Strawman: A Batch In Situ Visualization and Analysis Infrastructure for Multi-Physics Simulation Codes

DOE-CGF

Presented By: Matt Larsen

April 28th, 2016

Matt Larsen (LLNL), Eric Brugger (LLNL),
Jim Elliot (AWE), Hank Childs (University of Oregon + LBL),
Kevin Griffin (LLNL) & Cyrus Harrison (LLNL)

LLNL-PRES-681296

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

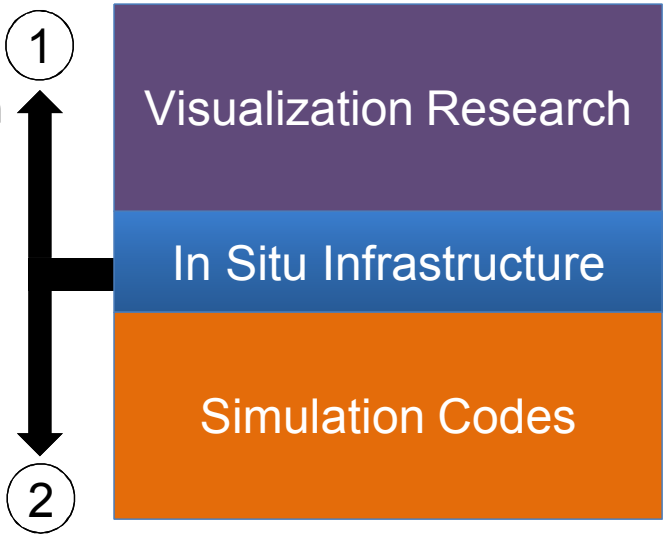
About Me

- **Ph.D. Candidate at the University of Oregon**
  - Advisor: Hank Childs
- **Research**
  - Data-Parallel Rendering (e.g., EAVL and VTK-m)
    - Ray Tracing within a Data-Parallel Framework (Pacific Vis 2015)
    - Volume Rendering Via Data-Parallel Primitives (EGPGV 2015)
  - In Situ Visualization
    - Strawman (ISAV 2015)
- **Full-Time Staff Position at LLNL (January 2016)**
  - Member of the VisIt team
  - Projects:
    - Strawman and VTK-m
    - Unstructured volume rendering and radiography

Strawman is an in situ visualization system with two primary aims:

1. **Serve as a community proxy app:**
   - Lightweight, easy to modify
   - Includes three simulation codes for experimentation
   - In situ model: tightly-coupled, batch-oriented

2. **Model an approach that minimizes burden on simulation codes:**
   - Our approach is distilled from diverse set of simulation code team requirements (including performance and integration time)



About the name….

Motivation

- Simulation customers will need a **range** of production in situ visualization and analysis capabilities on next-generation platforms.
  - Many simulation codes will need to integrate in situ solutions.
  - Are our in situ solutions easy to use, lightweight? Is their performance good enough?

The future is coming quickly and the path is still unclear …

Goals

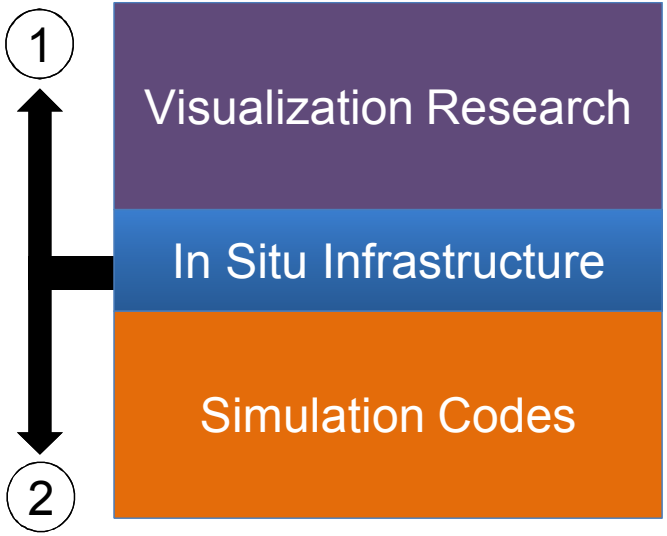
1. Establish lightweight, tightly coupled in situ system for rapid prototyping
   - Help enable growth in tightly-coupled techniques
2. Understand best techniques for integrating with a diverse set of simulation codes
   - Establish common requirements over many different code groups

We feel a shareable proxy application will be an effective mechanism to help the community reach these goals.

“The Most Important Slide”

	Simulation Codes		
	LULESH	Kripke	CloverLeaf3D
Data Description	15	21	39
Action Descriptions	14	14	14
Strawman API Calls	7	7	9
Total Lines of Code	**36**	**42**	**62**

These proxy simulation codes are “mini-apps” and are easy to modify (~1000 lines of code)



It is easy to integrate Strawman into simulation codes
(and Strawman will be released with 3 integrated proxy simulations)

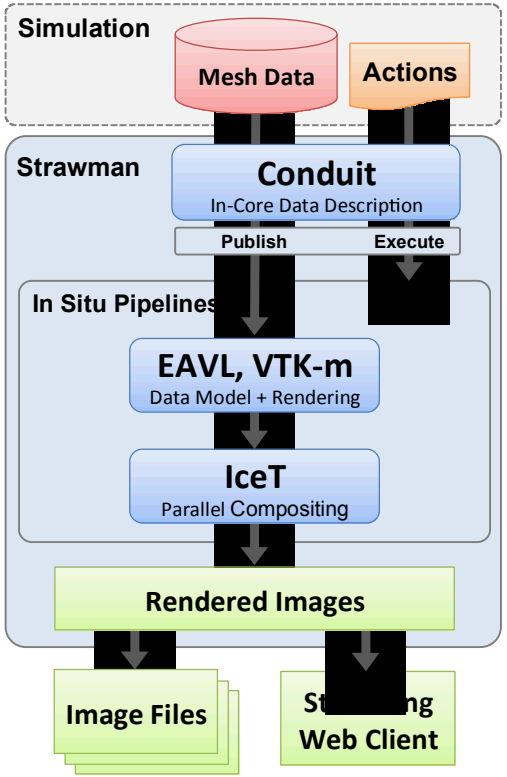
We collected requirements for tightly coupled in situ use cases.

- **3 Categories**
  - Portability
    - Architectures, languages, mesh types
  - Usability
    - Reduce integration time, data ownership, run-time control, easy to consume results
  - Minimal burden on simulation
    - Execution time, memory usage

See ISAV2015 paper for full list of requirements

Strawman heavily leverages a few external libraries.

- **Conduit**
  - Data description
- **EAVL, VTK-m**
  - Visualization and analysis toolkits
- **IceT**
  - Parallel image compositing



Conduit is used for in-core data description.

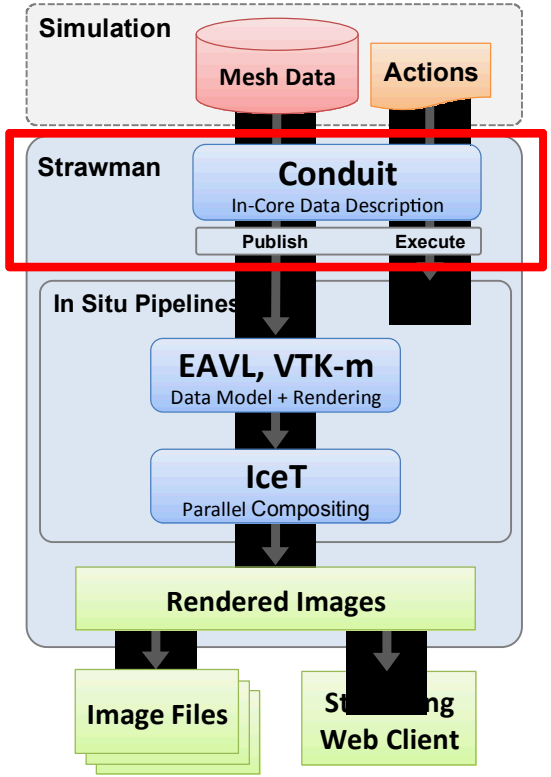
https://github.com/llnl/conduit

Conduit Provides:

- JSON style object model
- Type standardization (e.g., float64)
- Separates data and description
- Run-time focused
- Multiple language APIs

Conduit addresses requirements related to:

- Usability



The EAVL and VTK-m visualization and analysis toolkits are used for rendering.

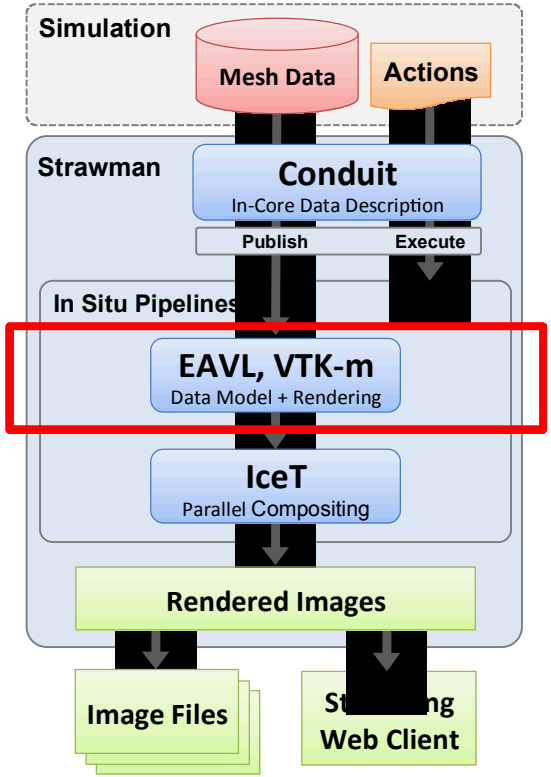
EAVL: http://ft.ornl.gov/eavl/index.html
VTK-m: http://http://m.vtk.org/

These toolkit's provide:

- Node level parallelism
  - Data-parallel primitives
    - Map, reduce, scan
  - Programming for portable performance
- Flexible and efficient mesh data models
- Common algorithms
- Rendering infrastructure

They address requirements related to:

- Portability
- Minimal burden (efficiency)



IceT is used for distributed-memory parallel image compositing.

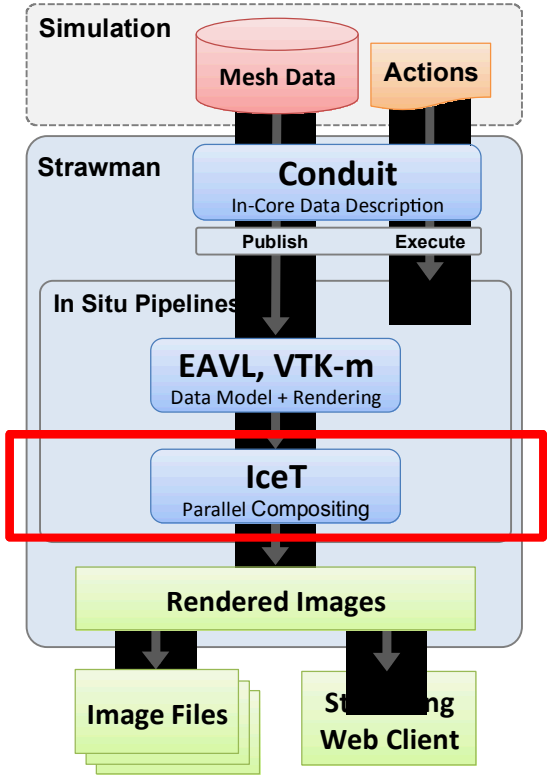
https://gitlab.kitware.com/icet/icet

IceT Provides:

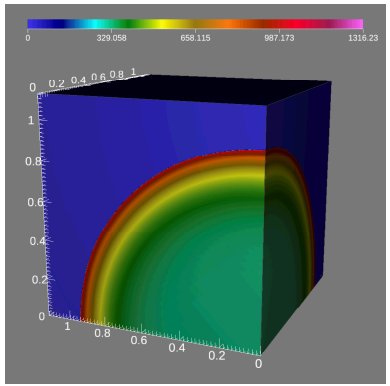
- Distributed memory image compositing for surfaces and volumes
- Scalable communication
  - Demonstrated on 100K+ cores

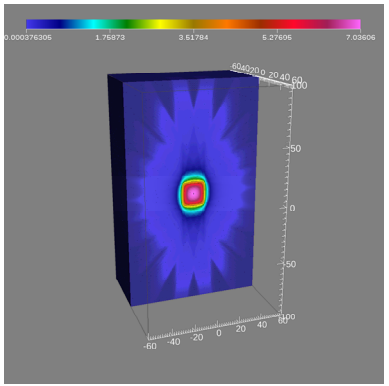
IceT addresses requirements related to:

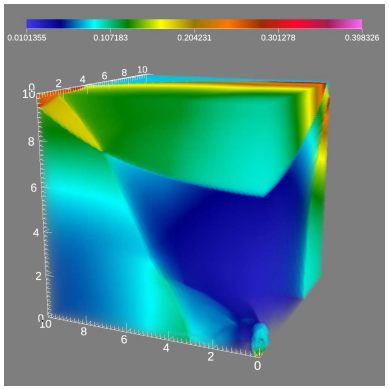
- Minimal burden (efficiency)



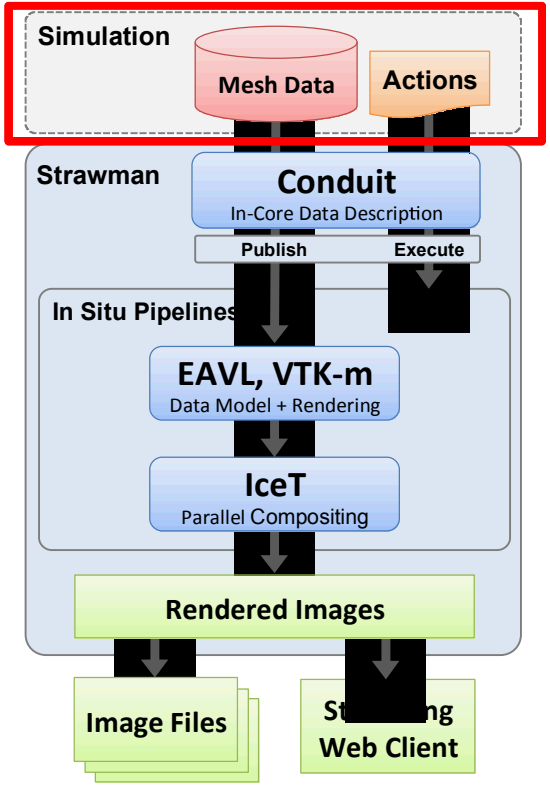
Strawman includes integration examples for 3 built-in proxy simulations.







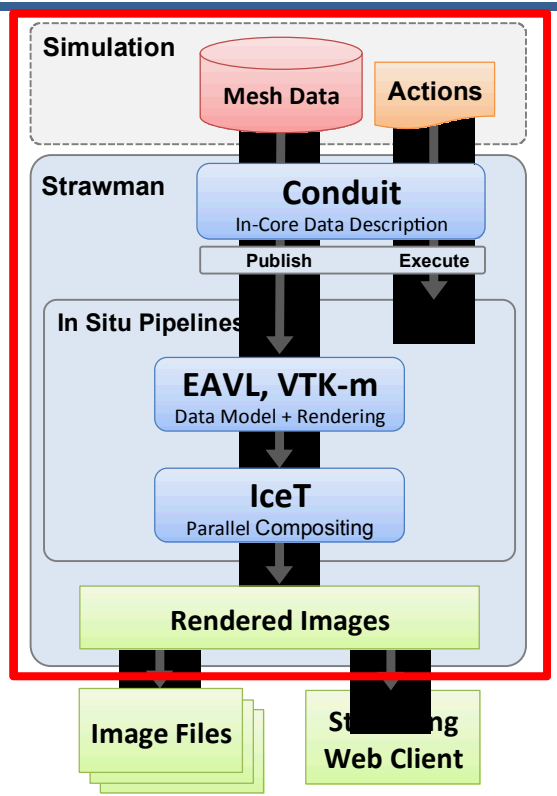
LULESH	Kripke	CloverLeaf3D
Hydrodynamics	Neutron Transport	Hydrodynamics
Unstructured	Uniform	Rectilinear
C++	C++	FORTRAN



We ran a small parallel study demonstrating Strawman's EAVL rendering pipeline.

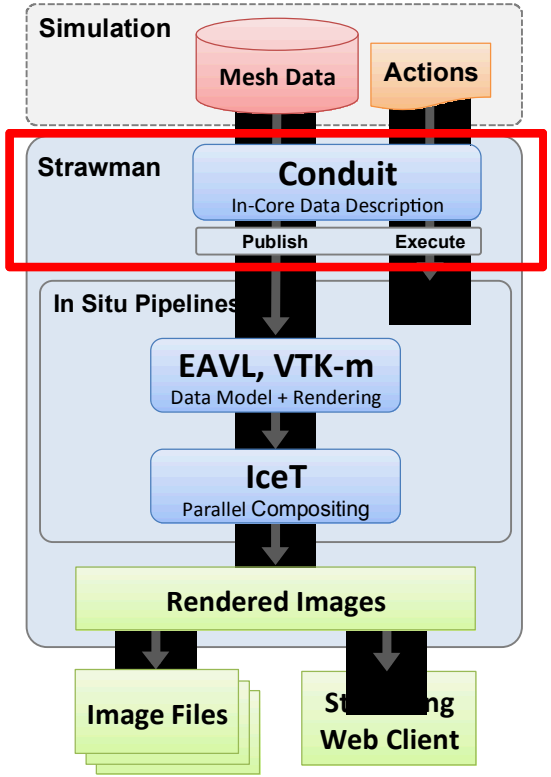
	Vis	Sim
CloverLeaf3D 4B Cells (Ray Tracing)	1.68	5.77
Kripke 4B Cells (OSMesa)	3.80	16.55
LULESH 8B Cells (Vol. Ren.)	30.85	12.62

- Average time per cycle (seconds)
- Simulation burden for:
  - 256 nodes / 4096 cores
  - ~ 1 / 2 M cells per core
- Performance was comparable with previous research results ("do no harm")



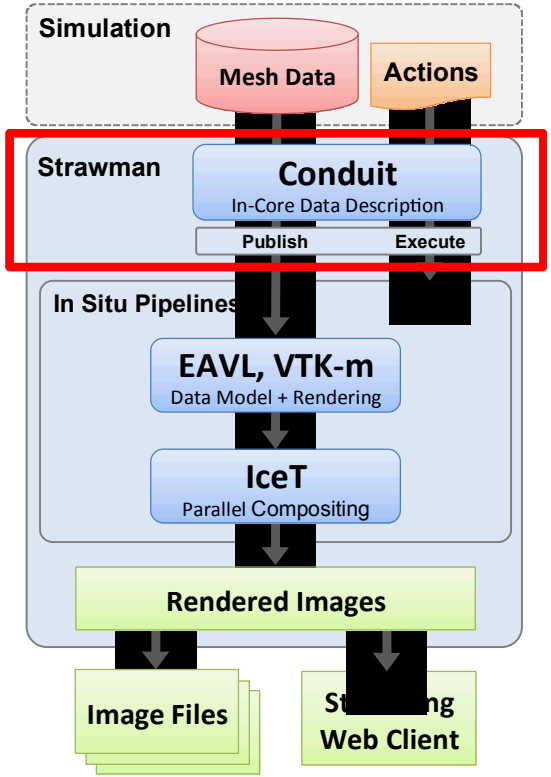
We used lines of code to gauge the effort required to integrate Strawman into these 3 simulations.

	Simulation Codes		
	LULESH	Kripke	CloverLeaf3D
Data Description	15	21	39
Action Descriptions	14	14	14
Strawman API Calls	7	7	9
Total Lines of Code	**36**	**42**	**62**



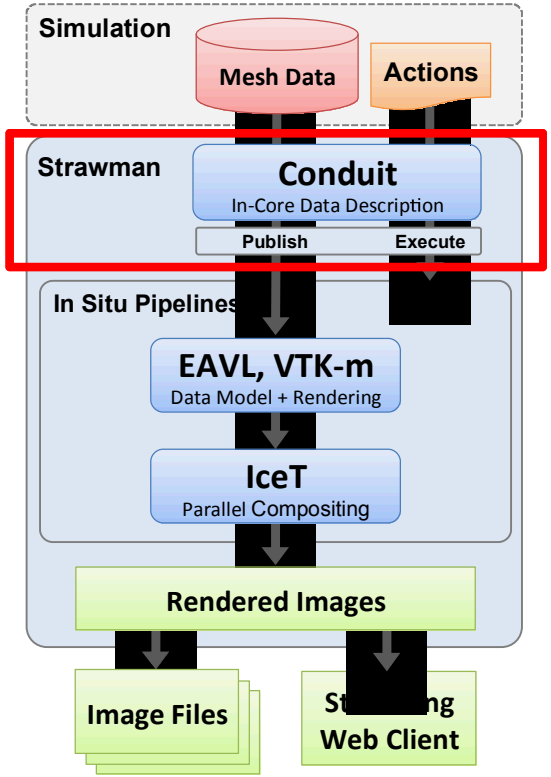
Integration Example: Strawman API calls

```
Strawman strawman;
Node options;
options["mpi_comm"] = mpi_comm_handle(MPI_COMM_WORLD);
strawman.Open(options);
strawman.Publish(data);
strawman.Execute(actions);
strawman.Close();
```



Integration Example: Describing LULESH's data

```
conduit::Node data;
data["state/time"].set_external(&time);
data["state/cycle"].set_external(&cycle);
data["state/domain"] = my_mpi_rank;
data["coords/type"] = "explicit";
data["coords/x"].set_external(x);
data["coords/y"].set_external(y);
data["coords/z"].set_external(z);
data["topology/type"] = "unstructured";
data["topology/coordset"] = "coords";
data["topology/elements/shape"] = "hexs";
data["topology/elements/connectivity"].set_external(nodelist);
data["fields/e/association"] = "element";
data["fields/e/type"]      = "scalar";
data["fields/e/values"].set_external(e);
```



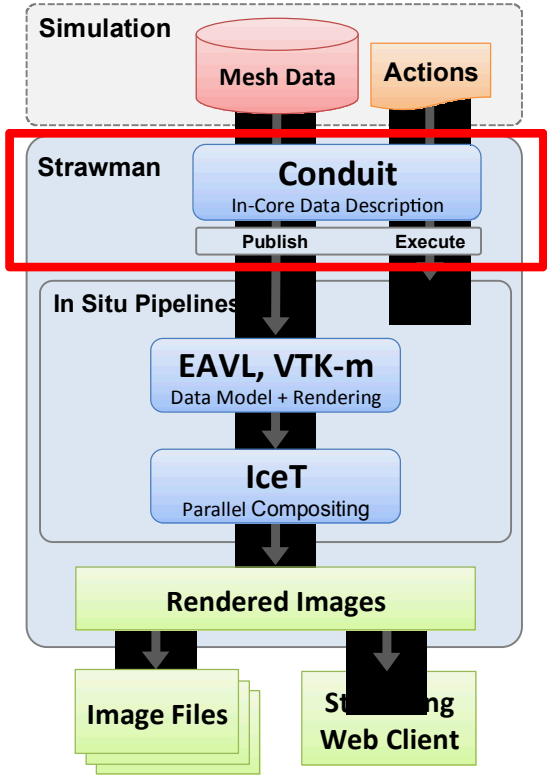
Meshes are described using the Conduit Mesh "Blueprint".

- **Coordinate Sets:**
  - Implicit: Uniform, Rectilinear
  - Explicit
- **Topologies:**
  - Implicit: Uniform, Rectilinear, Structured
  - Unstructured
    - Zoo Elements + Polygons and Polyhedra
- **Fields:**
  - Centerings and DoFs

The Blueprint provides a general set of conventions that allow us to easily target concrete APIs (VTK, EAVL, VTKm, Silo, ADIOS, etc)

Integration Example: Describing in situ actions

```
conduit::Node actions;
conduit::Node &add = actions.append();
add["action"]   = "AddPlot";
add["var"]      = "p";
conduit::Node &draw = actions.append();
draw["action"] = "DrawPlots";
conduit::Node &save = actions.append();
char output_filename[30];
sprintf(output_filename,"image%04d",cycle);
save["action"] = "SaveImage";
save["fileName"] = output_filename;
save["format"] = "png";
save["width"]  = 1024;
save["height"] = 1024;
```



A Snapshot of Our Current Work.

- **Preparing for Strawman public release**
- **Rendering Pipelines:**
  - VTK-m (in progress)
    - DPP Ray Tracer (ported from EAVL)
    - Structured Volume Render (ray casting)
    - Unstructured Volume Renderer (connectivity-based ray casting)
    - Basic Rasterizer
  - RAJA (Planned)

The VTK-m Ray Tracer Major Components.

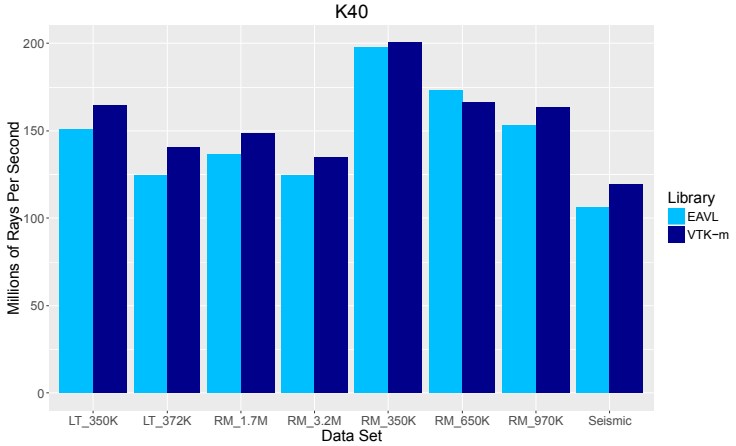
- **Acceleration Structure**
- **Intersector**

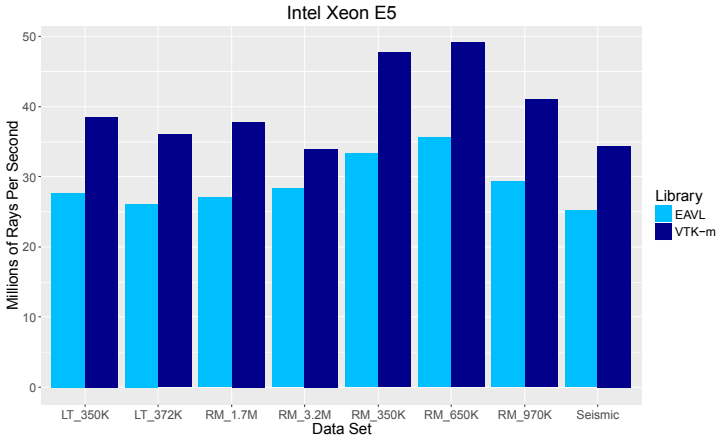
The Current BVH Choice.

- **LBVH Variant**
  - "Maximizing Parallelism in the Construction of BVHs, Octrees, and k-d Trees" (Karras, 2012)
  - Fast, but low quality tree
    - ~10% slower than SBVH for structured data
  - Straight forward data-parallel implementation
  - Key VTK-m feature: atomics
- **We plan on implementing other acceleration structures**
  - Cinema use case

Triangle Intersection Comparison With EAVL.

- **Traversal and Intersection**
  - Key VTK-m feature: texture memory
  - Currently exploring VTK-m vectorization
  - Results:





Volume Rendering Package for Multiple Use Cases.

- **Structured ray caster**
  - Single map operation
- **Unstructured ray caster** (Work in progress)
  - Construct connectivity
    - Exploring ways to generate faster and manage space-time tradeoffs
  - Traverse cell-face-cell
    - Still loosing rays (water-tight ray triangle intersection)
    - GPU kernel uses too many registers (double precision)
      - Breaking up into smaller steps
  - Generalize to support radiography use case
    - Material opacities and energy groups
    - Monte Carlo

Basic Rasterizer (VTK-m Pipeline)

- **Sample triangles using barycentric coordinates**
- **Map operation over objects**
  - Using atomics for depth test
  - Current issue:
    - Inefficient with large triangles
- **Investigate a task-based pipeline**

One More Planned Pipeline Using RAJA.

- **Portable performance abstraction over loops**
  - Designed to integrate with existing simulations
    - Example ALE3D:
      - ~20K for loops
      - 1M+ lines of code

Conclusion

- **We discussed Strawman:**
  - A new lightweight, tightly-coupled in situ proxy-app
- **We are using Strawman to understand stakeholder requirements:**
  - Including performance and usability
- **Future work:**
  - Complete Strawman's VTK-m rendering pipeline
  - Full Scaling Studies
  - ~Spring Open Source Release
- **Contact:**
  - Cyrus (cyrush@llnl.gov), Eric (brugger1@llnl.gov), Matt (larsen30@llnl.gov)
- **ISAV15 Paper:**
  - http://cdux.cs.uoregon.edu/pubs/LarsenISAV.pdf