In Situ Analysis and Visualization with Ascent and ParaView Catalyst

[Tutorial Introduction]

SC23 Tutorial
Monday November 13th, 2023

Cyrus Harrison, Lawrence Livermore National Laboratory (LLNL)

Jean M. Favre, Swiss National supercomputing Centre (CSCS)

Corey Wetterer-Nelson, Kitware Inc.

Nicole Marsaglia, Lawrence Livermore National Laboratory (LLNL)

In Situ Analysis and Visualization with Ascent and ParaView Catalyst

SC23 Tutorial

Monday November 13th, 2023

Cyrus Harrison (LLNL)
Jean Favre (CSCS)
Corey Wetterer-Nelson (Kitware)
Nicole Marsaglia (LLNL)

LLNL-PRES-856413
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

Welcome! Today you will learn:

- About in situ scientific visualization paradigms and use cases
- How to use ***Conduit***, a shared interface for describing in-memory mesh-based data
- How to use two in situ HPC scientific visualization tools:

  Ascent, ParaView Catalyst 2

CONDUIT

http://software.llnl.gov/conduit

Ascent

http://ascent-dav.org

ParaView Catalyst

https://catalyst-in-situ.readthedocs.io/en/latest/

Tutorial Outline

- **Introduction** [Lecture, 30 minutes]
  - Scientific Visualization and In Situ Processing Concepts
  - Ascent and Catalyst Project Overviews
- **Hands-on Cloud Environment Setup** [Hands-on, 10 minutes]
- **Using Conduit to Describe Mesh Data** [Hands-on, 50 minutes]
- ***Break*** [30 minutes]
- **Learning Ascent** [Hands-on, 40 minutes]
- **Learning Catalyst** [Hands-on, 40 minutes]
- **Closing Remarks and Questions** [10 minutes]

Introduction:
Scientific visualization and In Situ Processing Concepts

Scientific visualization tools transform, analyze, and render mesh-based data from HPC simulations

Mesh Data

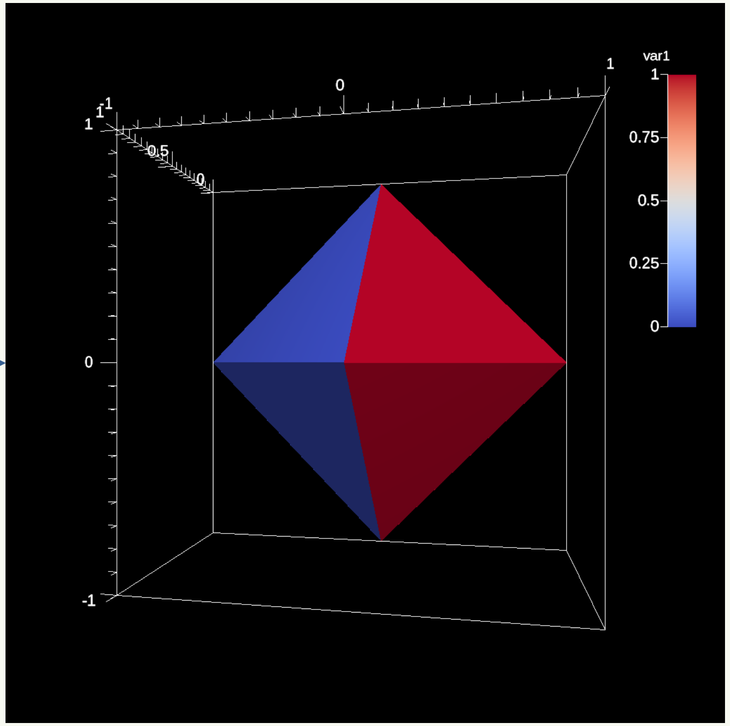
```
coordsets:
  coords:
    type: "explicit"
    values:
      x: [-1.0, 0.0, 0.0, 0.0, 1.0]
      y: [0.0, -1.0, 0.0, 1.0, 0.0]
      z: [0.0, 0.0, 1.0, 0.0, 0.0]
topologies:
  topo:
    type: "unstructured"
    coordset: "coords"
    elements:
      shape: "tet"
      connectivity: [0, 1, 3, 2, 4, 3, 1, 2]
fields:
  density:
    association: "element”
    topology: ”topo"
    values: [1.0, 2.0]
```

Actions

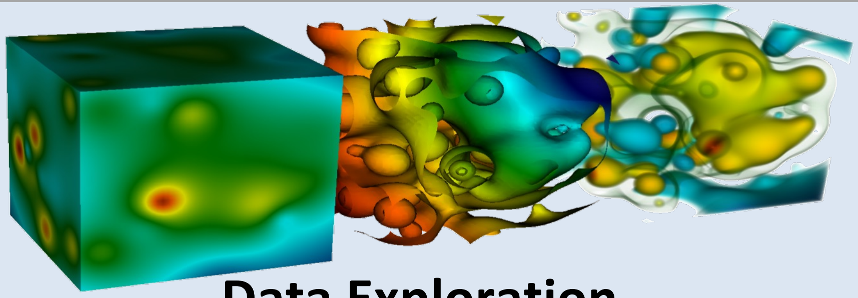
```
-
  action: "add_scenes"
  scenes:
    my_scene:
      plots:
        my_plot:
          type: "pseudocolor"
          field: "var1"
```

Visualization Tool

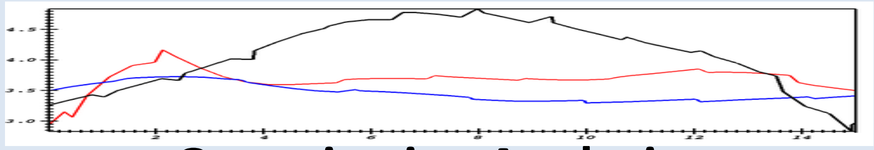
Transform, Analyze, Render



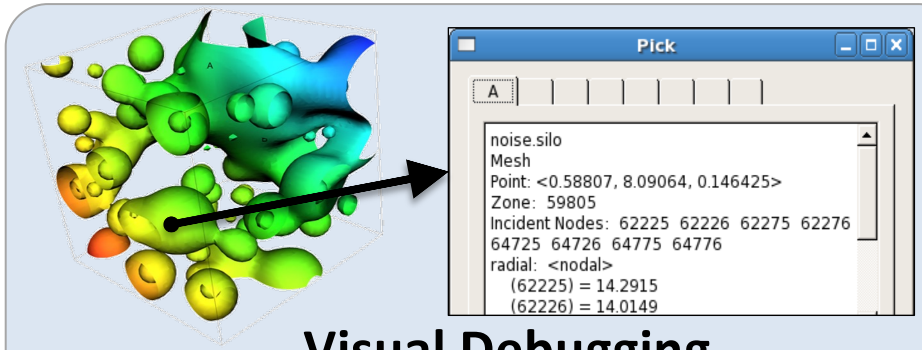
Scientific visualization tools support a wide range of use cases



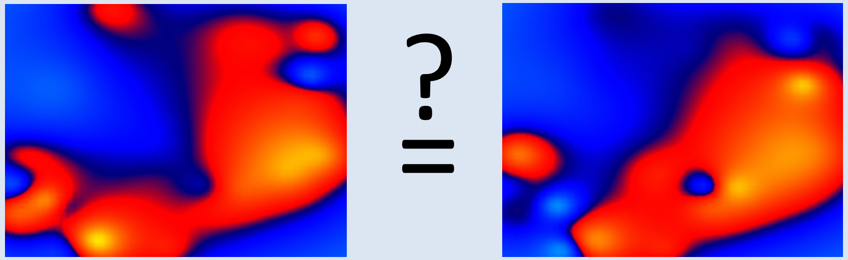
Data Exploration



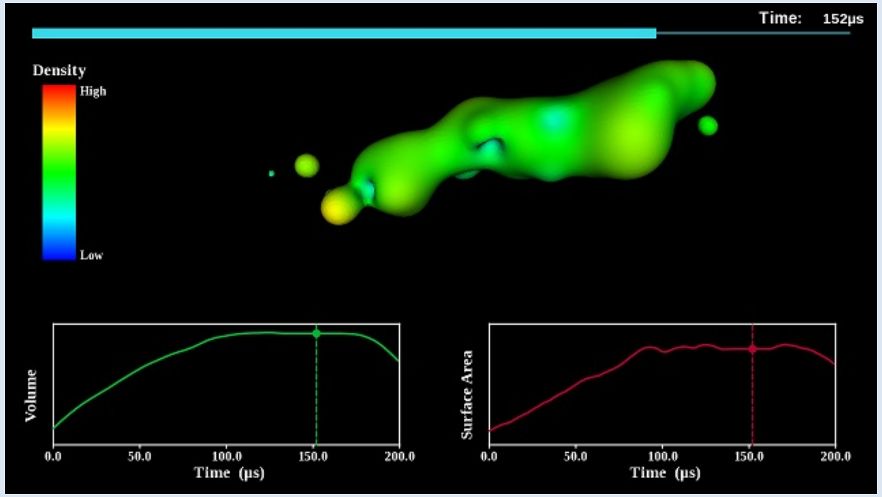
Quantitative Analysis



Visual Debugging

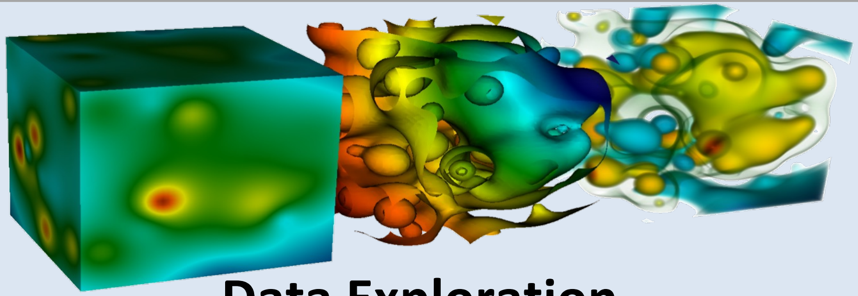


Comparative Analysis

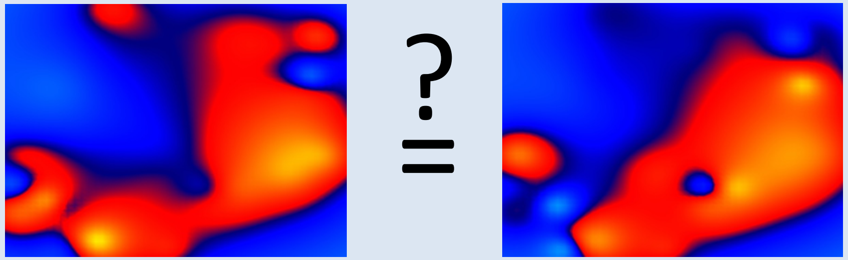


Presentation Graphics

Scientific visualization tools support a wide range of use cases

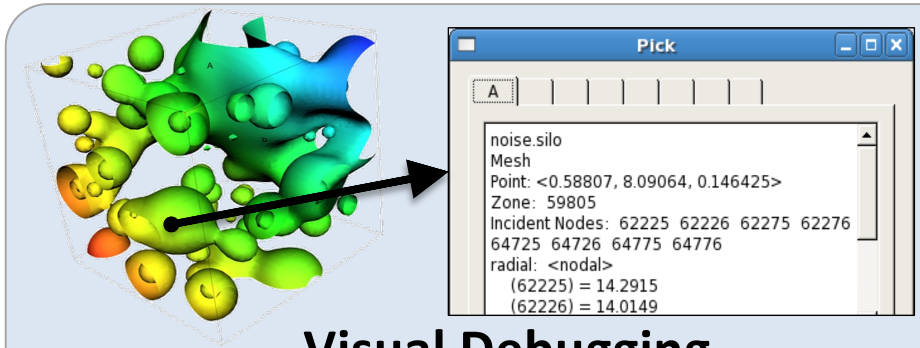


Data Exploration

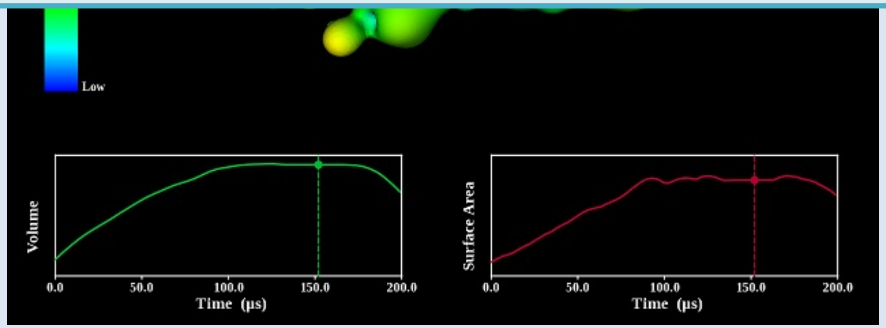


Comparative Analysis

These tools are used daily by scientists to digest and understand HPC simulation results



Visual Debugging



Presentation Graphics

Scientific visualization tools are used both *post hoc* and *in situ*

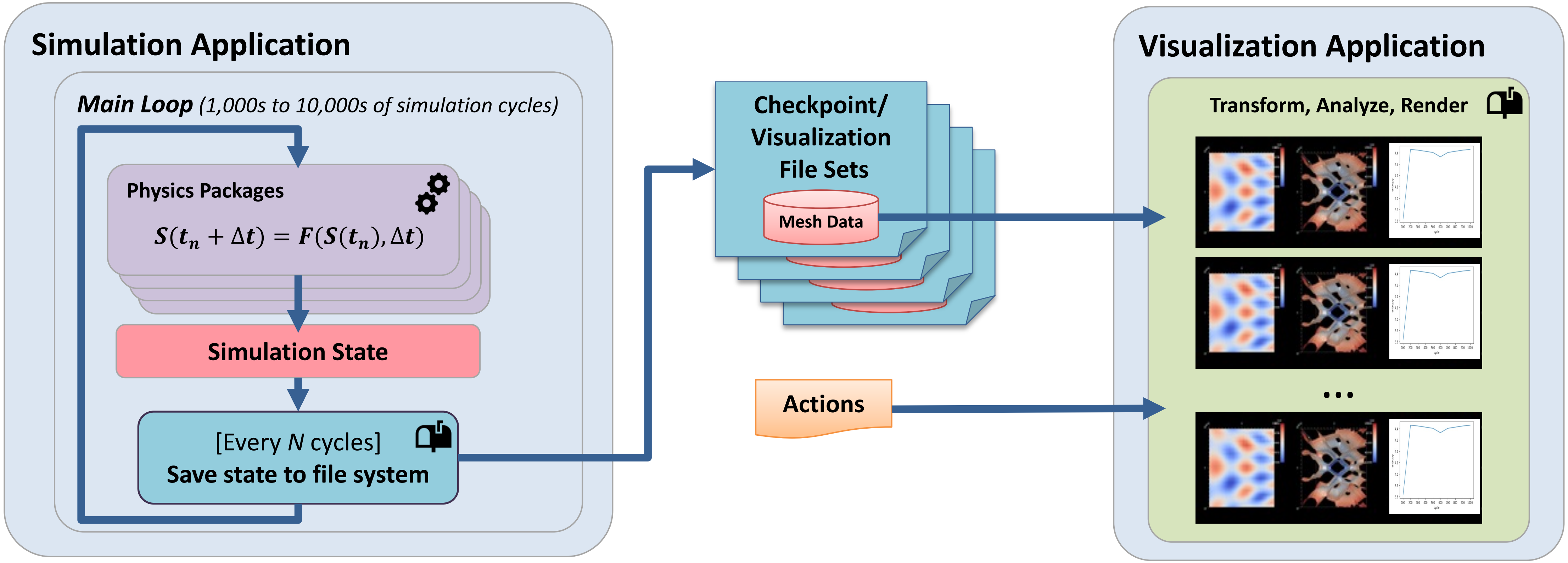
Post Hoc

Simulation data is processed after the simulation is run using distinct compute resources.

In Situ

Simulation data is processed while it is generated, sharing compute resources with the simulation application.

Post Hoc visualization is the most widely used paradigm to process simulation results

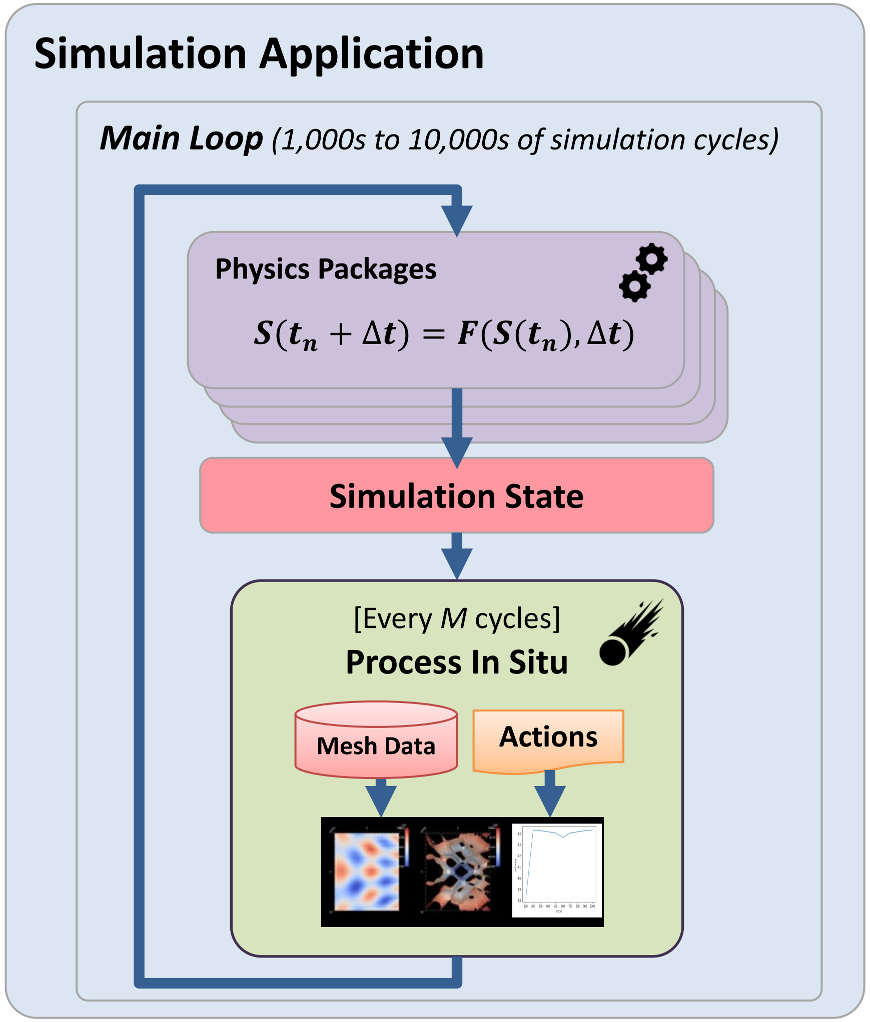


In situ processing expands the data we can access

- In situ tools couple visualization and analysis routines with the simulation application (avoiding file system I/O)

- Pros:
  - No or greatly reduced I/O vs post hoc processing
  - Can access all simulation data
  - Computational power is readily available
  - Results are ready after simulation completes

- Cons:
  - More difficult when lacking a priori knowledge of what to visualize/analyze
  - Increasing complexity
  - Constraints (memory, network)



HPC Compute vs I/O speed ratios can favor in situ processing

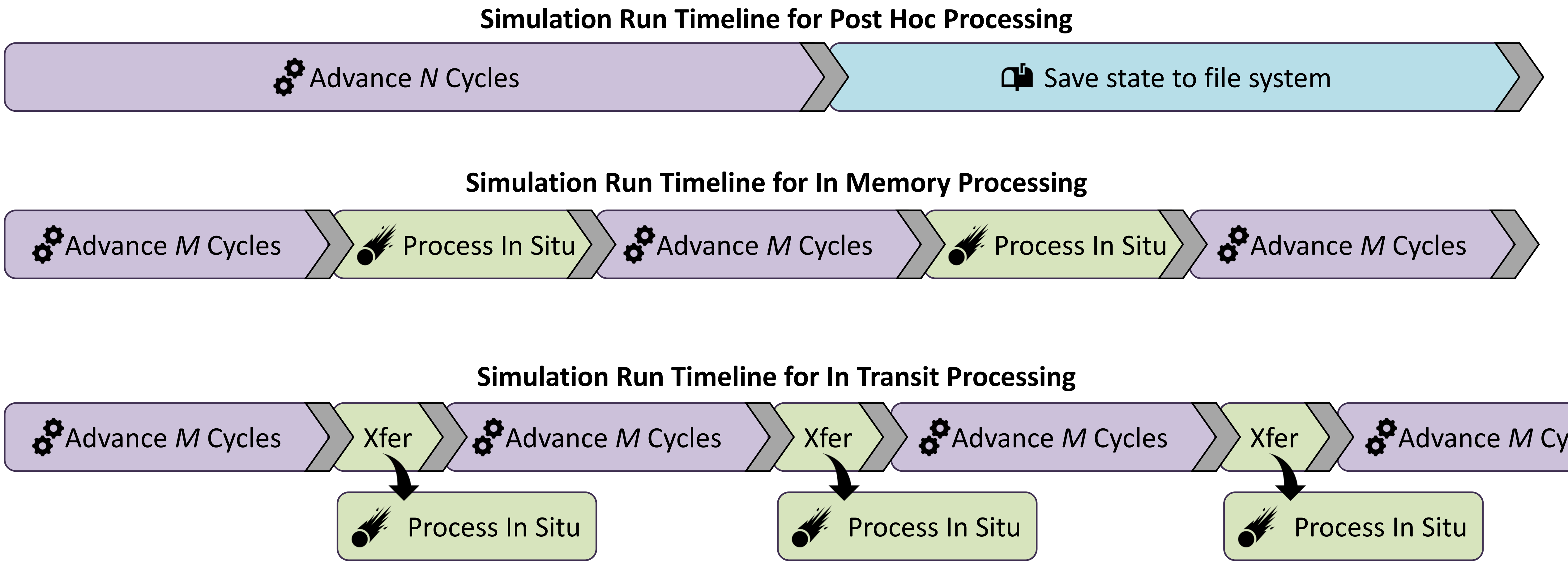
Simulation Run Timeline for Post Hoc Processing

Advance *N* Cycles → Save state to file system

Simulation Run Timeline for In Memory Processing

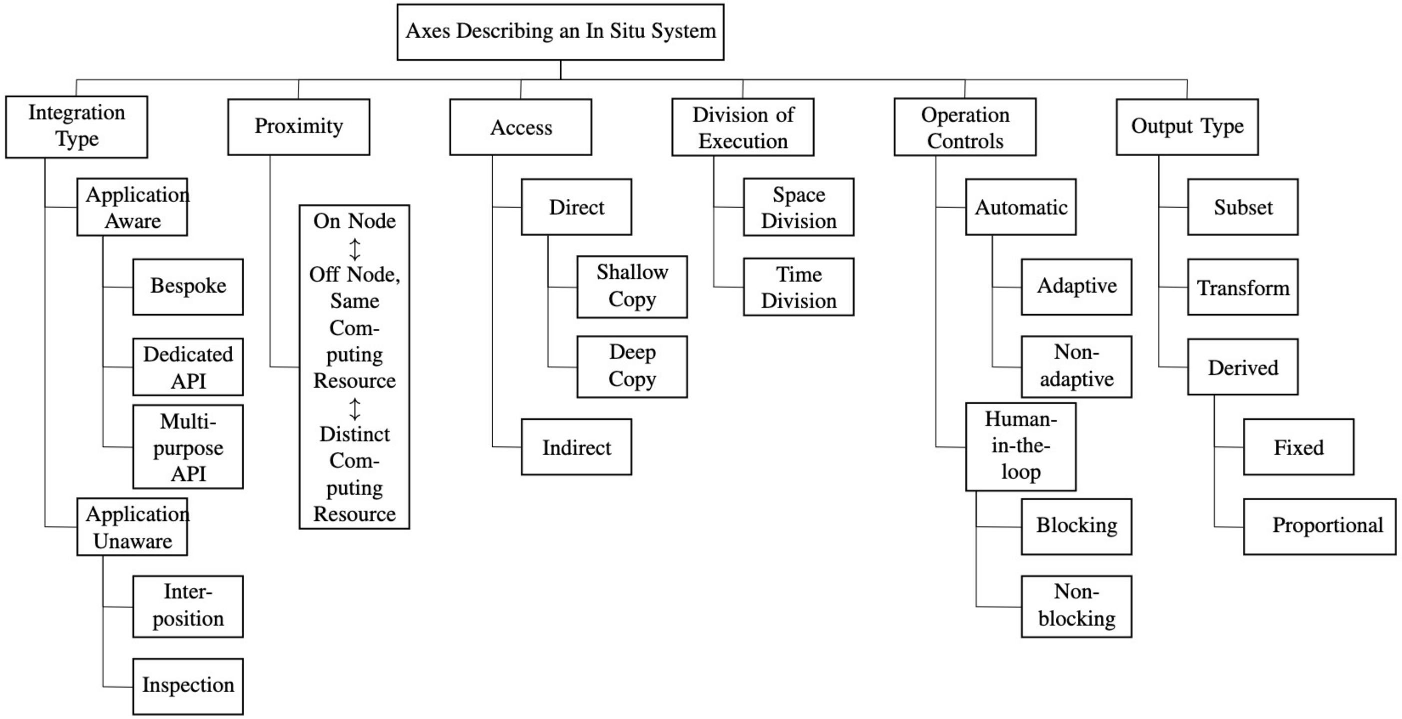
Advance *M* Cycles → Process In Situ → Advance *M* Cycles → Process In Situ → Advance *M* Cycles

In transit is a flavor of in situ processing that can use additional resources to improve runtime



There are many considerations and flavors of in situ processing

Question: How deep does the rabbit hole go?



Answer: "A Terminology for In Situ Visualization and Analysis Systems", H. Childs, et al.
https://cdux.cs.uoregon.edu/pubs/ChildsIJHPCA.pdf

We need to pass simulation mesh data and user actions to our in situ visualization tools

Mesh Data

```
coordsets:
  coords:
    type: "explicit"
    values:
      x: [-1.0, 0.0, 0.0, 0.0, 1.0]
      y: [0.0, -1.0, 0.0, 1.0, 0.0]
      z: [0.0, 0.0, 1.0, 0.0, 0.0]
topologies:
  topo:
    type: "unstructured"
    coordset: "coords"
    elements:
      shape: "tet"
      connectivity: [0, 1, 3, 2, 4, 3, 1, 2]
fields:
  density:
    association: "element”
    topology: ”topo"
    values: [1.0, 2.0]
```

Actions

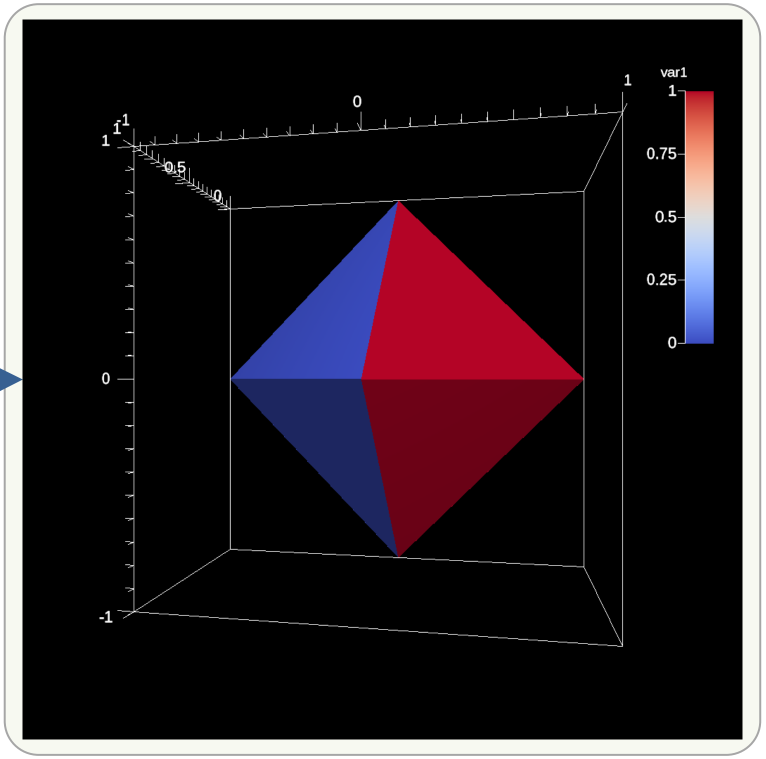
```
-
  action: "add_scenes"
  scenes:
    my_scene:
      plots:
        my_plot:
          type: "pseudocolor"
          field: "var1"
```

In Situ Visualization Tools

Transform, Analyze, Render

Ascent

ParaView Catalyst



We need to pass simulation mesh data and user actions to our in situ visualization tools

```
coordsets:
  coords:
    type: "explicit"
    values:
      x: [-1.0, 0.0, 0.0, 0.0, 1.0]
      y: [0.0, -1.0, 0.0, 1.0, 0.0]
      z: [0.0, 0.0, 1.0, 0.0, 0.0]
topologies:
  topo:
    type: "unstructured"
    coordset: "coords"
```

```
    values: [1.0, 2.0]
```

Mesh Data

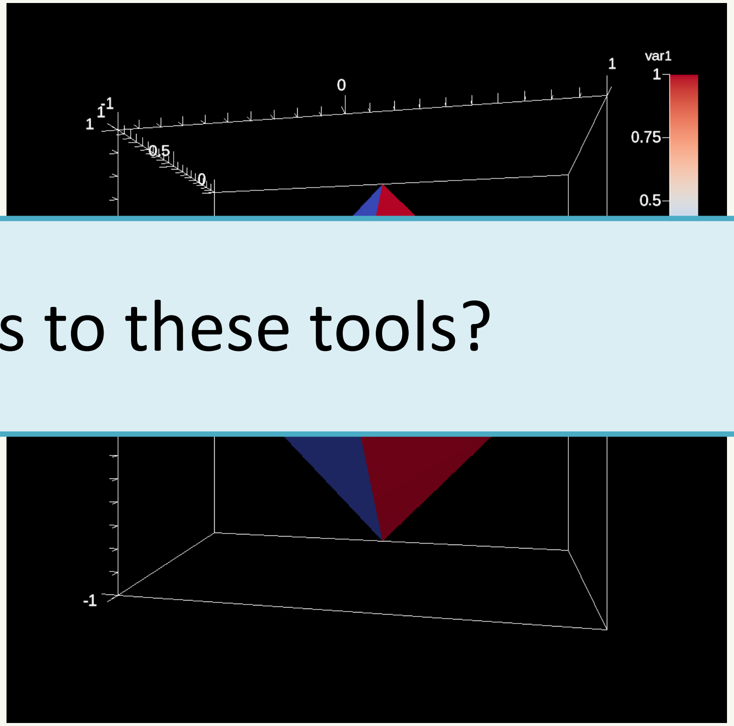
Actions

```
-
  action: "add_scenes"
  scenes:
    my_scene:
      plots:
        my_plot:
          type: "pseudocolor"
          field: "var1"
```

In Situ Visualization Tools

Ascent

ParaView Catalyst



Question 1: How do we pass simulation meshes to these tools?

HPC simulation applications implement and leverage a wide range of mesh data structures and APIs

- A variety of simulation codes leverage their own bespoke in-memory mesh data models.

- Other tools leverage a range of mesh-focused toolkits, frameworks, and APIs including: VTK, VTK-m, MFEM, SAMRAI, AMReX, (and many more …)

- A wide set of powerful analysis tools are mesh agnostic (NumPy, PyTorch, etc) and recasting mesh data into these tools is a challenge

- *A single full-fledged API will never cover all use cases across the ecosystem*

HPC simulation applications implement and leverage a wide range of mesh data structures and APIs

- A variety of simulation codes leverage their own bespoke in-memory mesh data models.

- Other tools leverage a range of mesh-focused toolkits, frameworks, and APIs

***Conduit Mesh Blueprint* provides a strategy to describe and adapt mesh data between a wide range of APIs**

  and recasting mesh data into these tools is a challenge

- *A single full-fledged API will never cover all use cases across the ecosystem*

Both Ascent and Catalyst use *Conduit* as a shared interface to describe and accept simulation mesh data

Mesh Data

CONDUIT

Actions

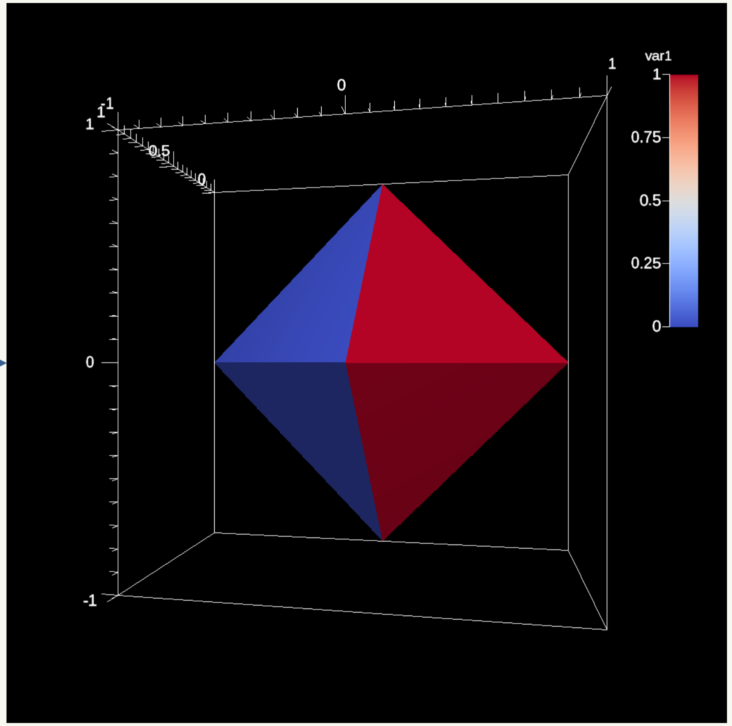
```yaml
-
  action: "add_scenes"
  scenes:
    my_scene:
      plots:
        my_plot:
          type: "pseudocolor"
          field: "var1"
```

In Situ Visualization Tools

Transform, Analyze, Render

Ascent

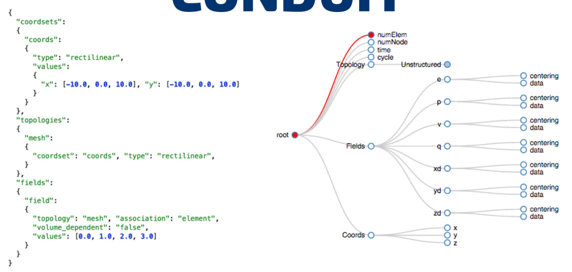
ParaView Catalyst



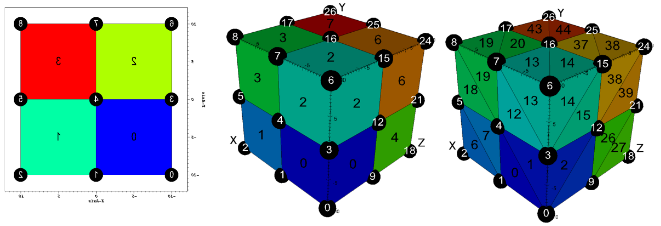
Conduit provides intuitive APIs for in-memory data description and exchange

- **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)

CONDUIT



Hierarchical in-memory data description



Conventions for sharing in-memory mesh data

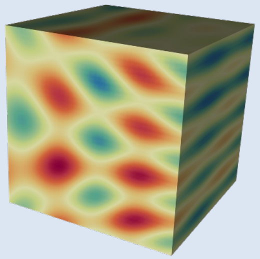
http://software.llnl.gov/conduit
http://github.com/llnl/conduit

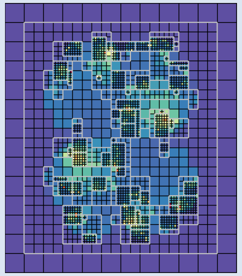
Website and GitHub Repo

The Conduit Blueprint library provides tools to share common flavors of data with Conduit

Blueprint

Supports shared higher-level conventions for using Conduit to represent data



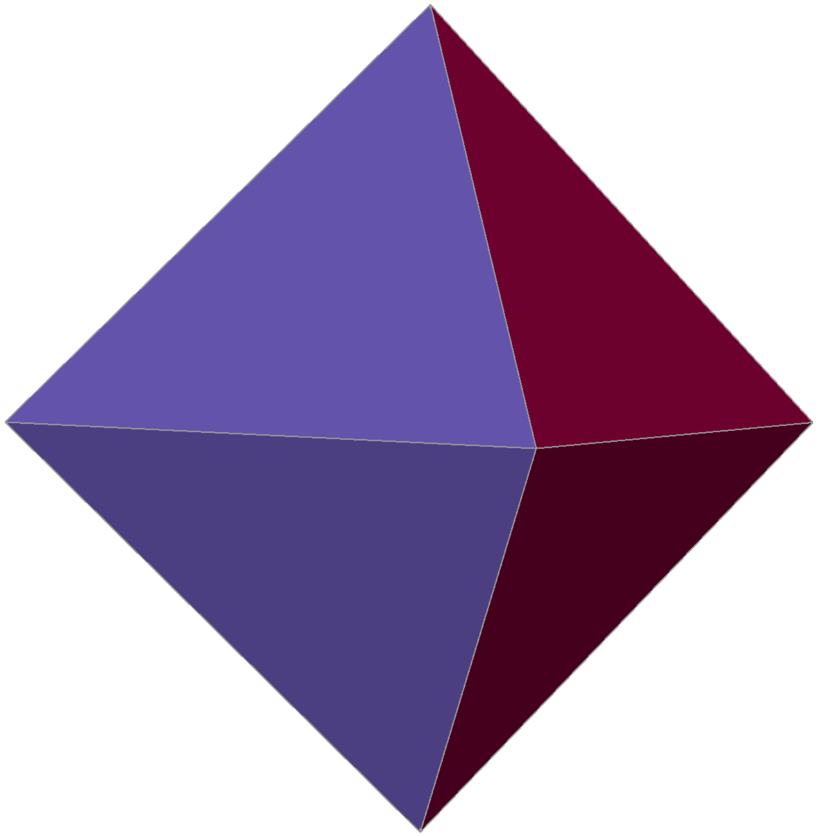


- Computational Meshes
- Multi-component Arrays
- One-to-many Relations
- Example Meshes
- Mesh Transforms

We will share several examples of Conduit "Blueprint" meshes in this tutorial


```yaml
coordsets:
  coords:
    type: "explicit"
    values:
      x: [-1.0, 0.0, 0.0, 0.0, 1.0]
      y: [0.0, -1.0, 0.0, 1.0, 0.0]
      z: [0.0, 0.0, 1.0, 0.0, 0.0]
topologies:
  topo:
    type: "unstructured"
    coordset: "coords"
    elements:
      shape: "tet"
      connectivity: [0, 1, 3, 2, 4, 3, 1, 2]
fields:
  density:
    association: "element”
    topology: ”topo"
    values: [1.0, 2.0]
```


Example YAML Output



An unstructured tet mesh

We need to pass simulation mesh data and user actions to our in situ visualization tools

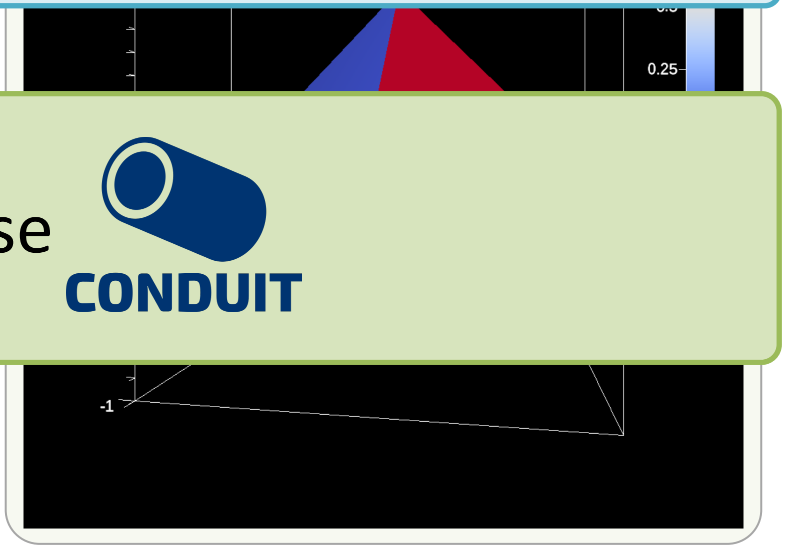
```yaml
coordsets:
  coords:
    type: "explicit"
    values:
```

Mesh Data

Question 1: How do we pass simulation meshes to these tools?

```yaml
    coordset: "coords"
    elements:
      shape: "tet"
```

In Situ Visualization Tools



Answer: Ascent and Catalyst both use CONDUIT

Actions

```yaml
-
  action: "add_scenes"
  scenes:
    my_scene:
      plots:
        my_plot:
          type: "pseudocolor"
          field: "var1"
```

Ascent

ParaView Catalyst

We need to pass simulation mesh data and user actions to our in situ visualization tools

Mesh Data

```yaml
coordsets:
  coords:
    type: "explicit"
    values:
      x: [-1.0, 0.0, 0.0, 0.0, 1.0]
      y: [0.0, -1.0, 0.0, 1.0, 0.0]
      z: [0.0, 0.0, 1.0, 0.0, 0.0]
topologies:
  topo:
    type: "unstructured"
    coordset: "coords"
```

```yaml
    topology: "topo"
    values: [1.0, 2.0]
```

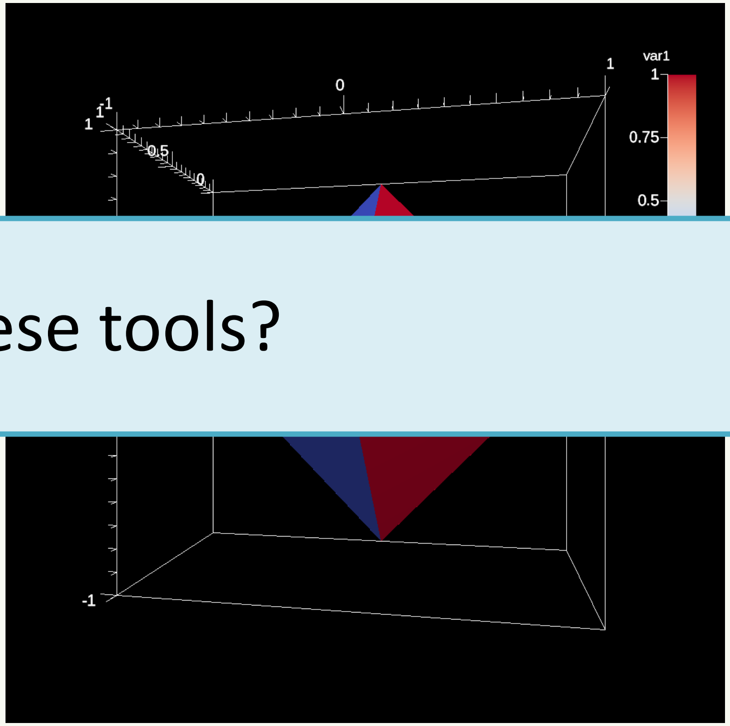
Actions

```yaml
-
  action: "add_scenes"
  scenes:
    my_scene:
      plots:
        my_plot:
          type: "pseudocolor"
          field: "var1"
```

In Situ Visualization Tools

Ascent

ParaView Catalyst



Question 2: How do we pass user actions to these tools?

We need to pass simulation mesh data and user actions to our in situ visualization tools

```
coordsets:
  coords:
    type: "explicit"
```

Mesh Data

Question 2: How do we pass user actions to these tools?

```
    coordset: "coords"
    elements:
      shape: "tet"
```

In Situ Visualization Tools

Answer: Ascent and Catalyst have their own APIs and multiple ways to leverage them (via C++, Fortran, Python, YAML, etc).

You will learn about both APIs in hands-on sessions.

```
      my_plot:
        type: "pseudocolor"
        field: "var1"
```

Catalyst

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

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

Lawrence Livermore National Laboratory