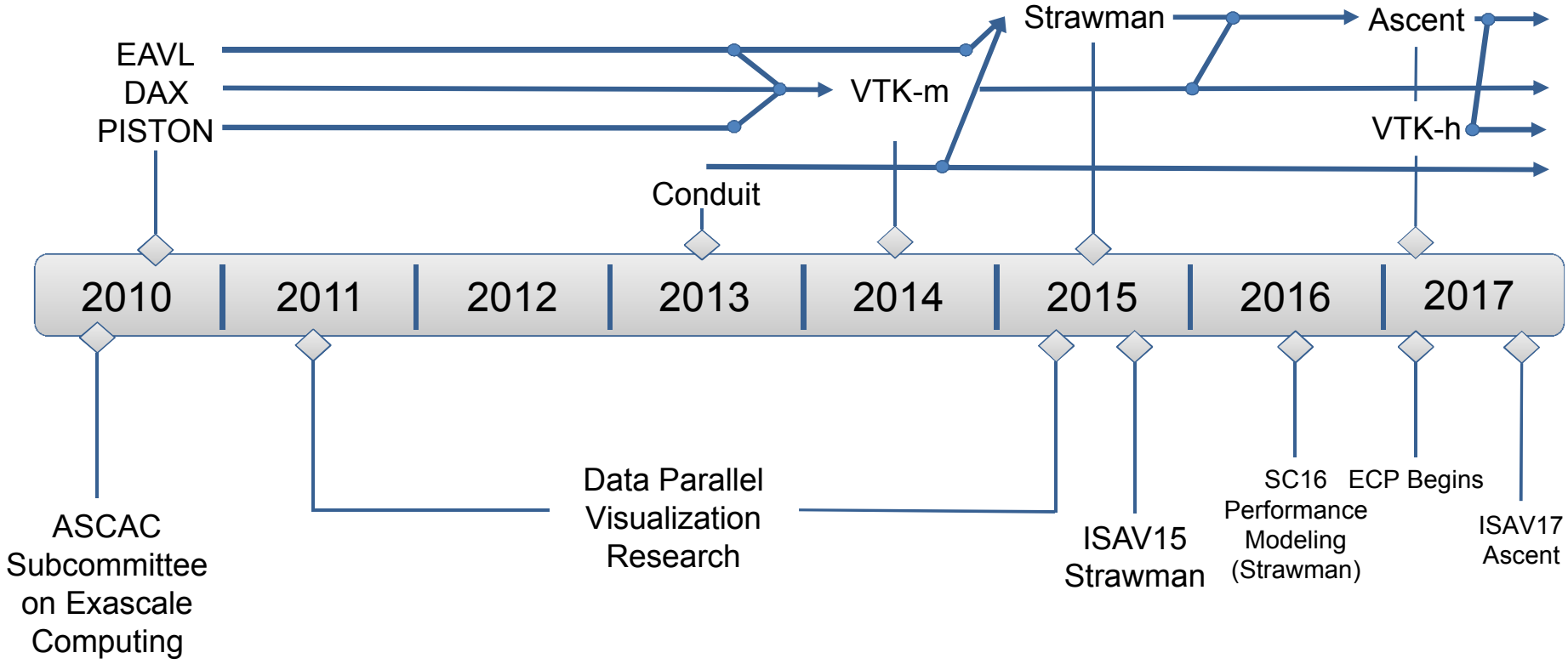
- Documentation:
  - https://alpine-dav.github.io/ascent



How to build Ascent

- Spack package under development
  - https://github.com/Alpine-DAV/spack
- Docker container:
  - src/examples/docker/ubuntu
- Build from source

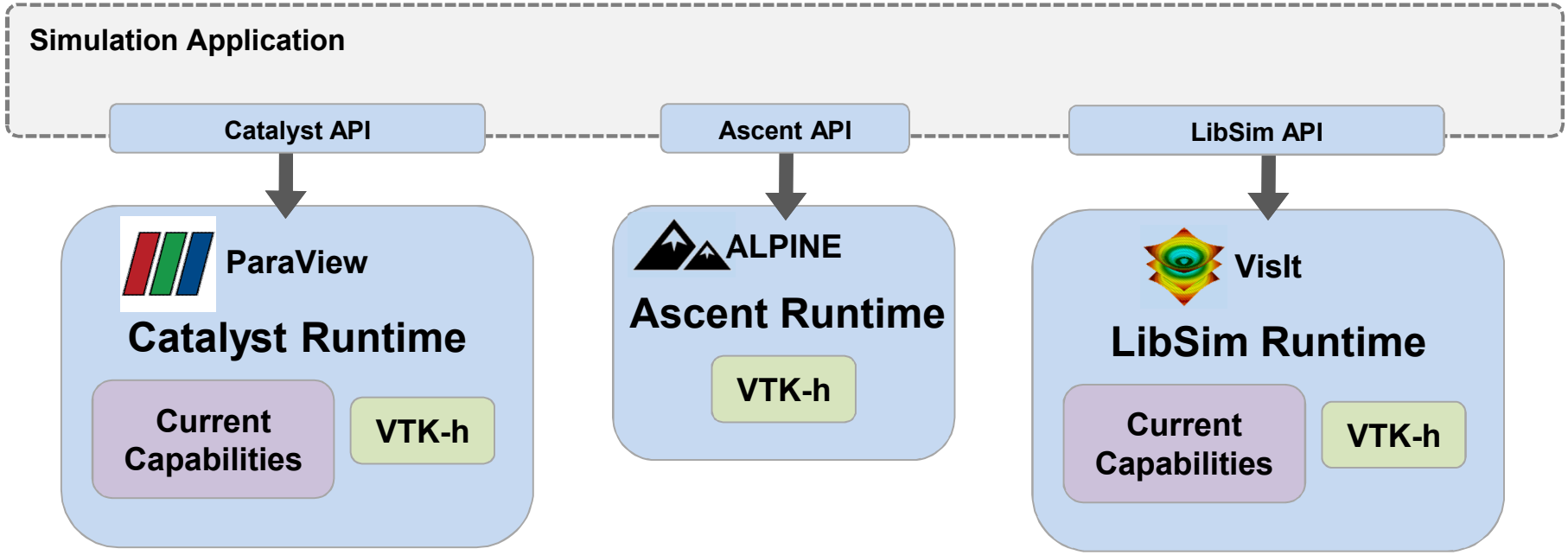
So, how did we get here?



Overview

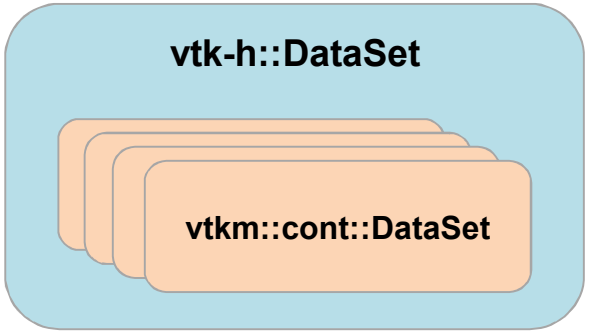
- ALPINE Project Overview
- **VTK-h Overview**
- Ascent Concepts and Examples

VTK-h enables sharing new visualization and analysis algorithms across multiple in-situ tools



VTK-h is a distributed-memory parallel layer on top of VTK-m

- VTK-h data model:
  - A collection of VTK-m data sets
    - VTK-m data sets hold “Domains” or ”Blocks” of the full data set
  - Distributed memory methods (e.g., GetGlobalScalarRange())



- Clear roles for VTK-m and VTK-h:
  - VTK-m (”m” = many-core): Data parallel node-local algorithms
  - VTK-h (“h” = hybrid) : Distributed memory coordination
- VTK-h provides a simple filter interface
- VTK-h does ***not*** provide an execution model for chaining together filters

VTK-h provides a simple filter interface

- Input and output:
  - void vtkh::SetInput(vtkh::DataSet *)
  - vtkh::DataSet * vtkh::GetOutput()

- Update()
  - Internally calls:
    - PreExecute()
    - DoExecute()
    - PostExecute()

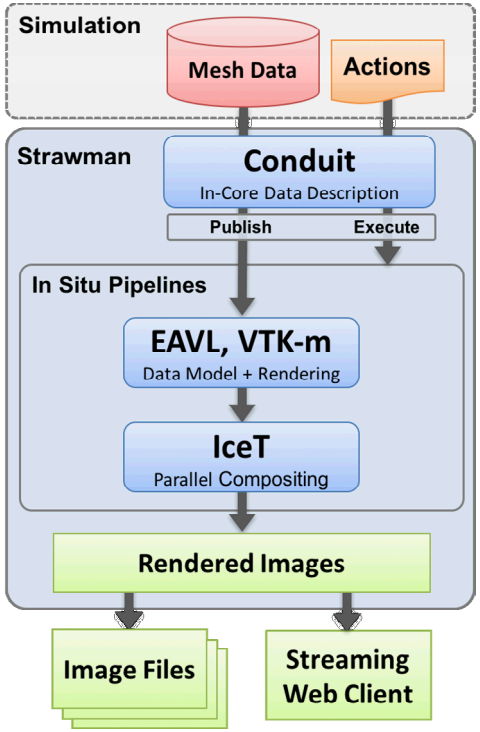
- Executing a filter passes through all fields unless specified

Overview

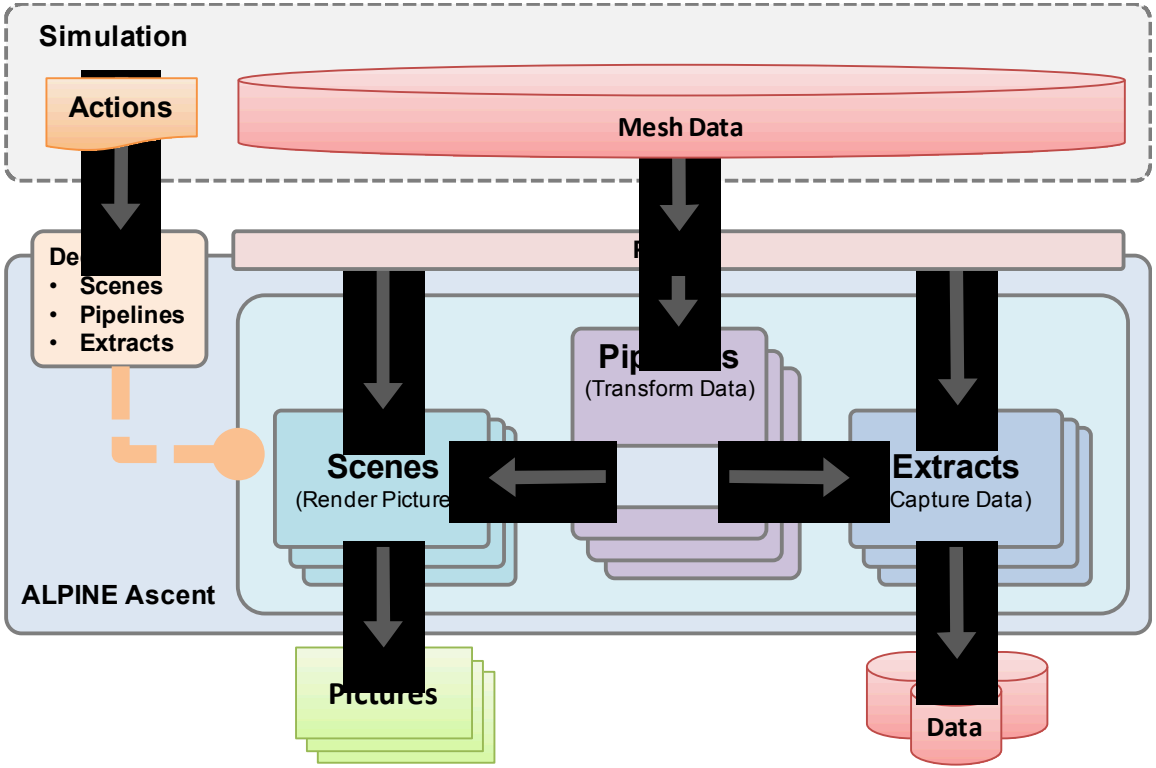
- ALPINE Project Overview
- VTK-h Overview
- **Ascent Concepts and Examples**

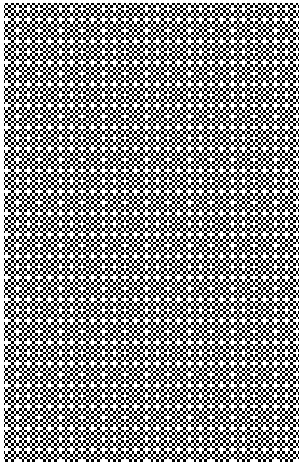
What did Strawman provide?

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



Ascent is a new flyweight in-situ runtime that supports rendering and analysis



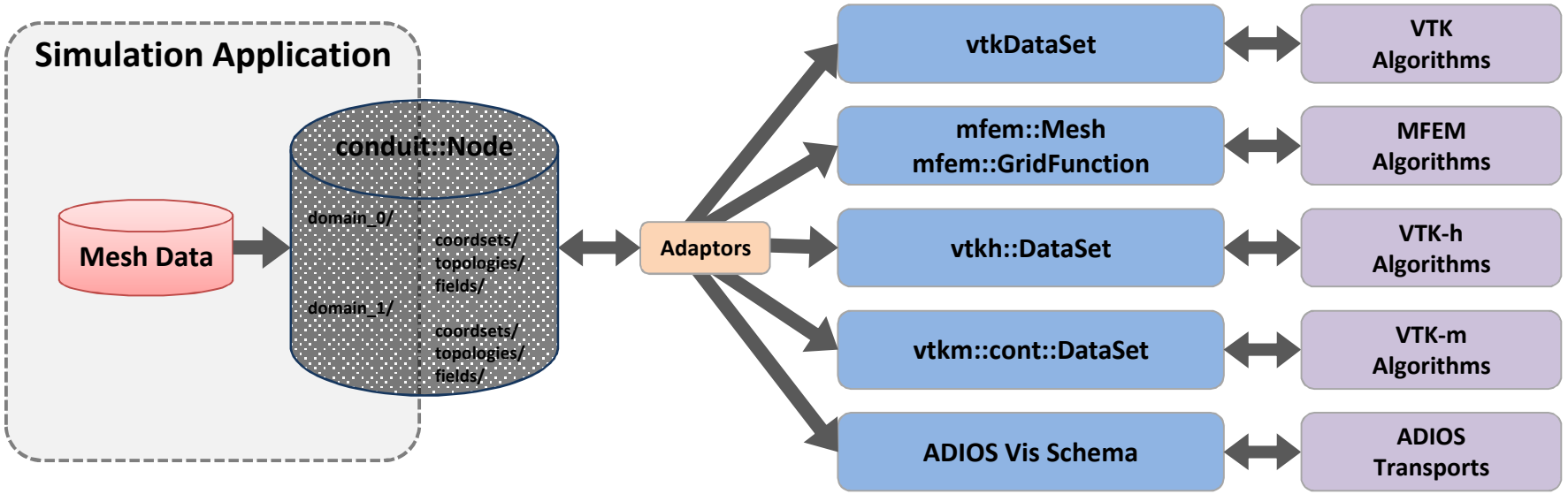


We are evolving Strawman into the Ascent API and Runtime

- **Ascent API**
  - Adopts Strawman Interface (*Open*, *Publish*, *Execute*, *Close*)
  - Mesh data is published using the Conduit Mesh Blueprint
  - *Execute* supports a new set ”actions” that allows users to:
    - Make pictures
    - Capture data
    - Transform data

- **Ascent Runtime**
  - Runtime that executes a DAG the implements the requested actions using Flow + VTK-h

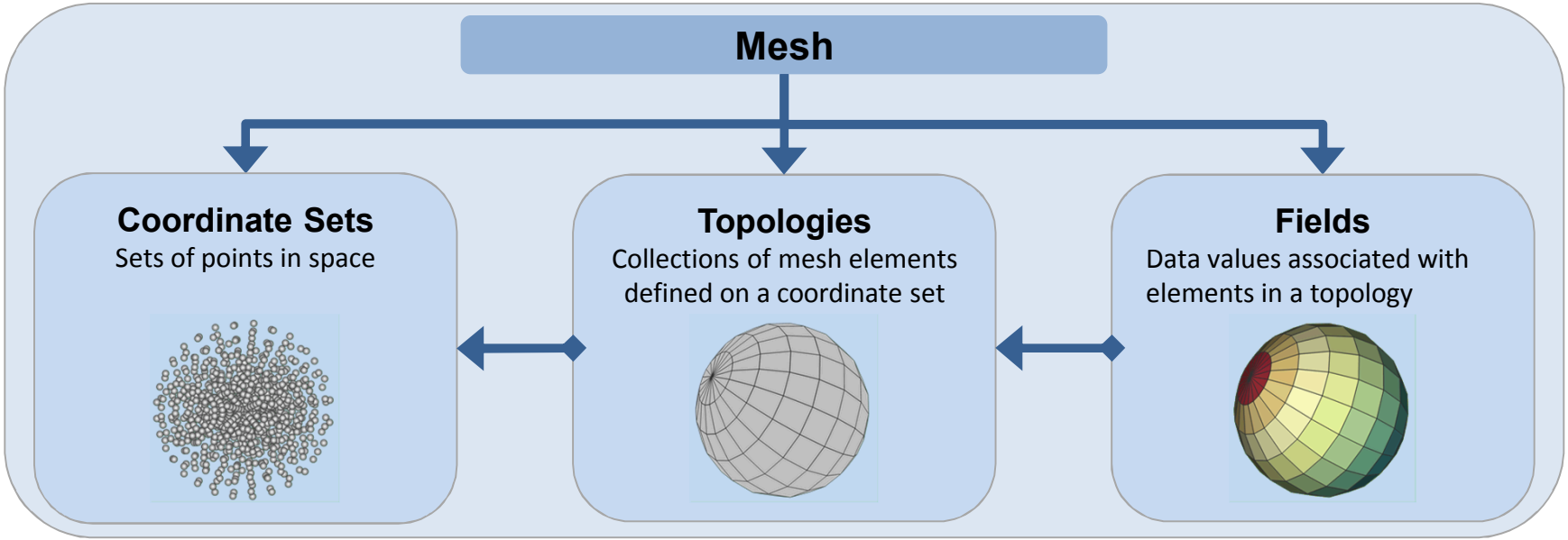
Mesh data is published to Ascent via Conduit Mesh Blueprint



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 fully feature 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

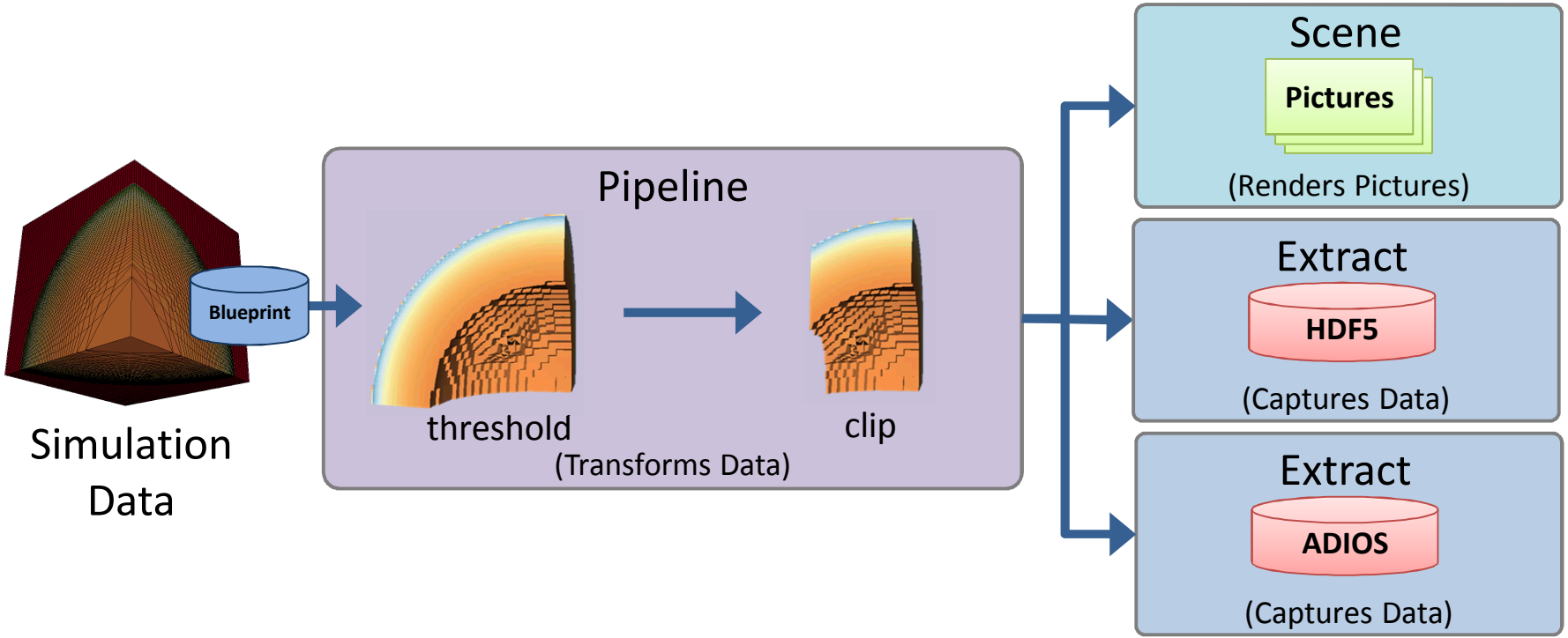
Example: Defining a uniform grid using Mesh Blueprint

```
conduit::Node node;
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);
```

Ascent's API provides three key building blocks

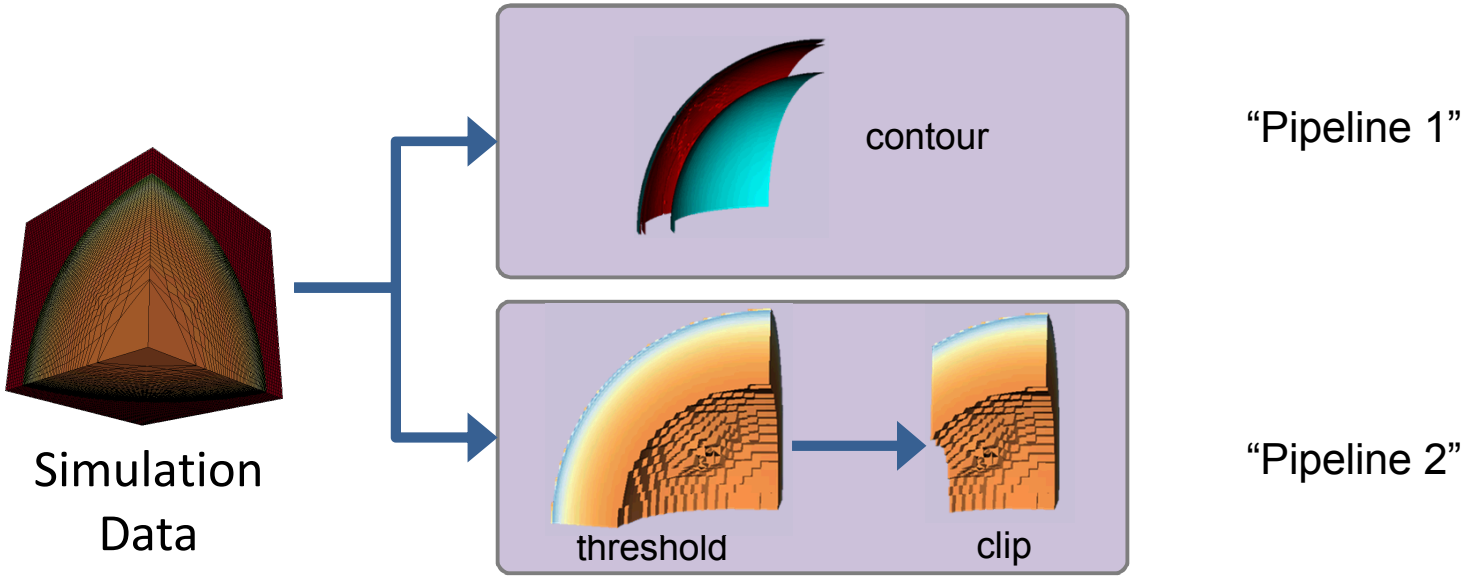
- Pipelines (transform data):
  - Allows users to describe how they want to transform their data
- Scenes(make pictures):
  - Allows users to describe the pictures they want to create
- Extracts (capture data):
  - Allows users to describe how they want capture data

Ascent end-to-end conceptual example



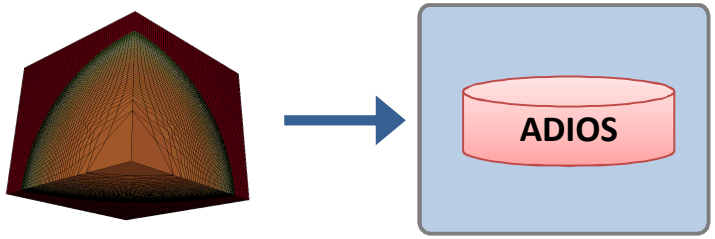
A pipeline is a series data transformations (i.e., filters)

- Ascent allows an arbitrary number of pipelines to be described

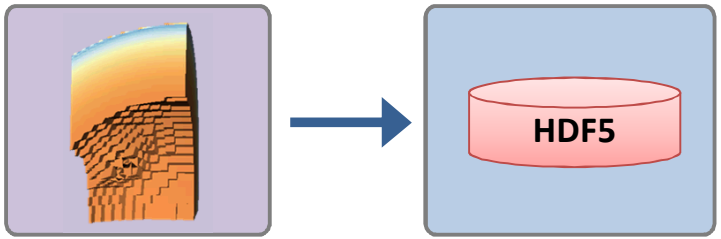


An extract is a way to capture data for use outside of Ascent

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

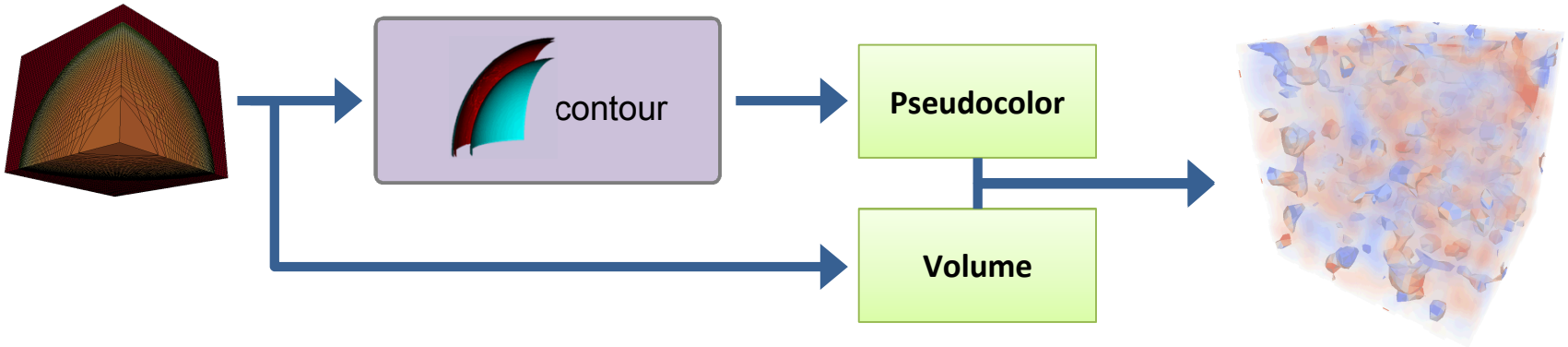


  - Export pipeline results to HDF5, ADIOS, etc.



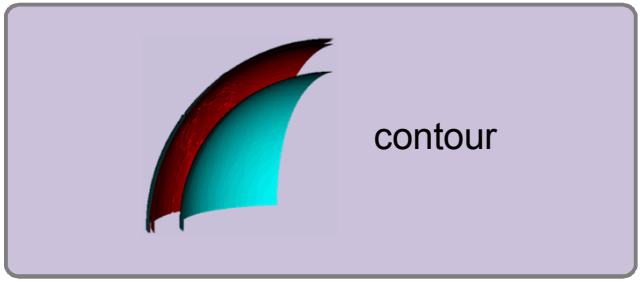
A scene is a way to render pictures

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



Example: Specifying pipelines

```
conduit::Node pipelines;
// pipeline 1
pipelines["pl1/f1/type"] = "contour";
// filter knobs
conduit::Node &contour_params = pipelines["pl1/f1/params"];
contour_params["field"] = "noise";
contour_params["iso_values"] = {0.0, 0.5};
```

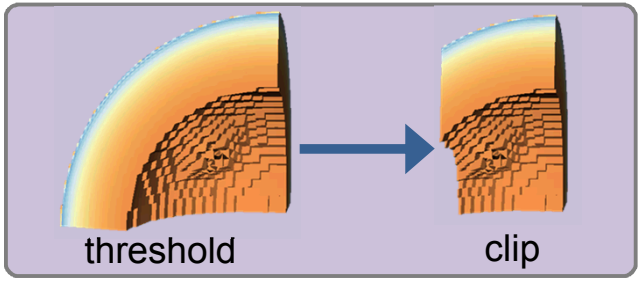


Pipeline 1
(pl1)

Example: Specifying pipelines

```
// pipeline 2
pipelines["pl2/f1/type"] = "threshold";
// filter knobs
conduit::Node &thresh_params = pipelines["pl2/f1/params"];
thresh_params["field"]  = "noise";
thresh_params["min_value"] = 0.0;
thresh_params["max_value"] = 0.5;

pipelines["pl2/f2/type"]   = "clip";
// filter knobs
conduit::Node &clip_params = pipelines["pl2/f2/params/"];
clip_params["topology"] = "mesh";
clip_params["sphere/center/x"] = 0.0;
clip_params["sphere/center/y"] = 0.0;
clip_params["sphere/center/z"] = 0.0;
clip_params["sphere/radius"]   = .1;
```

Filter order is executed in the order they are declared.



Pipeline 2
(pl2)

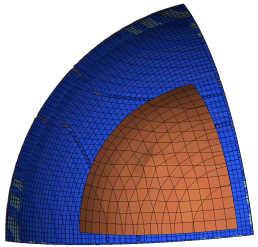
Example: Specifying scenes

```
conduit::Node scenes;

// scene 1, a single volume plot w/ default camera and output res
scenes["scene1/plots/plot1/type"]         = "volume";
scenes["scene1/plots/plot1/params/field"] = "noise";
```

scene1

Example: Specifying scenes

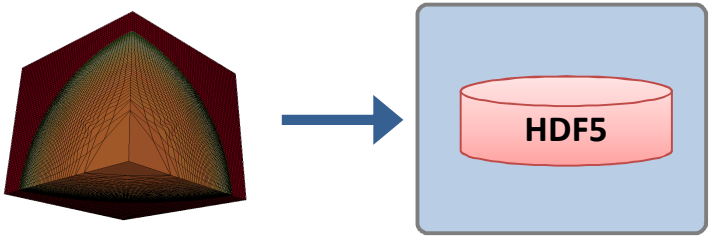
```
// scene 2, a pc plot + mesh plot of pipeline 'pl1'
// w/ default camera and output res
scenes["scene2/plot1/type"]          = "pseudocolor";
scenes["scene2/plot1/pipeline"]      = "pl1";
scenes["scene2/plot1/params/field"] = "noise";

scenes["scene2/plot2/type"]     = "mesh";
scenes["scene2/plot2/pipeline"] = "pl1";
```



scene2

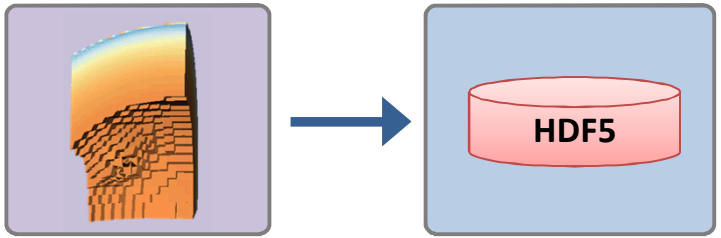
Example: Specifying extracts

```
conduit::Node extracts;

// use default pipeline (original mesh)
extracts["ex1/type"] = "hdf5";
```



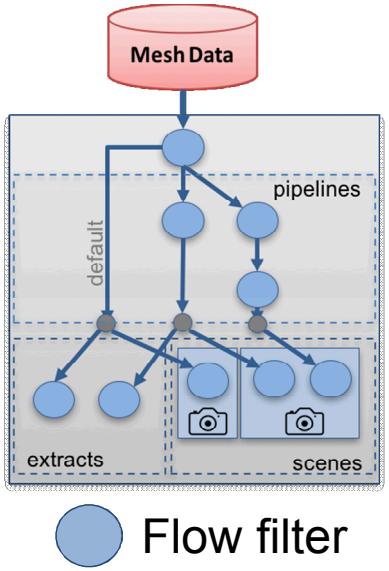
Example: Specifying extracts

```
// use the result of pipeline 2
extracts["ex2/type"] = "hdf5";
extracts["ex2/pipeline"]     = "p12";
```



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

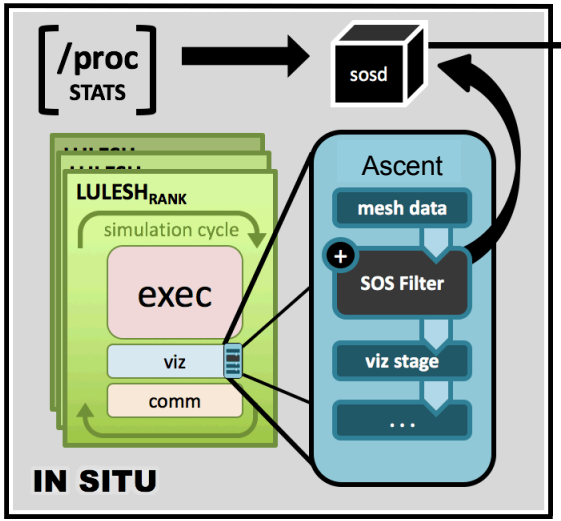
- *Recall:* VTK-h does not provide an execution model
- ParaView and VisIt have their own rich execution models
- VTK-h features in those tools will be exposed through their existing execution models
- Ascent needs a basic execution model to support complex user requests

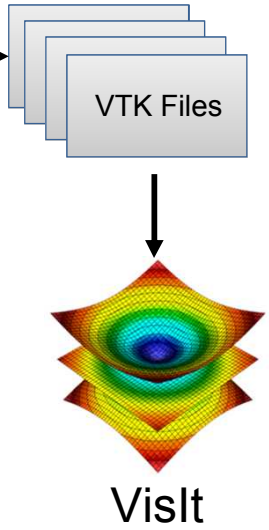


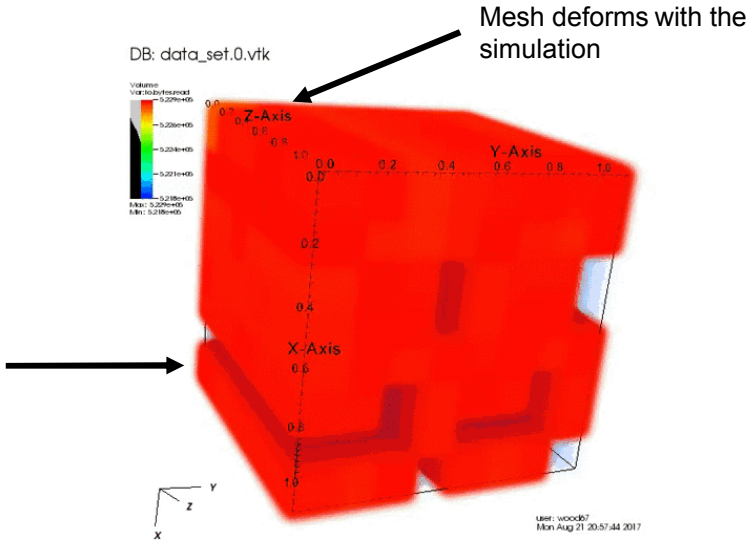
Flow provides the execution model for ALPINE Ascent.

Flow filters are flexible and customizable

- Compile and register filters outside of Ascent
- Example: SOS Flow filter (VPA 17)
  - Mapping performance data onto the mesh through Ascent







Acknowledgements

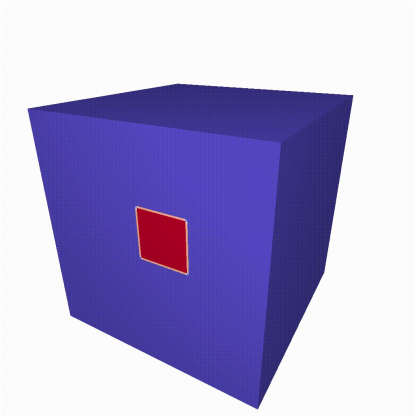
- Presentation released under
  - LLNL-PROC-741488

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

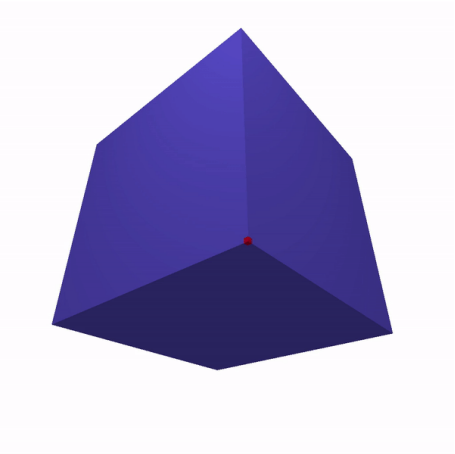
- This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52- 07NA27344. Lawrence Livermore National Security, LLC (LLNL-CONF-737832)

Questions?

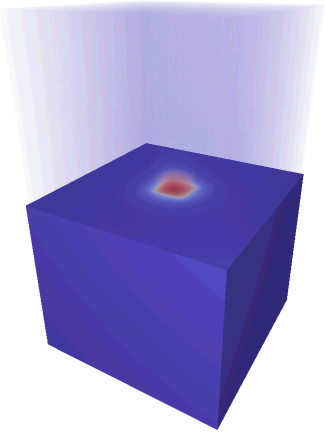
Proxy-applications included with Ascent



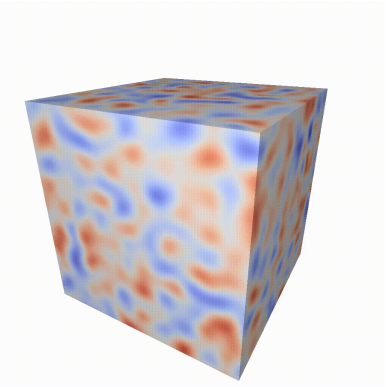
Cloverleaf3D



Lulesh



Kripke



Smooth Noise