LA-UR-13-26599
Approved for public release; distribution is unlimited.

Title: LANL CSSE L2: Case Study of In Situ Data Analysis in ASC Integrated Codes

Author(s):
Patchett, John M.
Ahrens, James P.
Nouanesengsy, Boonthanome
Fasel, Patricia K.
Oleary, Patrick W.
Sewell, Christopher Meyer
Woodring, Jonathan L.
Mitchell, Christopher J.
Lo, Li-Ta
Myers, Kary L.
Wendelberger, Joanne R.
Canada, Curtis V.
Daniels, Marcus G.
Abhold, Hilary M.
Rockefeller, Gabriel M.

Intended for: ASC LANL CSSE L2 Review, 8/13/2013

Issued: 2013-08-21

Abstract

This talk presents the overview of the 2013 CSSE ASC L2 Milestone: Case Study of In Situ Data Analysis in ASC Integrated Codes. The talk has 3 parts: An introduction to *in situ* analysis/visualization, a demonstration of paraview catalyst applied to xRage with timings, and a detail of the minor deliverables from the description of the milestone.

LANL CSSE L2

Case Study of *In Situ* Data Analysis in ASC Integrated Codes

Contributors: James Ahrens, Boonthanome Nouanesengsy, Patricia Fasel, Patrick O'leary, Chris Sewell, Jon Woodring, Christopher Mitchell, Ollie Lo, Kary Myers, Joanne Wendelberger, Curt Canada, Marcus Daniels, Hilary Abhold, Gabe Rockefeller.

John Patchett
patchett@lanl.gov
505 665 1110

The L2 Milestone

<table>
<tr><td colspan="3">Milestone (ID#): Case Study of In Situ Data Analysis in ASC Integrated Codes</td></tr>
<tr><td><b>Level</b>: 2</td><td><b>Fiscal Year</b>: FY13</td><td><b>DOE Area/Campaign</b>: ASC</td></tr>
<tr><td colspan="3"><b>Completion Date</b>: 9/30/13</td></tr>
<tr><td colspan="3"><b>ASC nWBS Subprogram</b>: CSSE</td></tr>
<tr><td colspan="3"><b>Participating Sites:</b> LANL</td></tr>
<tr><td colspan="3"><b>Participating Programs/Campaigns</b>: ASC</td></tr>
<tr><td colspan="3"><b>Description</b>: The massive, complex datasets generated by scientific simulations on next-generation computer architectures at extreme scale pose new challenges for data management and scientific understanding. A critical element in meeting these new challenges is in situ data analysis where information is reduced and analyzed as close to the point of generation as possible, for example, at run time or as the data stream to longer term storage.
This milestone will demonstrate in situ data analysis in xRAGE, an ASC Eulerian Radiation Hydrodynamics code. We will work with the Eulerian Applications Project to identify target problems that will exercise physics of interest to the program, and we will work closely with users to maximize the achieved usability and productivity improvements.
We will investigate enhancements to the in situ process that enable and improve techniques of data reduction and data triage in addition to improving in situ performance by using portable acceleration technology. We will report on their viability and performance in xRAGE or another ASC code.
These products will provide ASC IC developers with reference codes and tools that will benefit DSW by advancing the technology for predictive science at extreme scales.</td></tr>
<tr><td colspan="3"><b>Completion Criteria:</b> Demonstration of our in-situ visualization and analysis capability applied to xRAGE and/or other ASC codes with performance impacts documented.</td></tr>
<tr><td colspan="3"><b>Customer:</b> ASC IC development teams and DSW</td></tr>
<tr><td colspan="3"><b>Milestone Certification Method:</b> A program review is conducted and its results are documented. Professional documentation, such as a report or a set of viewgraphs with a written summary, is prepared as a record of milestone completion.</td></tr>
<tr><td colspan="3"><b>Supporting Resources:</b> System resources and participation of code teams and code users.</td></tr>
</table>

Outline of Presentation

- **Introduction – motivation and background**
- **Completion Criteria**
  - Demonstration of in situ in xRage
  - Document Performance Impacts
- **Description Deliverables**
  - Eulerian Applications Project to identify problems of interest
  - Work with Users
  - Enable/improve data reduction/triage
  - Run on GPU/alternative architectures
  - Provide developers with reference codes and tools

Motivation

In Situ – What and Why

- **In Situ is the process of transforming data at run time**
  - Analysis
  - Visualization
  - Reduction
  - Triage
- **In situ has the promise of**
  - Saving disk space
  - Saving time in computing
  - Saving time in analysis
  - Producing higher fidelity results
  - Saving more information dense data

The In Situ Process and Metrics

The In Situ Process:

```
While the simulation should keep running
    Advance the simulation
    Decide what data products to produce
    Produce data products
        In memory grid adaptor
        Create geometry
        Render/Composite
        Write Results
```

- **Measure all stages for time and disk space**
- **Data Product Sizes and Times for production**
  - Restart Dumps, viz dumps, vis/analysis plots/imagery, etc.
- **Softer Metrics**
  - Useability, Workflow times, etc …

ParaView and ParaView Catalyst

CATALYST
scalable in-situ analysis

ParaView and VTK

- 66 developers last year

VTK Dev Team: From Ohloh:

This is one of the largest open-source teams in the world, and is in the top 2% of all project teams on Ohloh.

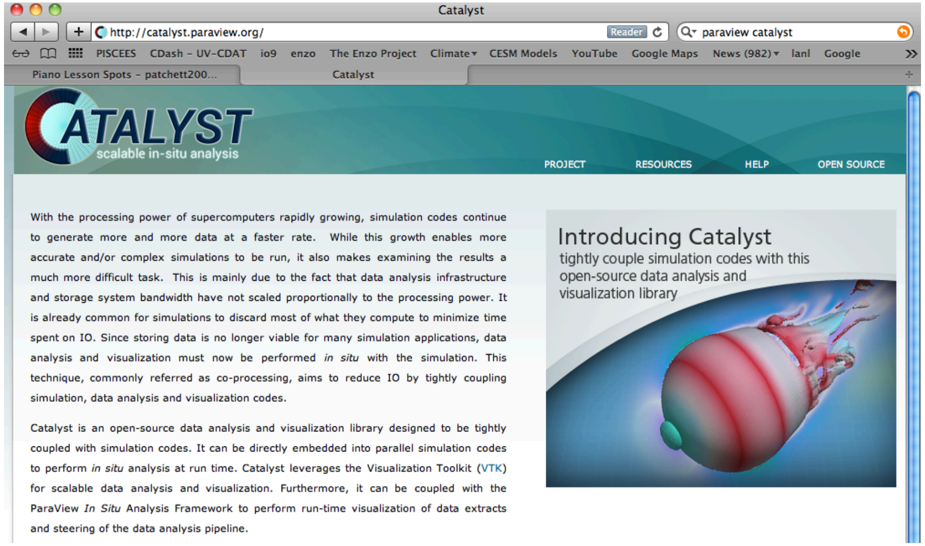
PITTSBURGH SUPERCOMPUTING CENTER
CSCS Swiss National Supercomputing Centre
Los Alamos NATIONAL LABORATORY EST. 1943
eDF
CD-adapco
ARL U.S. Army Research Laboratory
ge
TU Delft
Delft University of Technology
Kitware
THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL
Sandia National Laboratories
INDIANA UNIVERSITY
SCI www.sci.utah.edu
GEORGETOWN UNIVERSITY
cea
JOHNS HOPKINS UNIVERSITY
MIT Massachusetts Institute of Technology
Rensselaer

and many others...

ParaView Catalyst

Catalyst is an open-source data analysis and visualization library designed to be tightly coupled with simulation codes.

- http://catalyst.paraview.org/

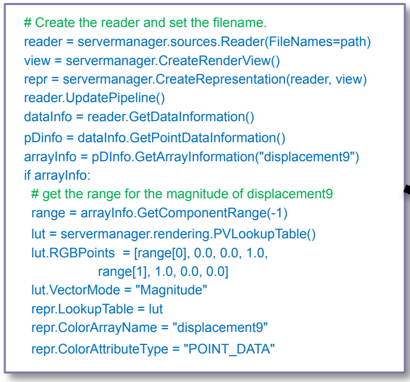


With the processing power of supercomputers rapidly growing, simulation codes continue to generate more and more data at a faster rate. While this growth enables more accurate and/or complex simulations to be run, it also makes examining the results a much more difficult task. This is mainly due to the fact that data analysis infrastructure and storage system bandwidth have not scaled proportionally to the processing power. It is already common for simulations to discard most of what they compute to minimize time spent on IO. Since storing data is no longer viable for many simulation applications, data analysis and visualization must now be performed *in situ* with the simulation. This technique, commonly referred as co-processing, aims to reduce IO by tightly coupling simulation, data analysis and visualization codes.

Catalyst is an open-source data analysis and visualization library designed to be tightly coupled with simulation codes. It can be directly embedded into parallel simulation codes to perform *in situ* analysis at run time. Catalyst leverages the Visualization Toolkit (VTK) for scalable data analysis and visualization. Furthermore, it can be coupled with the ParaView *In Situ* Analysis Framework to perform run-time visualization of data extracts and steering of the data analysis pipeline.

Introducing Catalyst
tightly couple simulation codes with this open-source data analysis and visualization library

ParaView In Situ (ParaView Catalyst)



```
# Create the reader and set the filename.
reader = servermanager.sources.Reader(FileNames=path)
view = servermanager.CreateRenderView()
repr = servermanager.CreateRepresentation(reader, view)
reader.UpdatePipeline()
dataInfo = reader.GetDataInformation()
pDinfo = dataInfo.GetPointDataInformation()
arrayInfo = pDInfo.GetArrayInformation("displacement9")
if arrayInfo:
  # get the range for the magnitude of displacement9
  range = arrayInfo.GetComponentRange(-1)
  lut = servermanager.rendering.PVLookupTable()
  lut.RGBPoints  = [range[0], 0.0, 0.0, 1.0,
                   range[1], 1.0, 0.0, 0.0]
  lut.VectorMode = "Magnitude"
  repr.LookupTable = lut
  repr.ColorArrayName = "displacement9"
  repr.ColorAttributeType = "POINT_DATA"
```

Python Script

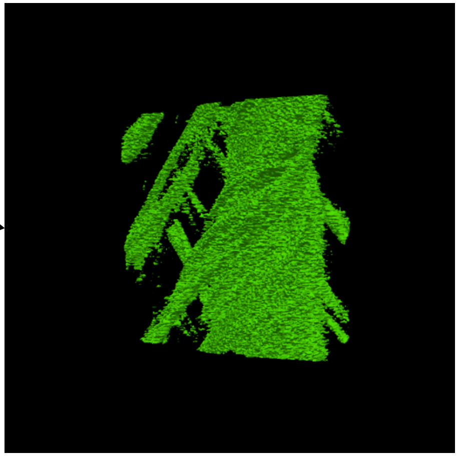
Simulation

xRage

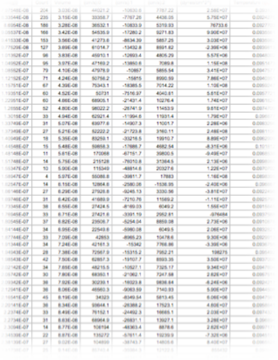
VPIC

Pagosa

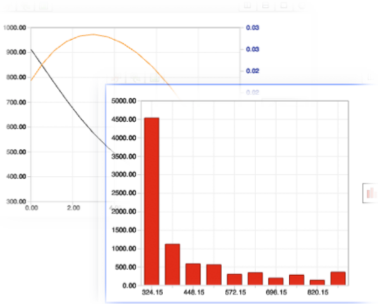
ParaView In Situ



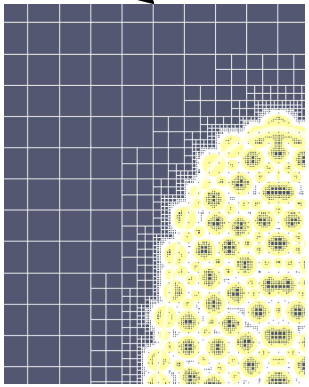
Rendered Images



Statistics

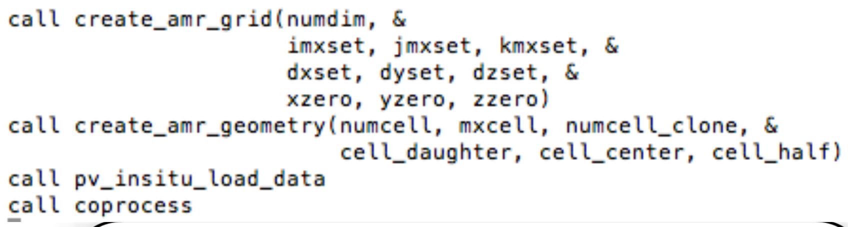


Series Data



Polygonal Output with Field Data

ParaView *In Situ* Checked into xRage



```
call create_amr_grid(numdim, &
                     imxset, jmxset, kmxset, &
                     dxset, dyset, dzset, &
                     xzero, yzero, zzero)
call create_amr_geometry(numcell, mxcell, numcell_clone, &
                         cell_daughter, cell_center, cell_half)
call pv_insitu_load_data
call coprocess
```

xRage

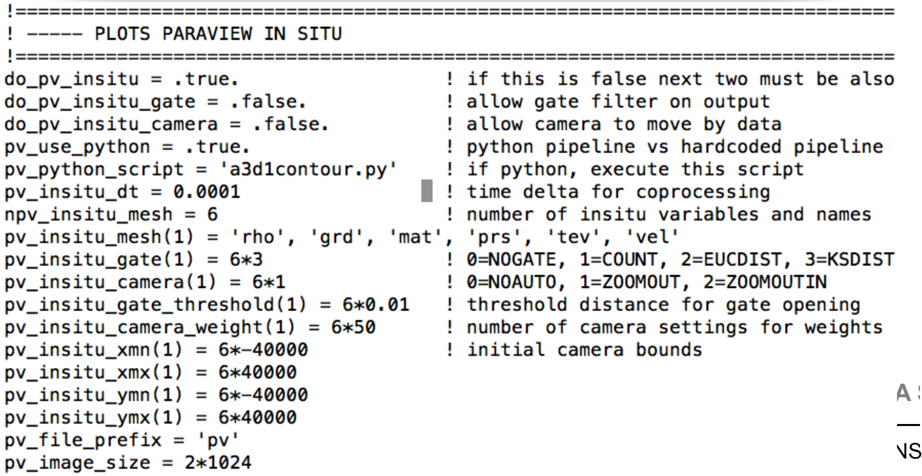
-Standard Module

(links to adaptor)

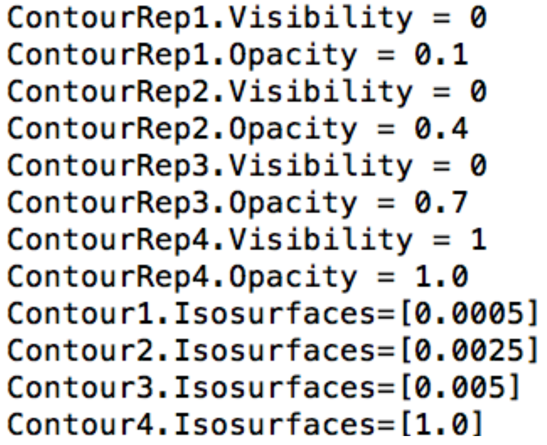
–Adaptor

(links to ParaView)

ParaView



```
!==============================================================================
! ----- PLOTS PARAVIEW IN SITU
!==============================================================================
do_pv_insitu = .true.                        ! if this is false next two must be also
do_pv_insitu_gate = .false.                  ! allow gate filter on output
do_pv_insitu_camera = .false.                ! allow camera to move by data
pv_use_python = .true.                       ! python pipeline vs hardcoded pipeline
pv_python_script = 'a3d1contour.py'          ! if python, execute this script
pv_insitu_dt = 0.0001                        ! time delta for coprocessing
npv_insitu_mesh = 6                          ! number of insitu variables and names
pv_insitu_mesh(1) = 'rho', 'grd', 'mat', 'prs', 'tev', 'vel'
pv_insitu_gate(1) = 6*3                      ! 0=NOGATE, 1=COUNT, 2=EUCDIST, 3=KSDIST
pv_insitu_camera(1) = 6*1                    ! 0=NOAUTO, 1=ZOOMOUT, 2=ZOOMOUTIN
pv_insitu_gate_threshold(1) = 6*0.01         ! threshold distance for gate opening
pv_insitu_camera_weight(1) = 6*50            ! number of camera settings for weights
pv_insitu_xmn(1) = 6*-40000                  ! initial camera bounds
pv_insitu_xmx(1) = 6*40000
pv_insitu_ymn(1) = 6*-40000
pv_insitu_ymx(1) = 6*40000
pv_file_prefix = 'pv'
pv_image_size = 2*1024
```

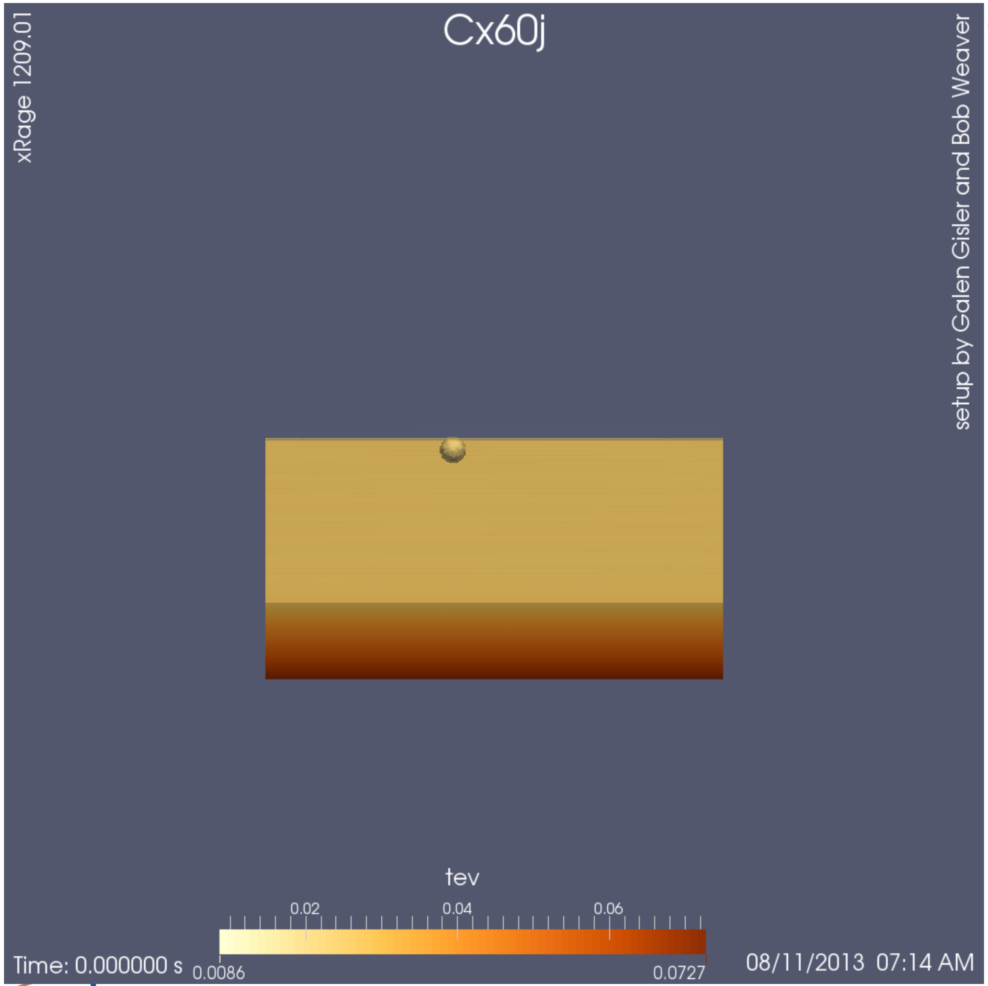


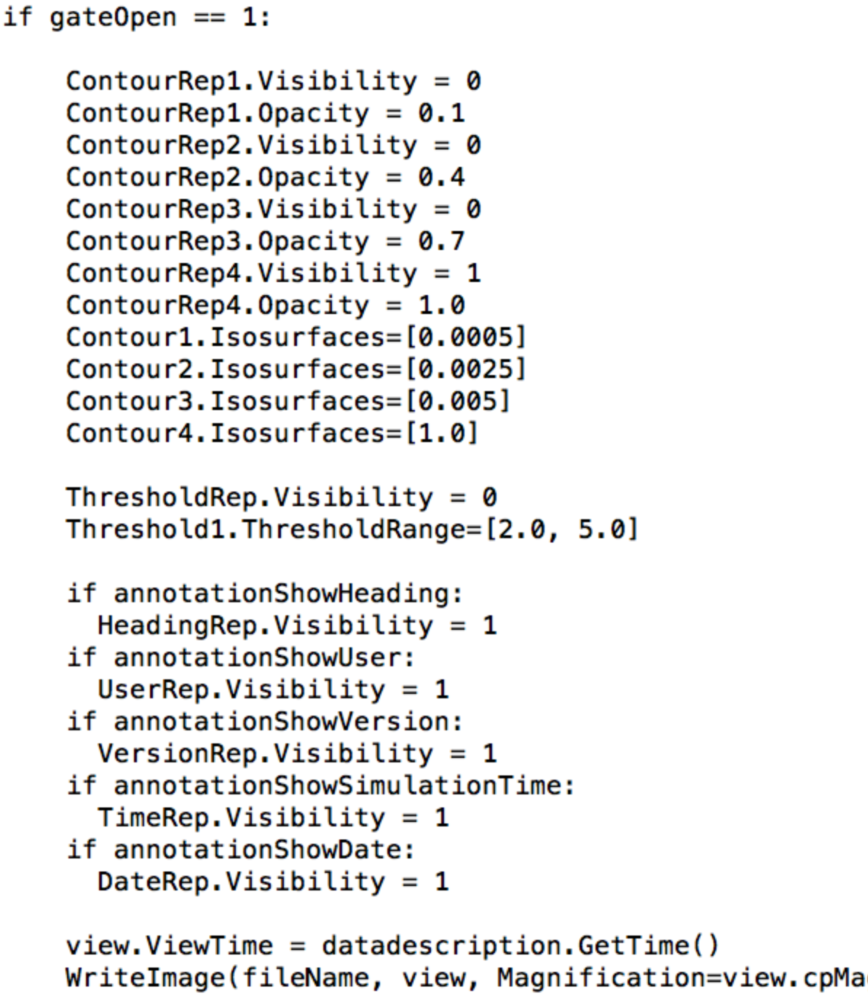
```
ContourRep1.Visibility = 0
ContourRep1.Opacity = 0.1
ContourRep2.Visibility = 0
ContourRep2.Opacity = 0.4
ContourRep3.Visibility = 0
ContourRep3.Opacity = 0.7
ContourRep4.Visibility = 1
ContourRep4.Opacity = 1.0
Contour1.Isosurfaces=[0.0005]
Contour2.Isosurfaces=[0.0025]
Contour3.Isosurfaces=[0.005]
Contour4.Isosurfaces=[1.0]
```

Completion Criteria

Demonstration of our *in situ* visualization and analysis capability applied to xRAGE and/or other ASC codes with performance impacts documented.

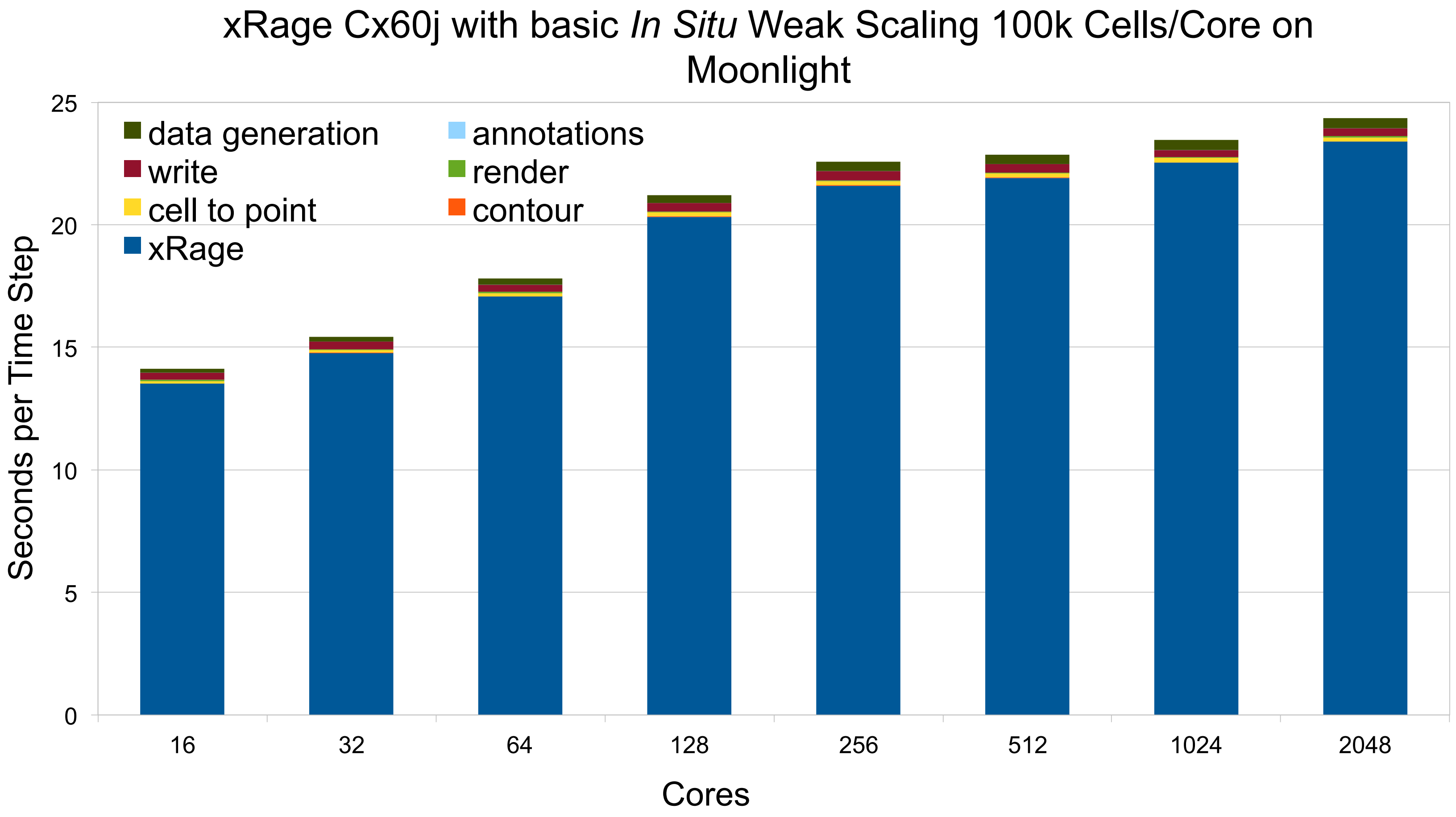
Demonstration of 3D *In Situ* in xRAGE





```
if gateOpen == 1:

    ContourRep1.Visibility = 0
    ContourRep1.Opacity = 0.1
    ContourRep2.Visibility = 0
    ContourRep2.Opacity = 0.4
    ContourRep3.Visibility = 0
    ContourRep3.Opacity = 0.7
    ContourRep4.Visibility = 1
    ContourRep4.Opacity = 1.0
    Contour1.Isosurfaces=[0.0005]
    Contour2.Isosurfaces=[0.0025]
    Contour3.Isosurfaces=[0.005]
    Contour4.Isosurfaces=[1.0]

    ThresholdRep.Visibility = 0
    Threshold1.ThresholdRange=[2.0, 5.0]

    if annotationShowHeading:
      HeadingRep.Visibility = 1
    if annotationShowUser:
      UserRep.Visibility = 1
    if annotationShowVersion:
      VersionRep.Visibility = 1
    if annotationShowSimulationTime:
      TimeRep.Visibility = 1
    if annotationShowDate:
      DateRep.Visibility = 1

    view.ViewTime = datadescription.GetTime()
    WriteImage(fileName, view, Magnification=view.cpMa
```

xRAGE Weak Scaling



UNCLASSIFIED

Costs of *in situ* xRAGE

Time of Processing	Type of File	Size per File	Size per 1000 time steps	Time per File to Write at Simulation
Post	Restart	1,300 MB	1,300,000 MB	1-20 seconds
Post	Ensight Dump	200 MB	200,000 MB	> 10 seconds
In Situ	PNG	.25 MB	250 MB	< 1 second

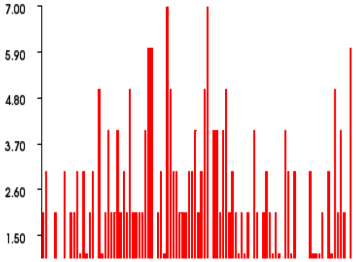
Cores	Auto-Data	Auto-Camera
16 - 256	0.03 s / time step	0.6 s / data product

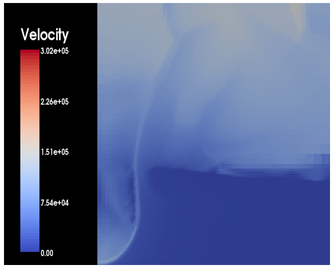
Details from L2 Description

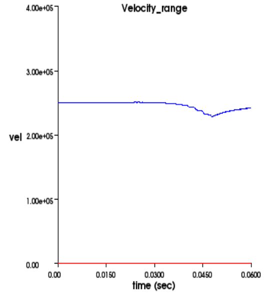
- We have shown completion criteria has been met.
- Next: Details listed in the L2 Description

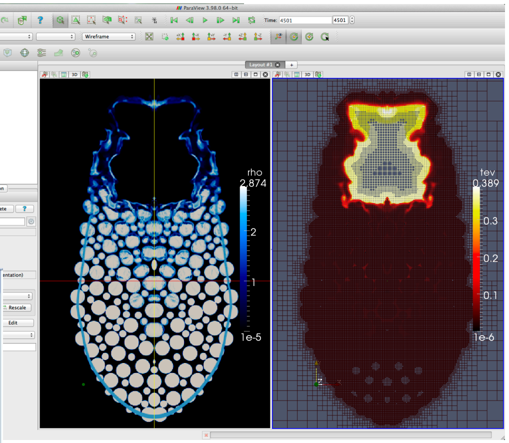
Work with the Eulerian Applications Project to identify target problems that will exercise physics of interest to the program

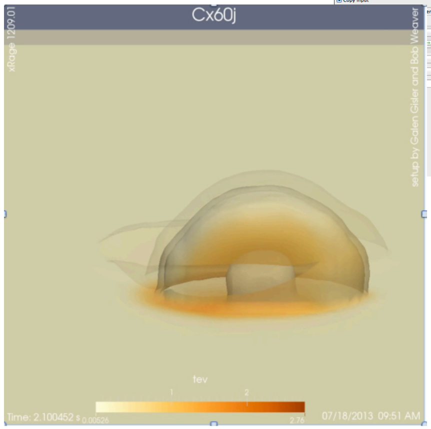
- Originally developed in xRage using 2D LCross problem from Kathy Plesko
- Bob Weaver provided 2D Asteroid (Armageddon Scenario) input deck for development of automatic data algorithms
- Marcus Daniels provided the 3D Chicxulub Asteroid Impact study (Cx60j) that was developed by Bob Weaver and used for 3D scaling studies











Work with users to maximize the achieved usability and productivity improvements

- **Ongoing Process**
- **Annotation:**

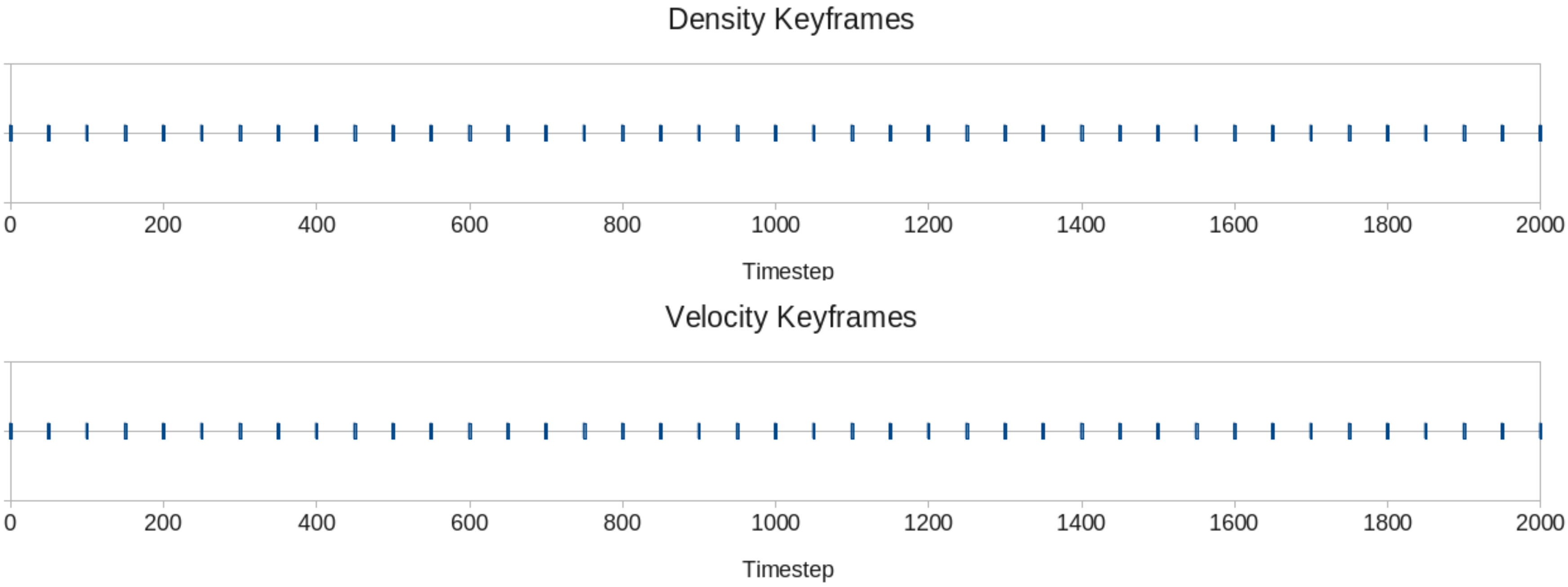
  Bob Weaver noted the importance of annotation for the purposes of understanding and provenance.
  - Simulation name
  - Time stamps
  - Attribution for setup and runner
  - Simulation version ID
- **Automatic Data**

  Bob Weaver pushed for automatic data productization
  - We developed this capability for both the xRage HDF pipeline and ParaView
- **Python Interface**

  Aimee Hungerford steered toward development of Python Pipelines
  - Steer away from custom languages in input decks
  - Creates greater flexibility
- **Work with community to support in generalized framework**
  - Annotations are available in public catalyst
  - Parallel optimizations – Point to Cell and Annotations

Decision Making When to Save and What

- **Simulations typically Sample over time**
  - Simulated time modulo some time
  - Simulated time index module some number



Decision Making When to Save and What

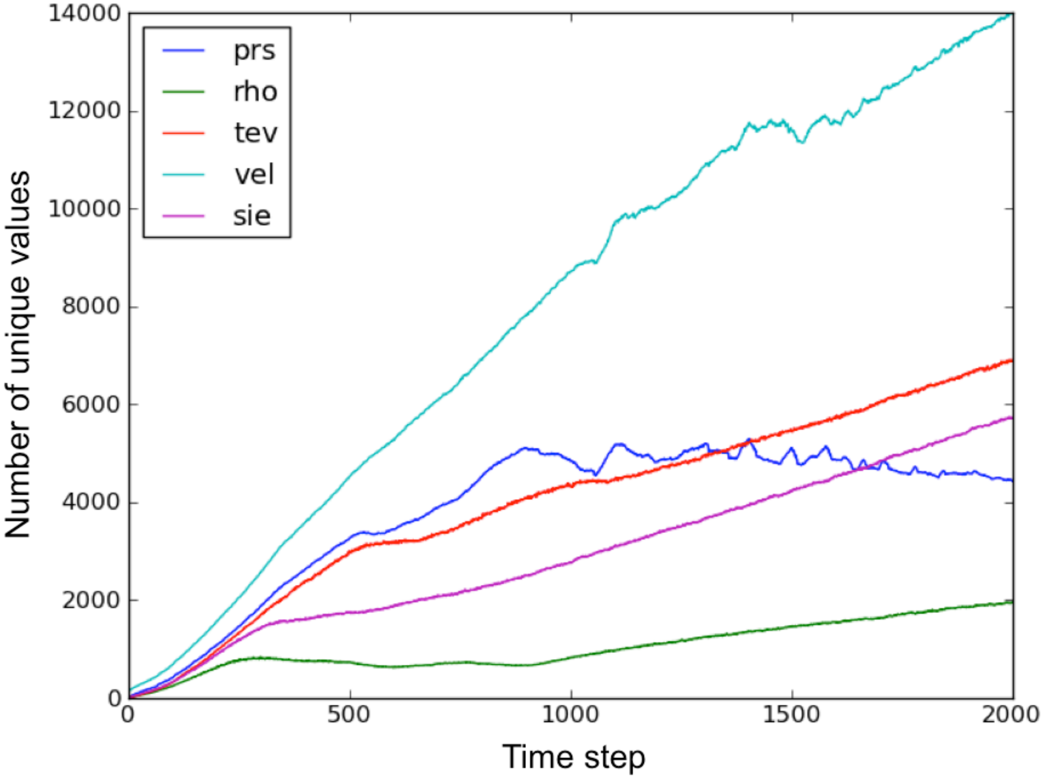
- **Create a metric to indicate state of information**
  - Trigger In Situ when the metric indicates sufficient change vs. last data product creation
  - Change Metric can be applied per variable

- **Pseudo Code**

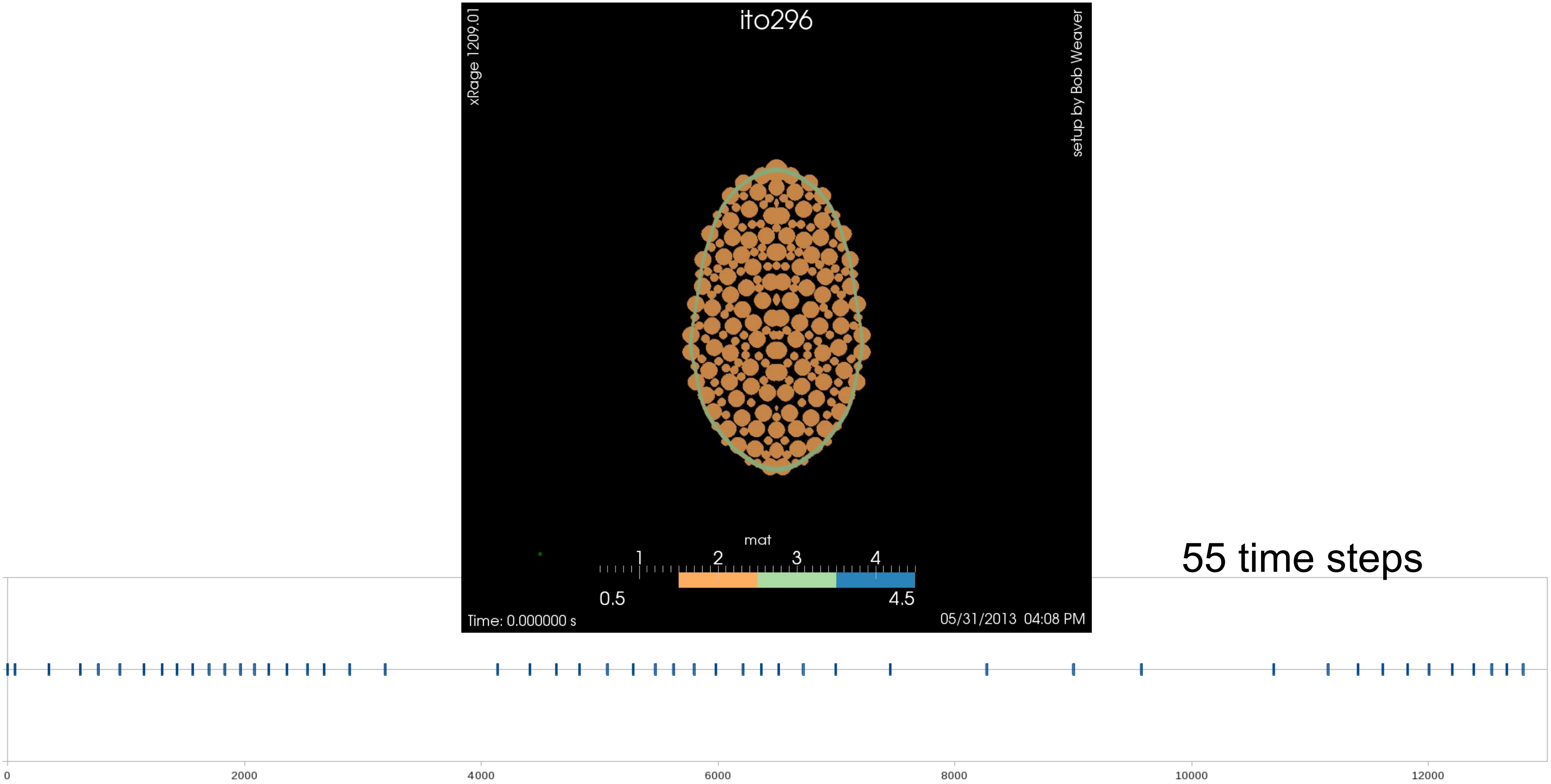
```
for each scalar field
   calculate a metric
   difference the metric from last save
   if difference > threshold
      produce data products defined for that scalar field
```

Leveraging Statistical Expertise - Selecting Key Data

- **Based on the number of unique data values**
  - *u* - *N*umber of unique histogram values (with an epsilon tolerance) in data space for one timestep
    - Histogram across all axises (spatial, value, multivariate)
  - If the difference between the current *u* and previous *u* exceeds a threshold, select image as keyframe
- **Currently exploring other more sophisticated statistical metrics**
  - Kolmogorov–Smirnov distance metric

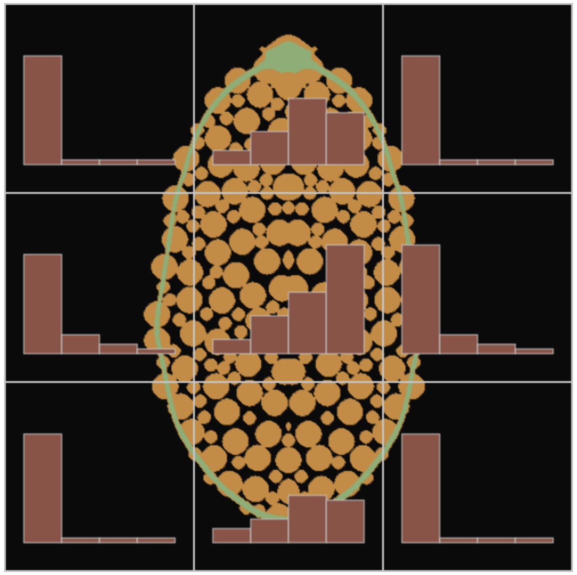


Automatic Data – Sampled Over Time

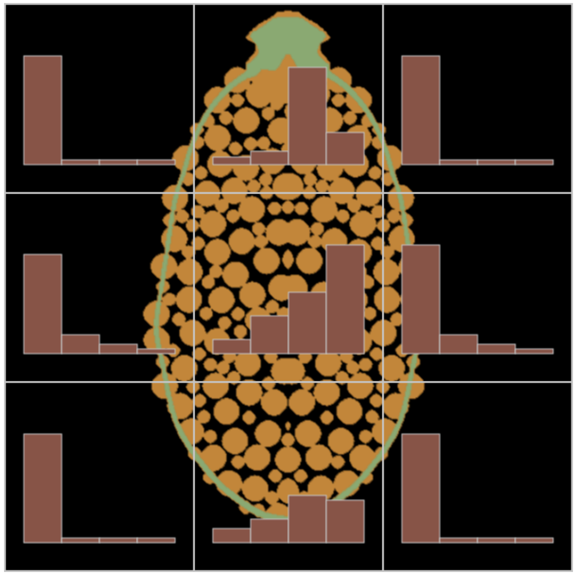


55 time steps

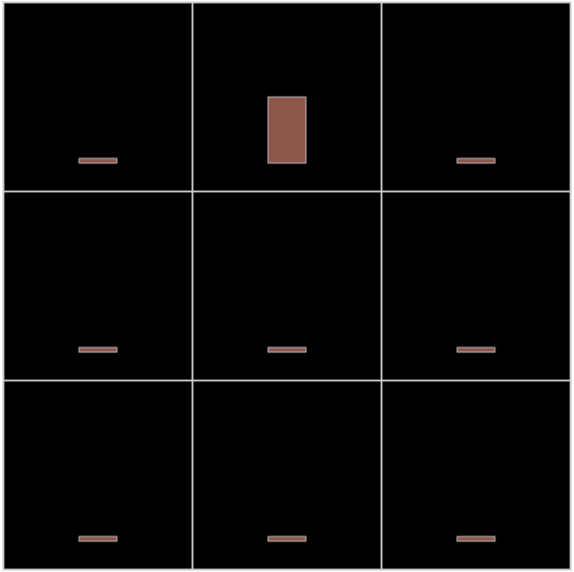
Automatic Camera Placement



-



=



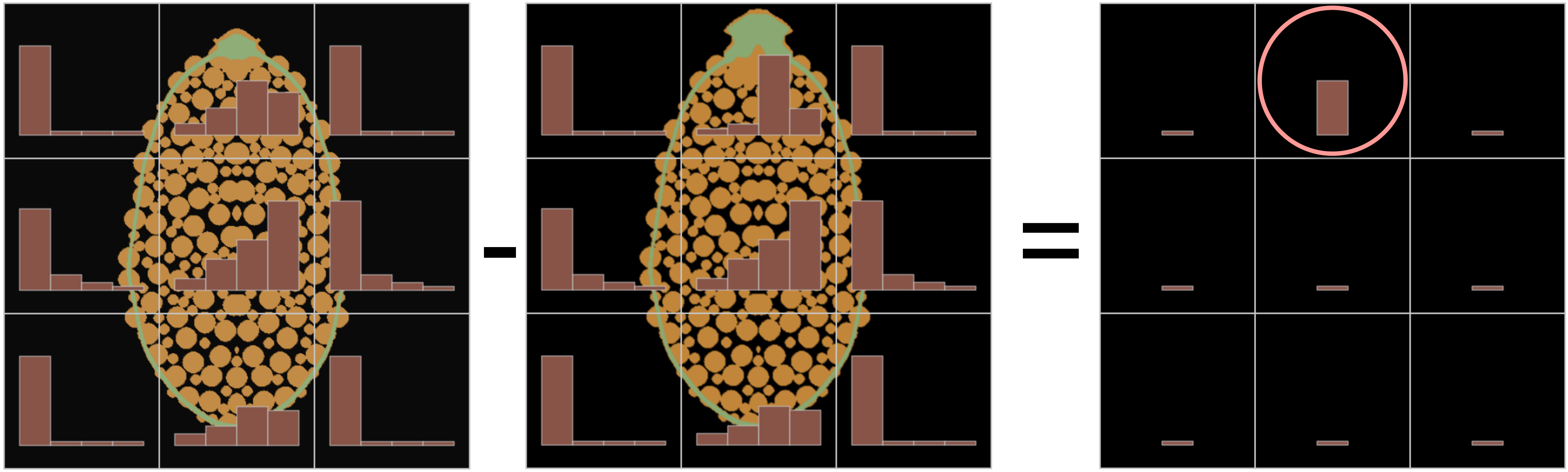
timestep_n

timestep_{n+1}

difference histogram

For each spatial bin, subtract the previous histogram and the current histogram, resulting in a difference histogram.

Automatic Camera Placement



timestep$_n$

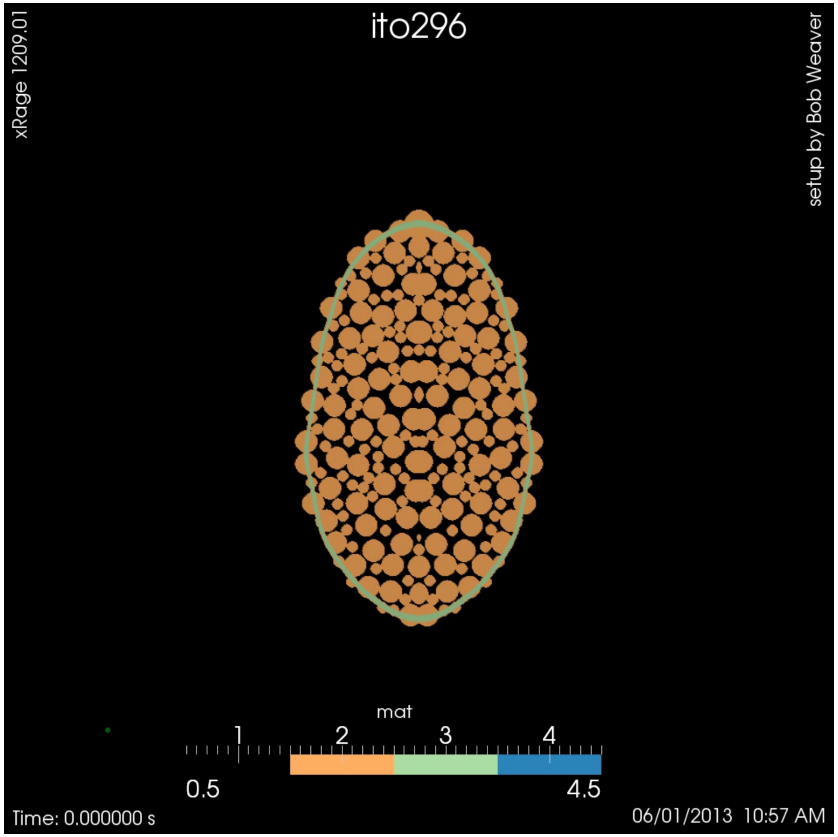
timestep$_{n+1}$

difference histogram

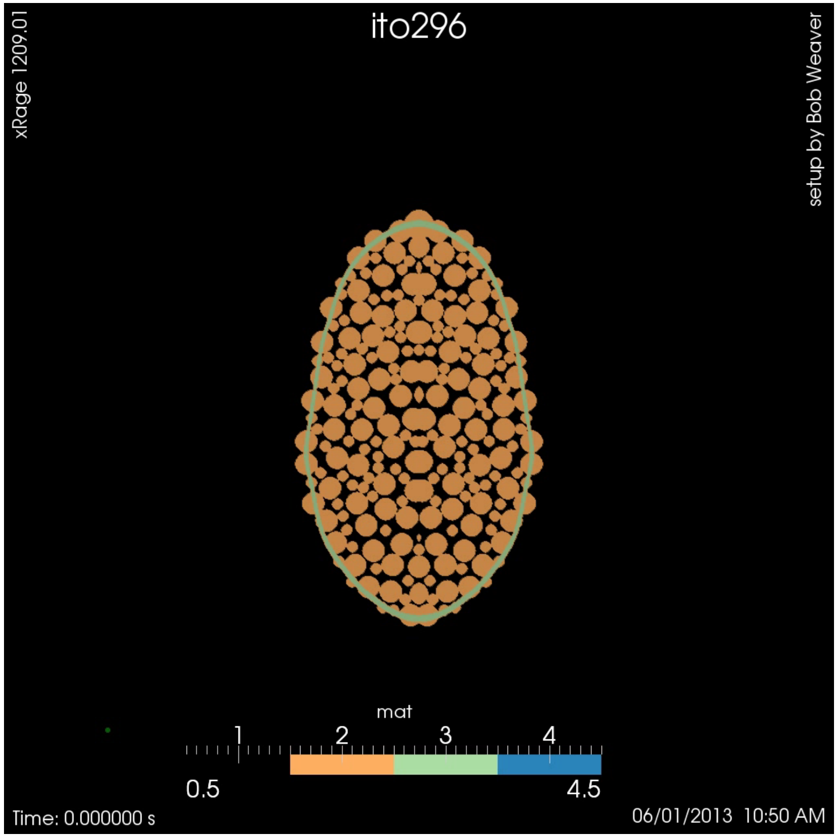
For spatial bins where the difference exceeds a threshold, mark bins as interesting.

Automatic Camera Examples

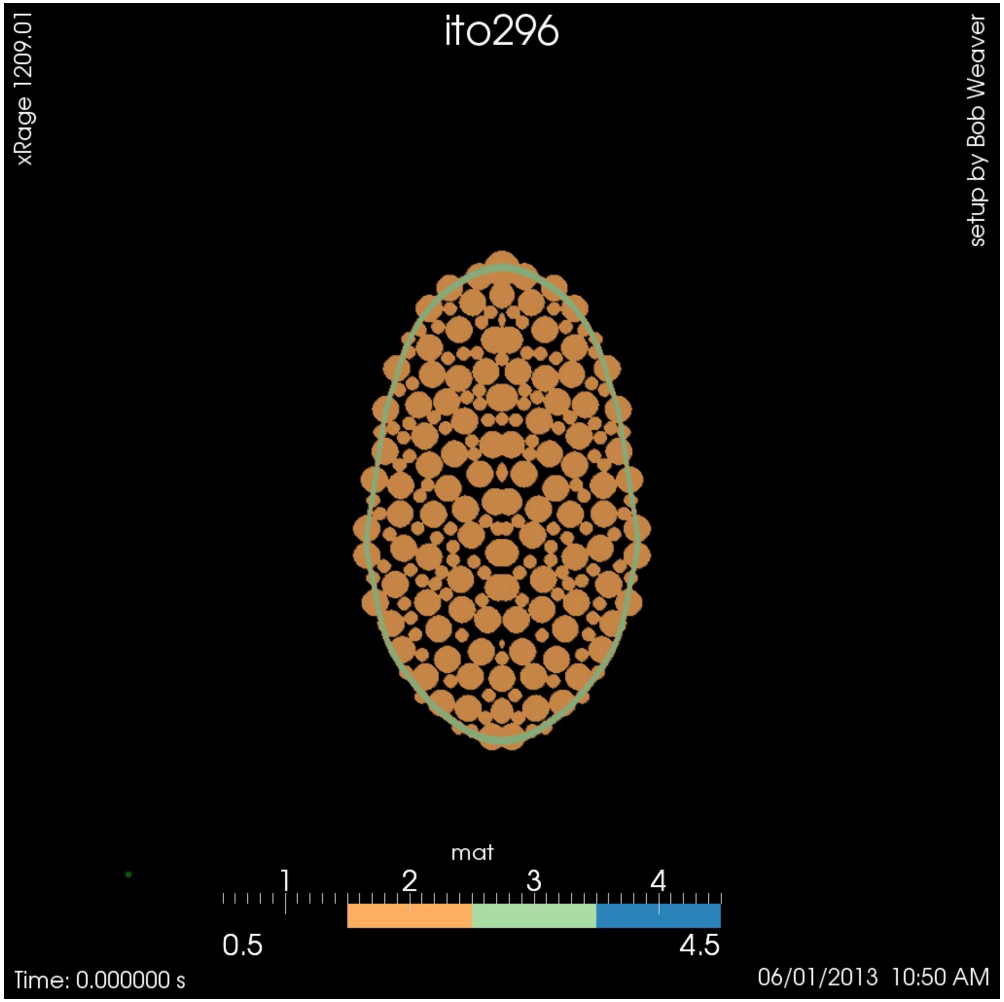
Zoom Out Only



Zoom In and Out

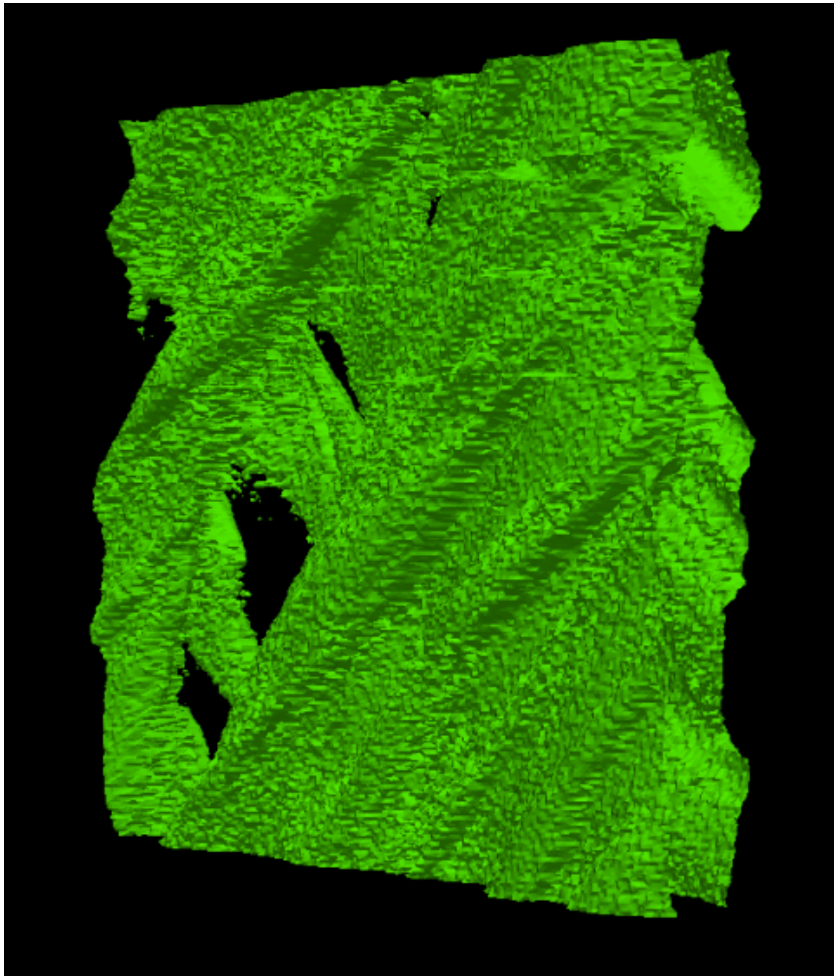


Demonstration of our *in situ* visualization and analysis capability applied to xRAGE



PISTON: A Portable Data-Parallel Framework

- **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, transform, etc.)
- **Implementation**
  - Requires architecture-specific optimizations for a small set of primitives
  - Built on top of NVIDIA's Thrust library
  - We can run algorithms on different multi and many core architectures (e.g, GPUs, multi-core CPUs) using the exact same operator code by compiling to different backends



Contour produced in-situ using PISTON in VPIC simulation

PISTON integration with VTK and ParaView

PISTON Performance and viability

- **Performance of PISTON is similar to non PISTON**
- **PISTON will be set to leverage performance on codes that have data on coprocessing elements like GPUs**
  - CoGL, HACC, PINION
- **We continue to work with Kitware on integrating PISTON into VTK**
- **This is viable**
- **From the VTK 6.0.0 Announcement June 24, 2013:**

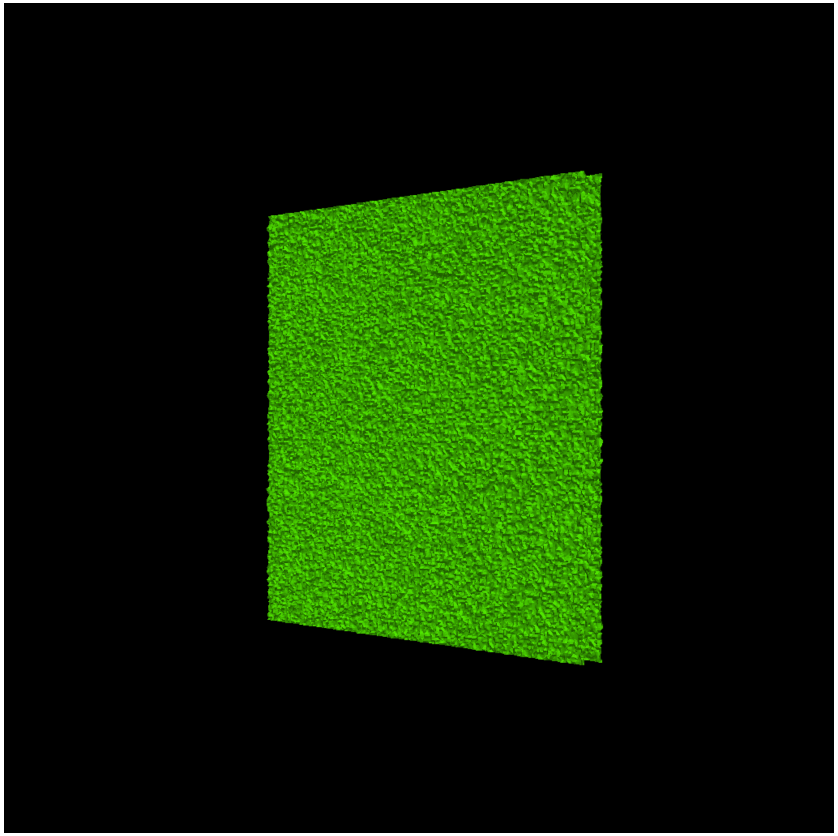
*** This release marks a first foray into GPGPU processing within VTK. AcceleratorsPiston is a module that calls into LANL's Piston library to enable Thrust-based accelerated data processing filters.**

VPIC and Pagosa

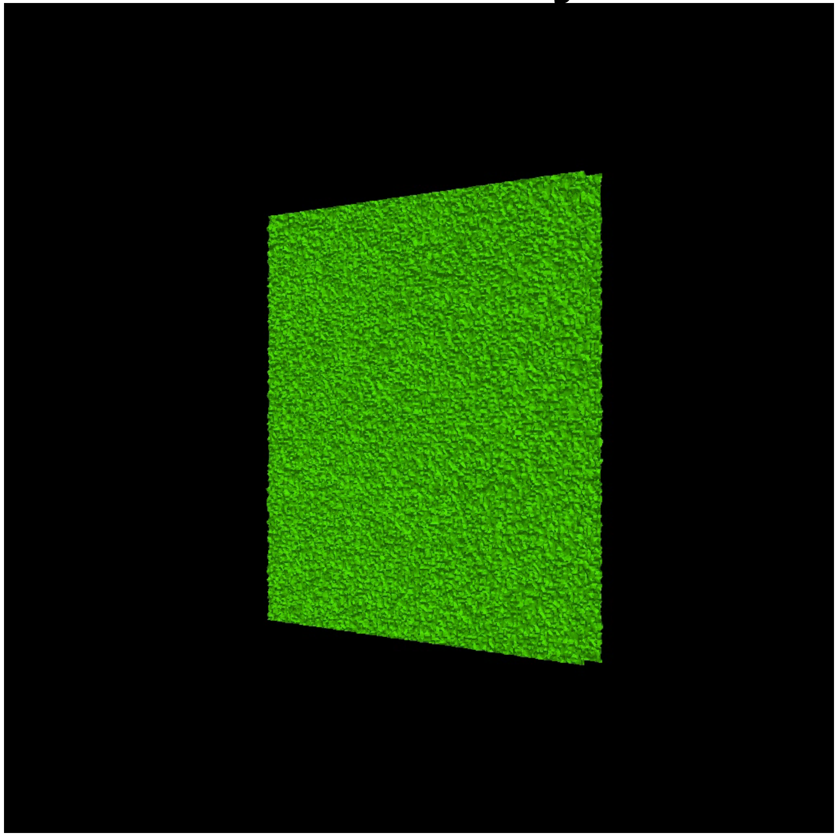
In situ studies applied to other ASC Codes

Demonstration of our *in situ* visualization and analysis capability applied to ASC Code VPIC

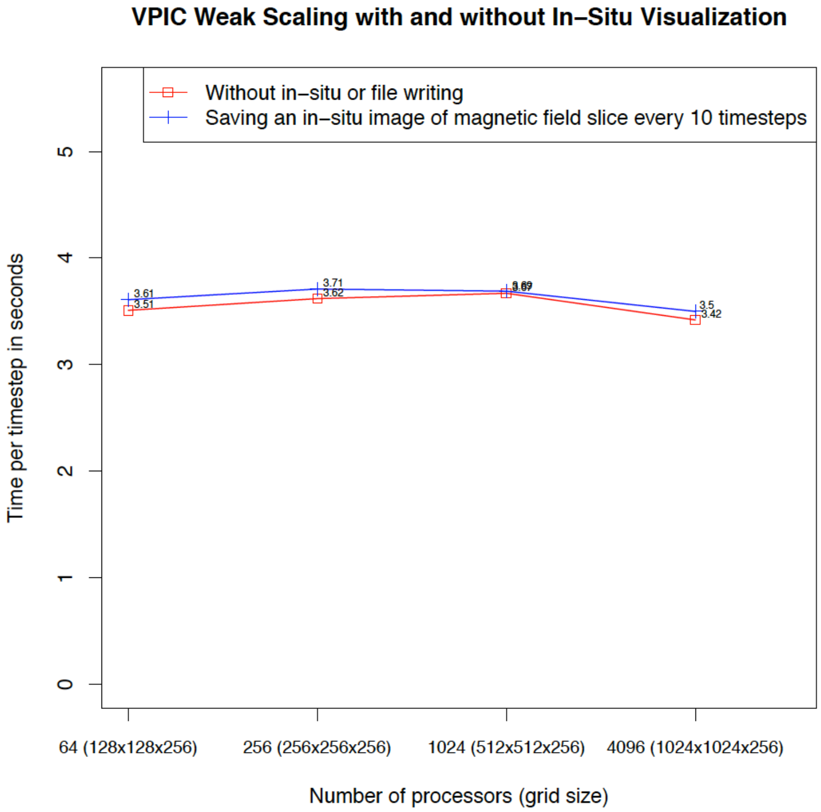
8 frames



346 frames – every 10th

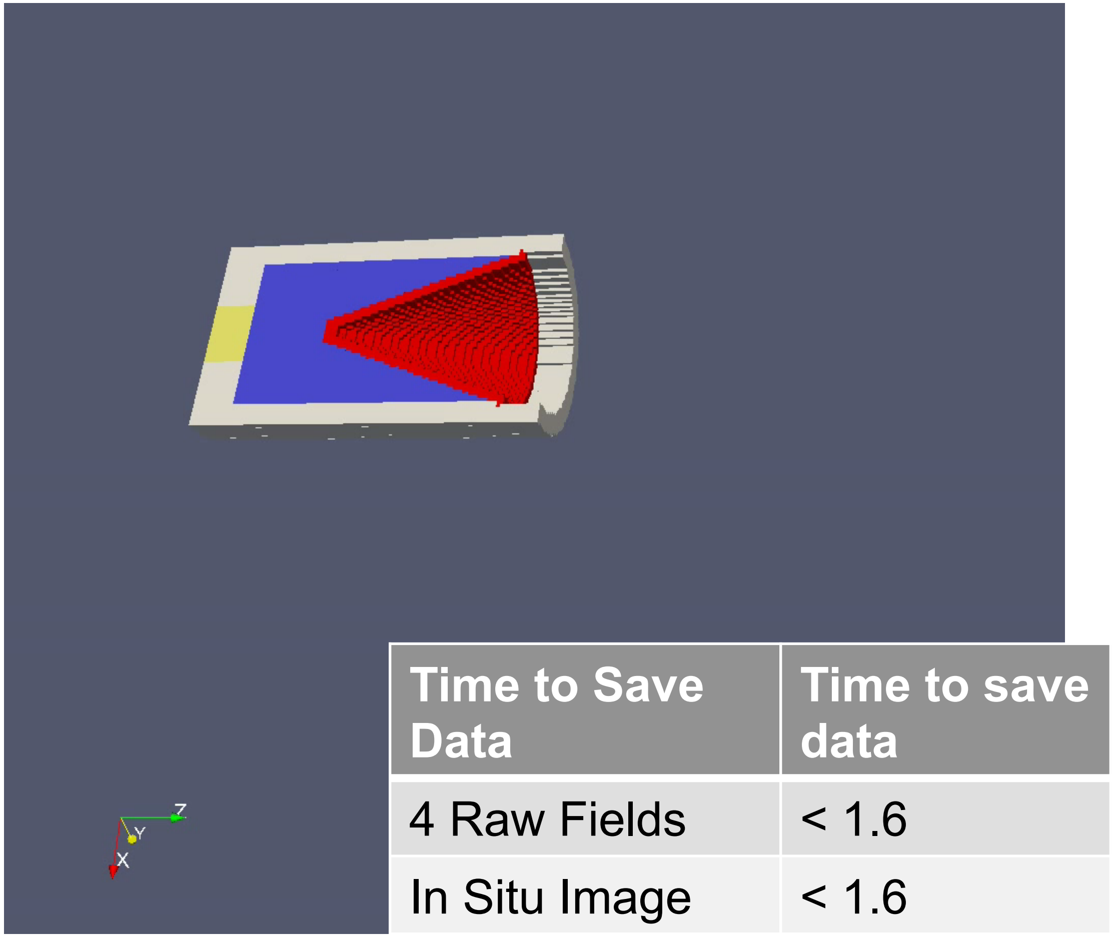


VPIC Weak Scaling



Time to Save Data	Time to save data	Size on Disk
Raw VPIC Field	< 7 seconds	> 100 MB
In Situ Image	< 1 seconds	< 1 MB

Demonstration of our in-situ visualization and analysis capability applied to ASC Code Pagosa



Time to Save Data	Time to save data	Size on Disk
4 Raw Fields	< 1.6	> 80 MB
In Situ Image	< 1.6	< .02 MB

Possible Directions for Future work

- **Work with Developers and Users**
  - For better productization, documentation
- **Support transition of support**
- **Build on Cielo**
- **Testing integration**
- **Documentation**
- **Zero Copy in xRage - memory savings**
- **Improve HDF performance**
- **Annotation and metadata in hdf files, ParaView output files, etc…**
  - Username, xRage version, data transformations…

Thank You

John Patchett

patchett@lanl.gov

505 665 1110

Many thanks to all who have contribute to this work including: James Ahrens, Boonthanome Nouanesengsy, Patricia Fasel, Patrick O'leary, Chris Sewell, Jon Woodring, Christopher Mitchell, Ollie Lo, Kary Meyers, Joanne Wendelberger, Curt Canada, Marcus Daniels, Hilary Abhold, Gabe Rockefeller.