The role of PISTON and LANL in VTK-m and the xVis Project

Chris Sewell
Li-ta Lo

Los Alamos National Laboratory

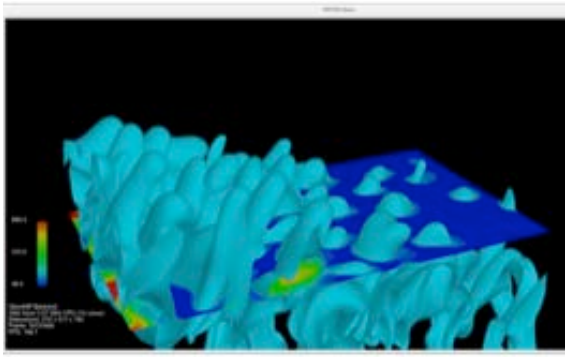
November 17, 2014

LA-UR-14-28771

PISTON Overview

- Our goal: Portability and performance for visualization and analysis operators and for simulations on current and next-generation parallel architectures
- 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
- PISTON uses NVIDIA's Thrust library, which provides an STL-like interface for memory management and backend implementations of data-parallel primitives for CUDA, OpenMP and TBB

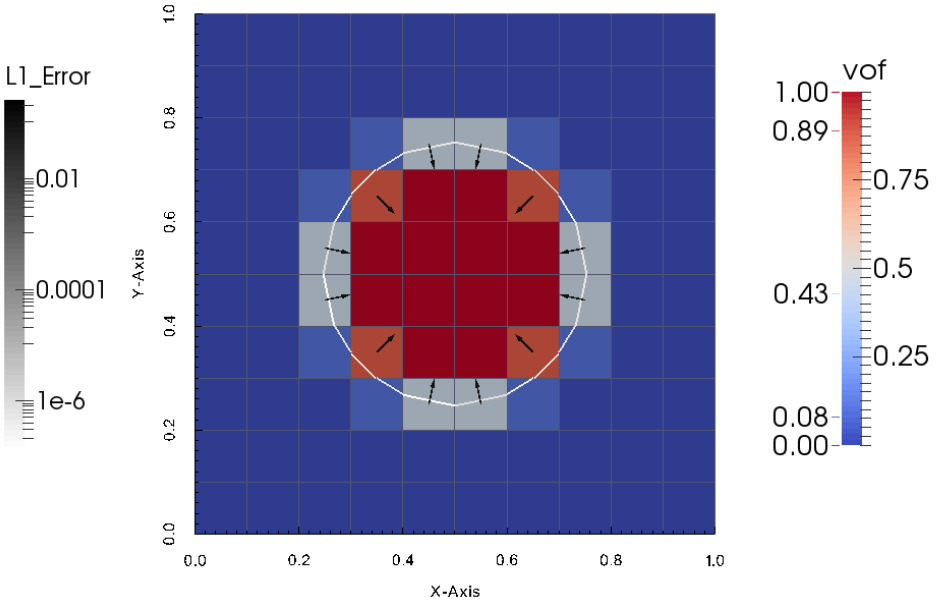
Isosurface of ocean temperature on curvilinear grid



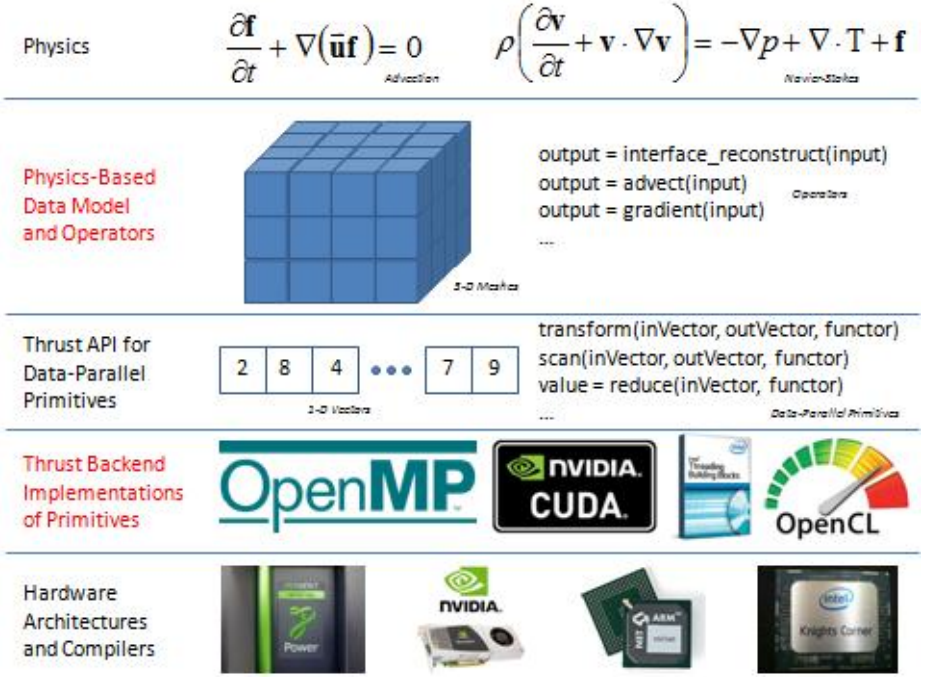
Isosurface and cut plane for Rayleigh-Taylor instability data set

PINION: Data-Parallel Physics Simulations (ASC)

- A portable, data-parallel software framework for physics simulations
- Data structures for the problem domain rather than dealing directly with 1D host/device vectors
- Operators that provide data-parallel implementations of functions often used in physics simulations
- Backends that optimize data parallel primitives for emerging architectures (e.g., Xeon Phi)



Interface reconstruction for a volume tracking simulation



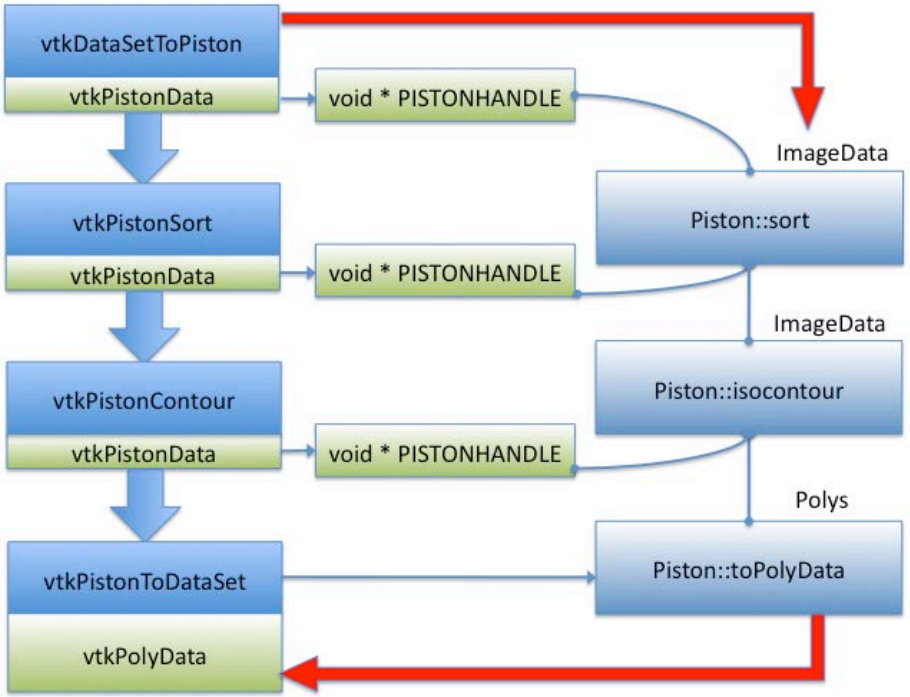
Relationship among the physics, data model, Thrust API, Thrust backends, and hardware.

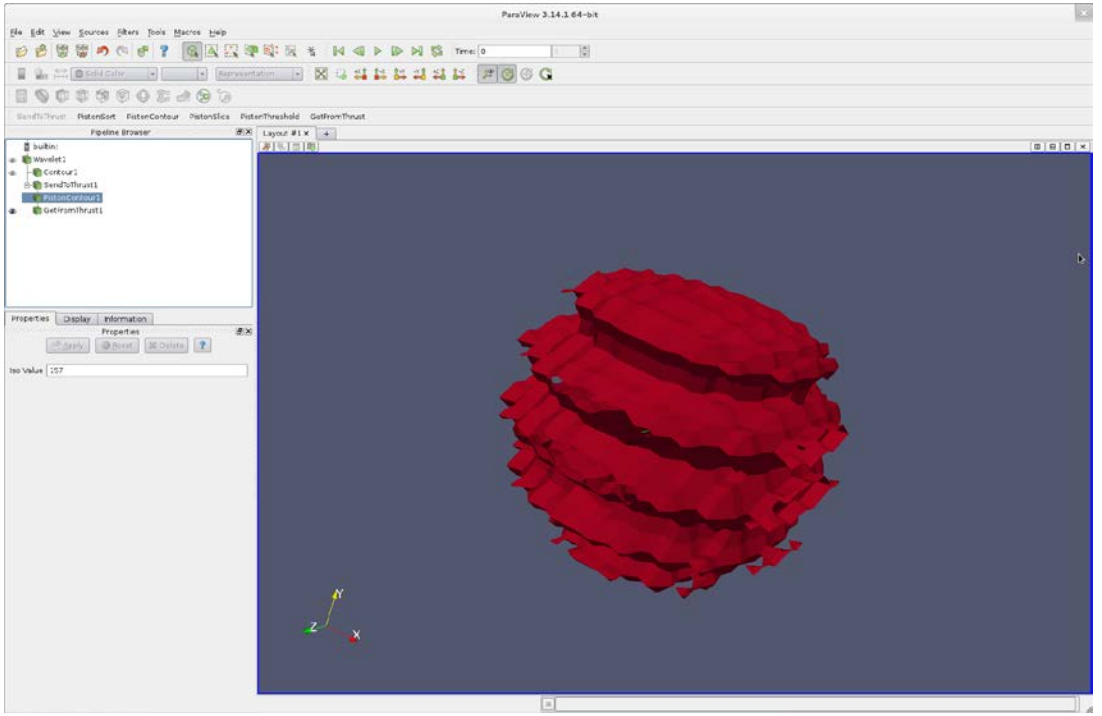
How PISTON/PINION Leverage Thrust

- Thrust provides:
  - An STL-like interface for memory management (host/device vectors) and data-parallel algorithms
  - Backend implementations of the data-parallel algorithms for CUDA, as well as slightly less-developed implementations for OpenMP and TBB
- PISTON/PINION intend to provide:
  - A library of visualization and analysis operators implemented using Thrust
  - A data model for simulation meshes (e.g., VTK structured grids, unstructured grids, AMR)
  - Simulation operators (e.g., advection, interface reconstruction, etc.)
- PISTON/PINION intend to enhance:
  - Non-CUDA backends (e.g., OpenCL prototype, optimize OpenMP for Xeon Phi, etc.)
  - Interface to support distributed memory operations

Integration with VTK and ParaView

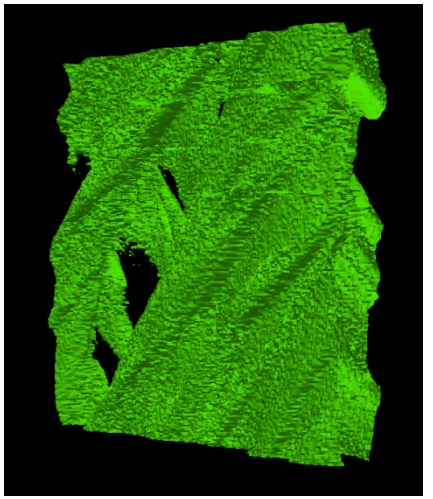
- Filters that use PISTON data types and algorithms integrated into VTK and ParaView
- Utility filters interconvert between standard VTK data format and PISTON data format (thrust device vectors)
- Supports interop for on-card rendering



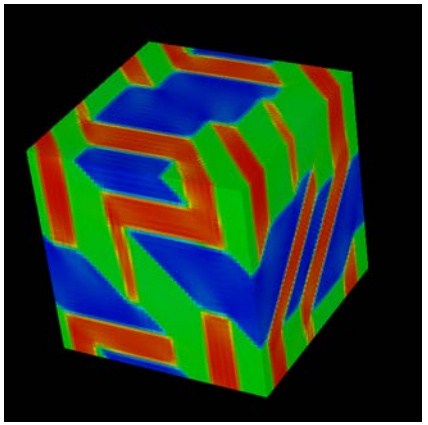


PISTON In-Situ

- VPIC (Vector Particle in Cell) Kinetic Plasma Simulation Code
  - Implemented an in-situ adapter based on Paraview CoProcessing Library (Catalyst)
  - PISTON contour pipeline using ParaView's PISTON integration
- 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 loading
  - Simulation code and in-situ viz implemented using PISTON



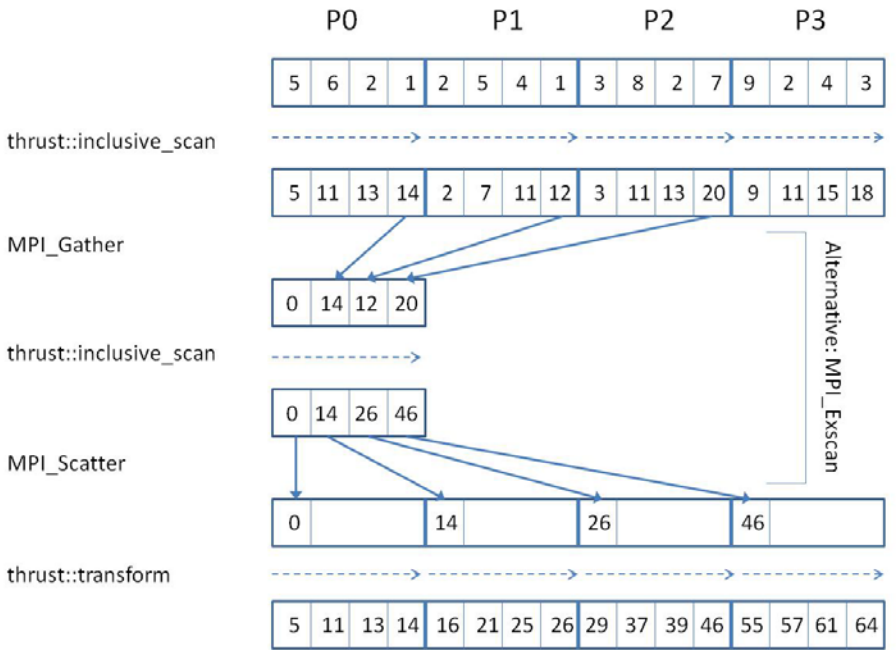
Output of PISTON contour filter on Hhydro charge density at one timestep of VPIC simulation



PISTON in-situ visualization of CoGLGinzburg-Landau simulation

Distributed Memory Parallelism

- Inter-node (distributed memory) parallelism
  - VTK Integration handles domain decomposition / image compositing
  - Distributed implementations of Thrust primitives using MPI
    - User can treat data as single vectors even though values are distributed across nodes
    - Regular Thrust primitives are called for on-node work, so it takes advantage of parallelism both on nodes and across nodes
    - Implemented isosurface and KD-tree construction algorithms using distributed PISTON



Distributed Scan Algorithm

Isosurface of 3600x2400x42 ocean temperature data computed on 4 GPUs

New Data-parallel Algorithms Accelerate Cosmology Data Analysis on GPUs

Objectives

Milestone

- Implement application-specific visualization and/or analysis operators needed for in-situ use by LCF science codes
- Use PISTON to take advantage of multi-core and many-core technologies

Target Application

- 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

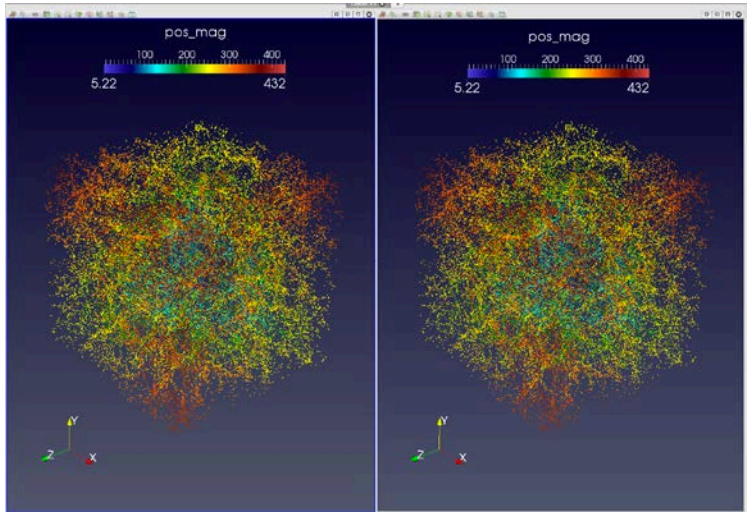
Impact

VTK-m framework

- The PISTON component of VTK-m develops data-parallel algorithms that are portable across many-core architectures for use by LCF codes
- PISTON consists of a library of visualization and analysis algorithms implemented using Thrust, and our extensions to Thrust

Halo and Center Finders

- Data-parallel algorithms for halo and center finding implemented using VTK-m (PISTON) allow the code to take advantage of parallelism on accelerators such as GPUs
- Can be used for post-processing or in-situ, with in-situ integration directly into HACC or via the CosmoTools library



Visual comparison of halos computed by the original HACC algorithms (left) and the PISTON algorithms (right). The results are equivalent, but are computed much more quickly on the GPU using PISTON.

Accomplishments

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
- **Portability of PISTON** allowed us to also run our algorithms on an Intel Xeon Phi
- 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

- Submission: "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

U.S. DEPARTMENT OF ENERGY | Office of Science

LA-UR-14-28771

SDAV Scalable Data Management, Analysis, and Visualization

LANL Milestones

- **Emerging Processor Technology**
  - Initial VTK-m Design, Year 1, Sandia, Kitware, ORNL, LANL: Provide the research and design for VTK-m functional operation and, in conjunction with SDAV, develop an initial implementation.
  - Hybrid Parallel, Year 2, LANL: Compare alternative models for the interaction of shared-memory and distributed-memory parallelism within VTK-m.
  - Additional Algorithms, Year 3, LANL: Develop algorithms for additional visualization and analysis filters in order to expand the functionality of the VTK-m toolkit to support less critical but commonly used operators.
- ***In Situ* Integration**
  - Memory Hierarchy Streaming, Year 2, LANL: Develop streaming out-of-core versions of key visualizations and analysis algorithms to efficiently use deep memory hierarchies with i*n situ* applications.
  - Fault Tolerant Primitive Functions, Year 3, LANL: Develop fault tolerant versions of key data-parallel primitives (such as scan, transform, reduce, etc.) used by VTK-m algorithms in order to transparently provide a level of resiliency within *in situ* applications.