Progress readying VisIt for VTK-m

Eric Brugger, Cameron Christensen, Jeremy Meredith, Kathleen Biagas, Kevin Griffin, Cyrus Harrison, Mark Miller

What is our approach?

We will change VisIt to base its processing and rendering infrastructure on VTK-m. We will take an incremental approach using the existing multicore and manycore toolkits to gain experience with such toolkits and prepare ourselves for a quick transition once VTK-m is ready.

Benefits of an incremental approach

- It will give us experience with these toolkits so that we are ready to make a quick transition once VTK-m is ready
- It will serve as a real world test bed for these toolkits
- We can provide feedback to the VTK-m developers based on our experiences with these toolkits
- We can make sure that VTK-m will meet the needs of VisIt when it is delivered
- It will allow us to deliver incremental functionality to the users over time

Integration overview

Phase 1: Form abstractions layers for the portions of VisIt that will be impacted by the transition to operate independently of any toolkit. These include the data processing filters and the rendering infrastructure.
- Enhance our data set representation to internally support an arbitrary number of other data set abstractions
- Enhance all of our filters so that they can be implemented using one or more toolkits
- Enhance our execution pipeline to support pipelines consisting of filters that are implemented using different toolkits
- Enhance our rendering infrastructure to support multiple underlying rendering infrastructures.

Phase 2: Prototype a set of filters that are representative of all the filters with multiple toolkits. We will prototype the filters with the following toolkits.
- Dax Toolkit – The Data Analysis at Extreme toolkit
- EAVL – The Extreme-scale Analysis and Visualization Library
- PISTON – A Portable Cross-Platform Framework for Data-Parallel Visualization
- Kokkos – Enabling performance portability across manycore architectures
- RAJA – A lightweight mechanism for parallelizing loops across manycore architectures

Phase 3: Prototype using the EAVL ray caster in our rendering infrastructure

Phase 4: Provide feedback to the VTK-m developers from our prototyping efforts

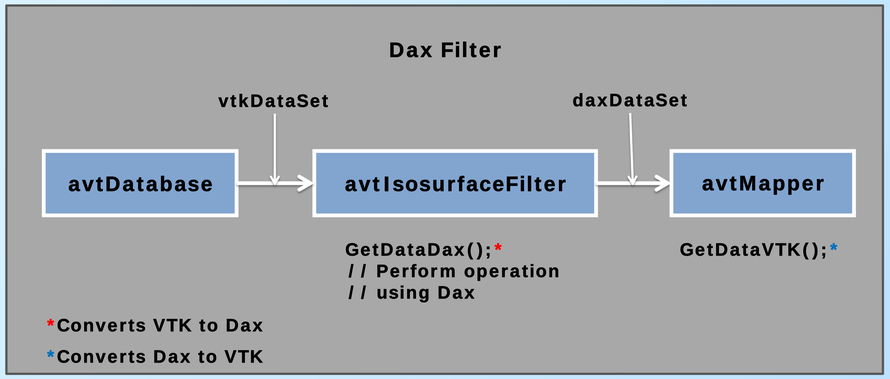
Phase 5: Incrementally convert the VisIt filters to VTK-m as functionality becomes available

Software engineering details

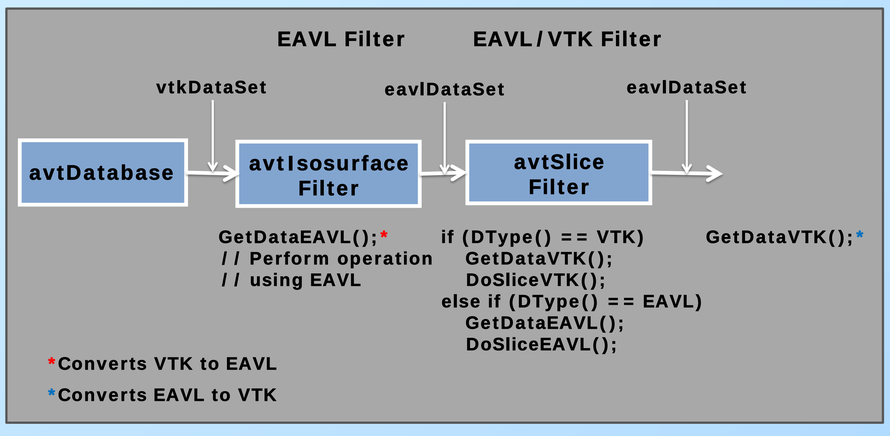
Software infrastructure changes

- We enhanced our avtDataRepresentation class to handle other dataset types
- We modified all our filters to operate on avtDataRepresentations
- We added into avtDataRepresentation the ability to convert between VTK datasets and toolkit datasets automatically
  - These are zero-copy in most situations
- We started to modify the filters to use the existing toolkits

An example pipeline using Dax



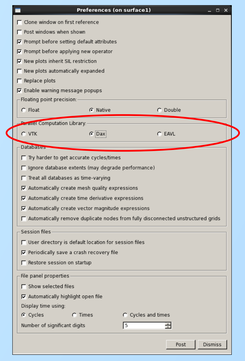
An example mixed filter pipeline using EAVL



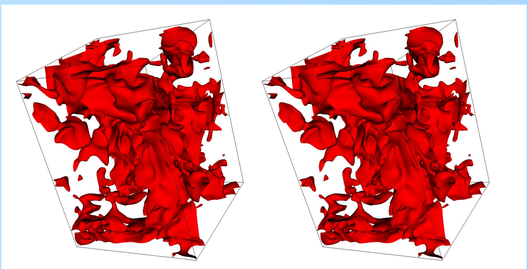
Current status

Software infrastructure changes completed

- We completed all the changes to support multiple toolkits within the filter infrastructure.
- We have completed the EAVL toolkit integration into the filter infrastructure and implemented a filter. This was released in VisIt 2.8.1.
- We have prototyped the Dax toolkit integration into the filter infrastructure and implemented a filter. This will be released in a VisIt 2.9.1.
- The user can select at runtime which toolkit to use.



The VisIt Preferences window for selecting the toolkit

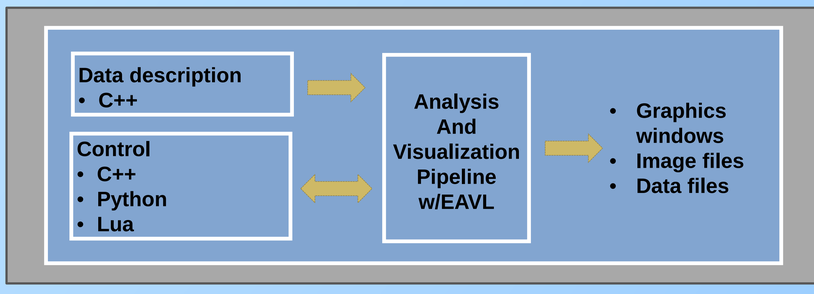


A contour surface generated by Dax and EAVL

Future plans

Finish implementing the contour filter using all of the toolkits and frameworks and provide feedback to the VTK-m developers on our experiences.

Prototype an in situ library using EAVL



This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344, Lawrence Livermore National Security, LLC

LLNL-POST-667660

SDAV Scalable Data Management, Analysis, and Visualization

U.S. DEPARTMENT OF ENERGY | Office of Science