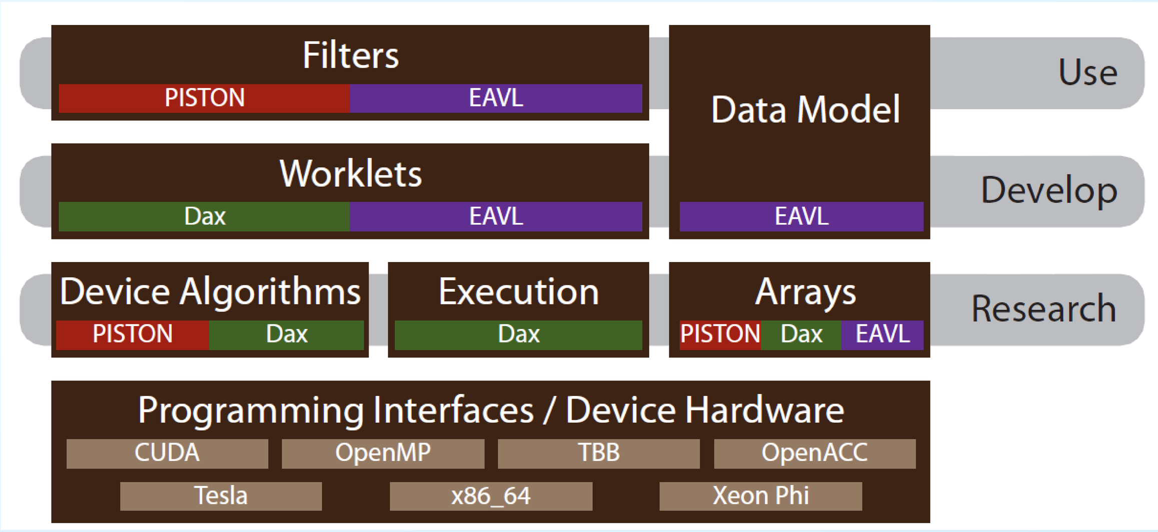
VTK-m: Accelerating the Visualization Toolkit for Multi-core and Many-core Architectures

Ken Moreland, Sandia National Laboratory
Robert Maynard and Berk Geveci, Kitware
Jeremy Meredith and Dave Pugmire, Oak Ridge National Laboratory
Christopher Sewell, Li-ta Lo, and James Ahrens, Los Alamos National Laboratory
Hank Childs and Matt Larsen, University of Oregon
Kwan-Liu Ma and Hendrik Schroots, University of California at Davis

VTK-m Goals

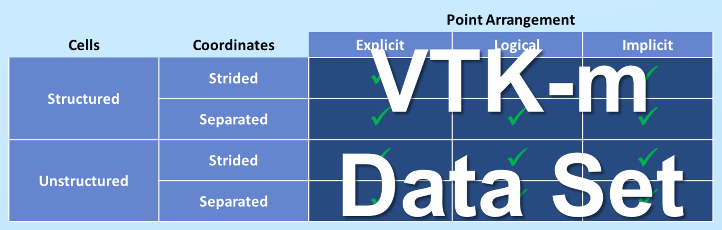
- A single place for the visualization community to collaborate, contribute, and leverage massively threaded algorithms
- Reduce the challenges of writing highly concurrent algorithms by using data parallel algorithms
- Make it easier for simulation codes to take advantage these parallel visualization and analysis tasks on a wide range of current and next-generation hardware
- Unify efforts in this area from Sandia (Dax), Oak Ridge (EAVL), and Los Alamos (PISTON)



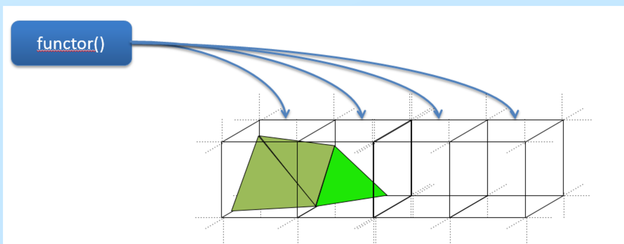
VTK-m infrastructure and use cases, with contributions from Dax, EAVL, and PISTON predecessor projects

VTK-m Status

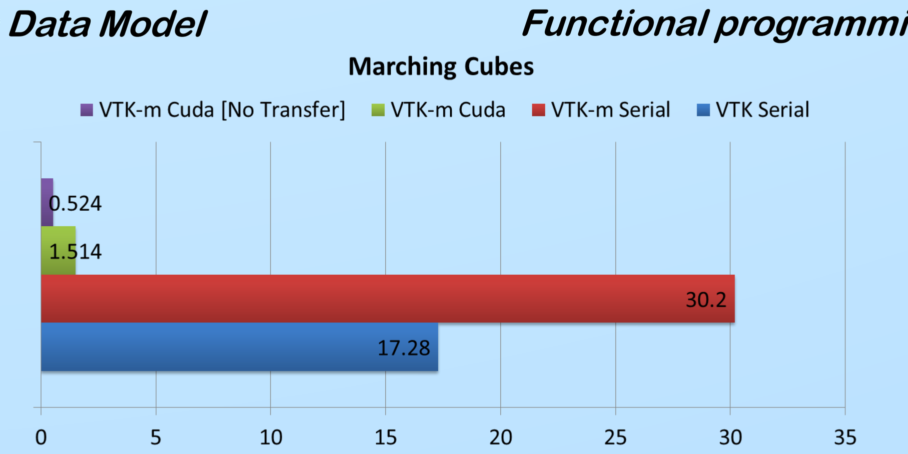
- Project infrastructure
  - Code repository: https://gitlab.kitware.com/vtk/vtk-m
  - Project webpage: http://m.vtk.org
- Features
  - Core Types
  - Statically Typed Arrays
  - Dynamically Typed Arrays
  - Device Interface (Serial, CUDA, TBB; OpenMP in progress)
  - Field and Topology Worklet and Dispatcher
- Data Model
  - 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
- Filters
  - Isosurface for structured grids
  - Statistical filters (histograms, moments, etc.)
  - In development: stream lines, stream surfaces, tetrahedralization



VTK-m Data Model



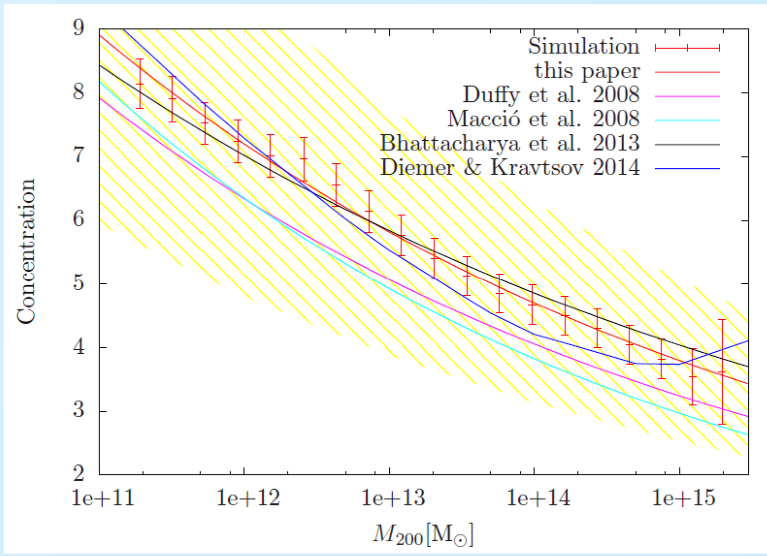
Functional programming paradigm



VTK-m Isosurface Performance (preliminary results)

Cosmology Applications

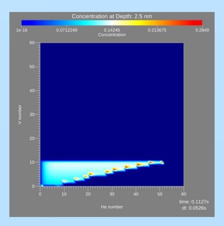
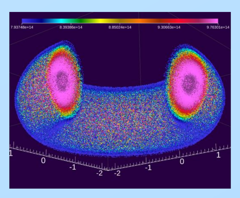
- Halo finding and halo center finding algorithms were written using PISTON, one of VTK-m's constituent projects
- On Titan, this enabled centers to be found on the GPU ~50x faster than using the pre-existing algorithms on the CPU (with one rank per node)
- This work allowed halo analysis to be completed on all time steps of 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
- The portability of VTK-m allowed us to run the same code on an Intel Xeon Phi
- This is the first time that the c-M relation has been measured from a single simulation volume over such an extended mass range
- To appear in the Astrophysical Journal: "The Q Continuum Simulation: Harnessing the Power of GPU Accelerated Supercomputers".



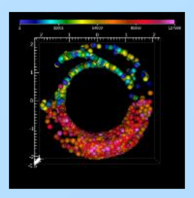
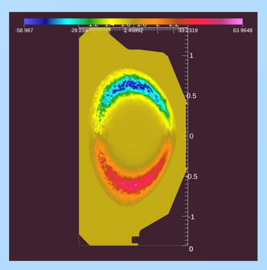
Concentration-mass relation over the full mass range covered by the Q Continuum simulation at redshift z = 0 (points with error bars) and the predictions from various groups. The yellow shaded region shows the intrinsic scatter. All predictions and the simulation results are well within that scatter.

In-situ Applications

- Tightly coupled in-situ with EAVL, one of VTK-m's constituent projects
  - 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
- 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

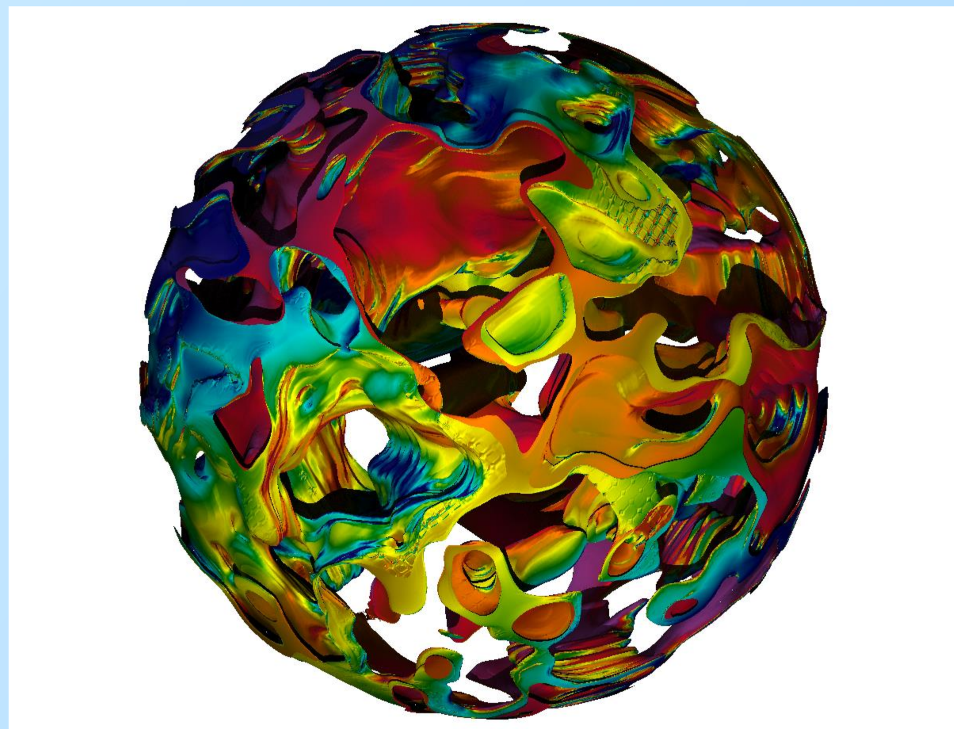
EAVL in-situ with Xolotl

EAVL in-situ with XGC

Hardware-Agnostic Ray Tracing

- VTK-m's hardware-agnostic approach gives comparable performance to hardware-specific approaches
- Since VTK-m is implemented in a hardware-agnostic way, we wanted to understand the corresponding sacrifice in performance
- We implemented a ray-traced renderer, which is computationally intensive and uses many unstructured memory accesses
- We then compared VTK-m's performance to NVIDIA's OptiX and Intel's Embree, two "guaranteed not to exceed" ray-tracing standards that are developed by teams of professionals
- Our study found that VTK-m performance was always within a factor of two of industry standards, and even outperformed them in some cases
- We concluded that VTK-m hardware-agnostic approach is viable - our single implementation performed comparably to multiple hardware-specific implementations

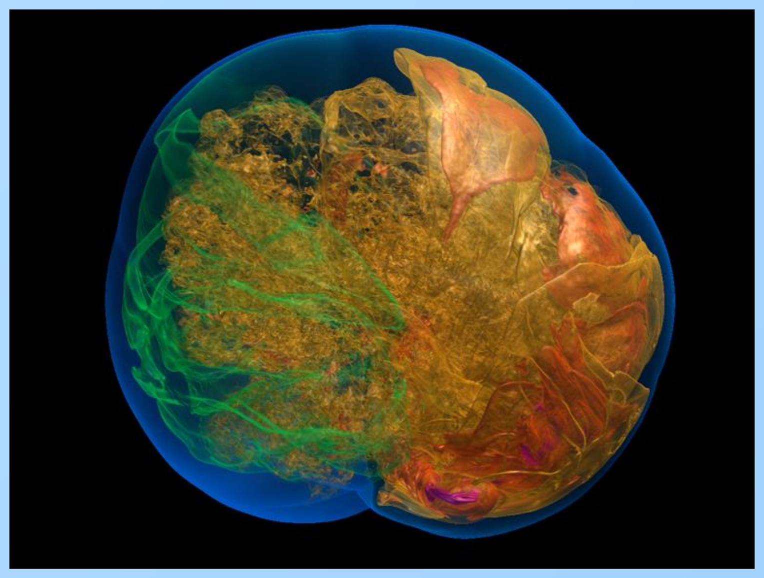


Ray-traced rendering of 6.2M triangles generated from SPECFEM3D. The data represents wave speed perturbations measured by seismograms and was provided by Oak Ridge National Laboratory.

Advanced Visualization Usability Study

- Implementation of both ray-casting and cell projection volume rendering algorithms using Dax, one of VTK-m's constituent projects
- Complied for CUDA, OpenMP, and Intel's Thread Building Blocks
- Comparative performance study on NVIDIA Titan X GPU, Intel Xeon, and Intel Xeon Phi
- VTK-m implementation in progress

Volume rendering of type Ia supernova simulation data set using ray-casting. Cell projection implementation using data parallel primitives renders comparable image in near sub-second times.



Acknowledgement

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program under the Institute of Scalable Data Management, Analysis and Visualization (SDAV).