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

Year-end report, FY15

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: Complete

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 https://gitlab.kitware.com/vtk/vtk-m. 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 VTK-m repository is maintained by a gitlab service that facilitates several aspects of our development procedures including managing design review pages and running regression tests before any modifications are committed to the main repository.

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, a multi-core CPU device (using the TBB library), 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 data model in VTK-m is based on an arbitrary collection and combination of cells, fields, and coordinate systems. The data model is abstract enough to flexibly represent a variety of structures but concrete enough to have clear semantics. The data model can also adapt to arbitrary array structures allowing VTK-m to interface directly with data defined in other software packages. We have currently implemented cell sets that represent either cells arranged in 1, 2, or, 3D structured array or unstructured cells with explicitly defined connections. We also have implementations for coordinate systems with either uniform axis-aligned spacing or arbitrary positions.

We have been defining and implementing a user-facing API to construct data sets of common mesh types, including regular grids, rectilinear grids, and unstructured grids. The goal of this API is provide an interface for users of VTK-m that masks the complexity of the underlying representation while providing a straightforward and efficient way to create VTK-m datasets from raw data arrays, or other representations. This includes translation from VTK datasets into VTK-m datasets, data readers, and more.

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. VTK-m currently supports two basic worklet patterns (with more planned for the future). The first worklet pattern is a simple map from an input array to an output array. The second worklet pattern can specify connections between two arbitrary topological elements (for example from points to cells or from cells to points) to give the worklet access to element-wide data. We have also added a generic cell shape mechanism and basic cell-wise operations such as parametric coordinates, interpolation, and derivatives.

We have also begun to develop filters within VTK-m, which implement specific visualization algorithms using the VTK-m arrays data sets, worklets, and device adapter algorithms. The algorithms currently integrated are cell average, point elevation, Marching Cubes (on regular grids), vertex clustering (for grid decimation), clip, and external faces. Several important algorithms are also in code review including statistics (moments, median, mean, variance, standard deviation, skew, and kurtosis), histogram, and tetrahedralization of both structured and unstructured grids. 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.

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.

The VTK-m development team held a code sprint on September 1-2 at Lawrence Livermore National Laboratory. There were over 25 participants that represented work from many different organizations including national laboratories (SNL, LANL, ORNL, LBNL, LLNL), universities (Oregon, UC Davis), and industry (Kitware, NVIDIA, Intelligent Light). This event allowed us to reach out to several interested developers to get them kick started with VTK-m development and also allowed us to make progress in several key areas of VTK-m and its algorithms.

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).

We consider the current state of the VTK-m software to be a demonstration of a working design and thus a completion of this milestone. However, development of VTK-m will continue throughout XVis and we will continue to report on its progress. We anticipate a prototype integration of VTK-m with VTK and ParaView to begin in FY16 Q1, and we plan to release version 1.0 of VTK-m in FY16 Q2.

Milestone 1.b Array Characterization (Year 2–SNL, Kitware) Automatically characterize how arrays are used and leverage that information to optimize memory hierarchy usage.

Expected Completion: FY16, Q4 Status: Preliminary work

During a design review with NVIDIA engineers we discussed the benefits of using texture memory when accessing global arrays. On NVIDIA hardware, global memory reads must be accessed in 32, 64, or 128 byte transactions. When a warp executes an instruction that uses global memory, the fetches are coalesced into the minimum number of transactions possible. If a warp is well coalesced a single 32, 64, or 128 byte transaction will suffice, otherwise more transactions will occur causing throughput to suffer. Texture memory is global memory backed by the L1 texture cache, allowing for higher throughput when there is 2D locality of the memory fetches.

In VTK-m uncoalesced memory access are very common when doing any algorithm that requires two different types of topological information, for example cells, and points or faces and edges. To solve this problem we have implemented custom classes that wrap all memory reads when executing on CUDA. These classes then use the provided CUDA command ldg allowing for texture memory reads from global memory accesses without explicitly constructing texture objects. This has resulted in a ~10% performance increase when executing Cell based algorithm that require Point based global memory reads.

Milestone 1.c Hybrid Parallel (Year 2–LANL) Compare alternative models for the interaction of shared-memory and distributed-memory parallelism within VTK-m.
Expected Completion: FY16, Q4 Status: Preliminary work

We have begun to explore the interplay between shared-memory data-parallelism and inter-node distributed-memory parallelism in the context of an algorithm for computing contour trees (Reeb graphs), which summarize the development of contours in a data set as the isovalue varies, as mentioned with regards to Milestone 1.a. Although topological analysis tools such as the contour tree and Morse-Smale complex are now well established, there is still a shortage of efficient parallel algorithms for their computation, in particular for massively data-parallel computation on a SIMD model. We developed a novel data-parallel algorithm for computing the fully augmented contour tree, using a quantized computation model. We then extended this to provide a hybrid data-parallel / distributed algorithm, allowing scaling beyond a single GPU or CPU, and tested its scaling using Earth elevation data from GTOPO30 across 16 nodes. Our implementation uses the portable data-parallel primitives provided by Nvidia's Thrust library, as well as MPI for inter-node communication.

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: Complete

The framework of the VTK-m data model is now in place. It 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 supports multiple cell sets to allow mixed-topology meshes, meshes multiple coordinate arrays to support meshes that live in multiple coordinate systems simultaneously, and meshes without coordinate systems entirely. These examples were all challenging to represent using traditional data models. It is generally more flexible as well, allowing, for example, hybrid meshes with regular points but unstructured cells, and overall, this can result in greater efficiency. We expect to continue to enhance and refine this data model throughout the project lifetime.

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: Delayed

Milestone 2.b explores architectural features currently beyond the expressivity of VTK-m. For this milestone, three architectural features are to be explored and evaluated, to potentially influence the VTK-m design. We have made excellent progress on our first architectural feature, which explores the performance improvements possible when using different types of memory. We performed this milestone within our ray-tracing code (now ported to VTK-m) and found that accessing GPU-specific memory significantly improved performance. This finding was a contributor in the expansion of VTK-m's improved memory. For our second architectural feature, we are exploring the usage of SSD for post-hoc exploration. A study is underway, which we expect to complete in the next three months. Finally, we decided to focus on deep memory hierarchies for the third architectural feature, and also to delay this study until more architectures are available (i.e., NVLink). This delay is consistent with our under spending and should not interfere with the completion of the project.

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. As a first step towards this goal, we developed the MappedDataArray and MappedDataSet classes, which allow for custom memory layouts. After further evaluation, our conclusion was the overhead introduced by the abstraction used in this approach is too high. We are currently working on a next generation version of this framework that depends on template based polymorphism rather than virtual dispatching. This approach gives us performance that is close to using raw pointers. 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.

We are in the process of developing a prototype that demonstrates how VTK-m can be integrated with VTK and ParaView Catalyst for in situ analysis. This work is done in collaboration with NVIDIA and our aim is to demonstrate it at SC15. We expect that Catalyst will form the foundation of our flyweight in situ framework.

Milestone 2.d Data Model Application (Year 2–ORNL) Explore application of new data models to novel architectures appropriate to in situ.
Expected Completion: FY16, Q3 Status: In progress

We have been exploring the integration of VTK-m in situ with several applications running on DOE LCFs. These have included several fusion codes, and a computational seismology code. Efforts have been focused on understanding the scientific workflows being used by these applications. This better allows us to target machine architectures and analysis products for use in situ to help scientists understand their simulations. Included in these efforts is work being done with XGC, a highly scalable physics code used to study plasmas in fusion tokamak devices. Recent work has explored using light-weight plugins to perform visualization both of the particles and the field variables using ADIOS and DataSpaces.

Milestone 2.e Memory Hierarchy Streaming (Year 2–LANL) Develop streaming out-of-core versions of key visualizations and analysis algorithms to efficiently use deep memory hierarchies within in situ applications.
Expected Completion: FY16, Q4 Status: Preliminary work

We have begun to experiment with the use of the STXXL library from Karlsruhe University for streaming data from disk into main memory and into accelerator memory for isosurface and KD-tree construction algorithms. We have also prototyped the combination of such external-memory algorithms with our distributed wrapper for Thrust as a first step towards enabling these algorithms to operate on data that is both distributed across nodes and too large on each node to fit into memory.

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: Complete

During this period, the UC Davis team continued studies in two areas. In one area, the aim is to evaluate the usability of VTK-m, a fine-grain parallel programming library, for realizing visualization operations. We have implemented a ray casting and cell projection volume renderer in both Dax and VTK-m using data parallel primitives to compare their performance on thee different hardware architectures: NVIDIA Titan X, Intel Xeon E5, and Intel Xeon Phi. Despite the portability provided by these frameworks, we observe that additional architecture specific modifications are necessary to achieve acceptable performance on some architectures. A paper presenting some of our experimental study results and findings will be presented at the Symposium on Visualization in HPC.

In the other area, we are developing new in situ visualization technologies and study their usability. The first technology that we have been developing, which we call Ximage, is based on our former work Explorable Images. We have extended Ximage to support image-space feature extraction and tracking. A paper reporting this work will be presented at LDAV 2015. The other technology we are developing is for supporting the need to study particle and field data together. We have developed a new data framework, which combines both the Eulerian and Lagrangian reference frames into a joint data format. By reorganizing Lagrangian information according to the Eulerian simulation grid into a "unit cell" based approach, we can provide an efficient out-of-core means of sampling, querying, and operating with both representations simultaneously. We also extend this framework to generate multi-resolution subsets of the full data to suit the viewer's needs and provide a fast flow-aware trajectory construction scheme. We are presently studying the effectiveness of this framework.

We will conduct a series of usability studies. One study will be using Ximage. The targeted simulation will be chosen from turbulent combustion, accelerator physics, or climate modeling. The other study will target the need in fusion research to study the behaviors of particles, ion and electrons. We will deploy our prototype libraries and tools into our science collaborator's workflow. The objective is to understand in what setting and to what extent our in situ methods present clear benefits over the conventional methods.

Milestone 3.b Prepare Usability Studies (Year 2–UC Davis, U Oregon) Identify participants and meet to discuss goals for the visualization and analysis tasks. Collect user data sets and design each user study according to code and hardware settings.
Expected Completion: FY16, Q2 Status: In progress

A UO Ph.D. student (James Kress) has relocated to Oak Ridge lab to embed with the XGC team in order to complete this milestone. Our plan is to evaluate their entire visualization & analysis pipeline and determine which pieces can be done in situ, which can be done post hoc, and which can be either way. After spending the summer working with XGC stakeholders (as an ORNL intern and not funded by XVis), James is now engaging with the XGC team and interviewing them and running small studies.

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: In progress, delayed

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.

The initial prototype for the Marching Cubes mini-app was implemented in FY15, Q4. The implementation will be hardened and contributed to Mantevo in FY16, Q1. This will give us an implementation to start working on subsequent milestones. A mini-driver for rendering and a mini-app for particle advection will follow soon after.

Milestone 4.b Validate Mini-App Characteristics (Year 2-ORNL) Validate behavior and resource usage of mini-applications against that of real applications and generate performance/resource models.
Expected Completion: FY16, Q4 Status: Preliminary work

In preparation for proxy analysis, 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, Catalyst and VTK-m,” Marcus Hanwell and Robert Maynard, GTC Lab, March 2015.

“VTK-m,” Kenneth Moreland, DOECGF, April 2015.

“VTK-m: Accelerating the Visualization Toolkit for Multi-core and Many-core Architectures,” Christopher Sewell, et al., SciDAC PI Meeting (poster), July 2015.

“Trends and Advanced Concepts for Scientific Visualization,” Kwan-Liu Ma, Keynote speech, China Scientific Data Conference, August 26, 2015.

“VTK-m Overview,” Kenneth Moreland, VTK-m Code Sprint, September 1, 2015.

“New Techniques for Visualizing Large-Scale Scientific Data,” Kwan-Liu Ma, Invited talk, Software Center for High Performance Numerical Simulation, Chinese Academy of Engineering Physics, Beijing, China, September 2, 2015.

“VTK-m,” Jeremy Meredith, FASTMath PI Meeting, September 2015.

“Advanced Concepts and Strategies for Visualizing Large-Scale, Complex Simulation Data,” Kwan-Liu Ma, Invited Talk, International Computational Accelerator Physics Conference (ICAP), October 14, 2015.

"Visualization and High Performance Computing," Kwan-Liu Ma, Keynote speech, Symposium on Visualization in HPC, SIGGRAPH Asia, November 2, 2015.

The 10th Workshop on Ultrascale Visualization, Workshop Co-Chair, Kwan-Liu Ma, SC15, November 16, 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.

U.S. DEPARTMENT OF ENERGY Sandia National Laboratories

SAND 2015-10879 R