SCALABLE DATA MANAGEMENT, ANALYSIS, AND VISUALIZATION INSTITUTE

Visualization Area

SDAV Institute Mid-Term Review
May 5-6, 2014

Context

SDAV Goals (from the proposal):

- Work directly with science teams to help them achieve breakthrough science.
- Provide technology in sdm, analysis, vis, that are broadly used within the computational science community.

Context, ctd.

How achieve goals? (from proposal)

- Actively engage science users running on big machines
- Work closely with science teams to help them integrate our technologies into their s/w ecosystem
- Incorporate ASCR research results into our portfolio
- Follow best practices in s/w engr, distribution, bug tracking, etc.

Context, ctd.

Mid-term review (May 5-6, 2014)

Focus on:

- Meeting milestones?
- Awareness: application, architecture, institute?
- Science impact?

Context, ctd. SDAV Participant Institutions

James Ahrens/LANL
E. Wes Bethel/LBNL
Eric Brugger/LLNL
Scott Klasky/ORNL
Ken Moreland/SNL-NM
Robert Ross/ANL

Alok Choudhary/NWU
Kwan-Liu Ma/UC Davis
Manish Parashar/Rutgers
Valerio Pascucci, Utah
Nagiza Samatova/NCSU
Karsten Schwan/Georgia Tech
Han-Wei Shen/Ohio State

Berk Geveci/Kitware

Context, ctd: This Presentation

For Visualization Area only

No context, background, etc (happens in earlier talks)

15 minutes + 5 for questions

Not much technical depth

Is a draft, your comments welcome/appreciated

Visualization Area Focus/Objectives

Provide production-quality visual data analysis and exploration software infrastructure for use by DOE science community on DOE SC platforms over 5-year horizon.

- Architectural challenges: increasing core/proc count, shrinking relative I/O capacity.

Respond to scientific knowledge discovery needs from science projects.

Visualization Area: Starting point

Technologies:

- Visualization applications: VisIt, ParaView
- Library: VTK (the foundation for VisIt, PV)
- Other sources: research programs
  - Data parallel visualization: Dax, EAVL, PISTON
  - Flow visualization/analysis
  - Rendering

Visualization Area Milestones

- VisIt and ParaView: ongoing SWE, deployment, support, evolve to new platforms, delivery vehicles for new technologies.
- VTK-m: our approach for realizing m-core across many technologies.
- Flow visualization: productize technology for enabling knowledge discovery.
- Rendering: *Ibid.*
- Ensembles, Uncertainty, Higher-dimensional methods: *Ibid.*

Also:

- Science team interactions/support

VisIt and ParaView Milestones

Visit and ParaView (Kitware, LANL, LBNL, LLNL, ORNL, SNL)

- Y1: Enhance Visit and Paraview to leverage multiple cores within a single MPI task
- Y2: Integrate VisIt and ParaView with ADIOS
- Y3: Demonstrate and evaluate *in situ* analysis methods with VisIt and ParaView

VisIt, ParaView and Multi-core

Objective:

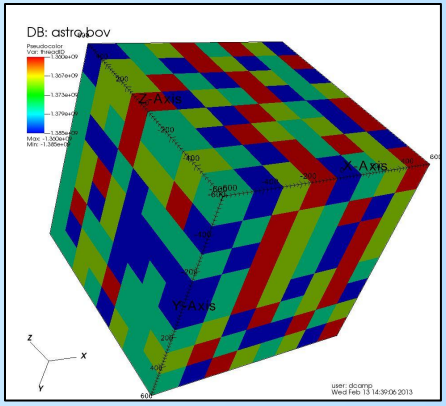
- Want to be able to take advantage of multi-core platforms, don't want to run MPI task-per core.

Approach:

- Focus on key infrastructure in VTK for coarse-grained parallelism.
- General-purpose threading interface to abstract back-end threads library (pthreads, TBB, etc.)
- Two coordinated teams: VisIt/VTK (LBNL/LLNL/Kitware), ParaView/VTK (LANL/SNL/Kitware)

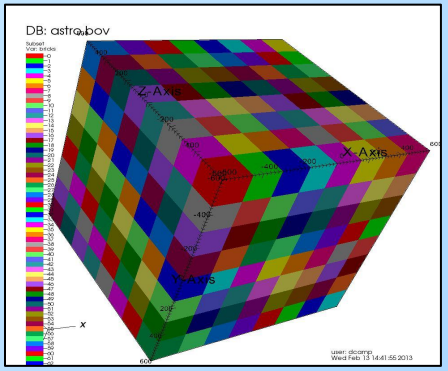
Accomplishments:

- 2012: Design/approach for key infrastructure.
- (Y1 milestone) Summer, Fall 2013: early releases of threaded VTK (2.6), VisIt (6.0) and PV (what version?)
- (Y2) VTK 6.1: vtkSMP class, Dax & Piston adapters
- (Y3) Summer 2014: VTK 2.8, VisIt 6.1, PV (what version?)
- Most embarrassingly parallel operators supported
- Others, not yet: streamlines, rendering.
- Runs faster, uses less memory.
- Broad community impact.



Top: 512-block astrophysics dataset colored by thread ID.

Bottom: serially processed 512-block dataset colored by data block ID.



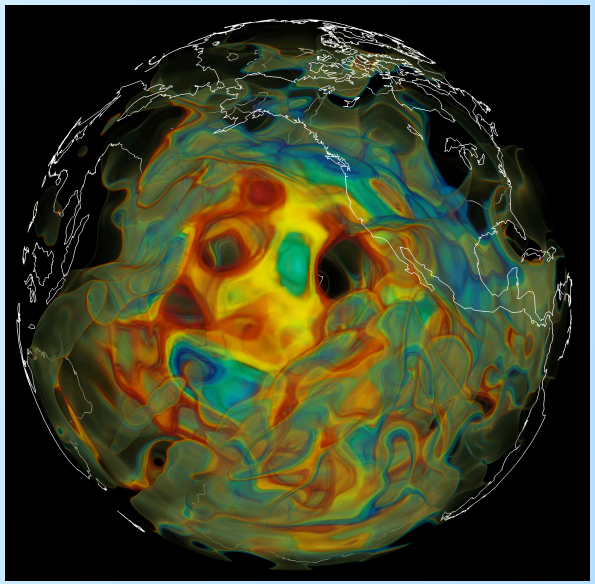
VisIt, ParaView, and ADIOS Integration

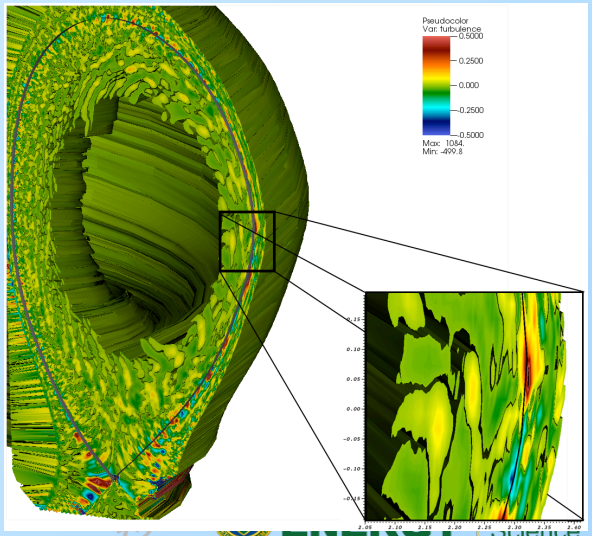
Problems: increasingly intractable to do full-resolution I/O, science being lost due to analysis of partial results.

Approach: integration, interoperation of SDAV technologies for *in situ* processing and vis/analysis; work with code teams to put this technology into practice.

Results:

- VisIt, PV ADIOS loaders.
- Code team interactions and deployment: SPECFM3D (comp. seismology, INCITE); XGC (fusion, SciDAC), others.





VisIt, ParaView and *In Situ*

Objective: provide *in situ* production quality vis, analysis capabilities to computational science community.

Approach: focus on app-facing infrastructure from VisIt, ParaView, help science teams make use of this infrastructure.

Accomplishments:

- Release of ParaView/Catalyst library: open source library for coupling codes to VTK-, PV-based vis/analysis.
- VisIt Libsim engineering to optimize memory footprint, etc.
- Custom PV+PISTON adaptor to support interactions between science teams and *in situ* tools.
- Worked with multiple applications/teams: VPIC (plasma physics), HACC (astro), Warp3D (accelerator), H3D (plasma physics), POP (climate)

VTK-m Milestones

VTK-m framework (ANL, Kitware, LANL, LBNL, LLNL, ORNL, SNL)

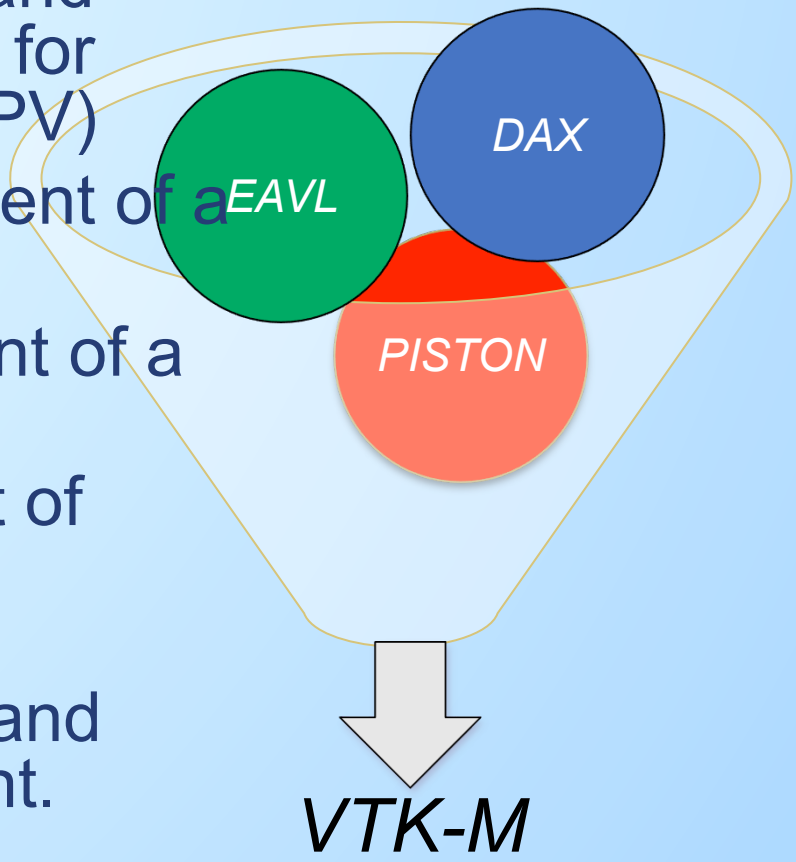
- Y1: Enhancements to existing multi/many-core technologies in anticipation of *in situ* analysis use cases with LCF codes
- Y2: Deployment and evaluation of existing technologies in prototype form.
- Y3: Initial implementation of VTK-m with an integration of existing packages

VTK-m Framework

- Objective: a single, community-based effort for achieving multi-/many-core vis/analysis (VTK-m), both *in situ* and *post hoc.*
- Benefit to scientists: These frameworks will make it easier for domain scientists' simulation codes to take advantage of the parallelism available on a wide range of current and next-generation hardware architectures, especially with regards to visualization and analysis tasks
- Approach: build on strengths of existing research, and deep body of experience; apply each where it makes sense; draw best of each into a single VTK-m "product."
- Projects
  - DAX, Sandia National Laboratory
  - DIY, Argonne National Laboratory
  - EAVL, Oak Ridge National Laboratory
  - PISTON, Los Alamos National Laboratory

VTK-m Constituent Technologies

- VTK is (was) a serial, single-threaded class library (data structures and algorithms) used as the basis for important applications (VisIt, PV)
- EAVL emphasizes the development of a new data model,
- Dax emphasizes the development of a new execution model,
- DIY provides a lightweight toolkit of commonly-used DM parallel functionality,
- PISTON emphasizes portability and parallel algorithm development.



Enhancements for LCF Codes

PISTON

- Integrate with VTK, PV for use within Catalyst pipelines
- PISTON/PV plug-in for moving data between PV & GPU
- Prototype use of vis/analysis operators from LCF codes (HACC, VPIC)

DAX

- Interface improvements: easier to code, and integrate with other s/w.
- Optimize geometry creation methods
- Accessible from VTK/PV pipelines

Enhancements for LCF codes, ctd.

EAVL

- Added support for zero-copy access
- Release engineering: eliminate 3rd party library dependencies, reduce binary size, speed compilation
- Optimizations: large mesh handling, better multi-core performance, more topology operations
- Prototype implementation with multiple codes: LULESH, two fusion codes (XolotI, PSI; XGC, ESPI).

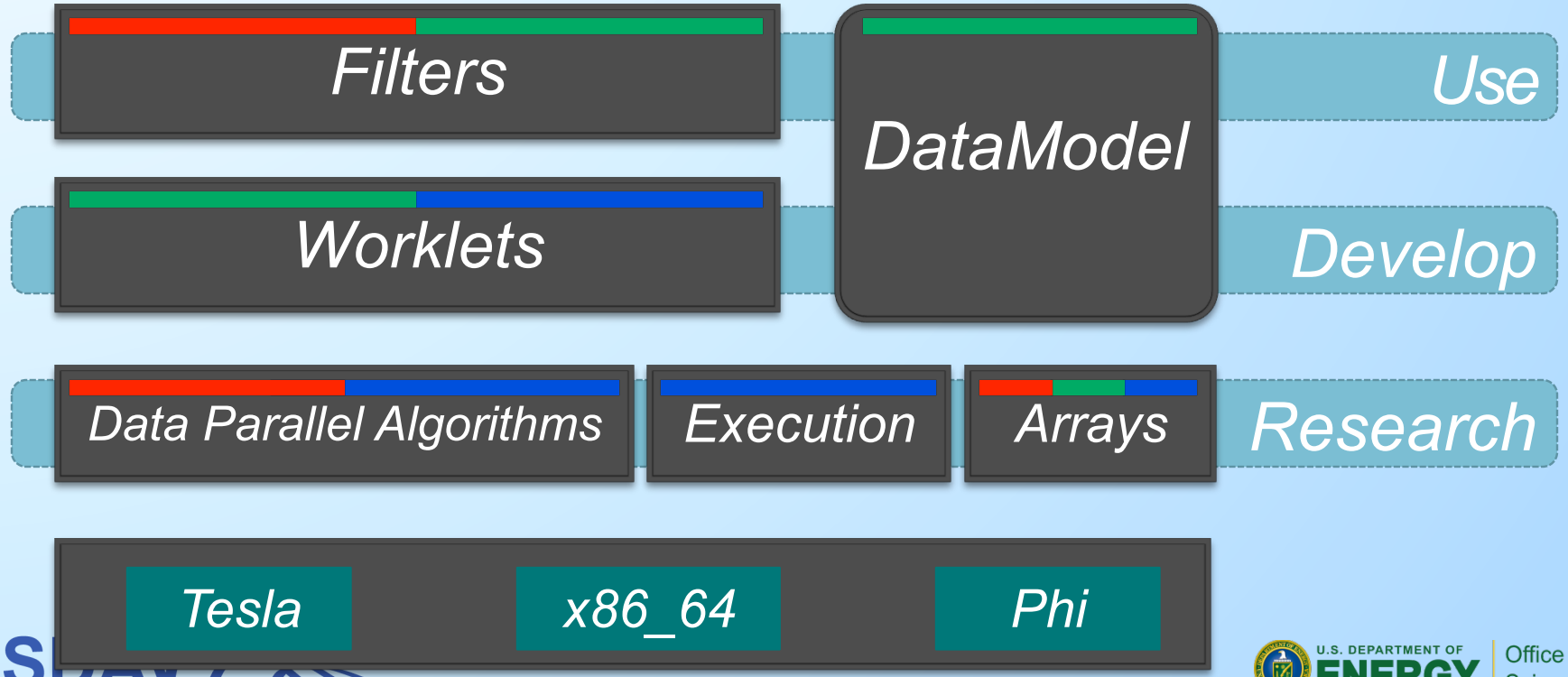
DIY

- Optimizations: load balancing, work-stealing, multiple domain decompositions.
- Prototype implementation with internal and external projects: flow analysis (OSUFlow), Voronoi tesselation (with HACC), topology.

VTK-m Architecture

Recent work:

- Design, code scaffolding, prelim key data parallel algorithms.
- vtkSMP class: early support for coarse- and fine-grained parallelism (VTK v6.1).



Flow Analysis

Problems: science needs the means to understand complex phenomena in flow fields; parallelizing flow analysis codes is difficult.

Approach: study methods for scaling, optimizing flow analysis/vis codes on multiple architectures; deliver new capabilities in production s/w to science community.

Results:

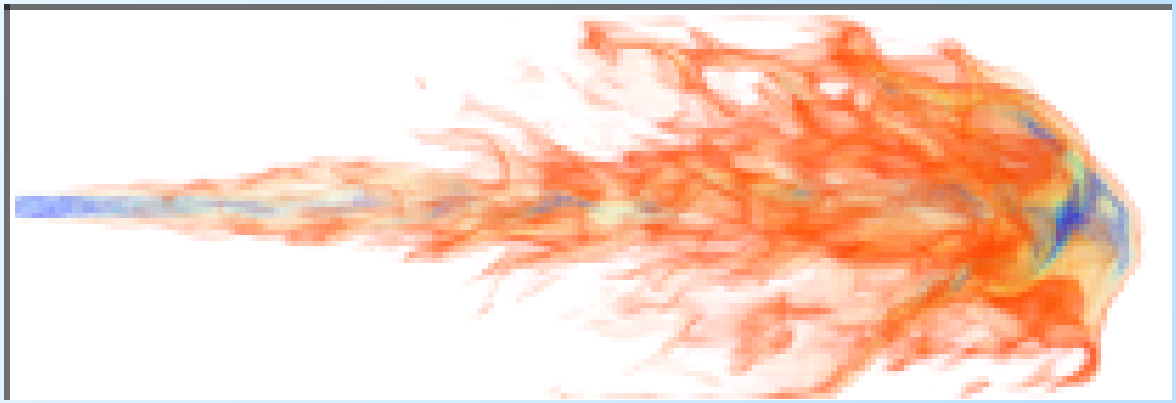
- Optimization of particle advection infrastructure (VisIt)
- Productization of parallel flow visualization code for use in VTK/PV (OSUflow)
- Flow visualization on GPUs

Rendering

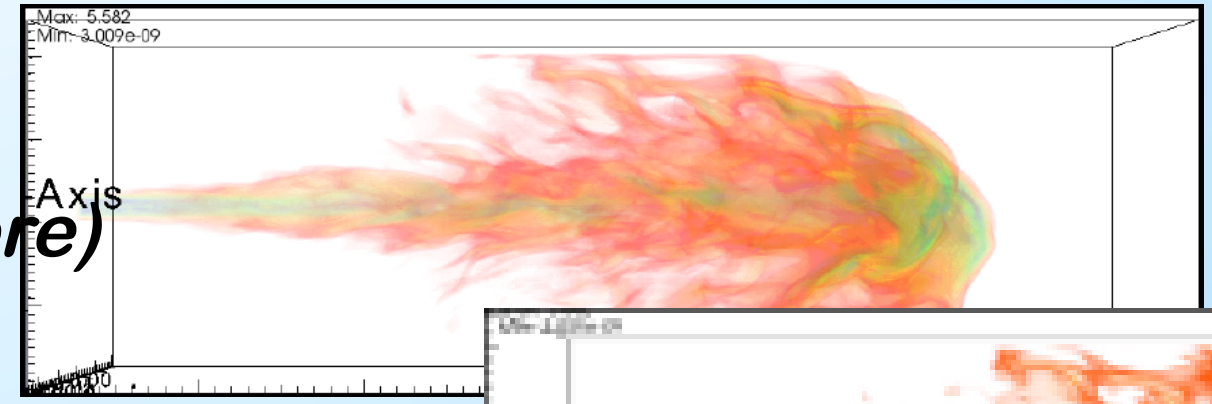
- Advanced Volume Rendering in VisIt
  - Improvements in GLSL-based rendering infrastructure incorporated into VisIt source tree for production release.

Advanced Volume Rendering in VisIt

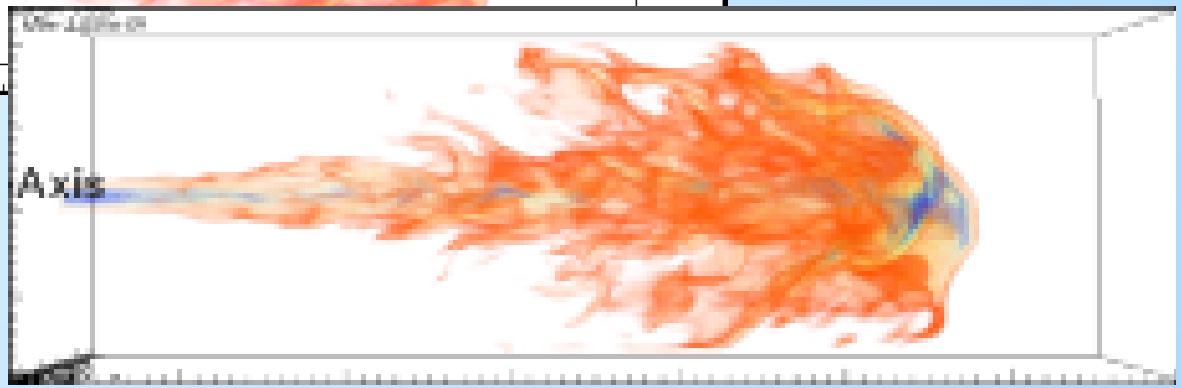
SCIRun:



VisIt (before)



VisIt (after)



Ensembles

Ensemble-vis: port to VisIt, PV

(Wes 4/25/2014: probably not going to include this slide in the deck.)

Application Projects Sampler

PISTON+HACC Halo Finder

(still deciding on which, if any, others to show at the review)

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

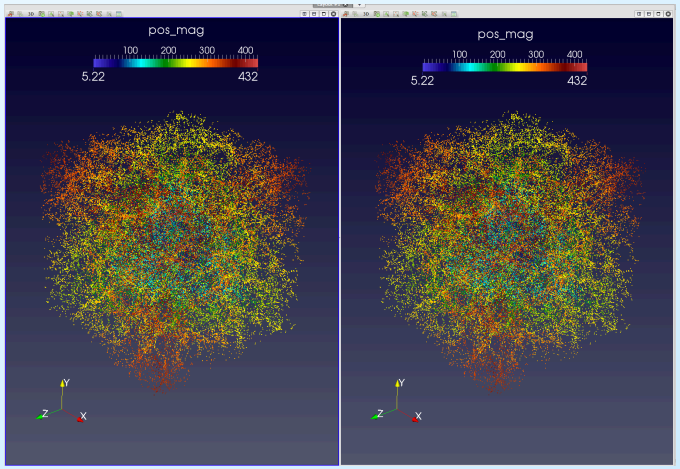
Impact

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

- *Submitted to SC14: "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*

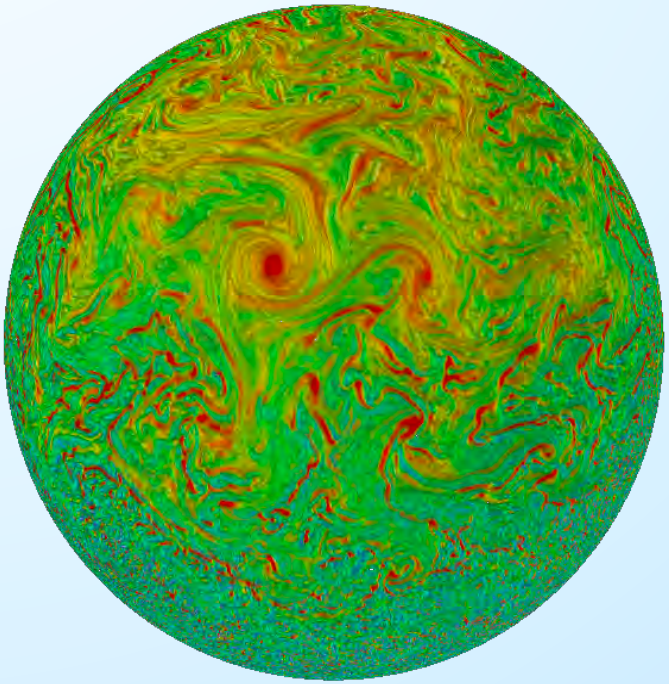
Visualization Area Software

Table showing releases/dates/capabilities of VisIt, PV, Catalyst, VTK (from DeMarle's DOECGF slides, if I ever get them ☺)

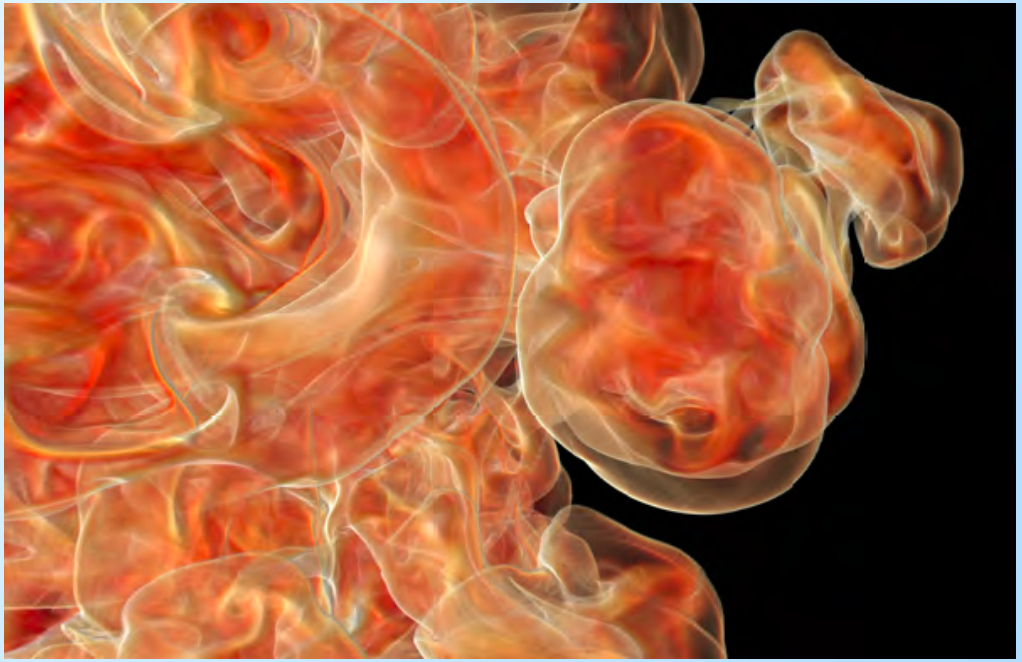
The End

Enhancements to existing multi/many-core technologies in anticipation of in situ analysis use cases with LCF codes – VTK-m Framework

Unstructured, and AMR Volume Rendering



Geodesic grid from GCRM



Rayleigh-Taylor turbulence modeling

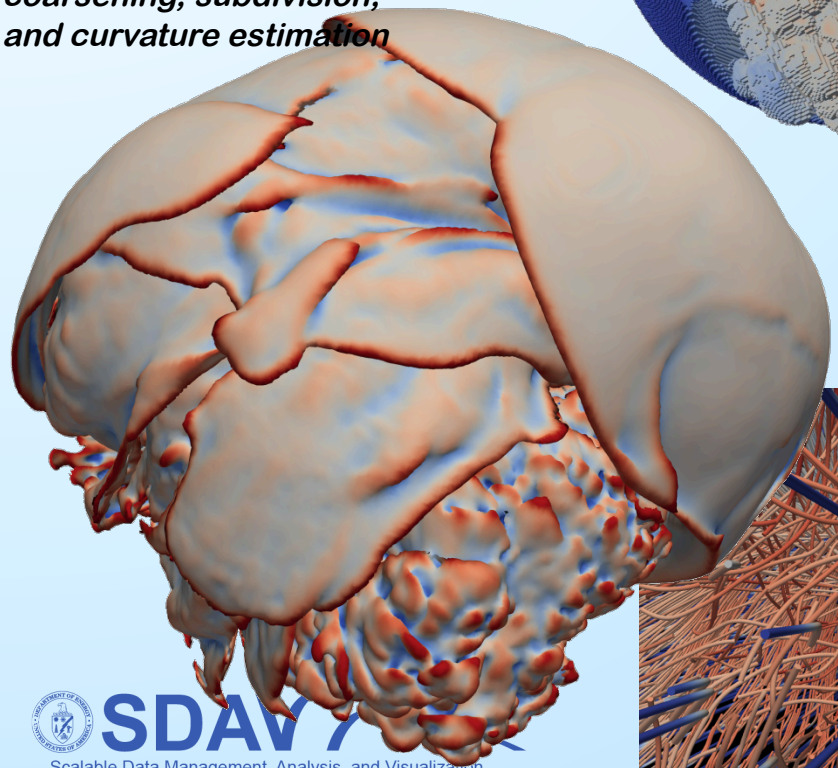
ax: A Toolkit for Analysis and Visualization at Extreme Sca

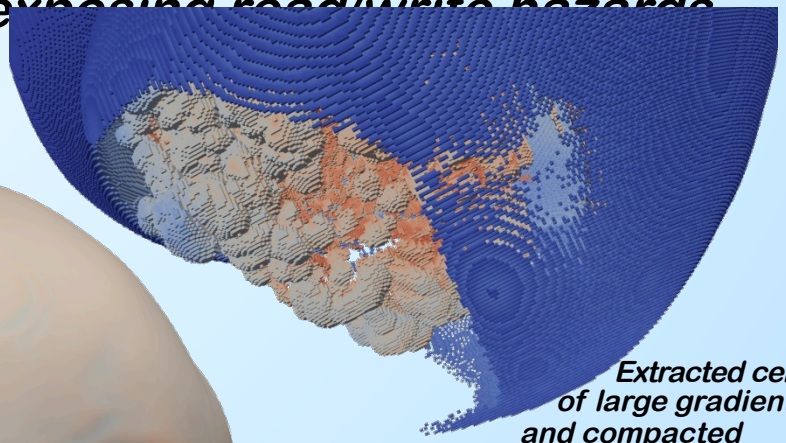
The primitives necessary to design finely-threaded algorithm

- ***"Worklets" ease design in serial, scheduled in parallel***
- ***Basic visualization design objects (think VTK for many-core)***
- ***Communicative operations provide neighborhood-wide operations without e***

http://daxtoolkit.or

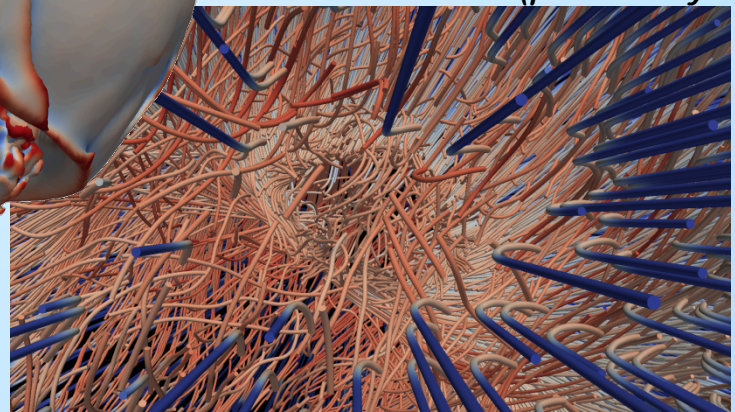
Contour with subsequent vertex welding, coarsening, subdivision, and curvature estimation

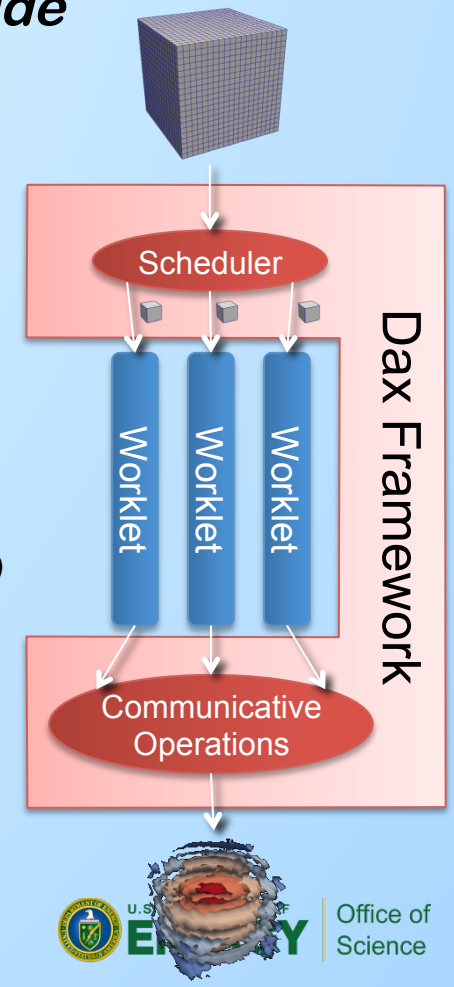




Extracted cells of large gradient and compacted points

Streamlines (preliminary work)





DIY (Do-It-Yourself): Overview

Main Ideas and Objectives

-Large-scale parallel analysis (visual and numerical) on HPC machines

-For scientists, visualization researchers, tool builders

-In situ, coprocessing, postprocessing

-Data-parallel problem decomposition

-MPI + threads hybrid parallelism

-Scalable data movement algorithms

-Runs on Unix-like platforms, from laptop to supercomputer (including all

ma

Features

-Parallel I/O to/from storage

-Domain decomposition

-Network communication

-Written in C++

-C bindings, can be called from Fortran, C, C++

-Autoconf build system

-Lightweight: libdiy.a 800KB

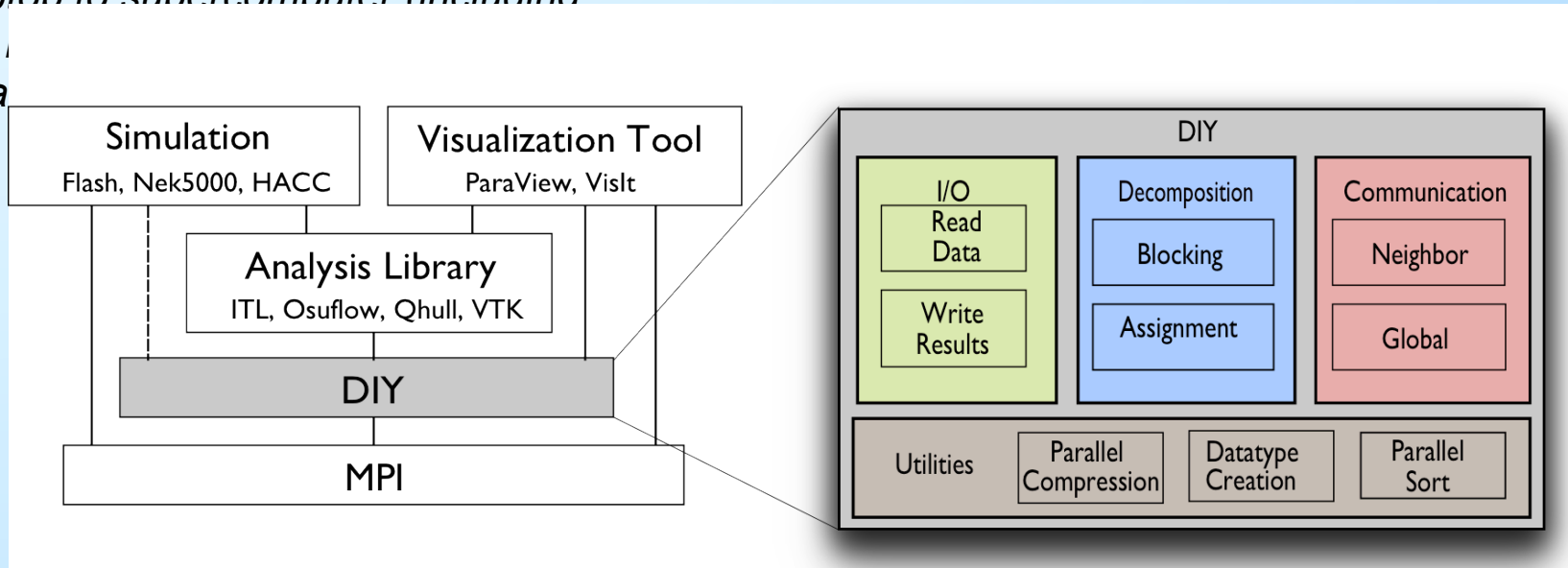
-Maintainable: ~15K lines of code

Benefits

-Researchers can focus on their own work, not on parallel infrastructure

-Analysis applications can be custom

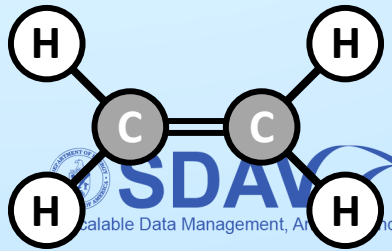
-Reuse core components and algorithms for performance and productivity

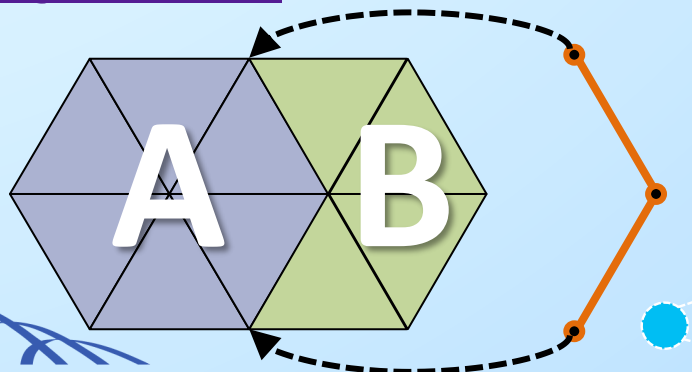


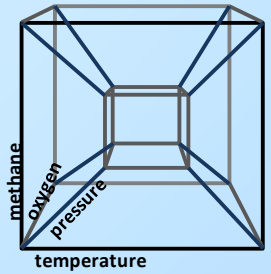
EAVL: Extreme-scale Analysis and Visualization Library

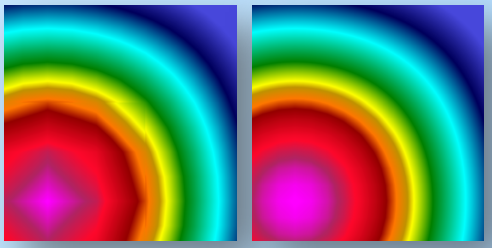
- Targets approaching hardware/software ecosystem:
- Update traditional data model to handle modern simulation codes and a wider range of data.
- Investigate how an updated data and execution model can achieve the necessary computational, I/O, and memory efficiency.
- Explore methods for visualization algorithm *developers* to achieve these efficiency gains and better support exascale architectures.

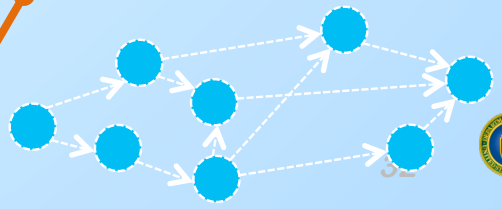
http://ft.ornl.gov/eavl

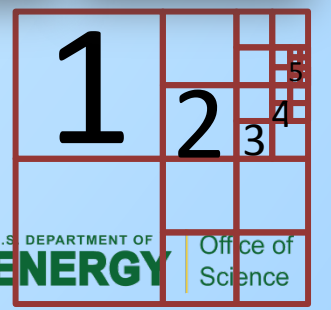












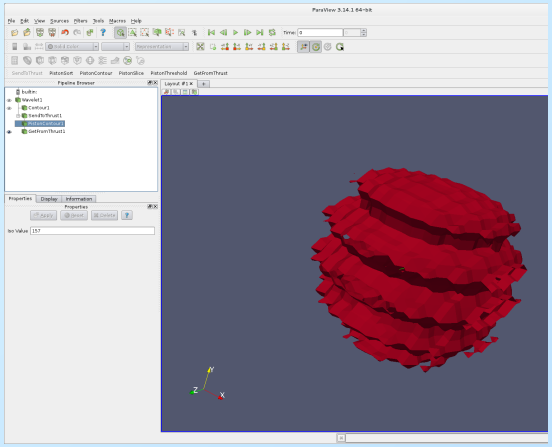
SDAV
Scalable Data Management, Analysis and Visualization

U.S. DEPARTMENT OF ENERGY | Office of Science

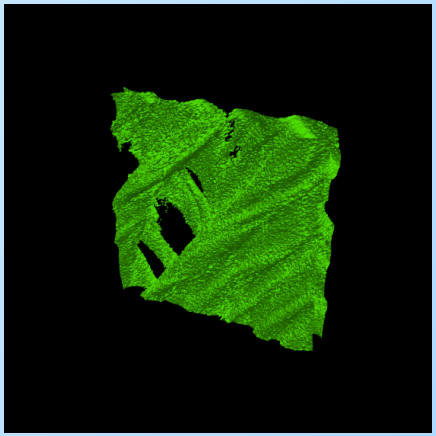
PISTON: A Portable Cross-Platform Framework for Data-Parallel Visualization Operators

- Goal: Portability and performance for visualization and analysis operators on current and next-generation parallel architectures
- Main idea: Write operators using only data-parallel primitives (scan, reduce, etc.)
- Requires architecture-specific optimizations for only for the small set of primitives
- PISTON is built on top of NVIDIA's Thrust library
- We have run visualization algorithms on GPUs and on multi-core CPUs using the exact same operator code by compiling to CUDA and to OpenMP backends
- PISTON has been integrated with VTK and included as a plug-in with ParaView
- Our in-situ adapter for VPIC (Vector Particle in Cell), an LCF kinetic plasma simulation code, makes use of PISTON via the ParaView Co-Processing Library (Catalyst)

Isosurface of ocean temperature data set on a



Isosurface generated using ParaView PISTON



Contour produced using PISTON in-situ

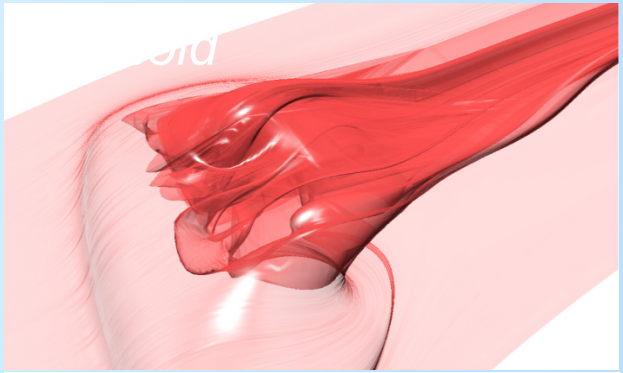
Parallel Stream Surfaces

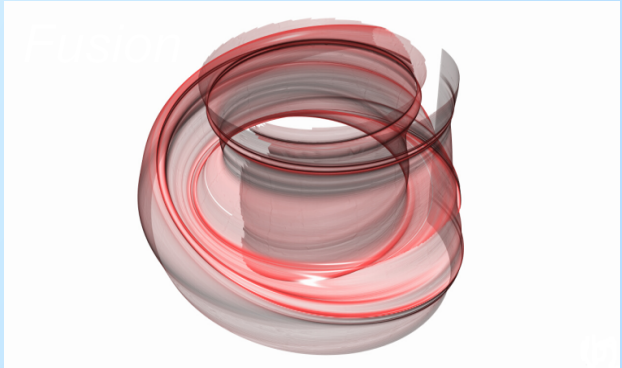
UCD Milestones
- vector field analysis and visualization
- scalable flow visualization methods

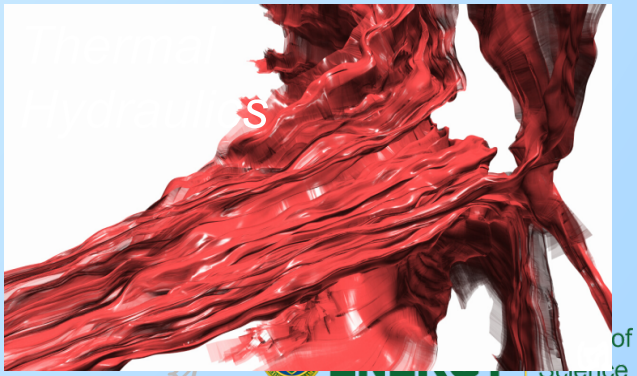
Results
- *Parallel Stream Surface Algorithm Development*
- *Integration with VisIt*

Partners
- *Lawrence Berkeley National Lab*
- *Visit users*

Future
- *Development of new integral curve methods for parallel flow visualization*

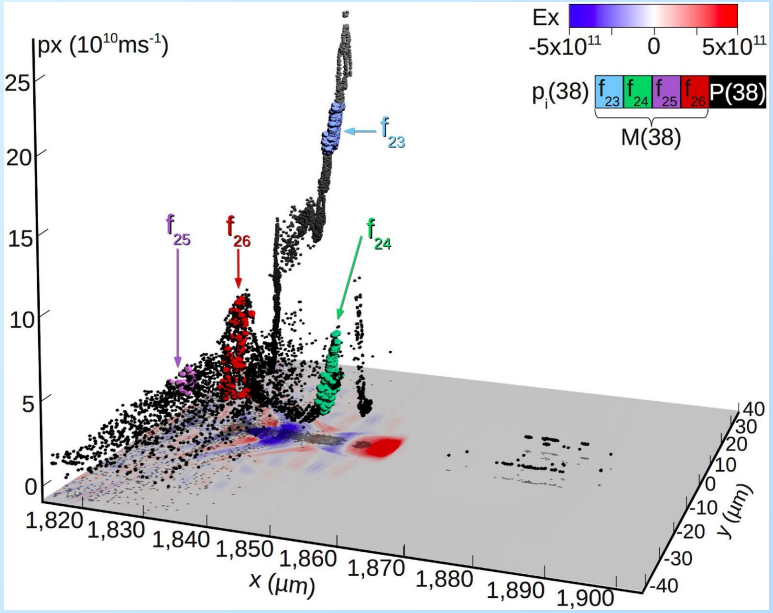






Application Support – Accelerator Modeling

- Particle Path Analysis
  - New methods for finding, tracking, and analyzing particles accelerated by the plasma wave generated by a laser pulse.
  - *Next steps: Apply the feature detection to perform analysis of large collections of simulations to enable efficient comparative analysis and parameter studies*
  - *Stakeholders: C.G.R. Geddes (LBNL/ComPASS)*
  - *Recent Publications:*



[1] Oliver Rübel, Cameron, G. R. Geddes, Min Chen, Estelle Cormier-Michel, and E. Wes Bethel, "Query-driven Analysis of Plasma-based Particel Acceleration Data," Poster Abstracts of IEEE VisWeek 2012, Seattle, WA, Oct 14-19, 2012 (Poster). LBNL-5906E.

[2] Oliver Rübel, Cameron, G. R. Geddes, Min Chen, Estelle Cormier-Michel, and E. Wes Bethel, "Feature-based Analysis of Plasma-based Particle Acceleration Data", IEEE TVCG, 2013.

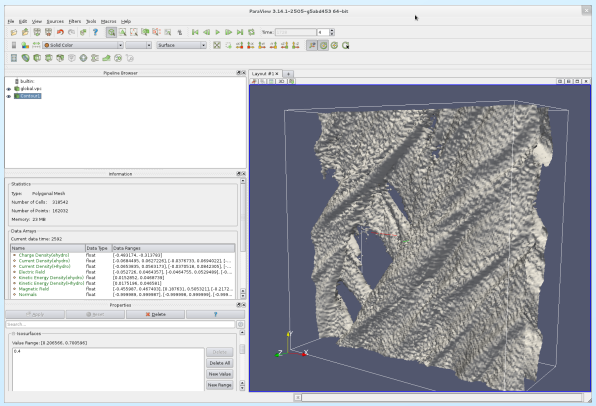
VPIC In Situ + PISTON

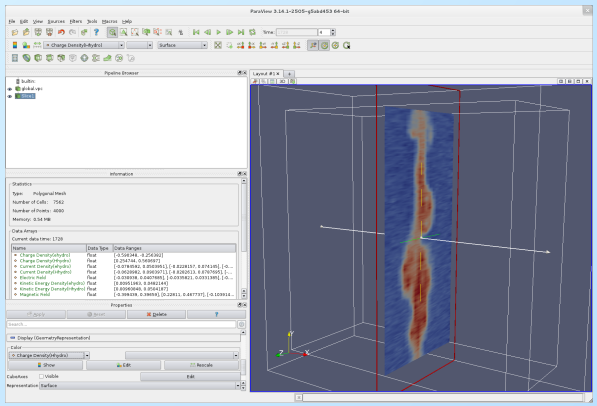
ParaView-Catalyst Enabling high temporal Resolution Visualization/Analysis

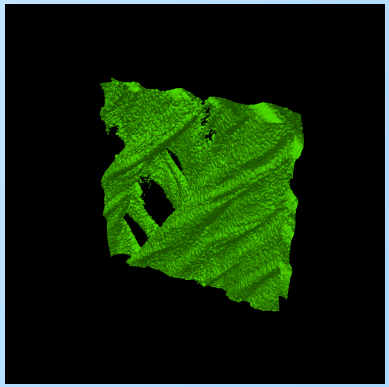
- "Hard Coded" operators
  - Surface line integral convolution, contour, slice
- PISTON Contour Operator

Surface line integral convolution

2D slices produced in-situ with VPIC







Contours produced using ParaView in post-processing

2D slides produced using ParaView in post-processing

Contours made using PISTON in-situ