SAND2015-2527R

LAB 14-1043

FWP 14-017566

Institution: **Sandia National Laboratories**

XVis: Visualization for the Extreme-Scale Scientific-Computation Ecosystem

Project PI: **Kenneth Moreland**, Sandia National Laboratories
Christopher Sewell, Los Alamos National Laboratory
Hank Childs, University of Oregon
Kwan-Liu Ma, University of California at Davis
Berk Geveci, Kitware, Inc.
Jeremy Meredith, Oak Ridge National Laboratory

Mid-year report, FY15 Q2

1 Project Description

The XVis project brings together the key elements of research to enable scientific discovery at extreme scale. Scientific computing will no longer be purely about how fast computations can be performed. Energy constraints, processor changes, and I/O limitations necessitate significant changes in both the software applications used in scientific computation and the ways in which scientists use them. Components for modeling, simulation, analysis, and visualization must work together in a computational ecosystem, rather than working independently as they have in the past. This project provides the necessary research and infrastructure for scientific discovery in this new computational ecosystem by addressing four interlocking challenges: emerging processor technology, in situ integration, usability, and proxy analysis.

Emerging Processor Technology One of the biggest recent changes in high-performance computing is the increasing use of accelerators. Accelerators contain processing cores that independently are inferior to a core in a typical CPU, but these cores are replicated and grouped such that their aggregate execution provides a very high computation rate at a much lower power. Current and future CPU processors also require much more explicit parallelism. Each successive version of the hardware packs more cores into each processor, and technologies like hyperthreading and vector operations require even more parallel processing to leverage each core's full potential.

XVis brings together collaborators from the predominant DOE projects for visualization on accelerators and combines their respective features in a unified visualization library named VTK-m. VTK-m will allow the DOE visualization community, as well as the larger visualization community, a single point to collaborate, contribute, and leverage massively threaded algorithms. The XVis project is providing the infrastructure, research, and basic algorithms for VTK-m, and we are working with the SDAV SciDAC institute to provide integration and collaboration throughout the Office of Science.

***In Situ* Integration** Fundamental physical limitations prevent storage systems from scaling at the same rate as our computation systems. Although large simulations commonly archive their results

before any analysis or visualization is performed, this practice is becoming increasingly impractical. Thus, the scientific community is turning to running visualization *in situ* with simulation. This integration of simulation and visualization removes the bottleneck of the storage system.

Integrating visualization *in situ* with simulation remains technically difficult. XVis leverages existing *in situ* libraries to integrate flyweight techniques and advanced data models to minimize resource overhead. Within our *in situ* visualization tools, XVis integrates existing visualization algorithms and those incorporating emerging processor technology. XVis also studies the latest techniques for new domain challenges and for post hoc interaction that reconstructs exploratory interaction with reduced data.

Usability A significant disadvantage of using a workflow that integrates simulation with visualization is that a great deal of exploratory interaction is lost. Post hoc techniques can recover some interaction but with a limited scope or precision. Little is known about how these limitations affect usability or a scientist's ability to form insight. XVis performs usability studies to determine the consequences of *in situ* visualization and proposes best practices to improve usability.

Unlike a scalability study, which is always quantitative, XVis' usability studies are mostly qualitative. Our goal is not to measure user performance; rather, we want to learn about the limitations and benefits of incorporating *in situ* methods in scientists' workflows. These studies reveal how the simulation, hardware, and users respond to a particular design and setting.

Proxy Analysis The extreme-scale scientific-computation ecosystem is a much more complicated world than the largely homogeneous systems of the past. There is significantly greater variance in the design of the accelerator architecture than is typical of the classic x86 CPU. *In situ* visualization also yields complicated interactions between the simulation and visualization that are difficult to predict. Thus, the behavior observed in one workflow might not be indicative of another.

To better study the behavior of visualization in numerous workflows on numerous systems, XVis builds proxy applications that characterize the behavior before the full system is run. We start with the design of mini-applications for prototypical visualization operations and then combine these with other mini-applications to build application proxies that characterize the behavior of larger systems. The proxy analysis and emerging processor technology work are symbiotic. The mini-applications are derived from the VTK-m implementations, and the VTK-m design is guided by the analysis of the mini-applications.

2 Progress Report

The XVis research plan specified in the proposal is divided into a set of milestones spread over the 3-year period of the project, divided among the projects research areas, and distributed among the participating institutions. Our report is similarly organized by giving progress on each of these milestones. Our report is abbreviated to include only those milestones with relevant work in the time period of this report.

2.1 Emerging Processors

Milestone 1.a, Initial VTK-m Design (Year 1–SNL, Kitware, ORNL, LANL) Provide the research and design for VTK-m functional operation and, in conjunction with SDAV, develop an initial implementation.

Expected Completion: FY15, Q4 Status: In progress

The VTK-m prototype is central to many of the activities in XVis. As such, a significant portion of the work in the early part of the project is dedicated to this milestone, and we have made a significant amount of progress.

We have established a central git repository hosted by Kitware. The URL for the repository is http://public.kitware.com/vtkm.git. We have established several procedures for managing the collaborative development of the project. This includes a weekly developers meeting to coordinate and communicate, a system of design documents (listed at http://m.vtk.org/index.php/Design_Documents), a branchy development workflow for coordinating concurrent contributions (http://m.vtk.org/index.php/Contributing_to_VTK-m), a set of coding conventions, and a large set of regression tests run nightly (reported at https://open.cdash.org/index.php?project=VTKM).

The basic foundations for VTK-m including the build system, package structure, and fundamental classes are implemented. VTK-m now includes a generic device adapter that implements the basic data parallel primitives and provides performance portability. VTK-m currently has implementations for a CUDA device and a serial device for debugging purposes.

VTK-m has a generic array interface that provides a single interface for direct access to data of any type made possible with static templating. This generic array interface simplifies zero-copy interfaces to other data structures. VTK-m also has a dynamic array wrapper that helps with handling data whose type is not known until compile time.

The mechanism for building and executing worklets is available. The mechanism is flexible in that it is straightforward to define new worklet algorithms, new worklet types, and new data handling mechanisms.

We have also begun to develop filters within VTK-m, which implement specific visualization algorithms using the VTK-m arrays, worklets, and device adapter algorithms. The first such algorithm to be developed was isosurface (for structured 3D grids, using Marching Cubes). Cut surface and threshold filters can also be constructed using the same core algorithm. Several variants of the algorithm have been developed in order to optimize and evaluate performance and memory usage using techniques such as fusing cells, slicing the data set, and parallelizing over the output triangles rather than the input voxels. As the infrastructure for data models further develops, and as new worklet types are introduced into VTK-m, the isosurface algorithm will co-evolve with them, becoming more efficient and versatile (supporting other data types). A variety of statistical analysis algorithms, making use of device adapter algorithms such as reduce and scan, have also been implemented, including a histogram filter and a filter for first, second, and third moment calculations. In addition to the aforementioned filters, which have been implemented based on standard existing algorithms, novel data-parallel algorithms are also being designed in collaboration with Hamish Carr from the University of Leeds for computing contour trees, which encode the topological changes that occur to the contour as the isovalue ranges between its minimum and maximum values.

The base VTK-m data model is in place, though still experimental, and will be moved to the master VTK-m branch after further hardening and additional features are added. It is built around the basic array interfaces and other infrastructure in VTK-m, and supports both regular and irregular grid topologies. A worklet mapping between topological elements in these grids is also in a working, but experimental, state, and will form the basis for a number of visualization and analysis algorithms.

In conjunction with the SDAV SciDAC institute, the VTK-m development team had a design review with engineers from NVIDA on March 4-5 for running VTK-m on CUDA-capable cards.

Highlights of the meeting include suggestions to introduce CUDA asynchrony/streaming, texture memory, layouts for unstructured mesh connectivity, and the possibilities of JIT compiling. In response to this review we have implemented texture memory support, and investigated better scheduling strategies for the CUDA device adapter, both of which improve performance.

To enable broad sharing of our code, we have received approval to assert copyright on our early implementation of VTK-m. VTK-m is officially released with a BSD 3-clause license.

Our VTK-m development effort is also focused on providing documentation to make our library accessible. We are maintaining a User's Guide with detailed information on using the features currently available in VTK-m. The current version of the VTK-m User's Guide is available from the VTK-m Wiki (http://m.vtk.org/images/c/c8/VTKmUsersGuide.pdf).

2.2 In Situ

Milestone 2.a Expand Data Models (Year 1–ORNL, Kitware) Expand visualization data models to encompass broader scope from new science domains.
Expected Completion: FY15, Q3 Status: In progress

As mentioned above, the basic VTK-m data model is in progress. As built, it already includes some advanced features necessary to support in situ analysis and modern architectures and simulation codes. Specifically, initial heterogeneous memory space support is available through the VTK-m array interfaces, and this array infrastructure has zero-copy support. The VTK-m data model also supports mesh features that were challenging in traditional data models, such as mixed-topology meshes, and is generally more flexible, leading to greater efficiency.

Milestone 2.b Post Hoc Interaction (Year 1–U Oregon) Implement three algorithms that use extreme-scale features such as non-volatile memory or knowledge of communication efficiencies.
Expected Completion: FY15, Q4 Status: In progress

We have made good progress on two of the three algorithms. The first involves using SSDs to do more effective compression by considering temporal compression. We have been using wavelets, and have a study in progress. The second algorithm involves understanding the performance limits of VTK-m's data-parallel primitives approach. We have considered both surface rendering and volume rendering and have observed performance comparable with community standards. We plan to do a further study on what performance we are missing out on by being hardware agnostic, and expect this to be completed by on time for the milestone.

Finally, we would like to delay the third algorithm until Year 2, when more architectures will be available (NVLINK, Knight's Landing) so we can do a study on deep memory hierarchies. This delay is consistent with UO's underspending of the budget.

Milestone 2.c Flyweight In Situ (Year 2–Kitware) Provide flyweight in situ visualization techniques into a feature-rich, general-purpose library.
Expected Completion: FY16, Q4 Status: In progress

Kitware has been investigating using non-standard memory layouts for arrays and data structures. VTK now has the concept of a MappedDataArray and MappedDataSet, which allow for custom memory layouts. Using STL-style iterators we allow complex algorithms to operate on iterators for mapped data arrays while continuing to use efficient raw memory access for standard VTK data arrays. While the main objective of these changes was to allow for tight coupling of VTK *in situ* with simulations, they also allow for things such as constant value arrays, implicit point arrays, and other efficient data model concepts that VTK-m also has. This work could be the

foundation for allowing VTK-m's data model to be used efficiently and seamlessly inside VTK with no memory copies.

2.3 Usability

Milestone 3.a Develop Techniques to be Studied (Year 1–UC Davis) Identify existing and new visualization techniques to be studied. Revise existing ones and implement new ones as needed.
Expected Completion: FY15, Q4 Status: In progress

During this period, the UC Davis team has been pursuing two independent studies. In one study, we aim to evaluate the usability of VTK-m, a fine-grain parallel programming library, for realizing visualization operations. We have first implemented two volume rendering algorithms: ray-casting and cell projection in Dax because VTK-m is not in its full configuration. The rendering performance achieved on NVIDIA GPU has not been satisfactory so we are still in the process of optimizing several aspects of data access and calculations. After the performance becomes acceptable, we will then test the implementations on Intel Xeon Phi processors using the testbed available at NERSC. We hope this experience will help us also create and evaluate some in situ visualization solutions in VTK-m.

In the second study, we are developing in situ visualization technologies that will be used in our proposed usability studies. The first technology that we are developing, which we call Ximage, is based on our former work Explorable Images. Because data reduction and different levels of approximation in visual transformation are inevitable when conducting visualization at extreme-scale, we would like to offer uncertainty-aware in situ visualization to users. We believe this feature will make in situation visualization more attractive and acceptable by scientists. Presently, we are still researching how to model uncertainty due to Explorable Image generation. The second technology we plan to develop is for supporting the need of studying particle and field data together. We have derived a preliminary design for it and will begin its implementation next.

2.4 Proxy Analysis

Milestone 4.a Initial Mini-App Implementation (Year 1–SNL, ORNL) An initial implementation of mini-applications based on visualization and in situ workloads.
Expected Completion: FY15, Q4 Status: Preliminary work

Although milestone 4.a was scheduled to be started at the beginning of the project, the majority of the work has been postponed in lieu of providing a VTK-m prototype (milestone 1.a), which is on the critical path. We plan the majority of the mini-app implementation in FY15, Q4. That said, there has been some related work progressing.

Kitware working with Intel has implemented two Mini-Apps to study the ability to parallelize common visualization algorithms using multithreading and CPU SIMD vector extensions. Kitware implemented multiple structured and unstructured grid contouring (Marching Cubes, and Marching Tetrahedra) algorithm variations to determine what approach performs better. This work is a good starting point for creating VTK-m based Mini-Apps, and as a reference implementation for how to design filters for emerging architectures such as the Xeon Phi.

In preparation for later proxy analysis milestones (and specifically 4.b: Validate Mini-App Characteristics in Year 2), we have been familiarizing ourselves with the Oxbow suite of application characterization tools and have performed some initial within-node characterization of a sequential contouring algorithm in VisIt and a data-parallel contouring algorithm in EAVL, one of the predecessor projects to VTK-m. These initial results point the way towards further

investigation and directions for mini-app implementations – for example, while there is no thread-level parallelism in VisIt, it was able to make use of integer SIMD arithmetic, while the highly-parallel EAVL algorithm was not.

3 Other Activities

3.1 Outreach

"Roadmap for Many-Core Visualization Software in DOE," Jeremy Meredith, GTC Presentation, March 2015.

"Visualization Toolkit: Faster, Better, Open Scientific Rendering and Compute", Robert Maynard and Marcus Hanwell, GTC Presentation, March 2015.

"Hands-on Lab: In-Situ Data Analysis and Visualization: ParaView, Calalyst and VTK-m" Marcus Hanwell and Robert Maynard, GTC Lab, March 2015.

4 Acknowledgement

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under contract number 14-017566.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.