Roadmap for Many-core Visualization Software in DOE

Jeremy Meredith
Oak Ridge National Laboratory

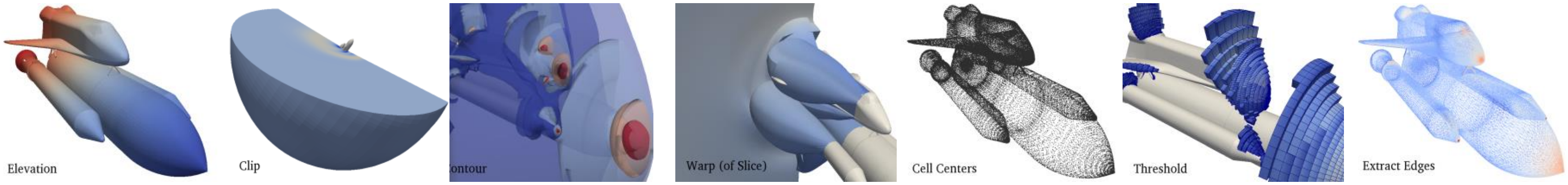
Supercomputers!

- Supercomputer Hardware Advances Everyday
  - More and more parallelism
- High-Level Parallelism
  - *“The Free Lunch Is Over”* (Herb Sutter)

	Jaguar – XT5	Titan – XK7	Exascale*
Cores	224,256	299,008 and 18,688 gpu	1 billion
Concurrency	224,256 way	70 – 500 million way	10 – 100 billion way
Memory	300 Terabytes	700 Terabytes	128 Petabytes

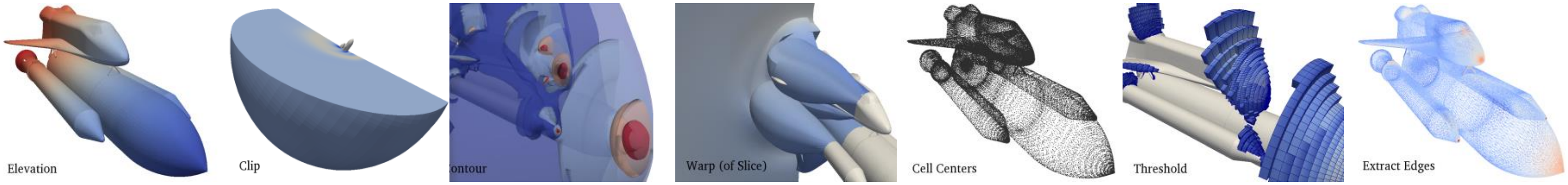
VTK-m Project

- Combines the strengths of multiple projects:
  - EAVL, Oak Ridge National Laboratory
  - DAX, Sandia National Laboratory
  - PISTON, Los Alamos National Laboratory



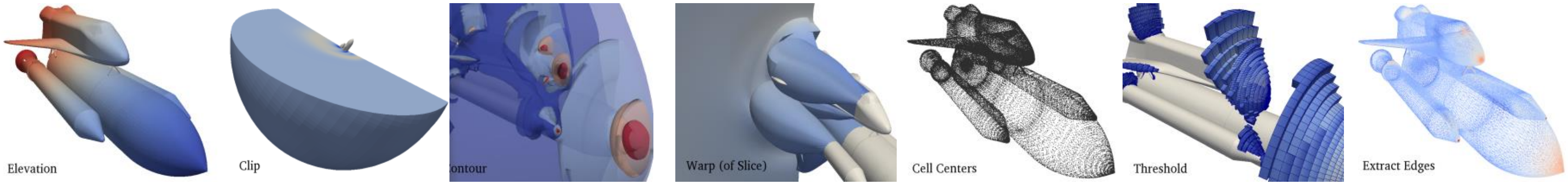
VTK-m Goals

- A single place for the visualization community to collaborate, contribute, and leverage massively threaded algorithms.
- Reduce the challenges of writing highly concurrent algorithms by using data parallel algorithms

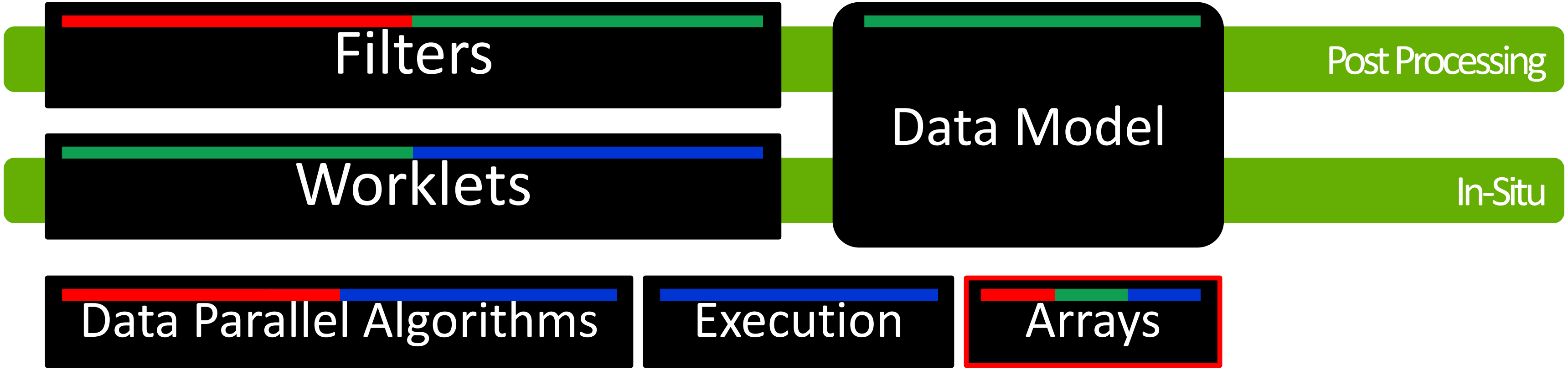


VTK-m Goals

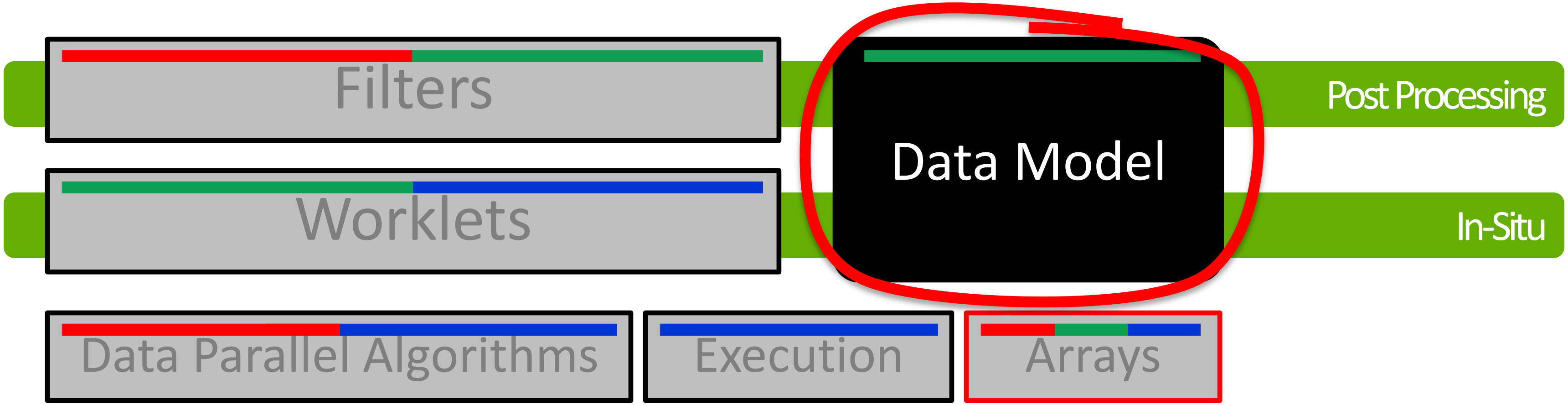
- Make it easier for simulation codes to take advantage these parallel visualization and analysis tasks on a wide range of current and next-generation hardware.



VTK-m Architecture



VTK-m Architecture



Extreme-scale Analysis and Visualization Library (**EAVL**)

EAVL enables advanced visualization and analysis for the next generation of scientific simulations, supercomputing systems, and end-user analysis tools.

New Mesh Layouts

- More accurately represent simulation data in analysis results
- Support novel simulation applications

Greater Memory Efficiency

- Support future low-memory systems
- Minimize data movement and transformation costs

Parallel Algorithm Framework

- Accelerator-based system support
- Pervasive parallelism for multi-core and many-core processors

In Situ Support

- Direct zero-copy mapping of data from simulation to analysis codes
- Heterogeneous processing models

J.S. Meredith, S. Ahern, D. Pugmire, R. Sisneros, "EAVL: The Extreme-scale Analysis and Visualization Library", Eurographics Symposium on Parallel Graphics and Visualization (EGPGV), 2012.

http://ft.ornl.gov/eavl

Gaps in Current Data Models

- Traditional data set models target only common combinations of cell and point arrangements
- This limits their expressiveness and flexibility

Cells	Coordinates	Point Arrangement: Explicit	Logical	Implicit	Hybrid
Structured	Strided	Structured Grid	?	Image Data	?
	Separated	?	Rectilinear Grid		?
	Hybrid	?	?		?
Unstructured	Strided	Unstructured Grid	?	?	?
	Separated	?	?		?
	Hybrid	?	?		?

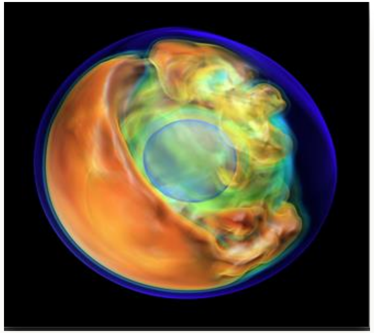
Arbitrary Compositions for Flexibility

- EAVL allows clients to construct data sets from cell and point arrangements that exactly match their original data
  - In effect, this allows for hybrid and novel mesh types
- Native data results in greater accuracy and efficiency

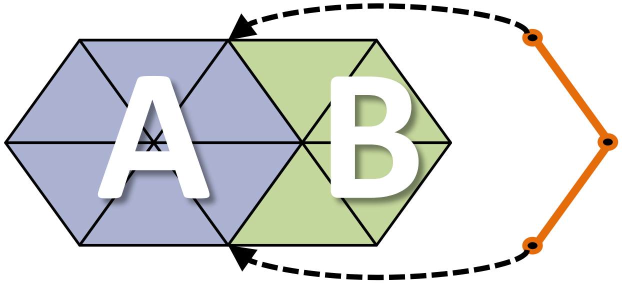
		Point Arrangement			
Cells	**Coordinates**	**Explicit**	**Logical**	**Implicit**	**Hybrid**
Structured	Strided	✓	✓	✓	✓
	Separated	✓	✓	✓	✓
	Hybrid	✓	✓	✓	✓
Unstructured	Strided	✓	✓	✓	✓
	Separated	✓	✓	✓	✓
	Hybrid	✓	✓	✓	✓

EAVL Data Set

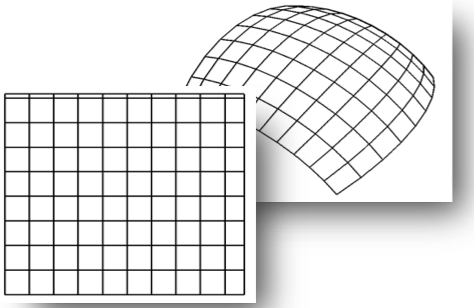
Other Data Model Gaps Addressed in EAVL



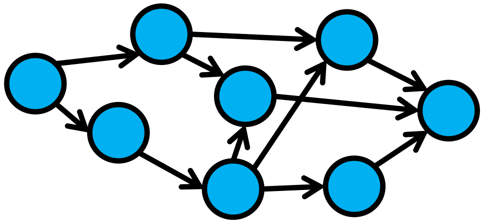
Low/high dimensional data
(9D mesh in GenASiS)



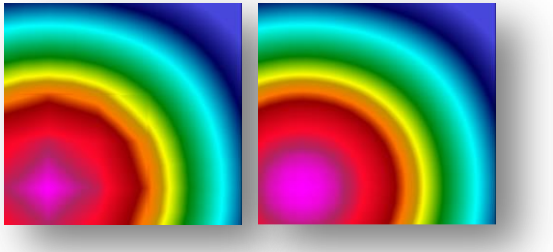
Multiple cell groups in one mesh
(E.g. subsets, face sets, flux surfaces)



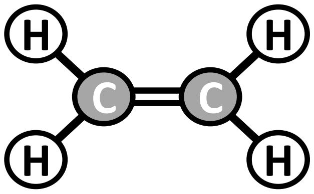
Multiple simultaneous
coordinate systems
(lat/lon + Cartesian xyz)



Non-physical data (graph,
sensor, performance data)



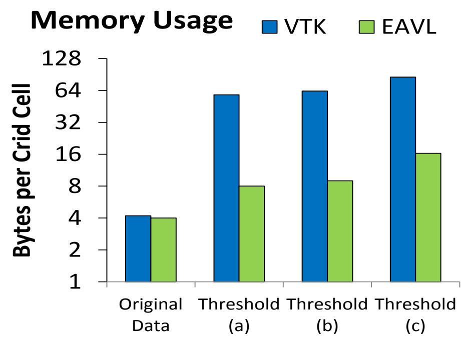
Novel and hybrid mesh types
(quadtree grid from MADNESS)

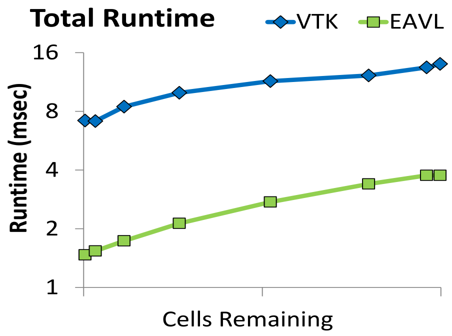


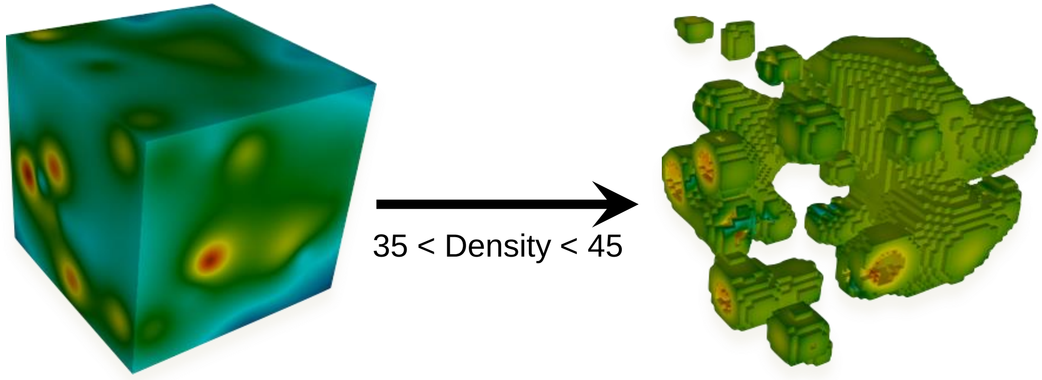
Mixed topology meshes
(atoms + bonds, sidesets)

Memory Efficiency in EAVL

- Data model designed for memory efficient representations
  - Lower memory usage for same mesh relative to traditional data models
  - Less data movement for common transformations leads to faster operation
- Example: threshold data selection
  - 7x memory usage reduction
  - 5x performance improvement



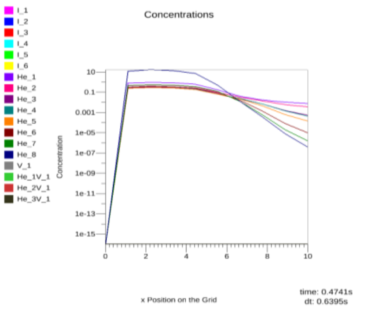




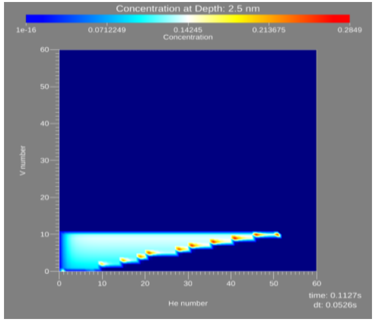
Tightly Coupled In Situ with EAVL

- Efficient in situ visualization and analysis
  - light weight, zero-dependency library
  - zero-copy references to host simulation
  - heterogeneous memory support for accelerators
  - flexible data model supports non-physical data types
- Example: scientific and performance visualization, tightly coupled EAVL with SciDAC *Xolotl* plasma/surface simulation

In Situ Scientific Visualization with Xolotl and EAVL

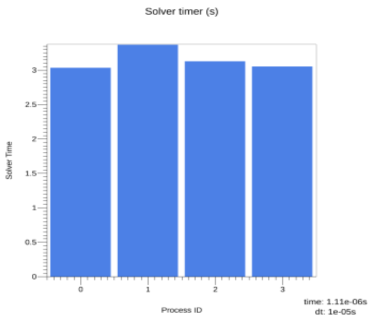


Species concentrations across grid

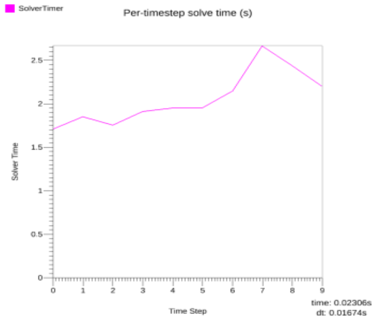


Cluster concentrations at 2.5mm

In Situ Performance Visualization with Xolotl and EAVL



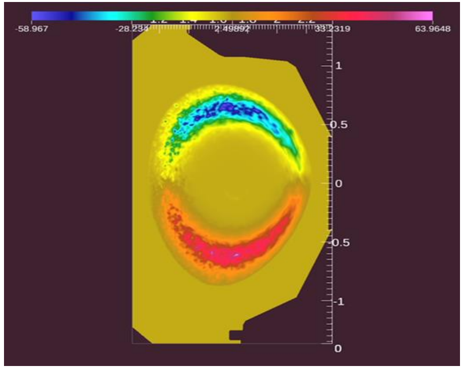
Solver time for each MPI task



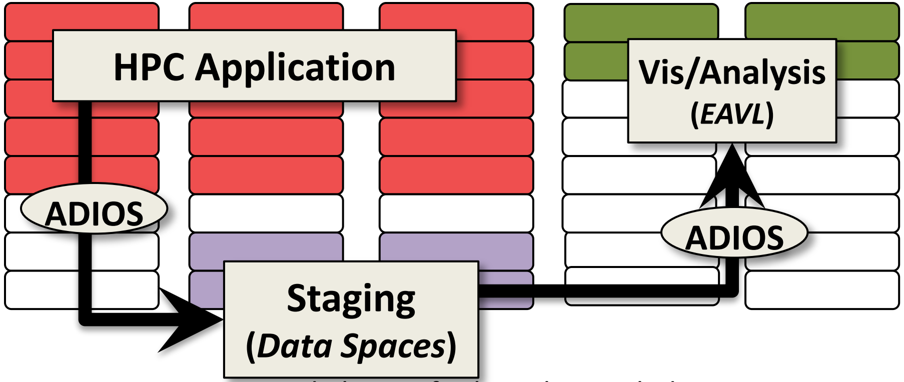
Solver time at each time step

Loosely coupled In Situ with EAVL

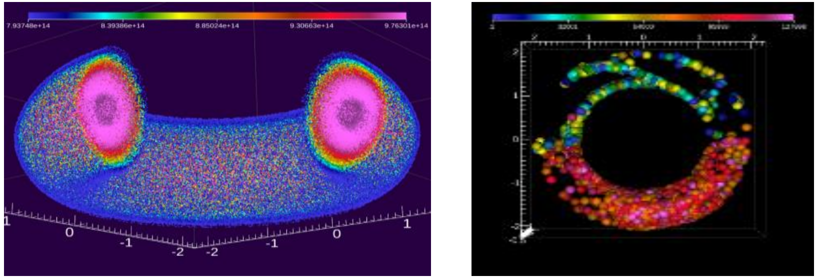
- Application de-coupled from visualization using ADIOS and Data Spaces
  - EAVL plug-in reads data from staging nodes
  - System nodes running EAVL perform visualization operations and rendering
- Example: field and particle data, EAVL in situ with XGC SciDAC simulation via ADIOS and Data Spaces



Visualization of XGC field data from running simulation

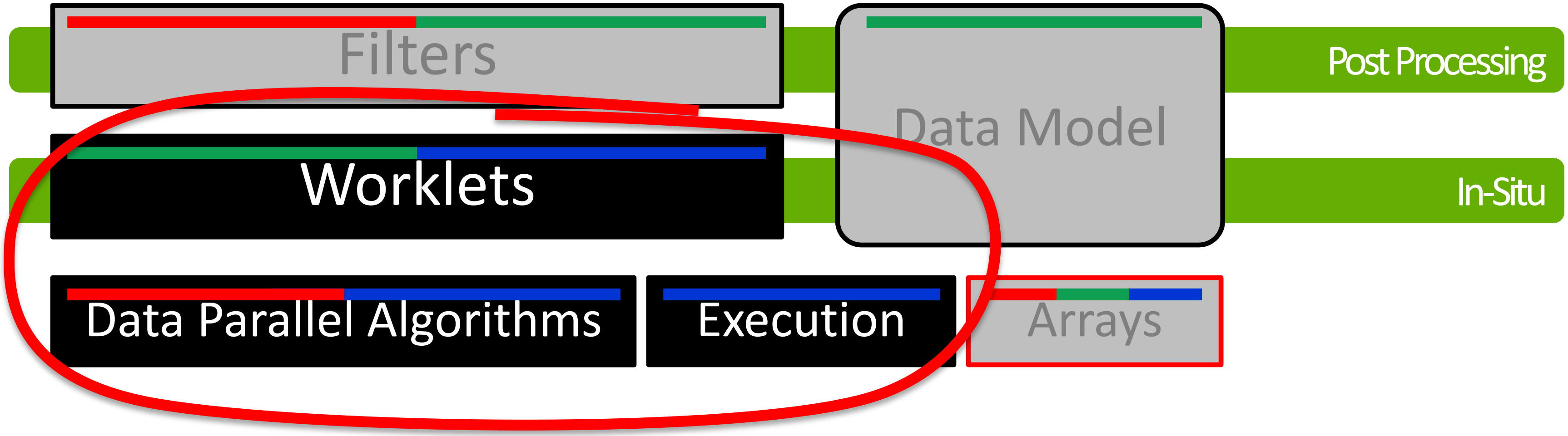


Supercomputer node layout for loosely coupled EAVL in situ



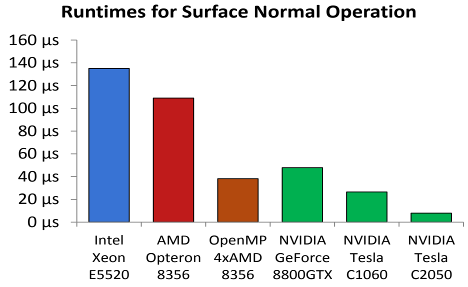
Visualization of XGC particles from running simulation. All particles (left), and selected subset of particles (right).

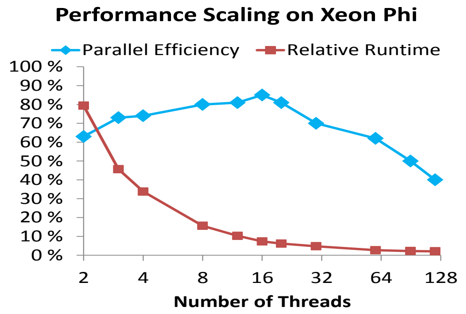
VTK-m Architecture



Data Parallelism in EAVL

- Algorithm development framework in EAVL combines productivity with pervasive parallelism
  - Data parallel primitives map functors onto mesh-aware iteration patterns
- Example: surface normal operation
  - strong performance scaling on multi-core and many-core devices (CPU, GPU, MIC/KNF)





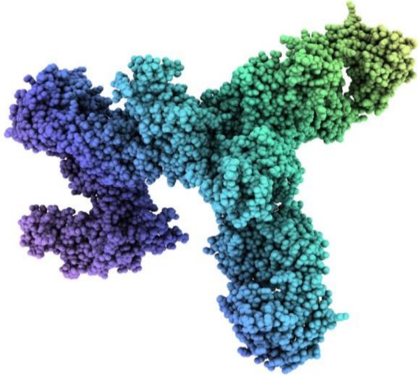
Publications:

- D. Pugmire, J. Kress, J.S. Meredith, N. Podhorszki, J. Choi, S. Klasky, “Towards Scalable Visualization Plugins for Data Staging Workflows”, 5th International Workshop on Big Data Analytics: Challenges and Opportunities (BDAC), 2014.
- C. Sewell, J.S. Meredith, K. Moreland, T. Peterka, D. DeMarle, L.-T. Lo, J. Ahrens, R. Maynard, B. Geveci, "The SDAV Software Frameworks for Visualization and Analysis on Next-Generation Multi-Core and Many-Core Architectures", Seventh Workshop on Ultrascale Visualization (UltraVis), 2012.
- J.S. Meredith, R. Sisneros, D. Pugmire, S. Ahern, "A Distributed Data-Parallel Framework for Analysis and Visualization Algorithm Development", Workshop on General Purpose Processing on Graphics Processing Units (GPGPU5), 2012.
- J.S. Meredith, S. Ahern, D. Pugmire, R. Sisneros, "EAVL: The Extreme-scale Analysis and Visualization Library", Eurographics Symposium on Parallel Graphics and Visualization (EGPGV), 2012.

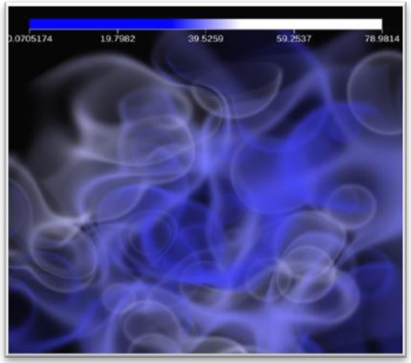
Advanced Rendering in EAVL

- Advanced rendering capabilities
  - raster/vector, ray tracing, volume rendering
  - all GPU accelerated using EAVL's data parallel API
  - parallel rendering support via MPI and IceT
- Examples: ambient occlusion lighting effects highlight subtle shape cues for scientific understanding
- Example: direct volume rendering achieves high accuracy images with GPU-accelerated performance

Shear-wave perturbations in SPECFEM3D_GLOBAL code

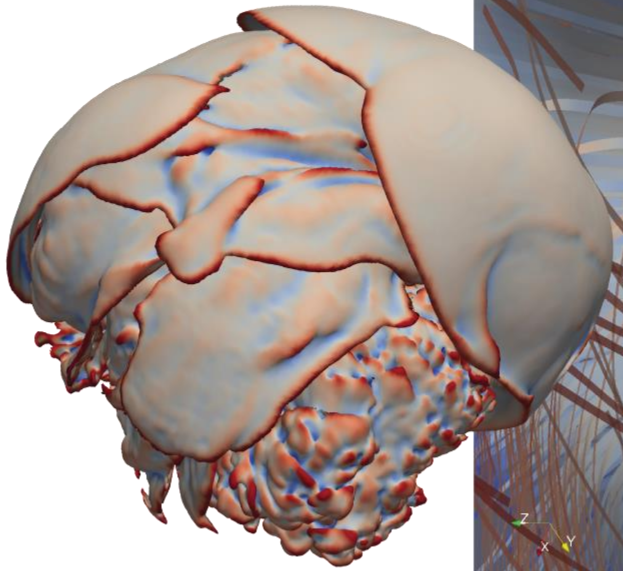


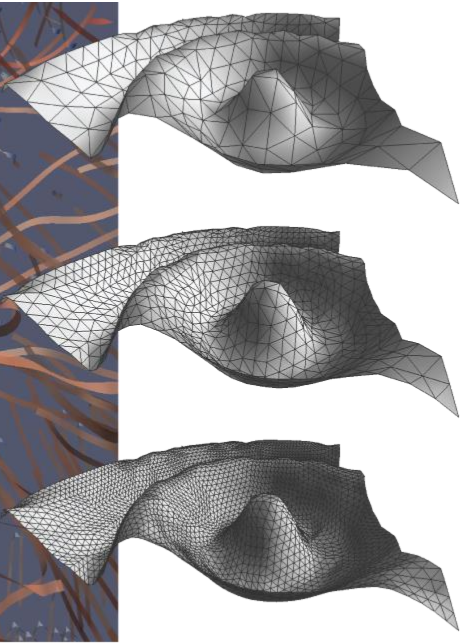
Ebola glycoprotein with proteins from survivor



Direct volume rendering from Shepard global interpolant

Dax: Data Analysis Toolkit for Extreme Scale

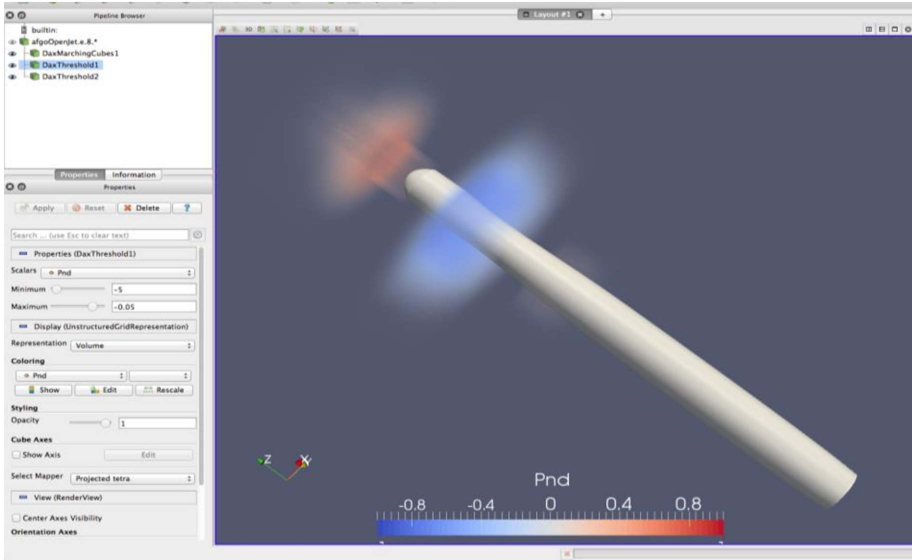




Kenneth Moreland Sandia National Laboratories
Robert Maynard Kitware, Inc.

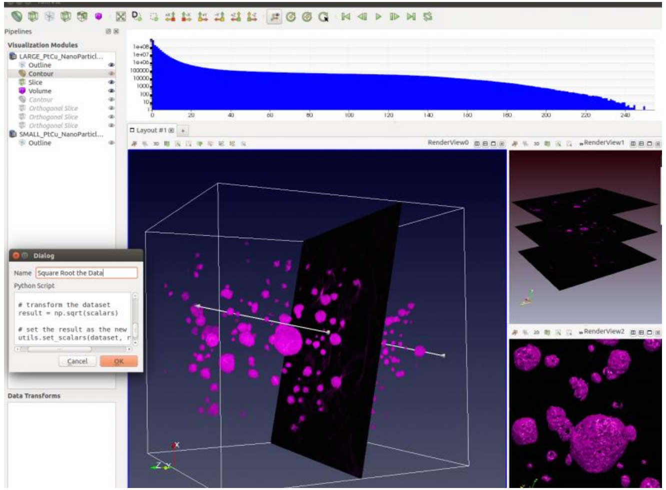
Dax Success

- ParaView/VTK
  - Zero-copy support for vtkDataArray
  - Exposed as a plugin inside ParaView
    - Will fall back to cpu version



Dax Success

- TomViz: an open, general S/TEM visualization tool
  - Built on top of ParaView framework
  - Operates on large (1024^3 and greater) volumes
  - Uses Dax for algorithm construction
- Implements streaming, interactive, incremental contouring
  - Streams indexed sub-grids to threaded contouring algorithms



```
dax::cont::ArrayHandle<dax::Scalar> inputHandle =
    dax::cont::make_ArrayHandle(input);
dax::cont::ArrayHandle<dax::Scalar> sineResult;

dax::cont::DispatcherMapField<Sine> dispatcher;
dispatcher.Invoke(inputHandle, sineResult);
```

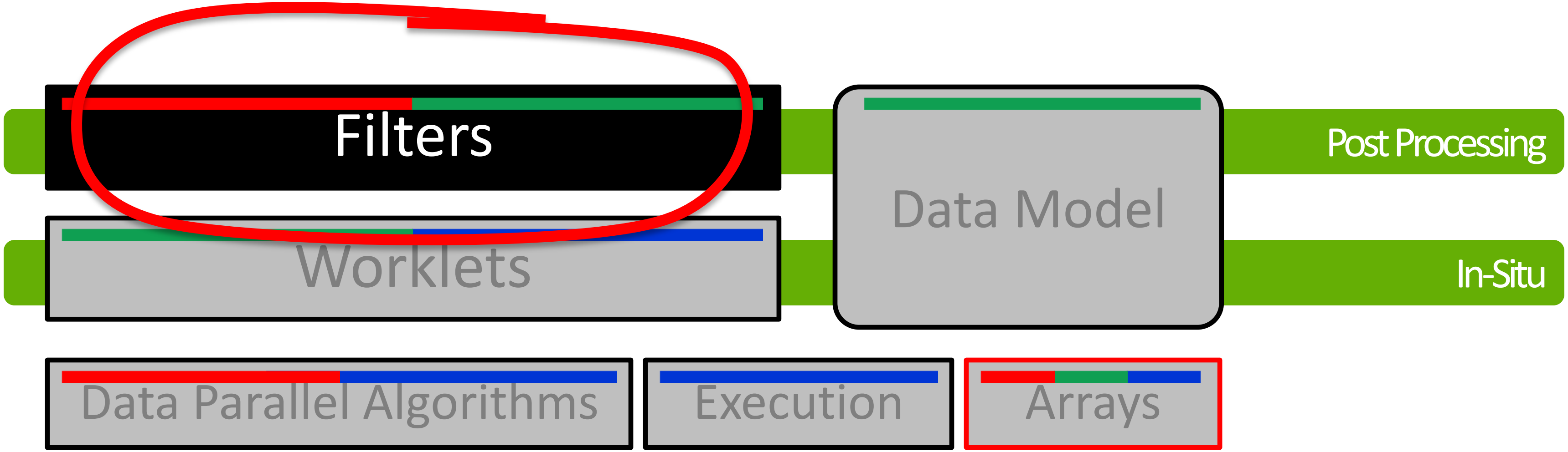
Control Environment

Execution Environment

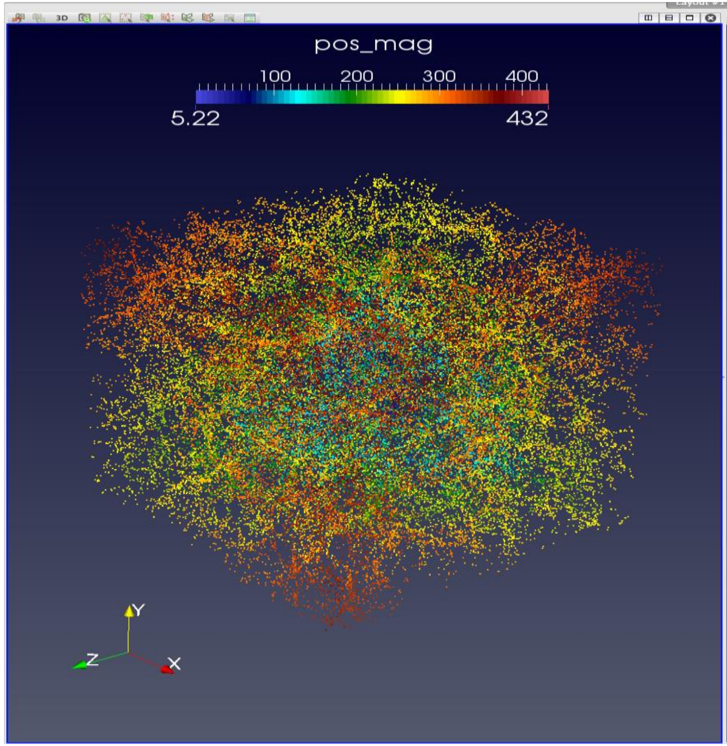
```
struct Sine: public dax::exec::WorkletMapField {
  typedef void ControlSignature(FieldIn, FieldOut);
  typedef _2 ExecutionSignature(_1);

  DAX_EXEC_EXPORT
  dax::Scalar operator()(dax::Scalar v) const {
    return dax::math::Sin(v);
  }
};
```

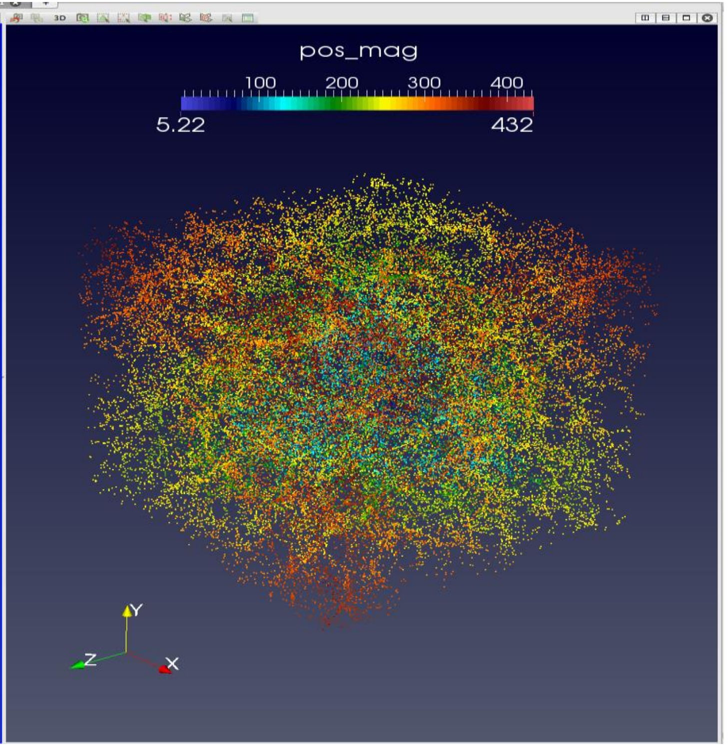
VTK-m Architecture



Results: Visual comparison of halos



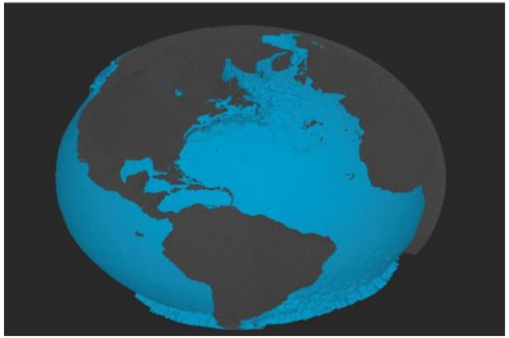
Original Algorithm



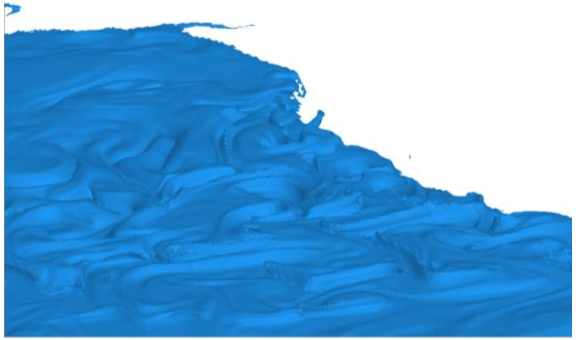
PISTON Algorithm

Piston

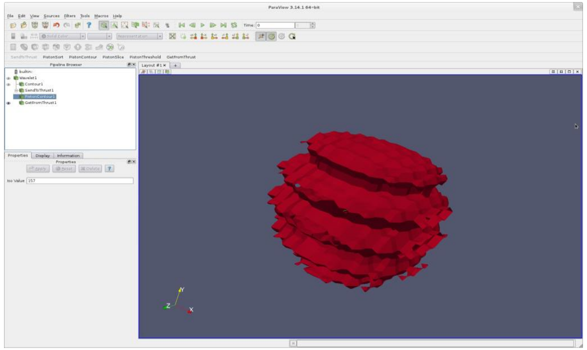
- Focuses on developing data-parallel algorithms that are portable across multi-core and many-core architectures for use by LCF codes of interest
- Algorithms are integrated into LCF codes in-situ either directly or though integration with ParaView Catalyst



PISTON isosurface with curvilinear coordinates



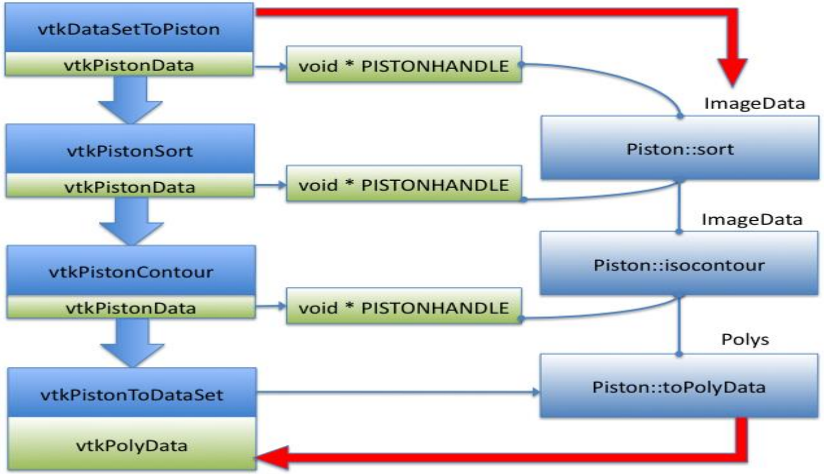
Ocean temperature isosurface generated across four GPUs using distributed PISTON

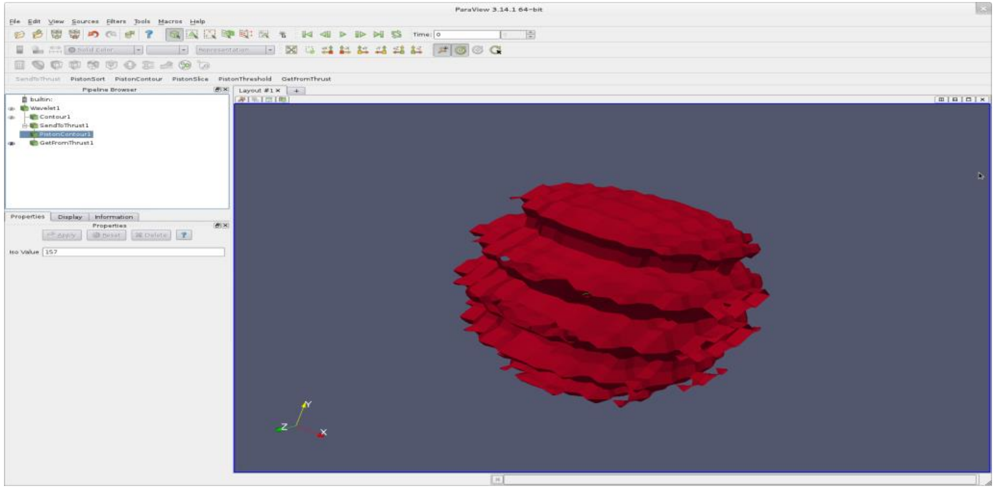


PISTON integration with VTK and ParaView

Integration with VTK and ParaView

- Filters that use PISTON data types and algorithms integrated into VTK and ParaView
- Utility filters interconvert between standard VTK data format and PISTON data format (thrust device vectors)
- Supports interop for on-card rendering





Distributed Parallel Halo Finder

- Particles are distributed among processors according to a decomposition of the physical space
- Overload zones (where particles are assigned to two processors) are defined such that every halo will be fully contained within at least one processor
- Each processor finds halos within its domain: Drop in PISTON multi-/many-core accelerated algorithms
- At the end, the parallel halo finder performs a merge step to handle “mixed” halos (shared between two processors), such that a unique set of halos is reported globally

Distributed Parallel Halo Finder

Performance Improvements

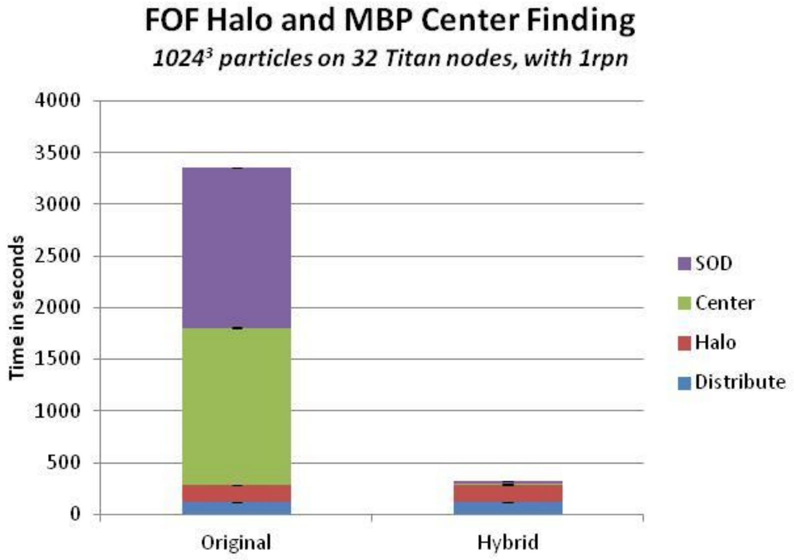
- On Moonlight with 1024^3 particles on 128 nodes with 16 processes per node, PISTON on GPUs was **4.9x** faster for halo + most bound particle center finding
- On Titan with 1024^3 particles on 32 nodes with 1 process per node, PISTON on GPUs was **11x** faster for halo + most bound particle center finding
- Implemented grid-based most bound particle center finder using a Poisson solver that performs fewer total computations than standard $O(n^2)$ algorithm

Science Impact

- These performance improvements **allowed halo analysis to be performed** on a very large **8192^3** particle data set across 16,384 nodes on Titan **for which analysis using the existing CPU algorithms was not feasible**

Publications

- Submitted to PPoPP15: "Utilizing Many-Core Accelerators for Halo and Center Finding within a Cosmology Simulation" Christopher Sewell, Li-ta Lo, Katrin Heitmann, Salman Habib, and James Ahrens



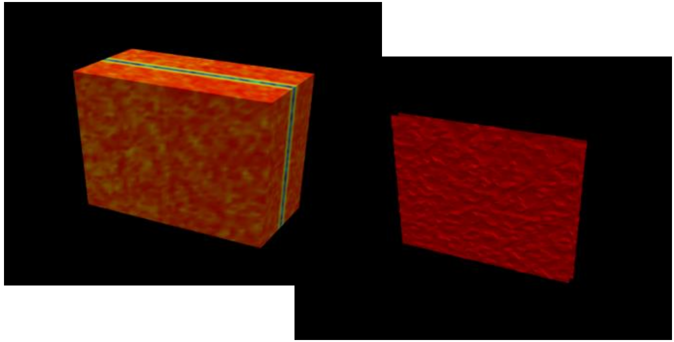
• This test problem has ~90 million particles per process.

• Due to memory constraints on the GPUs, we utilize a hybrid approach, in which the halos are computed on the CPU but the centers on the GPU.

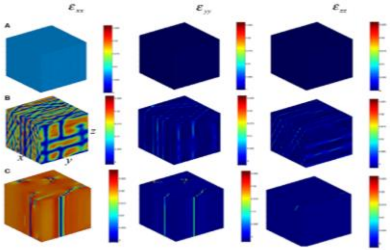
• The PISTON MBP center finding algorithm requires much less memory than the halo finding algorithm but provides the large majority of the speed-up, since MBP center finding takes much longer than FOF halo finding with the original CPU code.

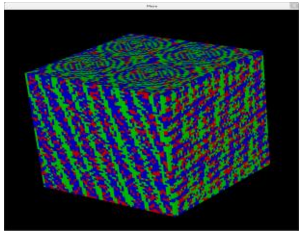
PISTON In-Situ

- VPIC (Vector Particle in Cell) Kinetic Plasma Simulation Code
  - Implemented first version of an in-situ adapter based on Paraview CoProcessing Library (Catalyst)
  - Three pipelines: vtkDataSetMapper, vtkContourFilter, vtkPistonContour
- CoGL
  - Stand-alone meso-scale simulation code developed as part of the Exascale Co-Design Center for Materials in Extreme Environments
  - Studies pattern formation in ferroelastic materials using the Ginzburg–Landau approach
  - Models cubic-to-tetragonal transitions under dynamic strain loadin
  - Simulation code and in-situ viz implemented using PISTON



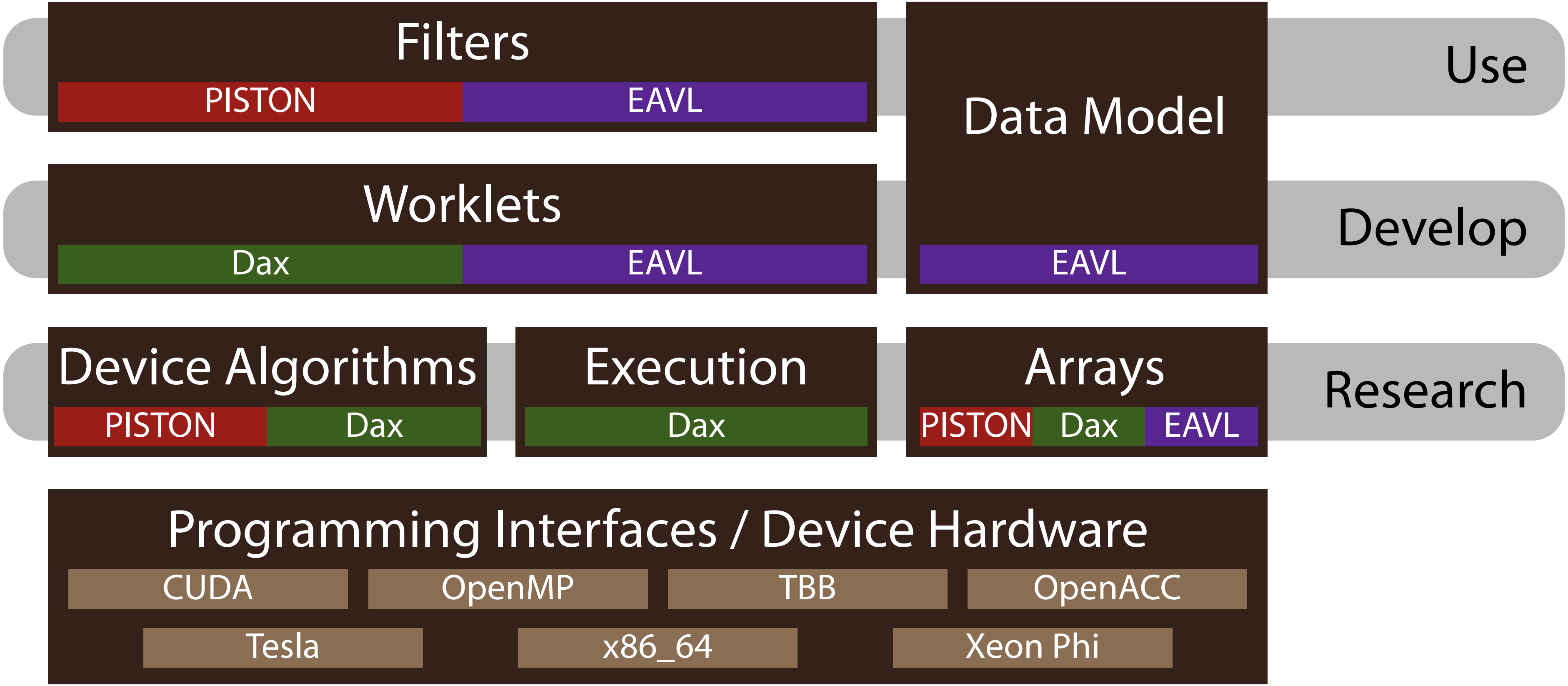
Output of vtkDataSetMapper and vtkPistonContour filters on Hhydro charge density at one timestep of VPIC simulation



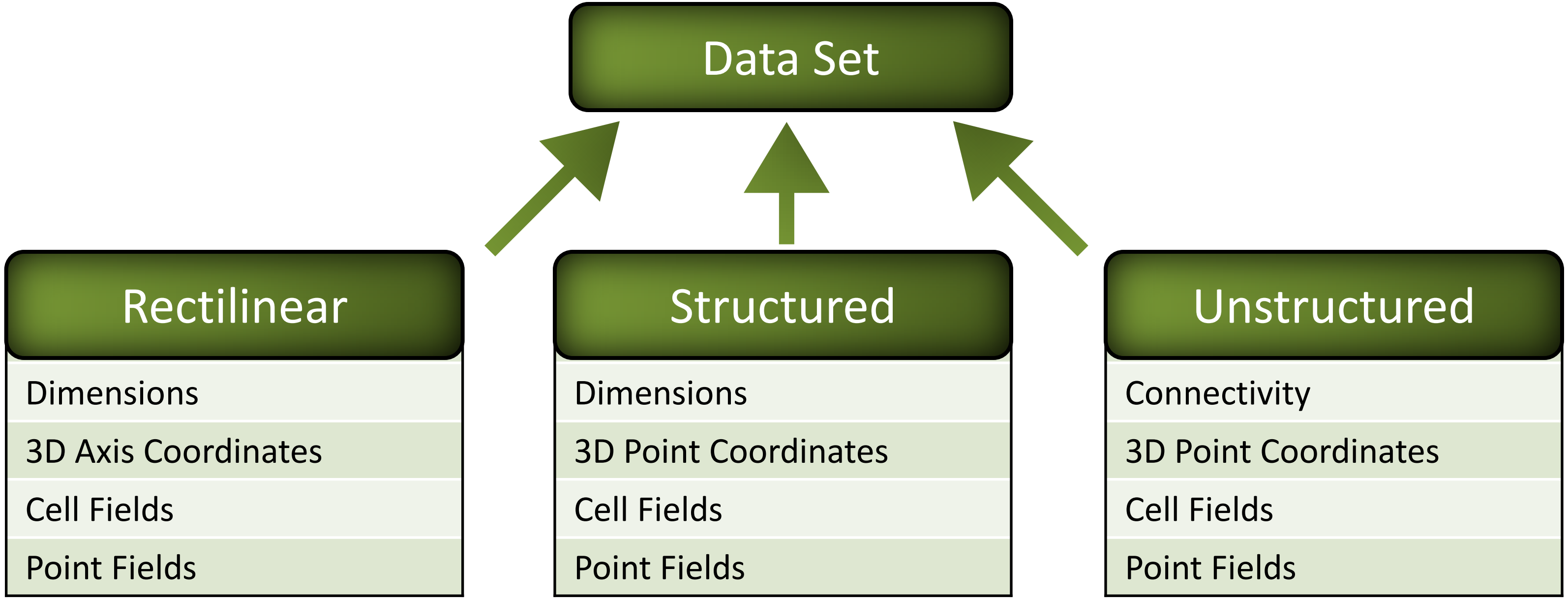


Strains in x,y,z (above); PISTON in-situ visualization (right)

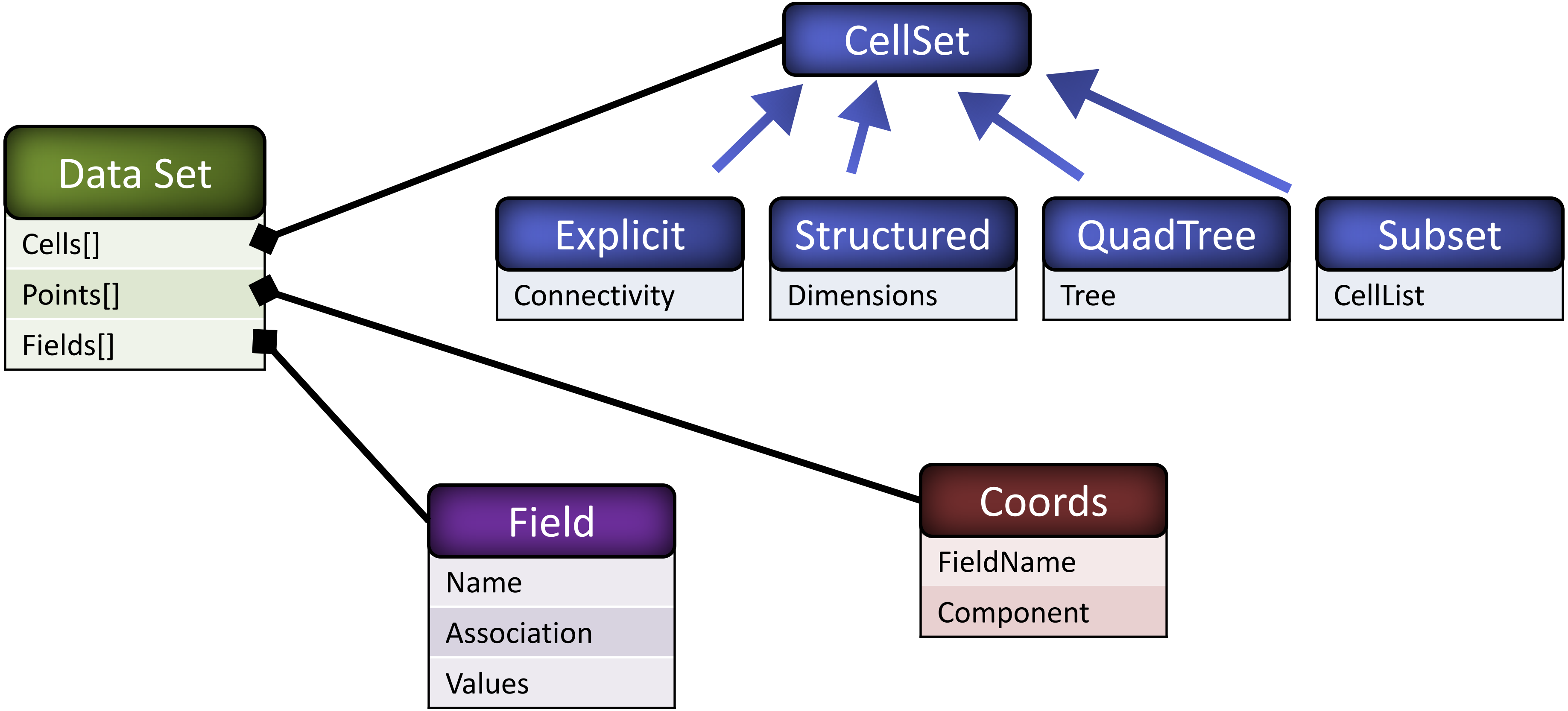
VTK-m Combines Dax, PISTON, EAVL



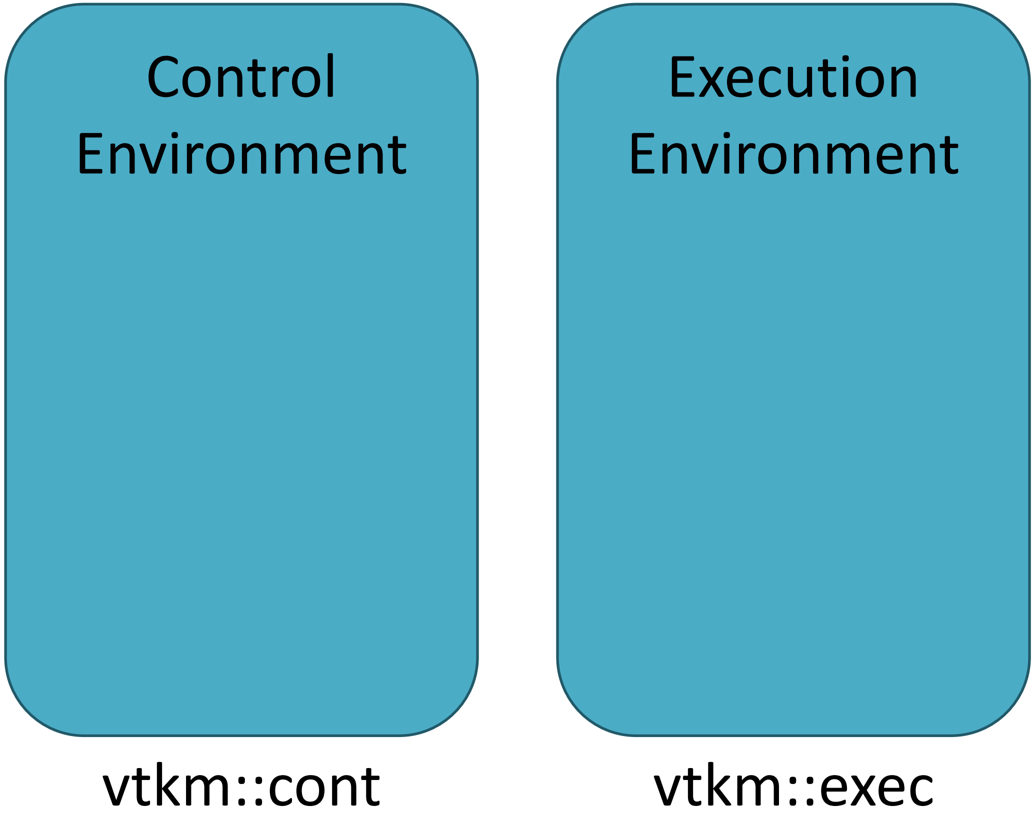
A Traditional Data Set Model



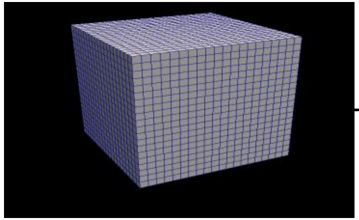
The VTK-m Data Set Model



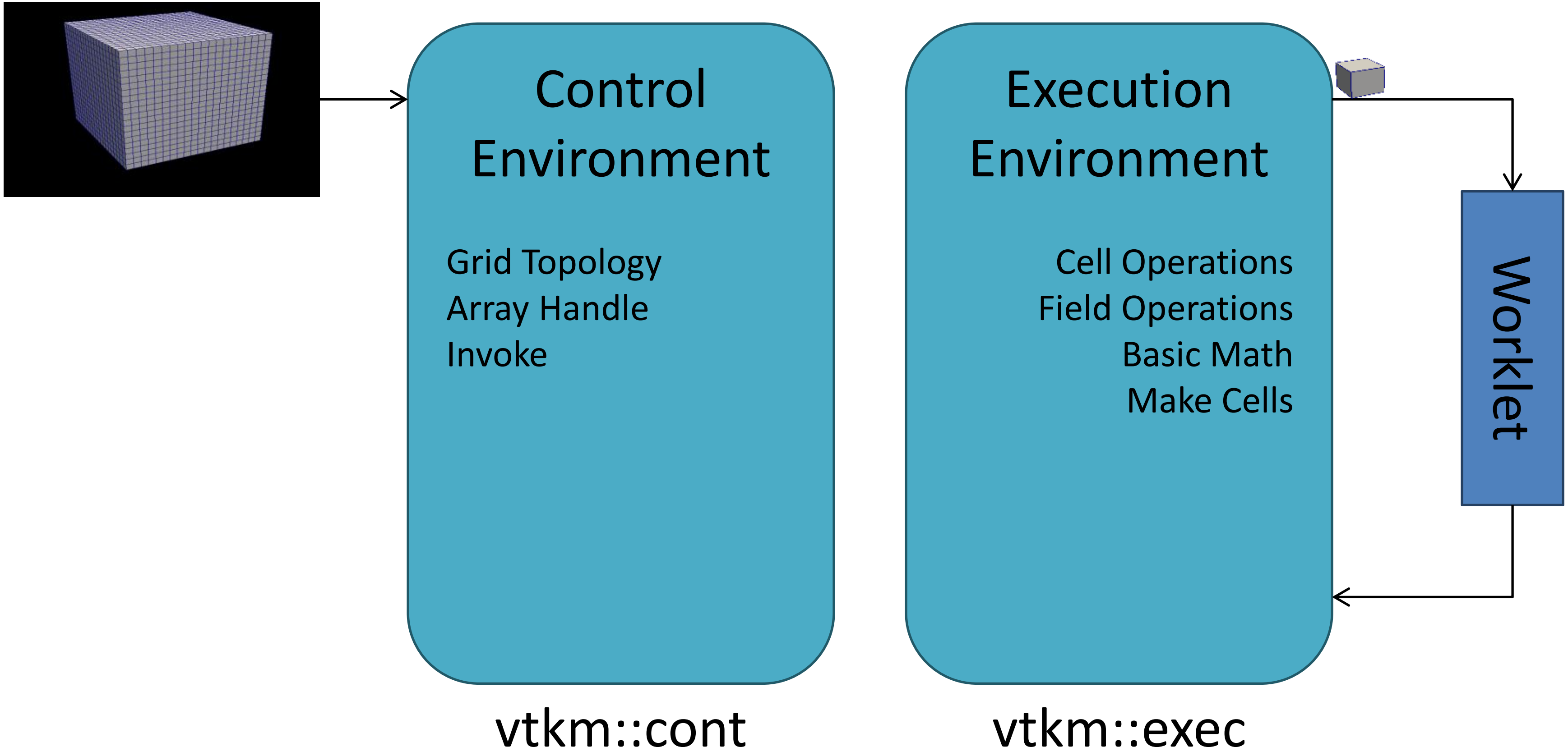
VTK-m Framework



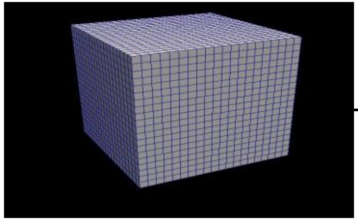
VTK-m Framework



VTK-m Framework



VTK-m Framework



Control Environment

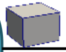
Grid Topology
Array Handle
Invoke

vtkm::cont

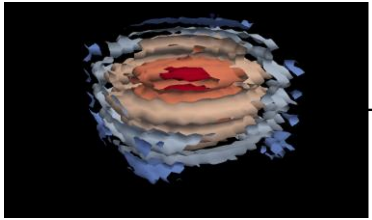
Execution Environment

Cell Operations
Field Operations
Basic Math
Make Cells

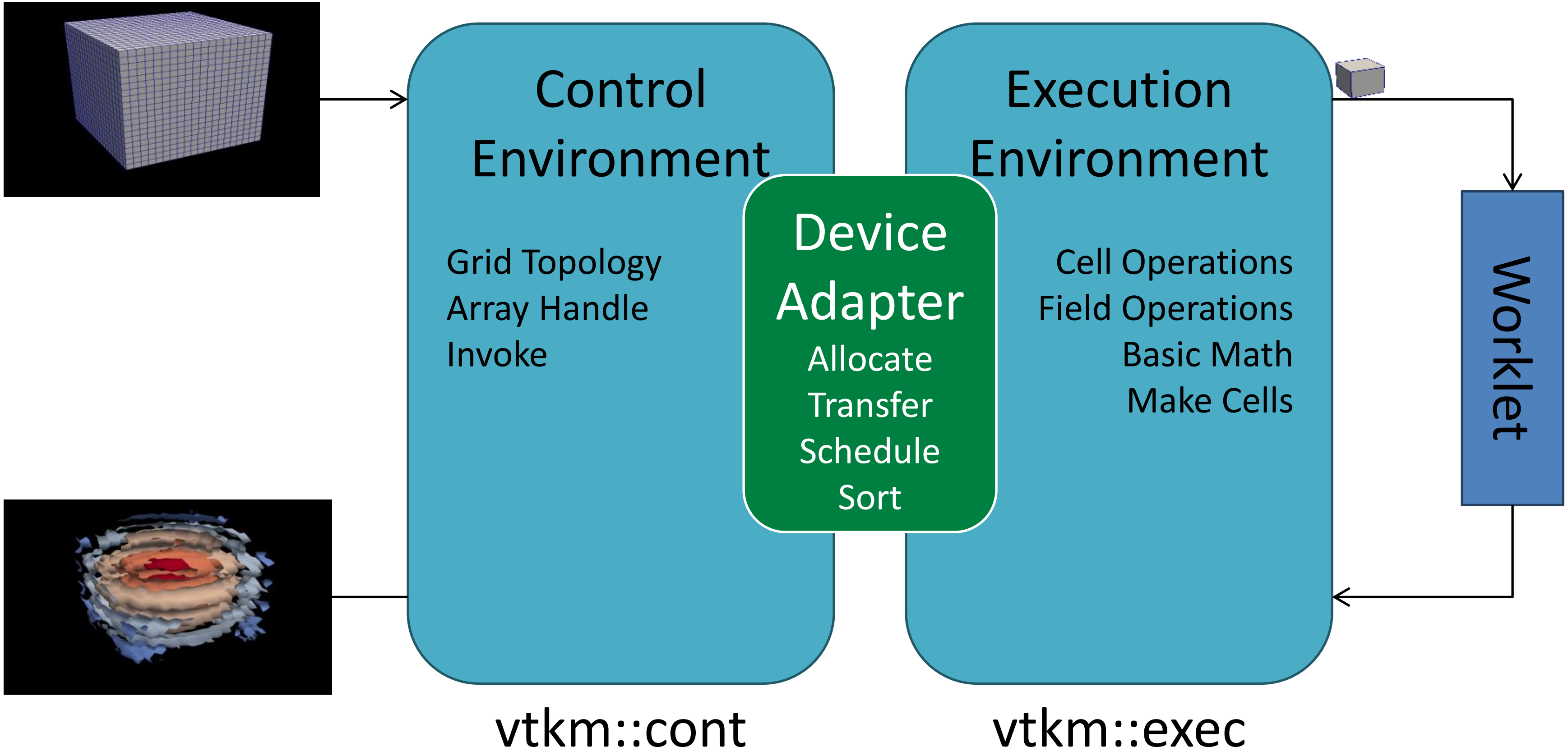
vtkm::exec



Worklet

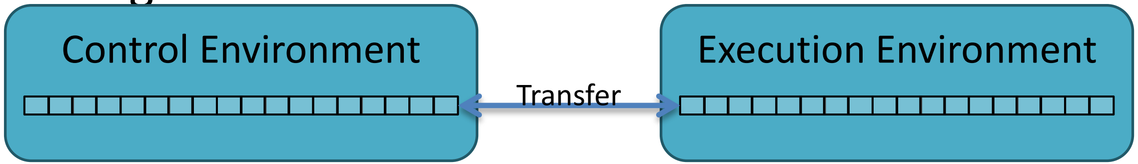


VTK-m Framework

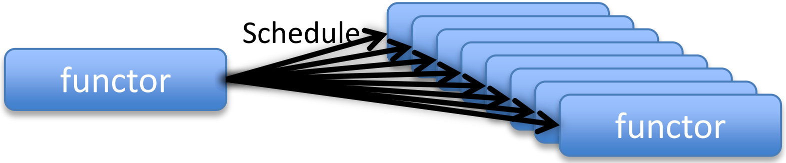


Device Adapter Contents

- Tag (`struct DeviceAdapterFoo {  };`)
- Execution Array Manager

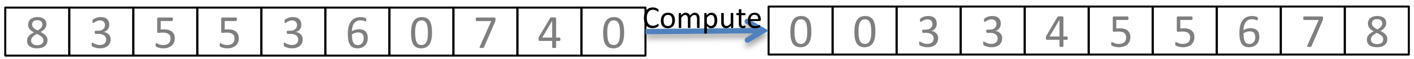


- Schedule



- Scan

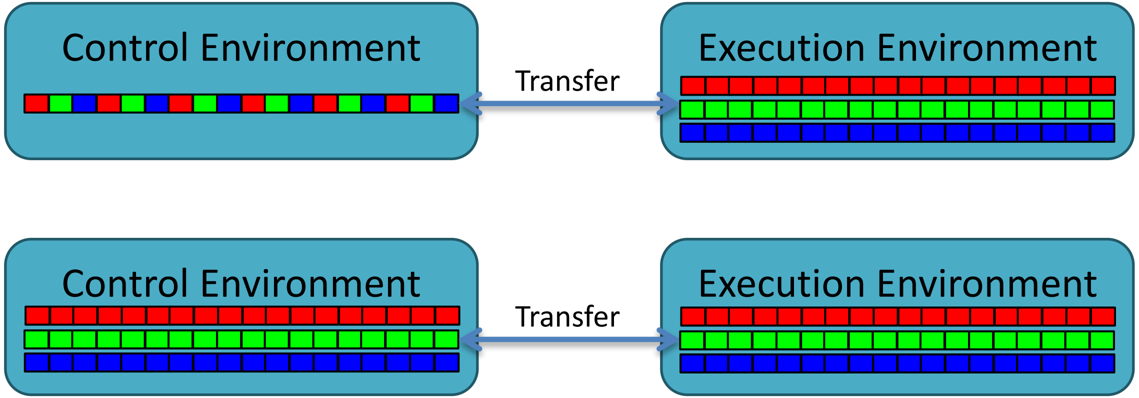
- Sort



- Other Support algorithms
  - Stream compact, copy, parallel find, unique

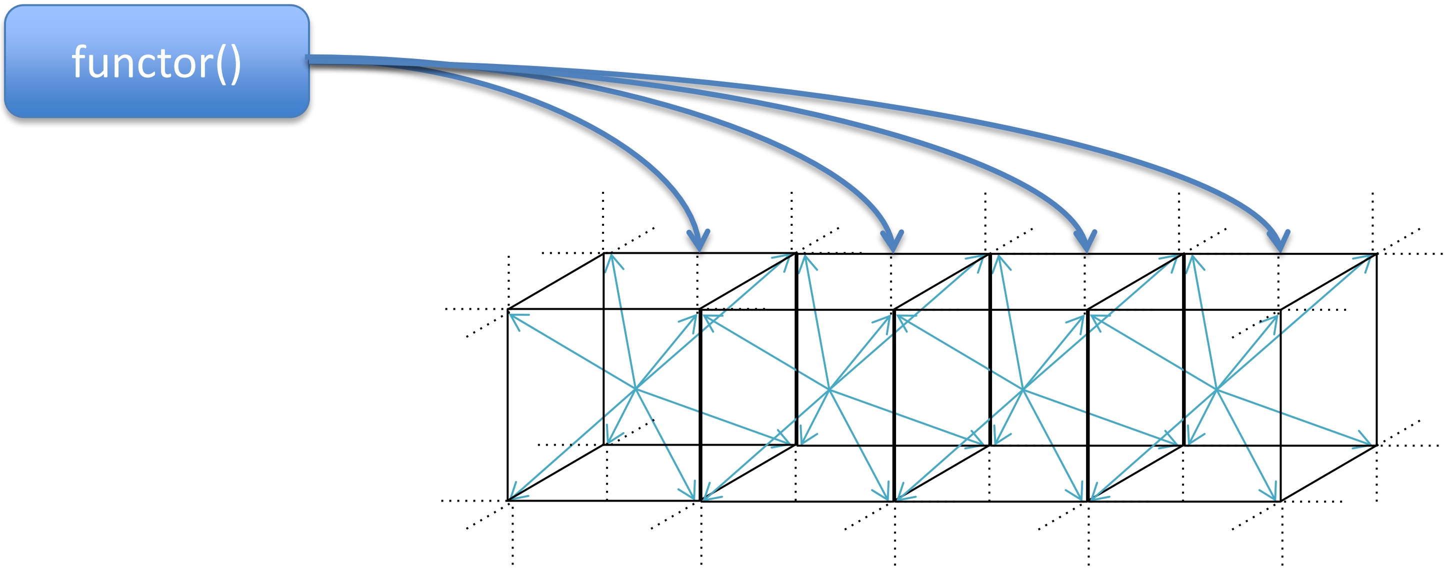
VTK-m Arbitrary Composition

- VTK-m allows clients to access different memory layouts through the Array Handle and Dynamic Array Handle.
  - –Allows for efficient in-situ integration
  - –Allows for reduced data transfer



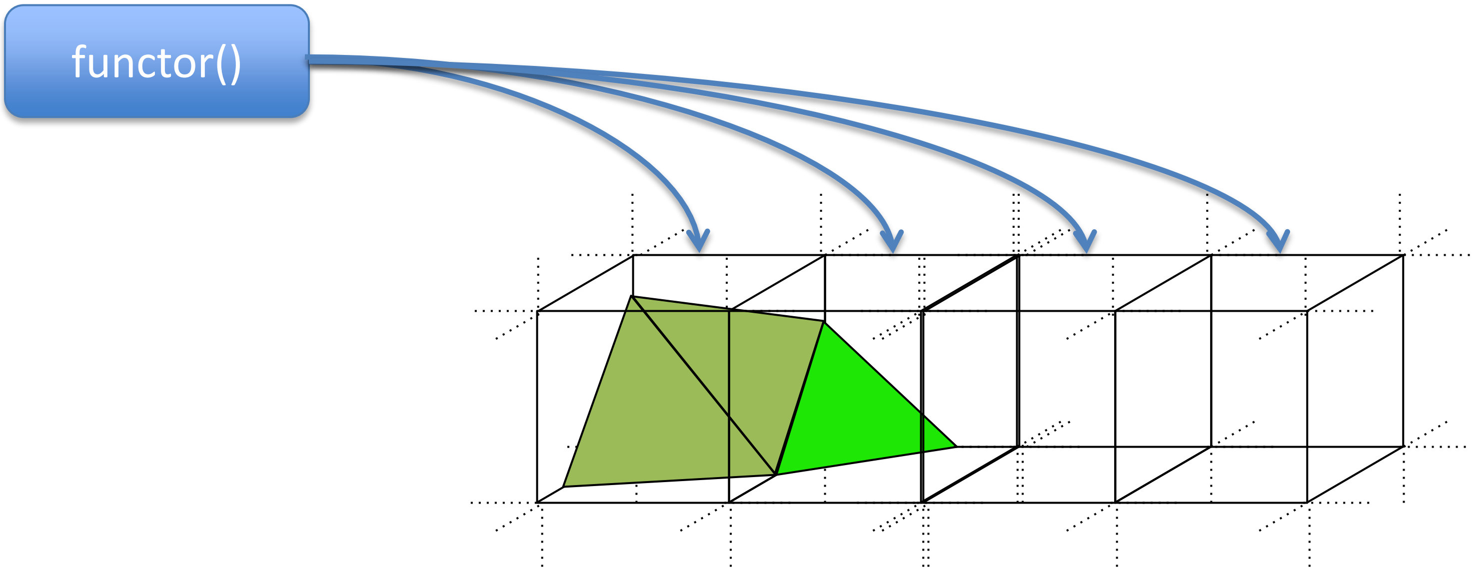
[Baker, et al. 2010]

Functor Mapping Applied to Topologies

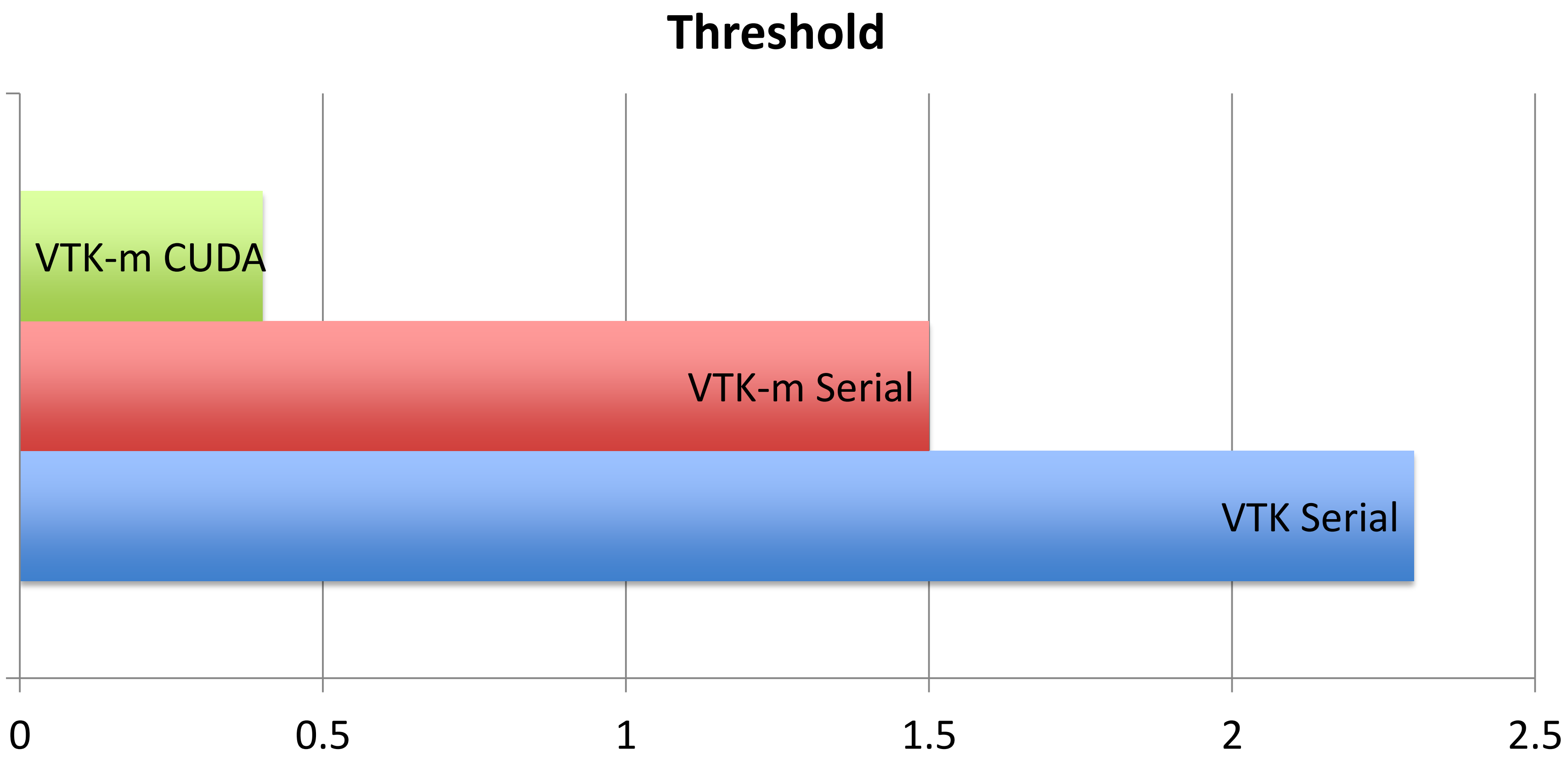


[Baker, et al. 2010]

Functor Mapping Applied to Topologies

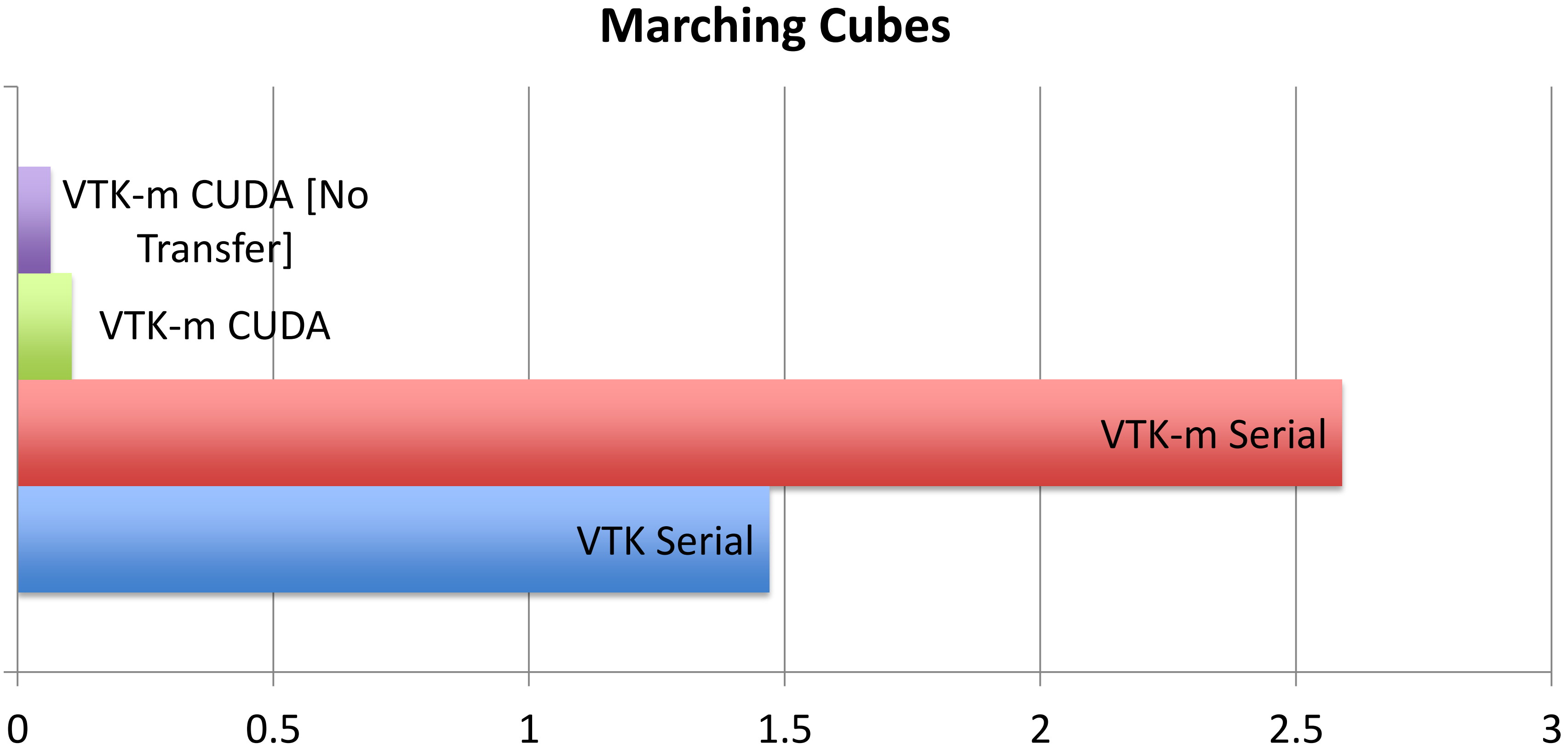


2 x Intel Xeon CPU E5-2620 v3 @ 2.40GHz + NVIDIA Tesla K40c



2 x Intel Xeon CPU E5-2620 v3 @ 2.40GHz + NVIDIA Tesla K40c

Data: 432^3



What We Have So Far

- Features
  - Core Types
  - Statically Typed Arrays
  - Dynamically Typed Arrays
  - Device Interface (Serial, CUDA, and TBB)
  - Basic Worklet and Dispatcher

What We Have So Far

- Compiles with
  - gcc (4.8+), clang, msvc (2010+), icc, and pgi
- User Guide work in progress
- Ready for larger collaboration

Questions?

m.vtk.org