Visualization and Analysis on Multi-/Many-Core Architectures

Using the PISTON Component of the SDAV VTK-m Project

Christopher Sewell, Li-ta Lo, John Patchett, and James Ahrens
Los Alamos National Laboratory
Science Collaborators: Katrin Heitmann and Salman Habib, Argonne National Laboratory

Motivation and Approach

- **Current visualization and analysis algorithms, including those in the popular VTK library, usually do not take advantage of shared-memory parallelism available on multi-core architectures and many-core accelerators**
- **Due to fundamental physics limitations, computers are getting faster by using more parallel cores rather than faster clock rates**
- **Algorithms traditionally have to be rewritten to run efficiently on each new architecture**
- **Standards such as OpenCL may allow a program to run cross-platform, but usually still requires many architecture specific optimizations to run well**
- **Our goal: Portability and performance for visualization and analysis operators and for simulations on current and next-generation parallel architectures**
- **The main idea: Write algorithms using only data-parallel primitives (scan, reduce, transform, etc.)**
- **This approach requires architecture-specific optimizations for only for the small set of data-parallel primitives**
- **Challenge for algorithm developer: Write operators in terms of these primitives only**
- **Reward for algorithm developer: Efficient, portable code**

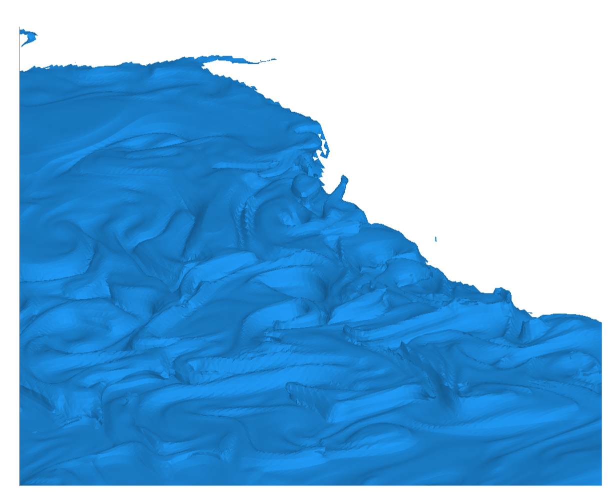
```
input                   4  5  2  1  3
-------------------------------------
transform(+1)           5  6  3  2  4
inclusive_scan(+)       4  9 11 12 15
exclusive_scan(+)       0  4  9 11 12
exclusive_scan(max)     0  4  5  5  5
transform_inscan(*2,+)  8 18 22 24 30
for_each(-1)            3  4  1  0  2
sort                    1  2  3  4  5
copy_if(n % 2 == 1)     5  1  3
reduce(+)                          15

input1                  0  0  2  4  8
input2                  3  4  1  0  2
-------------------------------------
upper_bound             3  4  2  2  3
permutation_iterator    4  8  0  0  2
```

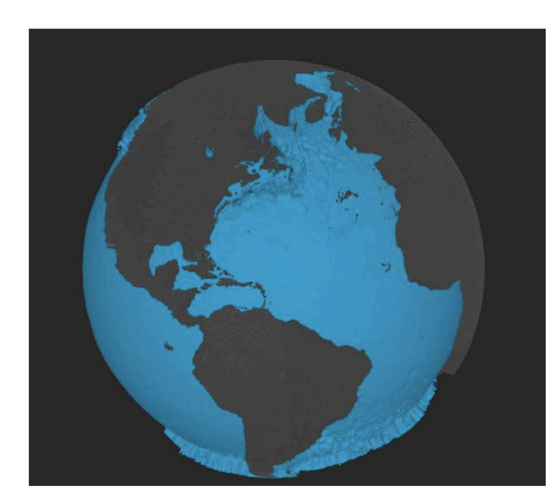
Examples of data parallel primitives

PISTON and the VTK-m Project

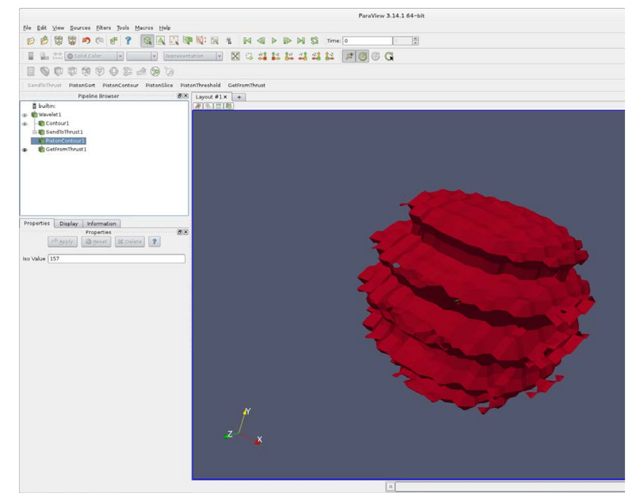
- **The goal of the VTK-m project within the Scalable Data Management, Analysis, and Visualization (SDAV) SciDAC Institute is to deliver multi-/many-core enabled visualization and analysis algorithms to scientific codes**
- **VTK-m leverages the strengths of the Dax project at Sandia National Laboratory, the EAVL project at Oak Ridge National Laboratory, and the PISTON project at Los Alamos National Laboratory**
- **The PISTON component of VTK-m focuses on developing data-parallel algorithms that are portable across multi-core and many-core architectures for use by LCF codes of interest**
- **PISTON consists of a library of visualization and analysis algorithms implemented using NVIDIA's Thrust library, as well as our set of extensions to Thrust**
- **PISTON algorithms are integrated into LCF codes in-situ either directly or though ParaView Catalyst**



Ocean temperature isosurface generated across four GPUs using distributed PISTON



PISTON isosurface of ocean temperature using curvilinear coordinates



PISTON integration with VTK and ParaView

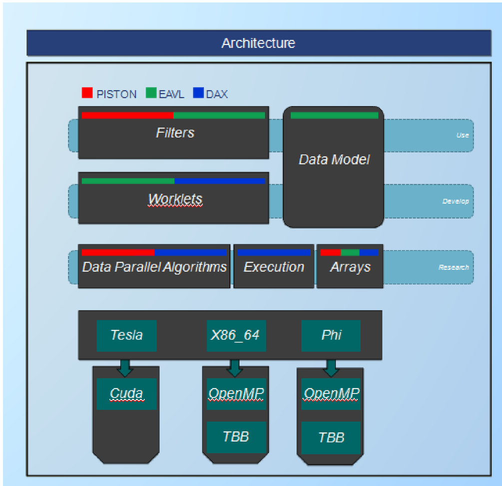
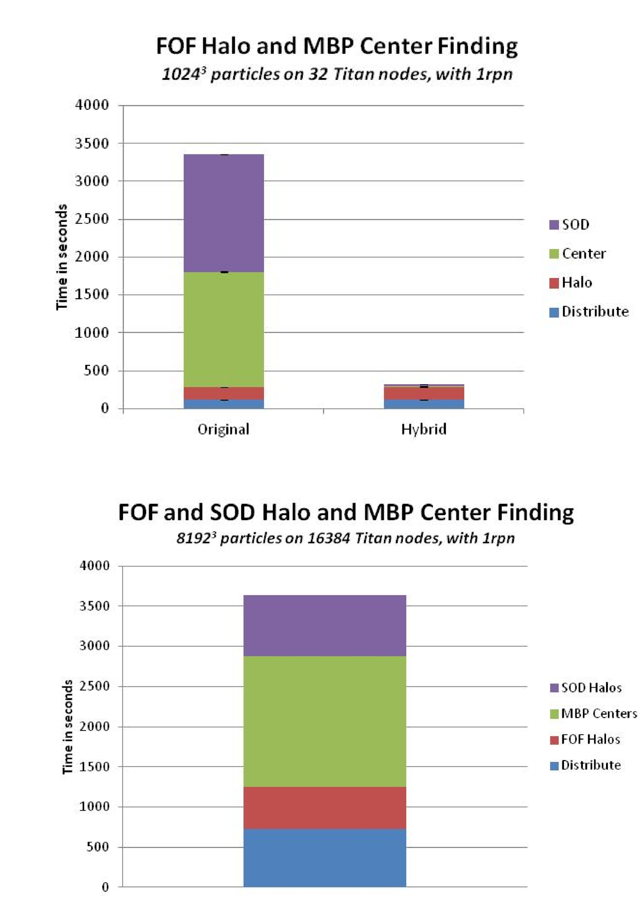


Diagram by Robert Maynard

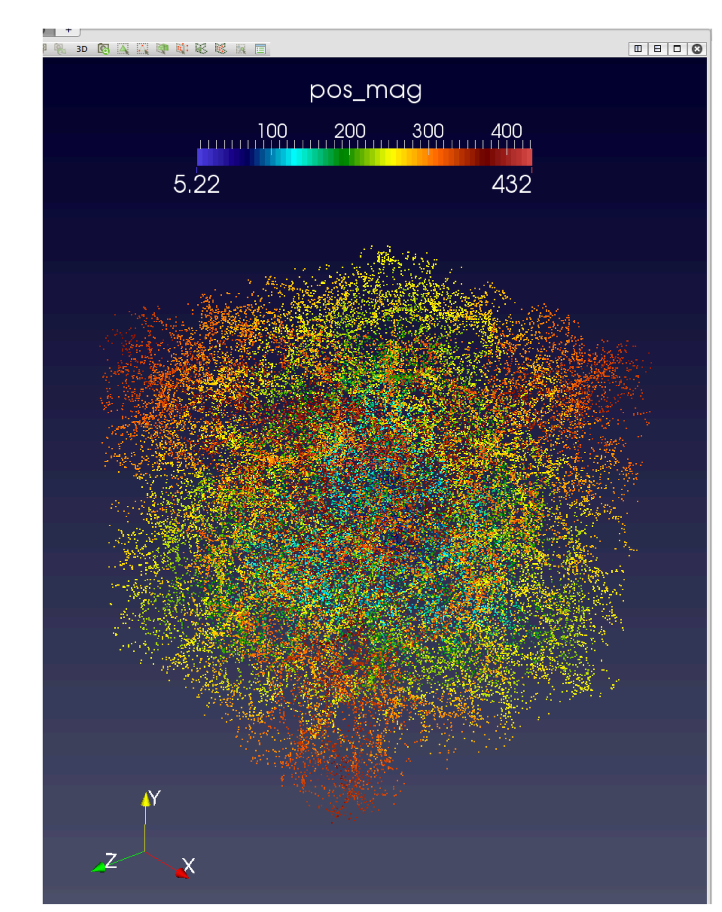
The Dax, EAVL, and PISTON components of VTK-m

Example Application to Science

- **The Hardware/Hybrid Accelerated Cosmology Code (HACC) simulates the distribution of dark matter in the universe over time**
- **An important and time-consuming analysis function within this code is finding halos (high density regions) and the centers of those halos**
- **Data-parallel algorithms for halo and center finding implemented using PISTON allow the code to take advantage of parallelism on accelerators such as GPUs**
- **PISTON on GPUs was 4.9x faster for halo + most bound particle center finding on Moonlight with 1024^3 particles on 128 nodes with 16 rpn; on Titan, a hybrid approach using PISTON center finding and CPU halo finding was 11x faster with 1024^3 particles on 32 nodes with 1 rpn**
- **Portability of PISTON also allowed running on Xeon Phi**
- **We also implemented a grid-based center finder using a Poisson solver**
- **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**



Performance of PISTON center finder on large data sets on Titan



Visualization of halo centers found using PISTON algorithm