The Conduit Mesh Blueprint: Drafting a New Way to Share Simulation Meshes

DOE CGF 2019

Wednesday April 24th, 2019

Cyrus Harrison, Eric Brugger, Joe Ciurej, Richard Hornung, Adam Kunen, Matthew Larsen, Mark Miller, Robert Rieben, Brian Ryujin

LLNL-PRES-758581

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

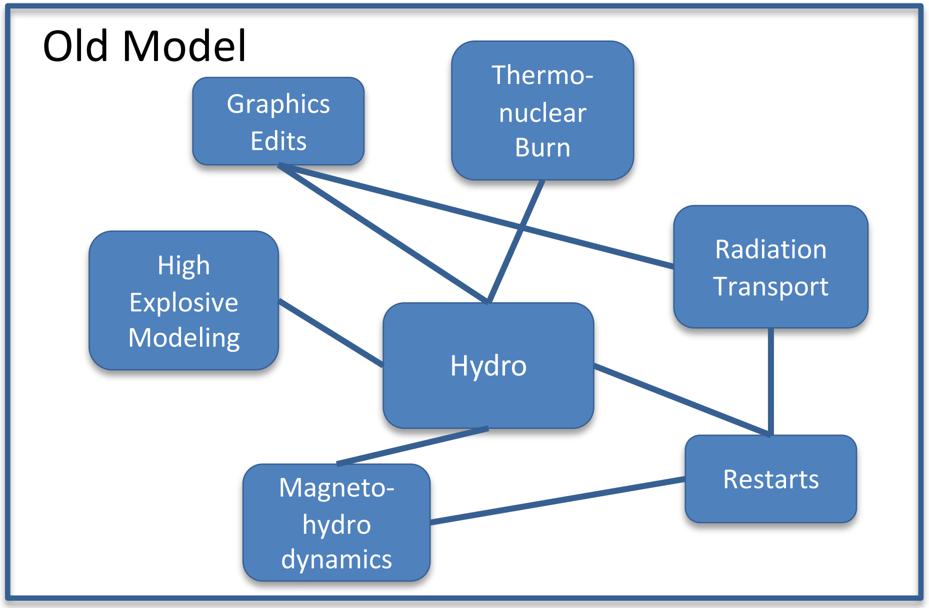
Abstract

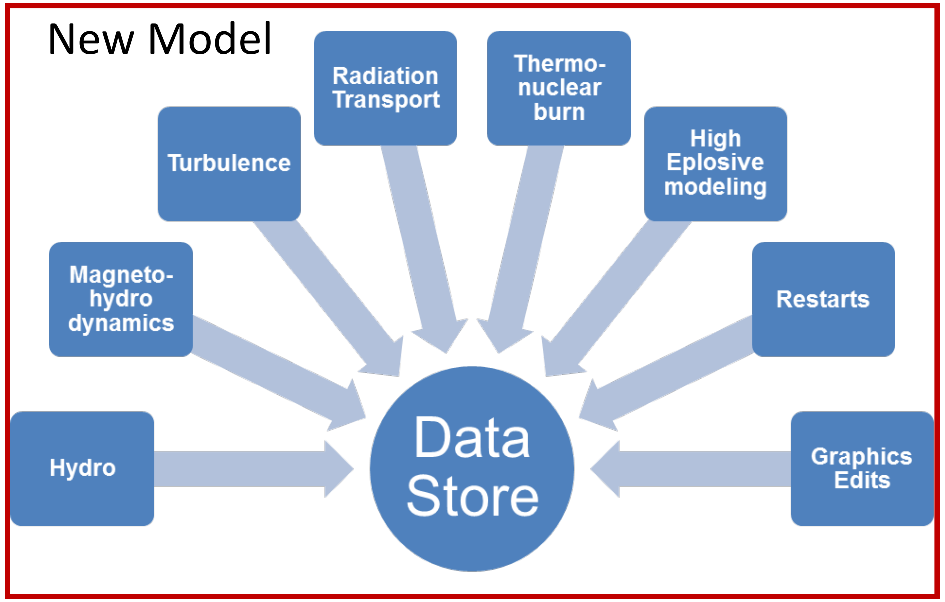
- The Conduit Mesh Blueprint is a set of conventions for sharing mesh-based simulation data both in-memory and via files. LLNL's Weapons Simulation and Computing (WSC) program is adopting the Mesh Blueprint as part of an overarching strategy for sharing mesh data between simulation components. The Mesh Blueprint was first introduced as a way to simplify describing mesh data for in-situ visualization. Following this successful demonstration, WSC's Axom Toolkit and MARBL simulation code adopted the Blueprint to describe mesh data for checkpoint restart and as an interface for in-situ mesh overlay. The Blueprint is also used to describe meshes to ALPINE Ascent, an ECP funded lightweight in situ visualization and data analysis library targeting next generation HPC platforms. This presentation provides an introduction to the Mesh Blueprint and background on its use in WSC projects and the ALPINE ECP project.

Introduction

LLNL's Weapons Simulation and Computing (WSC) program is investing in modularity as part of our code strategy

- 2014 Next-gen CS Working Group Recommendations:
  - Create a toolkit of modular components that provide infrastructure services
  - Increase modularization of physics packages





Simplifying in-memory sharing of simulation mesh data across tools is key to this strategy

We are developing the Mesh Blueprint to simplify sharing meshes across tools in the HPC physics simulation ecosystem

The Mesh Blueprint is a set of hierarchical conventions to describe mesh-based simulation data both in-memory and via files

The HPC simulation physics ecosystem includes a diverse set of applications, tools, and libraries

Multi-physics Simulation Applications

CS Infrastructure
- Input Parsing
- Steering
- Communication
- Parallelism Abstractions
- I/O
- In Situ Coupling

Physics Packages
- Hydrodynamics
- Chemistry
- Thermal radiation
- *{and many more …}*

Physics Libraries
- Material Properties
- Material Models

Numerical Libraries
- Linear Algebra
- Finite Elements

Workflow Applications

Problem Setup and Meshing
- Computational Geometry
- Mesh Generation
- Mesh Decomposition
- Mesh Overlay

Visualization and Analysis
- Mesh Rendering
- Feature Extraction
- Simulated Diagnostics

Uncertainty Quantification
- Ensemble Generation
- Parametric Studies
- Statistical Models

Simulation & Data Management
- Workflow Capture
- Data Organization
- Provenance

The HPC simulation physics ecosystem includes a diverse set of applications, tools, and libraries

Multi-physics Simulation Applications

CS Infrastructure
- Input Parsing
- Steering
- Communication
- Parallelism Abstractions

Physics Packages
- Hydrodynamics
- Chemistry
- Thermal radiation
- *{and many more …}*

Physics Libraries
- Material Properties
- Material Models

Numerical Libraries

“Sharing” mesh data requires agreement on how to represent meshes ...

Workflow Applications

Problem Setup and Meshing
- Computational Geometry
- Mesh Generation
- Mesh Decomposition
- Mesh Overlay

Visualization and Analysis
- Mesh Rendering
- Feature Extraction
- Simulated Diagnostics

Uncertainty Quantification
- Ensemble Generation
- Parametric Studies
- Statistical Models

Simulation & Data Management
- Workflow Capture
- Data Organization
- Provenance

Components of the ecosystem implement and leverage a wide range of mesh data structures and APIs

- Most simulations leverage their own bespoke in-memory mesh data models
  - Growing adoption of MFEM and SAMRAI are notable exceptions

- Other tools leverage a range of mesh-focused toolkits, frameworks, and APIs including:
  - VTK, VTK-m, MFEM, Mint, and SAMRAI

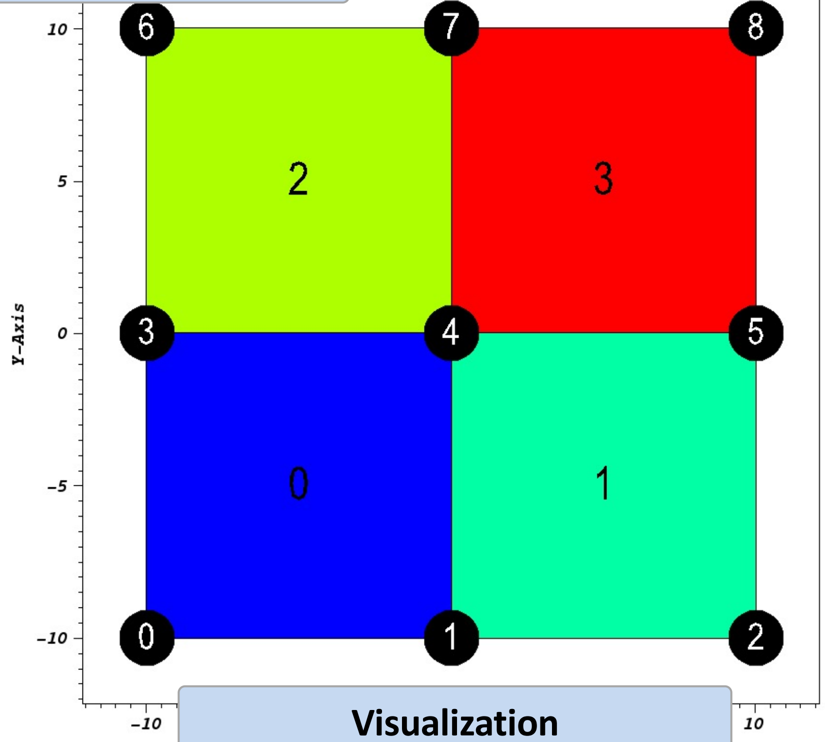
We never expect a single full fledged API to emerge that will cover all use cases across the ecosystem

The Mesh Blueprint is a set of conventions that outline a hierarchical structure (or schema) to describe mesh data

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

JSON Representation

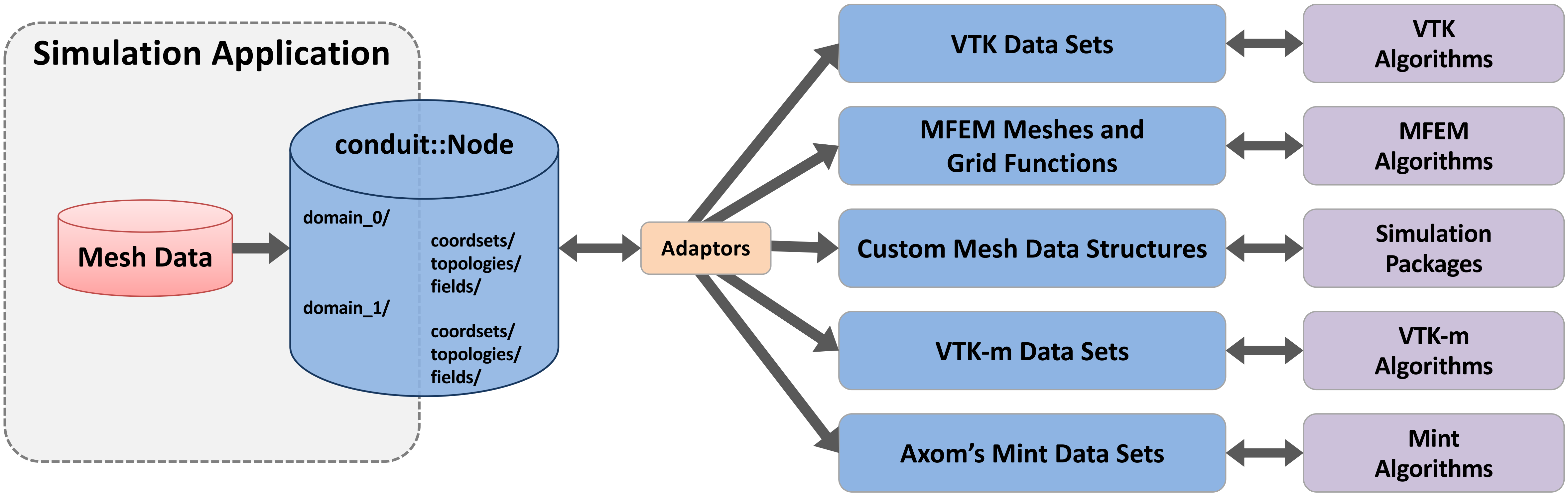
Mesh Blueprint Example: A Simple Rectilinear Grid



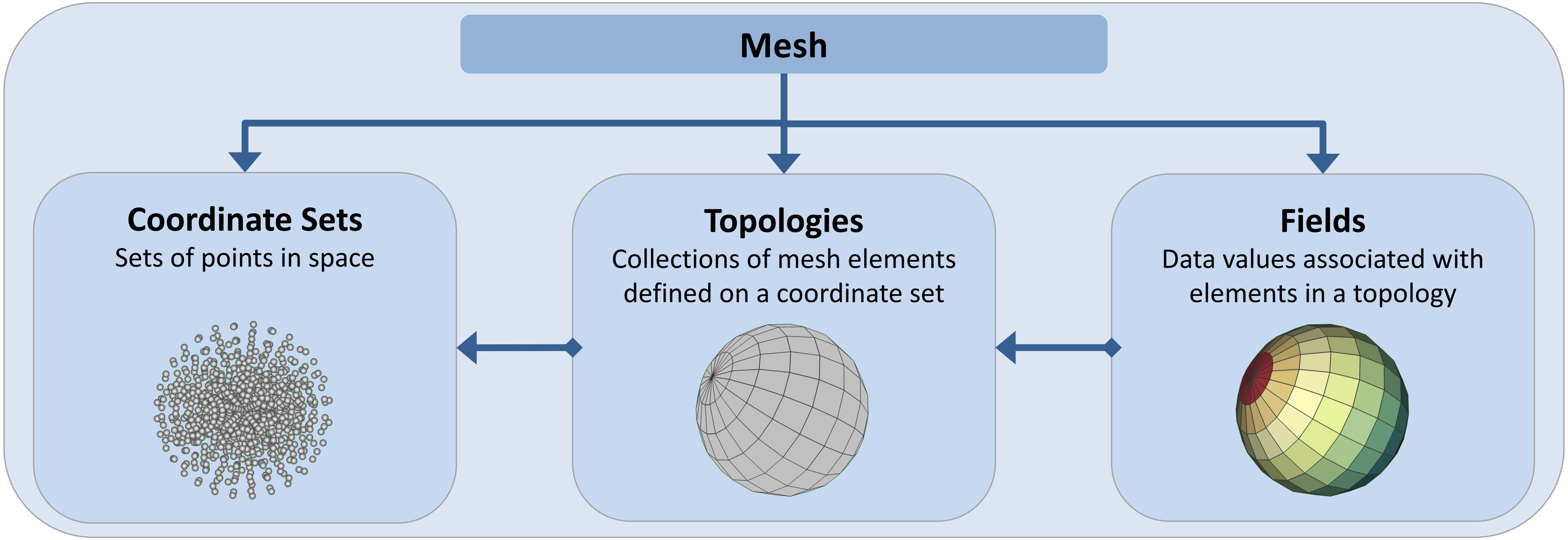
Visualization

Prescribing hierarchical structure in lieu of an API strikes a balance that gives the Blueprint the flexibility to meet the wide range of use cases across the ecosystem

Mesh Blueprint data can be adapted to a wide range of concrete mesh-focused toolkits and APIs



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

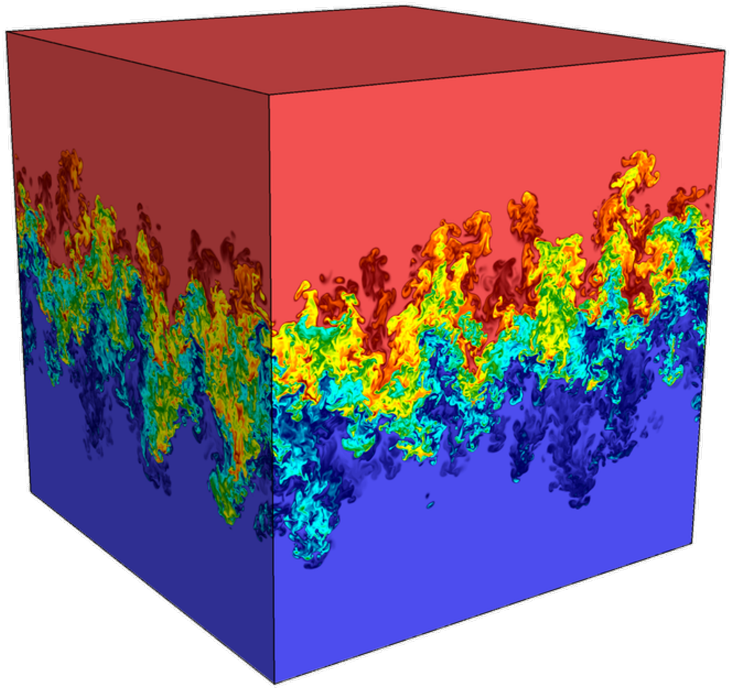


Ideas were shaped by surveying projects including: ADIOS, BoxLib, Chombo, Damaris, EAVL, Exodus, ITAPS, MFEM, SAF, SAMRAI, Silo, VisIt’s AVT, VTK, VTK-m, Xdmf.

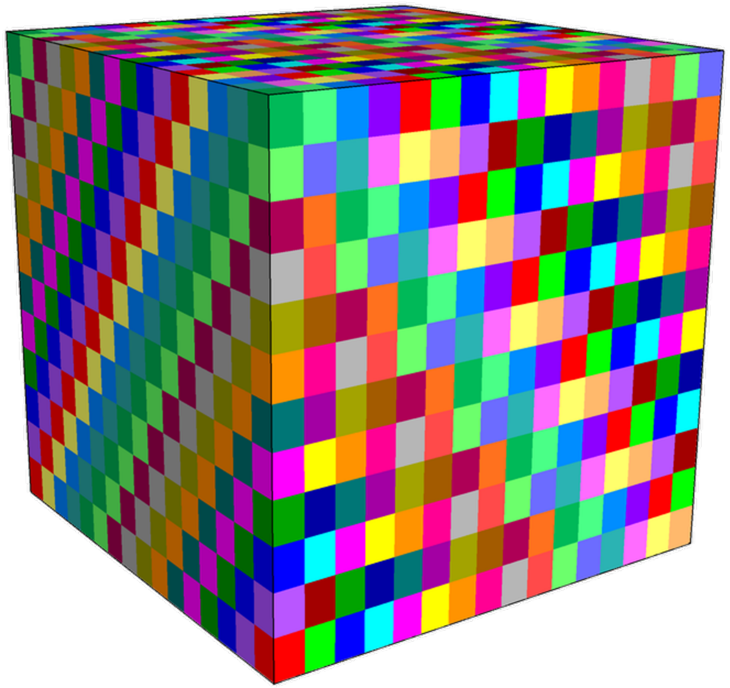
We are steadily filling out the Blueprint to cover the wide range of mesh descriptions required by the ecosystem

- **Coordinate Sets**
  - 1D/2D/3D
  - Cartesian, Cylindrical, Spherical
  - Implicit: Uniform, Rectilinear
  - Explicit
- **Topologies**
  - Implicit: Points, Uniform, Rectilinear, Structured
  - Unstructured
    [Points, Lines, Quads, Tris, Tets, Hexs]
  - Optional MFEM Grid Function support
  - Arbitrary Polygonal and Polyhedral
    (Active development)
  - Unstructured heterogenous element shapes
    (Planned for future)
- **Fields**
  - Vertex or Element associated
  - Multi-component field arrays
  - Optional MFEM Grid Function Basis support
  - Multi-dimensional field arrays
    (Planned for future)
  - Sparse representations for field arrays
    (Planned for future)
- **Domain Decomposition Info**
  - Basic State Info [Domain Ids]
  - Domain Adjacency Info for Unstructured Meshes
  - Domain Adjacency Info for Structured Meshes
    (Planned for future)
  - Nesting Info for Block-Structured AMR Meshes
    (Active development)

The structure of the Blueprint is designed with distributed-memory parallelism in mind



Full Dataset



Domain Decomposition

Any info required to describe to domain decomposition, nesting, or abutment is local

Software infrastructure supporting Mesh Blueprint conventions

The Mesh Blueprint is part of Conduit, an open source project focused on simplifying in-memory data exchange

CONDUIT

Project Info

- Languages: C++, Python, C, Fortran
- Docs: https://software.llnl.gov/conduit
- GitHub Repo: https://github.com/llnl/conduit
- License: BSD Style
- Builds with Spack: https://spack.io/

Project Timeline

- 2011 – 2013: Neurons start firing and developing concepts
- November 2013: Code development starts at a LLNL Hackathon
- Fall 2014 – Spring 2015: Harvey Mudd Clinic Project
- January 2015: Released open source
- 2015: Early use in LLNL simulation codes and Strawman In Situ proxy
- 2016 – Present : Adoption in LLNL simulation codes, VisIt support, use in ALPINE ECP Project

The Conduit project manages the official Blueprint conventions and provides basic software infrastructure supporting their use

Conduit Library

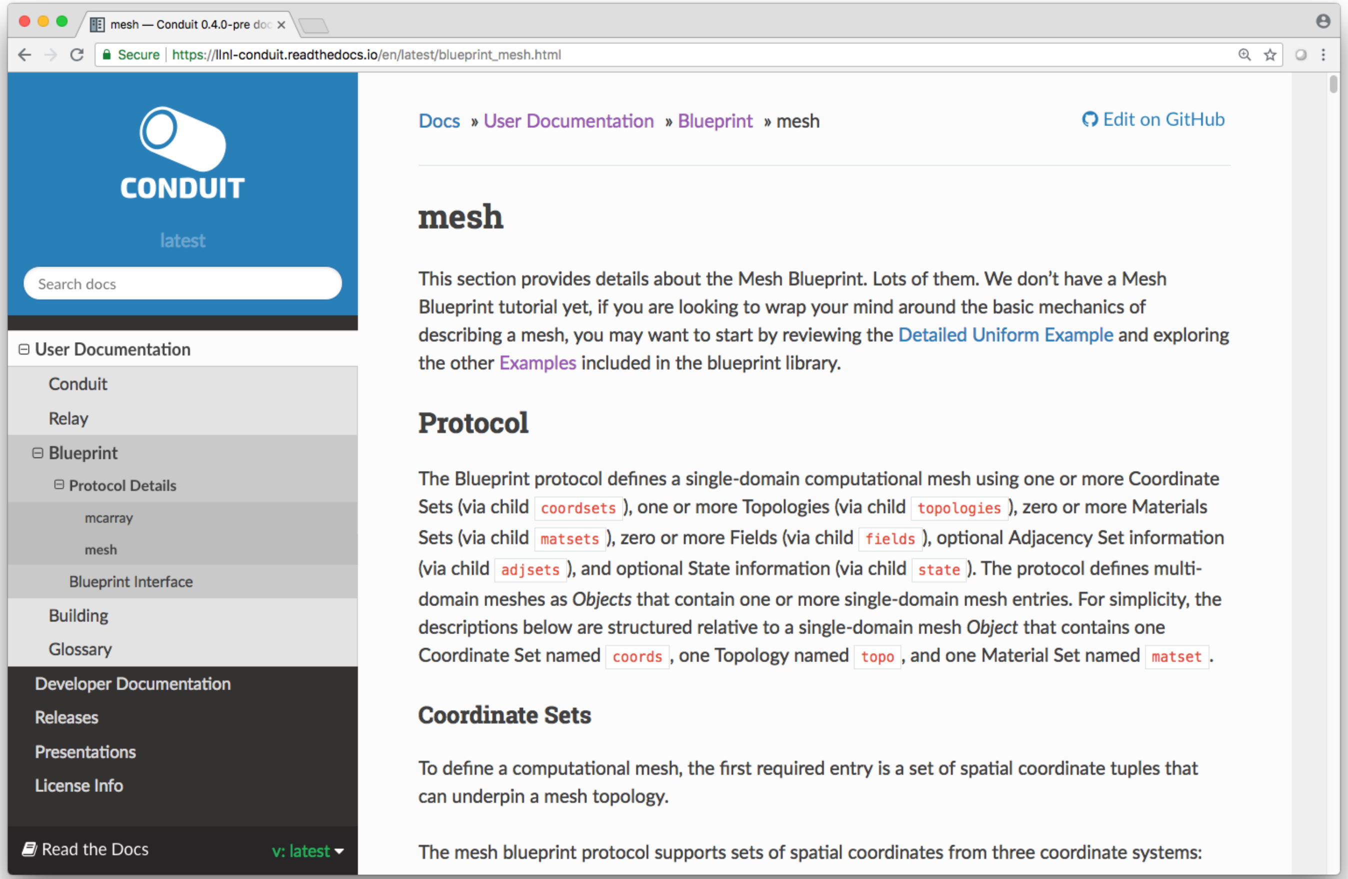
Implements interfaces to Conduit's in-memory data model

- Core Objects
- JSON parsing
- Basic I/O and Serialization

Conduit Blueprint Library

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

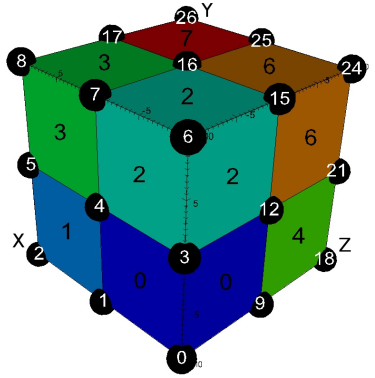
- Computational Meshes
- Multi-Component Arrays

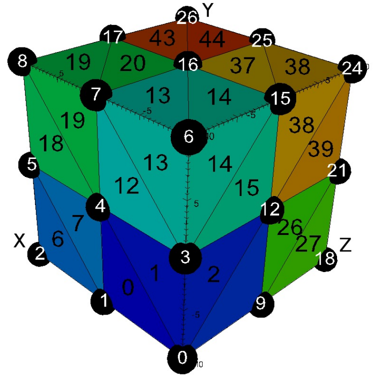


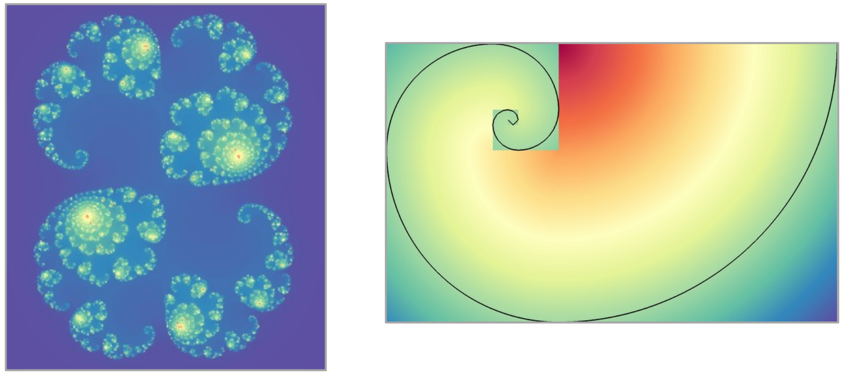
https://llnl-conduit.readthedocs.io/en/latest/blueprint_mesh.html

The Conduit Blueprint library facilitates using Mesh Blueprint data via three important capabilities

- Methods which verify if data conforms to blessed conventions at runtime
  - Provides detailed information for non-conforming data
- Methods that apply basic data transforms to conforming data, including:
  - Coordinate Set and Topology transforms (e.g. Implicit Uniform to Explicit Coordinates)
  - Memory layout transforms (e.g. Contiguous to Interleaved to array layouts)
- Methods that generate mesh examples, aiming to cover the range of supported meshes





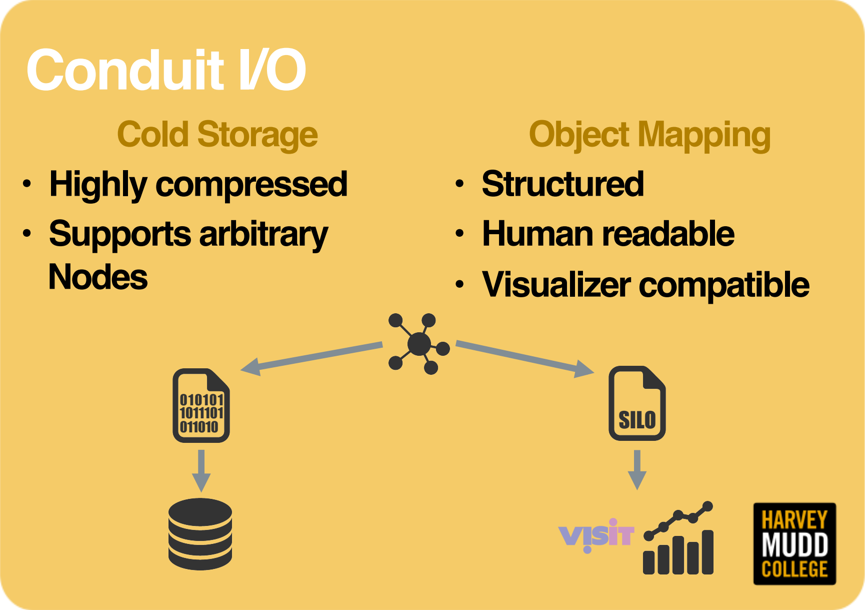


Mesh Blueprint Examples generated by the Conduit Blueprint Library

The evolution and adoption of the Mesh Blueprint

The Blueprint started with experiments using Conduit to describe meshes for visualization

- Fall 2014 – Spring 2015: As part of Harvey Mudd Clinic project focused on Conduit, students explored creating Conduit trees of meshes and converting them to Silo files for visualization in VisIt



Conduit I/O

Schema of Node required for Object Mapping

```
VNODE: {
    numElem: int: number of elements;
    numNode: int: number of compute nodes;
    cycle: float: current cycle number of the computation;
    time: float: time of the computation cycle;
    Topology: {
        Unstructured: {
            hexes: int32*: connectivity;
        };
    };
    Coords: {
        x: (float32*|float64*): x coordinate data;
        y: (float32*|float64*): y coordinate data;
        z: (float32*|float64*): z coordinate data;
    };
    Fields: {
        [fieldname]: {
            data: (float32*|float64*): field data;
            centering: string: centering of the data
                must match /(zone|node)/;
    };
};
```

HARVEY MUDD COLLEGE

Conduit I/O

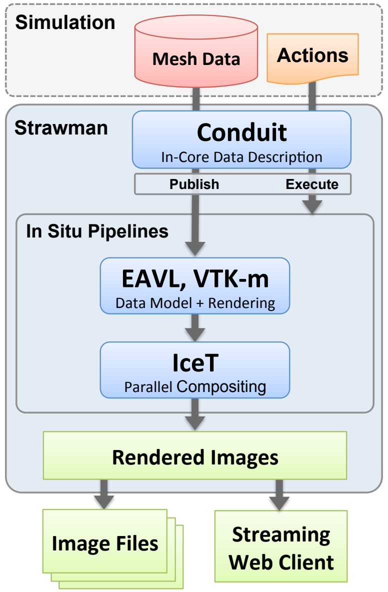
Schema of Node required for Object Mapping

```
VNODE: {
    numElem: int: number of elements;
    numNode: int: number of compute nodes;
    cycle: float: current cycle number of the computation;
    time: float: time of the computation cycle;
    Topology: {
        Unstructured: {
            hexes: int32*: connectivity;
        };
    };
    Coords: {
        x: (float32*|float64*): x coordinate data;
        y: (float32*|float64*): y coordinate data;
        z: (float32*|float64*): z coordinate data;
    };
    Fields: {
        [fieldname]: {
            data: (float32*|float64*): field data;
            centering: string: centering of the data
                must match /(zone|node)/;
        };
    };
};
```

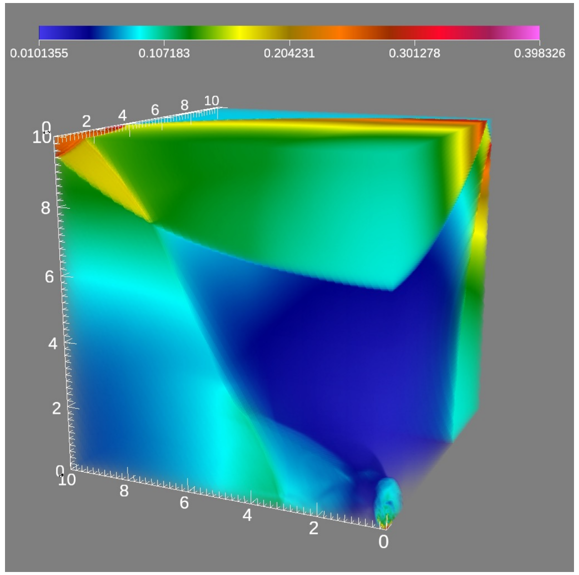
HARVEY MUDD COLLEGE

The Blueprint started with experiments using Conduit to describe meshes for visualization

- Summer 2015: The VisIt team expanded these ideas to create a mesh interface for a new in-situ visualization proxy called “Strawman”



“Strawman” Software Architecture



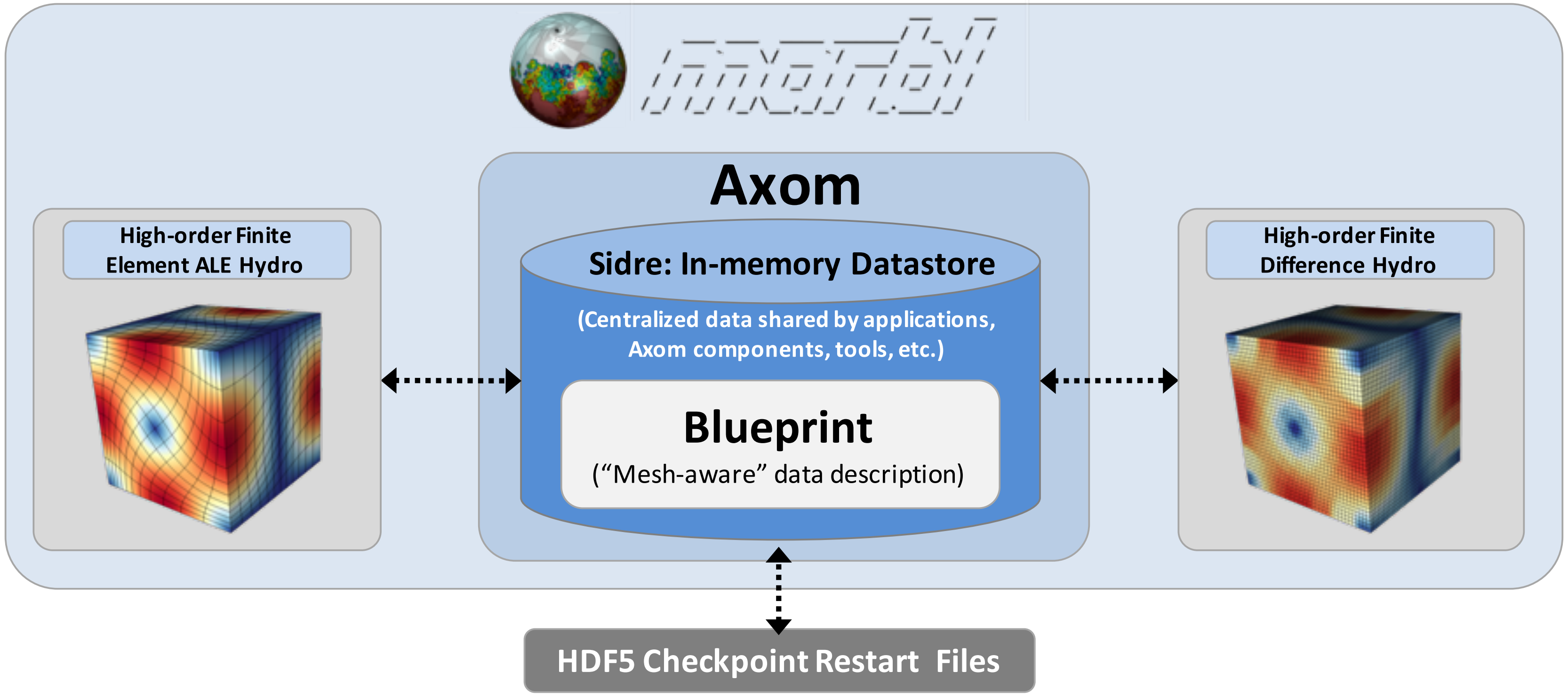
In-situ render of Cloverleaf3D data

Inspiration for Viz Proxy App

“Strawman: A Batch In Situ Visualization and Analysis Infrastructure for Multi-Physics Simulation Codes”
In Proceedings of ISAV 2015 (SC15)Workshop, Austin TX, November 2015

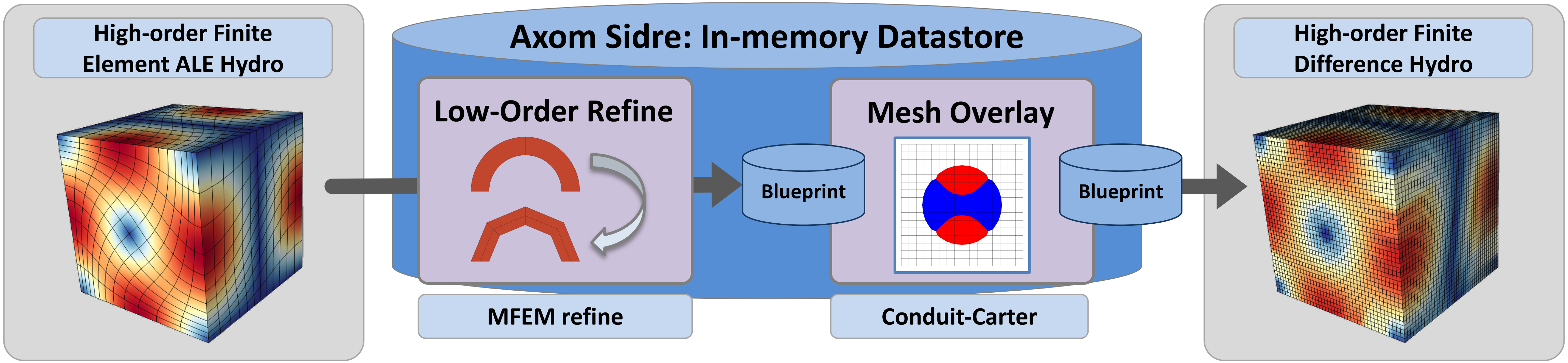
Success with visualization use cases helped demonstrate that the Blueprint was viable for use directly in simulations

- Fall 2015 – Fall 2016: The Axom and MARBL teams adopted and helped expand Blueprint to describe meshes for checkpoint restarts supporting both of MARBL's hydrodynamics packages (Supported an ATDM FY16 L2)



Success with visualization use cases helped demonstrate that the Blueprint was viable for use directly in simulations

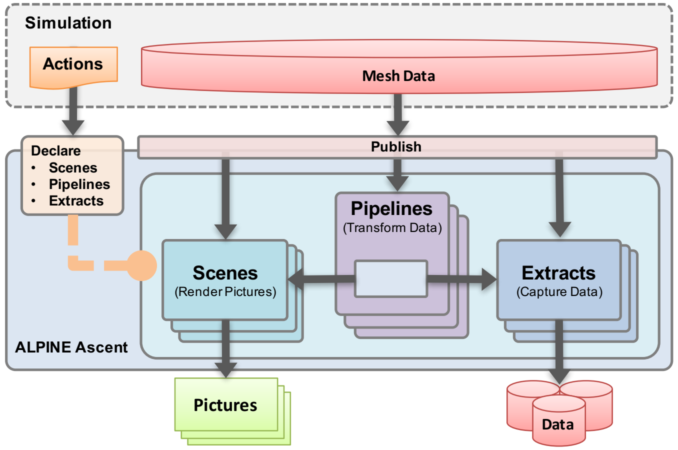
- Fall 2016 – Fall 2017: A Blueprint interface for the Carter mesh overlay tool was developed, which enabled in-situ mesh overlay from MARBL's high-order ALE hydrodynamics package to MARBL's high-order finite difference hydrodynamics package (Supported an ATDM FY17 L2)



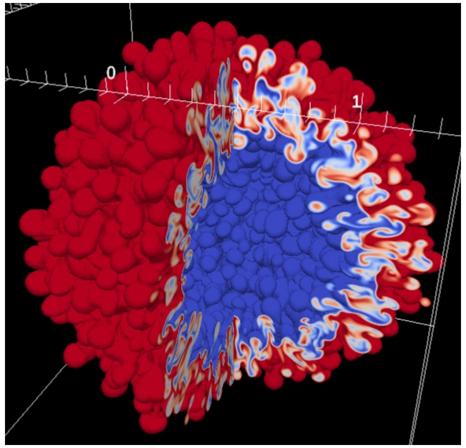
Full Circle: Adoption of the Blueprint in MARBL bolstered the case for using Blueprint in the ALPINE ECP project

- Fall 2017 – Present: Blueprint is the mesh interface for Ascent, an in-situ visualization and analysis infrastructure developed as part of the ALPINE ECP project

Ascent



Ascent Software Architecture



In-situ rendering from Ares

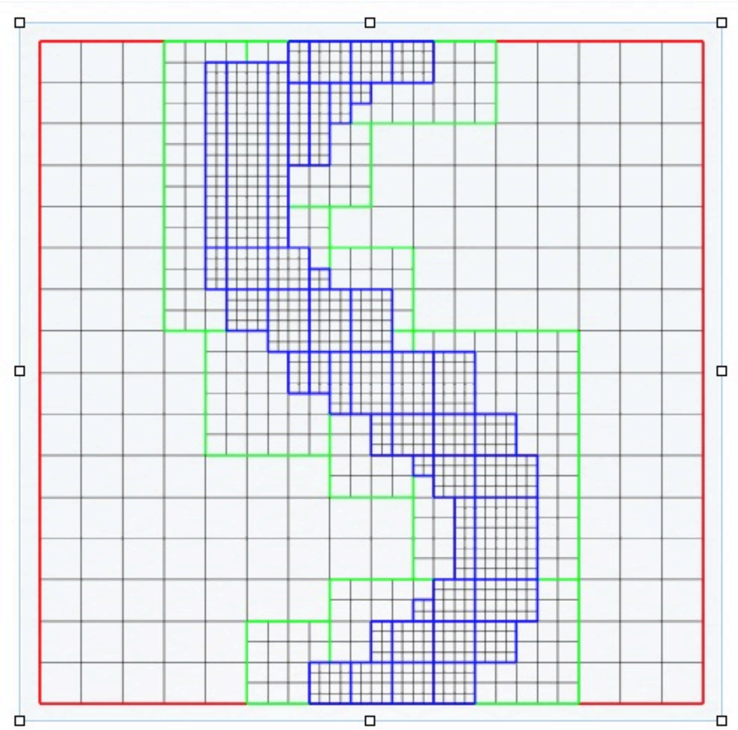
Evolved from "Strawman"

"The ALPINE In Situ Infrastructure: Ascending from the Ashes of Strawman"
In Proceedings of ISAV 2017 (SC17) Workshop, Denver CO, November 2017

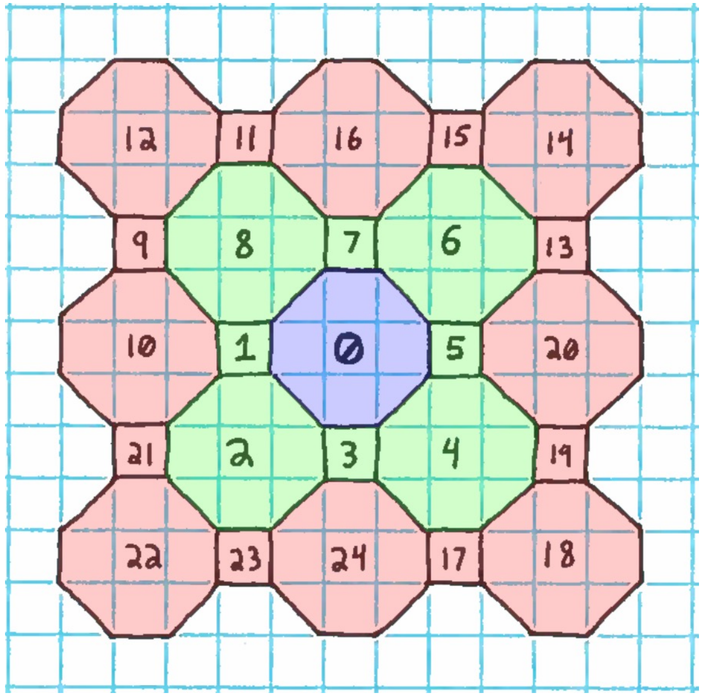
https://github.com/alpine-dav/ascent
https://ascent.readthedocs.io/

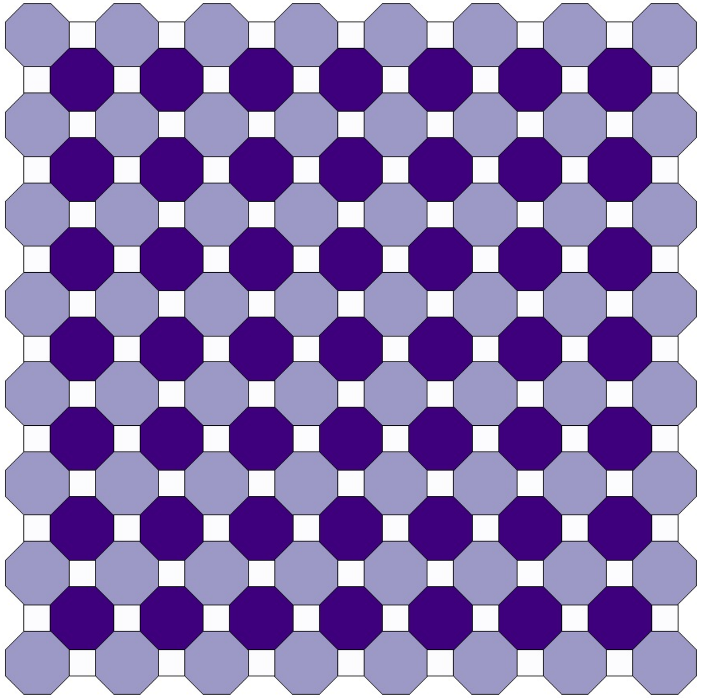
Recent WSC development has focused on extensions to the Blueprint to describe more complex mesh constructs

- Spring 2018 – Present: Conventions for AMR Nesting information and Polygonal and Polyhedral topologies are under development



AMR Nesting Relationships





Polygonal Topologies

Conclusion

- The Mesh Blueprint is a set of hierarchical conventions to describe mesh-based simulation data both in-memory and via files
- We are developing and vetting the Blueprint in both visualization and simulation contexts
- We are working across teams to develop these shared conventions

The Mesh Blueprint is supporting WSC's modular code development strategy by helping simplify in-memory sharing of simulation mesh data

Lawrence Livermore National Laboratory