SANDIA REPORT

SAND2019-9297
Unlimited Release
Printed August 2019

XVis: Visualization for the Extreme-Scale Scientific Computation Ecosystem, Final Report

Kenneth Moreland, David Pugmire, David Rogers, Hank Childs, Kwan-Liu Ma, and Berk Geveci

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.osti.gov/bridge

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2019-9297
Unlimited Release
Printed August 2019

XVis: Visualization for the Extreme-Scale Scientific Computation Ecosystem, Final Report

Kenneth Moreland
Sandia National Laboratories
kmorel@sandia.gov

David Pugmire
Oak Ridge National Laboratory
pugmire@ornl.gov

David Rogers
Los Alamos National Laboratory
dhr@lanl.gov

Hank Childs
University of Oregon
hank@cs.uoregon.edu

Kwan-Liu Ma
University of California at Davis
ma@cs.ucdavis.edu

Berk Geveci
Kitware, Inc.
berk.geveci@kitware.com

Abstract

Scientific computing is 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. The XVis 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. This report reviews the accomplishments of the XVis project to prepare scientific visualization for Exascale computing.

Acknowledgment

We would especially like to thank Lucy Nowell for her support and funding. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award Number 14-017566.

Thanks to Jeremy Meredith and Christopher Sewell for their help getting the XVis project started. Also thanks to Robert Maynard and Matthew Larsen for their extensive technical contributions.

Contents

List of Figures

List of Tables

Summary

The project *XVis: Visualization for the Extreme-Scale Scientific Computation Ecosystem*, or simply *XVis* for short, was a project funded by the Office of Advanced Scientific Computing Research in the DOE Office of Science. The project ran from October 2014 to September 2017 (with some no-cost extensions for some groups). XVis was a collaboration between Sandia National Laboratories, Oak Ridge National Laboratory, Los Alamos National Laboratory, Kitware, Inc., the University of California at Davis, and the University of Oregon.

The primary purpose of the XVis project was to provide fundamental research and development to enable scientific visualization at the Exascale. Scientific visualization is a critical component for enabling scientific discovery through computation. This lofty goal is well beyond the scope of any single project. Thus, the XVis project focused on changes in the Exascale supercomputer "ecosystem." That is, how different software and hardware components of the HPC system must change their nature, behavior, and, most importantly, their relationship with each other to perform efficiently on what we expect an Exascale machine to look like. The XVis project approached this problem by addressing four interlocking challenges: emerging processor technology, *in situ* integration, usability, and proxy analysis. This executive summary briefly reviews the results achieved for each of these thrusts.

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 hyper-threading and vector operations require even more parallel processing to leverage each core's full potential.

This change in the basic nature of processor technology for HPC systems left scientific visualization software in a precarious state. The parallel code designed to be highly scalable in distributed memory systems simply will not perform well with the fine, lighter-weight threading paradigm introduced in new processors.

The XVis project developed techniques to enable scientific visualization on advanced processors and accelerators. Although parallel algorithm design was a significant part of this effort, the main thrust of the work was in providing more abstract building blocks that could be applied to numerous scientific visualization features. The main approach used was the engagement of *data parallel primitives*, which are a small set of basic operations (such as prefix sum, sort, and reduce) that are optimized to run in parallel. From these data parallel primitives we built higher level constructs commonly encountered in scientific visualization algorithms such as topology representations, permutation of elements, scheduling of connected structures, and finding coincident elements.

This research in parallel algorithm techniques was encapsulated in a software framework named VTK-m. The XVis project was careful to use well-established collaborative software development methods for VTK-m. Today, VTK-m has become a critical component in DOE's scientific visualization software stack. It plays a predominant role in ECP (`https://www.exascaleproject.org/`), a large DOE project to get to Exascale computing.

In Situ Integration

Although the overall computation rate of HPC systems continues to increase at an exponential rate, the bandwidth and capacity of disk storage systems increases at a much more moderate rate. Consequently, there is a growing divergence between the amount of data that is generated from computation and the amount of that data that can be captured on the storage system. At the Petascale, it is becoming the case for many existing runs that it is not possible to save all the data required for a proper analysis, and this problem will get much worse at the Exascale.

The way around this problem is to incorporate *in situ* visualization, which processes data while it still resides in a running simulations memory without the need to push that data to a disk storage system. We see this technology as critical at the Exascale and spent much time in XVis addressing the problem. However, there are many issues that make *in situ* visualization challenging [22].

To make *in situ* visualization more effective, XVis investigated ways in which we could reduce the footprint of the visualization code and better integrate with the simulation. Together, we consider these features *flyweight in situ* techniques.

One important point that XVis addressed was the need in a tightly coupled visualization to be able to directly access simulation data without copying the data from one format to another. This required the visualization code to adapt to different memory layouts. XVis achieved this goal by introducing abstract array interface components that could be customized for different memory layouts (for example an "array of structures" versus a "structure of arrays"). Such an array interface component was built into VTK-m, which used templating parameters to allow the compiler to automatically adjust an algorithm's code to use customized data structures. XVis also introduced similar array concepts into VTK. Using a combination of virtual methods and template-based polymorphism, this allows VTK algorithms to give performance close to that of raw pointers while allowing for dynamic specification of data. These VTK classes can then be leveraged from *in situ* libraries like Libsim and Catalyst that use VTK for their algorithm implementations.

XVis also partnered with several other projects to deliver *in situ* implementations to multiple frameworks and science applications [2]. XVis helped integrate VTK-m into computational science workflows implemented in projects like Legion, SENSEI, and ALPINE. XVis also provided direct *in situ* support for applications in combustion, cosmology, and fusion.

Usability

A significant disadvantage of using a workflow that integrates simulation with visualization is that a great deal of exploratory interaction is lost. All visualization parameters must be established before the simulation starts or, at best, the user can explore transient data that changes with the state of the simulation. Little is known about how these limitations affect usability or a scientist's ability to form insight.

The XVis project investigated usability of visualization with a focus on maintaining usability when applying *in situ* visualization techniques. The XVis team approached the problem by first collaborating with application scientists to understand their needs. The team then designed customized algorithms to extract useful information that required less disk storage but allowed post hoc analysis. The teams then worked with application scientists to gauge the effectiveness of the new *in situ* technique.

Combustion Working with the S3D combustion code and Jacqueline Chen's research group at Sandia National Laboratories, the XVis team developed specialized techniques to characterize groups of data with probability distribution functions. Using a comprehensive visualization tool designed by XVis, combustion scientists are able to explore these probability distribution functions, quickly find distributions relevant to physical phenomena, and filter based on this information.

Cosmology Working with Salman Habib's research group at Argonne National Laboratory, the XVis team developed and deployed tools to make effective use of parallel GPU clusters to provide interactive visualization of merger tree and particle data. This work included improving the scalability of halo tag processing, which is critical for interactive analysis of halo structure.

Fusion Working closely with the XGC team at Oak Ridge National Laboratory, the XVis team analyzed the data needs of the application scientists and explored ways to capture data. Focusing on the XGC1 and Xolotl codes, XVis applied the aforementioned flyweight *in situ* techniques to integrate VTK-m code into their workflow. This collaboration lead to a new binning technique that was able to capture data from millions of particles in a grid structure with more than 80% reduction of data while incurring less than 1% error.

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 in 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 address this issue, XVis engaged in employing proxy analysis to provide simplified tools for assessing performance on HPC systems.

The XVis project developed two visualization "mini-apps" that can be used as examples for system performance. The first mini-app, miniIsosurface, demonstrates the creation of contours from three dimensional volumes. Contouring is one of the fundamental operations of scientific visualization. There are many contouring algorithms and miniIsosurface encapsulates many of them. The second mini-app, miniGraphics, demonstrates parallel rendering. Rendering is, naturally, fundamental for visualization, and HPC visualization demands a parallel rendering approach. miniGraphics encapsulates many of these parallel rendering algorithms.

During the course of XVis, independent projects also worked on similar small visualization systems. One such example is a simple *in situ* rendering library named Ascent (formally Strawman), which has since become part of ECP. Ascent was also well aligned with XVis' proxy application goals, and hence was used for multiple measurement purposes.

Chapter 1

Introduction

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.

Chapter 2

Progress

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.

2.1 Emerging Processors

Milestone 1.a, Initial VTK-m Design Provide the research and design for VTK-m functional operation and, in conjunction with SDAV, develop an initial implementation.

The VTK-m prototype [26] 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 remained a consistent theme throughout the project. To enable broad sharing of the code, VTK-m is released with an open BSD 3-clause license.

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 branchy development workflow for coordinating concurrent contributions (`https://gitlab.kitware.com/vtk/vtk-m/blob/master/CONTRIBUTING.md`), a set of coding conventions (`https://gitlab.kitware.com/vtk/vtk-m/blob/master/docs/CodingConventions.md`), and a large set of regression tests run nightly (reported at `https://open.cdash.org/index.php?project=VTKM`). We are also making use of the GitLab issue tracker, which helps both with tracking desired changes and capturing design decisions.

The basic foundations for VTK-m were completed in the first year. These include the build system, package structure, and fundamental classes. VTK-m now includes a generic device adapter that implements the basic data parallel primitives and provides performance portability [24]. XVis provided 3 implementations for this generic device adapter: 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.

XVis implemented the initial mechanism for building and executing worklets. The mechanism is flexible in that it is straightforward to define new worklet algorithms, new worklet types, and new data handling mechanisms. XVis implemented two basic worklet patterns. 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.

Through XVis we designed and implemented the original “filter” interface in VTK-m. This is a high-level API to run algorithms in VTK-m. It allows users of the toolkit to create and manipulate data without having to learn the details of data structures and worklet invocations. XVis provided the initial implementation of several filters in VTK-m.

- cell average (point to cell)
- point average (cell to point)
- point elevation

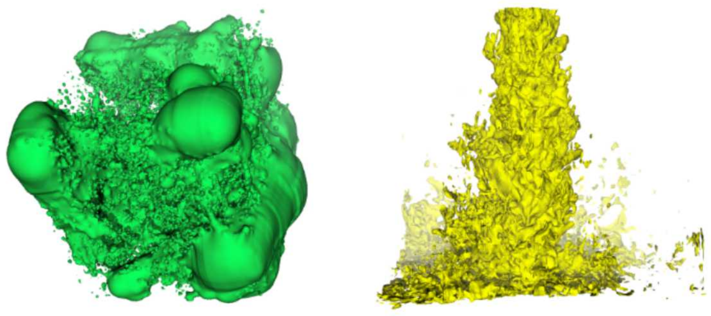


Figure 2.1. Examples of the output from the external faces algorithm.

- Marching Cubes (on hexahedral meshes)
- vertex clustering (for grid decimation)
- clip
- external faces [14, 15]
- histogram

Additionally, experiments with using the VTK-m framework to implement other algorithms, some of which are outside the strict scientific visualization domain, were made — cliques [17] and computer vision [16].

XVis implemented a lightweight rendering infrastructure in VTK-m. Although limited in functionality, VTK-m's rendering library can be used without depending on external features. The rendering infrastructure supports a basic but full set of rendering features including cameras, windows, basic scene graphs, and annotations such as axes and colorbars. In this case, the render is based wholly on parallel algorithms designed within VTK-m itself. This rendering library makes it much easier to support *in situ* rendering [9]. It also provides a good use case for comparing a generic algorithm written in VTK-m with heavily optimize rendering from other libraries. Figure 2.2 provides an example rendering from VTK-m's rendering library.

XVis held several code sprints for the VTK-m software. The first code sprint was hosted at Lawrence Livermore National Laboratory on September 1–2, 2015. 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). The second code sprint was hosted at Kitware, Inc. on August 2–3, 2016, and the third code sprint was hosted at the University of Oregon on April 4–6, 2017. These events

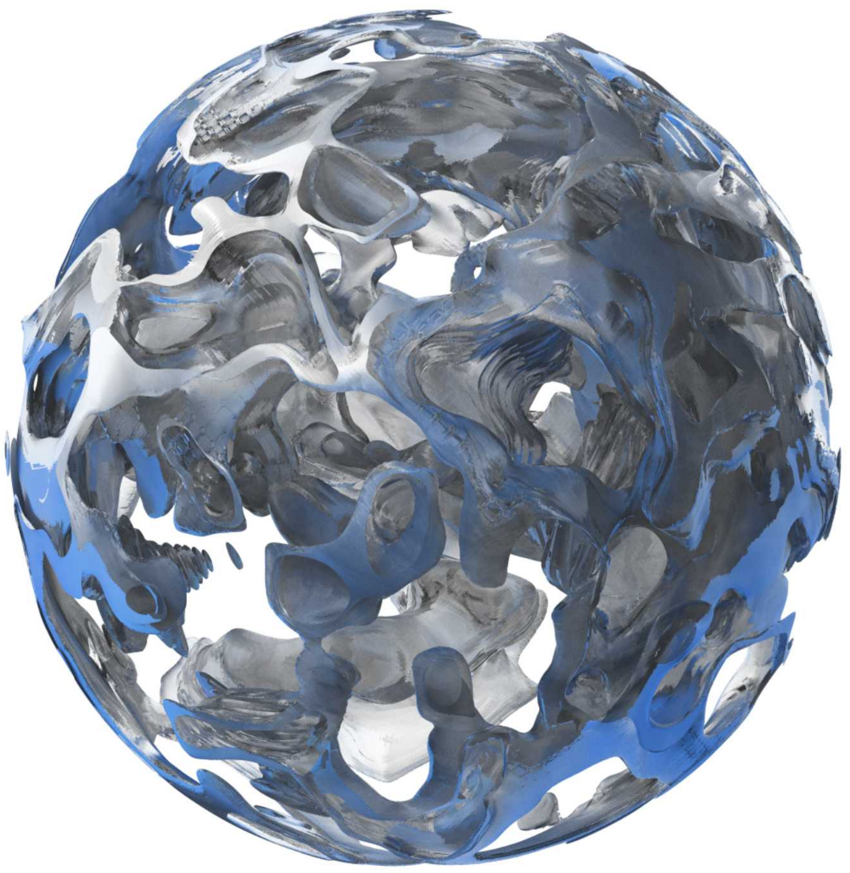


Figure 2.2. An example rendering from VTK-m's ray-casting library.

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.

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 [23].

As the XVis project ended, the DOE's ECP (`https://www.exascaleproject.org/`) started. VTK-m has become a large part of ECP. A project under ECP is dedicated to increase the functionality and support VTK-m while several other projects are relying on VTK-m to provide visualization services. We consider this one of the larger successes of the XVis project.

Milestone 1.b Array Characterization Automatically characterize how arrays are used and leverage that information to optimize memory hierarchy usage.

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 about 10% performance increase when executing Cell based algorithm that require Point based global memory reads.

We have also studied the performance on Intel-based architecture with respect to Array-of-Structures (AoS) versus Structure-of-Arrays (SoA). A typical array of vectors in VTK-m is stored as an array with each component containing a Vec structure (an AoS in this context). We found that this AoS could sometimes inhibit the engagement of vector operations on x86 processors. However, we also found that when coupled with loop tiling (investigated during Milestone 1.e), we could efficiently copy blocks of the AoS to an SoA efficiently. The improved caching and vectorization hide the overhead of the copy.

Milestone 1.c Hybrid Parallel Compare alternative models for the interaction of shared-memory and distributed-memory parallelism within VTK-m.

In collaboration with Hamish Carr from the University of Leeds, we explored the interaction between shared-memory data-parallelism and inter-node distributed-memory parallelism in the context of an algorithm for computing contour trees (Reeb graphs). Contour trees encode the topological changes that occur to the contour as the isovalue ranges between its minimum and maximum values. They can be used to identify the most "important" isovalues in a data set according to various metrics (e.g., persistence). 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 [5]. The original implementation used the portable data-parallel primitives provided by NVIDIA's Thrust library, as well as MPI for inter-node communication. This implementation was later ported to VTK-m.

The XVis project also collaborated with the project "A Unified Data-Driven Approach for Programming *In Situ* Analysis and Visualization," led by Pat McCormick, to evaluate the performance of the prototype integration of VTK-m with their prototype of Legion that they developed. Legion is a task-parallel runtime that can schedule tasks using a task graph based on the dependencies between the tasks. This is an alternative to the traditional bulk-synchronous MPI model, as used by VTK. We successfully compiled the code produced by McCormick's project on the Moonlight supercomputer at Los Alamos, using GASNet, OSMesa, VTK-m, and Legion.

Milestone 1.d Additional Algorithms Develop algorithms for additional visualization and analysis filters in order to expand the functionality of the VTK-m toolkit to support less critical but commonly used operators.

In collaboration with Hamish Carr from the University of Leeds and Gunther Weber from LBNL, XVis helped develop and implement a new data-parallel contour tree algorithm that does not quantize the contour values, allowing for more precise results and less memory usage. Our shared SMP algorithm for parallel contour tree computation has formal guarantees of $O(\log(n)\log(t))$ parallel steps and $O(n\log(n))$ work. It employs "parallel peak pruning," in which superarcs are created in the join tree by identifying peaks and finding their governing saddles, and recursively pruning the regions for each peak/saddle pair as shown in Figure 2.3. We implemented this algorithm natively in OpenMP and with the Thrust library, achieving up to 10x parallel speed up on multi-core CPUs and up to 50x speed up on NVIDIA GPUs compared to the serial version [6]. Performance was compared to the serial sweep-and-merge algorithm published by Carr in 2003, which was also used to verify the results. XVis also took the algorithm developed in the collaboration and converted it to VTK-m structures. The algorithm was subsequently released in VTK-m and demonstrated in numerous applications as shown in Figure 2.4.

Samuel Li, a graduate student working with Chris Sewell at Los Alamos over the summer of 2015, implemented 1D and 2D wavelet transformation worklets in VTK-m. A set of worklets are used to support the wavelet calculation, including boundary handling, decomposition of signals into approximation and detail coefficients, and reconstruction of the original signal from the wavelet coefficients. Coefficients resulting from the wavelet transforms can be used for compression. The simplest strategy is to threshold small coefficients, which has been implemented and achieves fairly good compression. These calculations can be applied recursively to the output to obtain better compression. This implementation of wavelet transforms uses filter banks, which has the advantage of flexibility to support more wavelet filters. Currently it supports four widely used wavelet filters for compression: CDF 9/7, CDF 8/4, CDF 5/3, and HAAR.

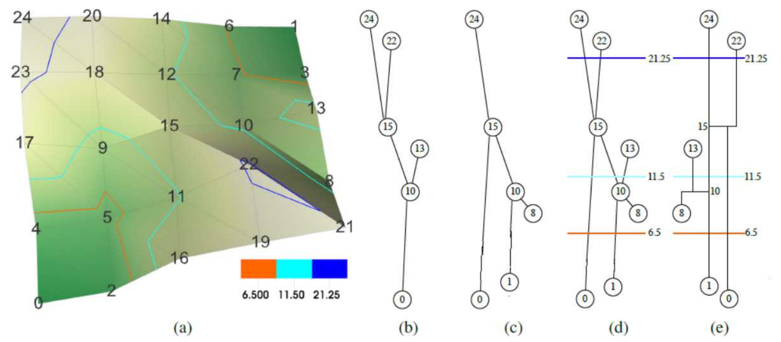


Figure 2.3. An example of performing parallel peak pruning on a simple terrain. From the terrain (a), a join tree (b) and a split tree (c) record the relationship between maxima and minima. The contour tree (d) combines both to obtain the connectivity of contours. Branch decomposition (e) orders contour tree features hierarchically based on importance.

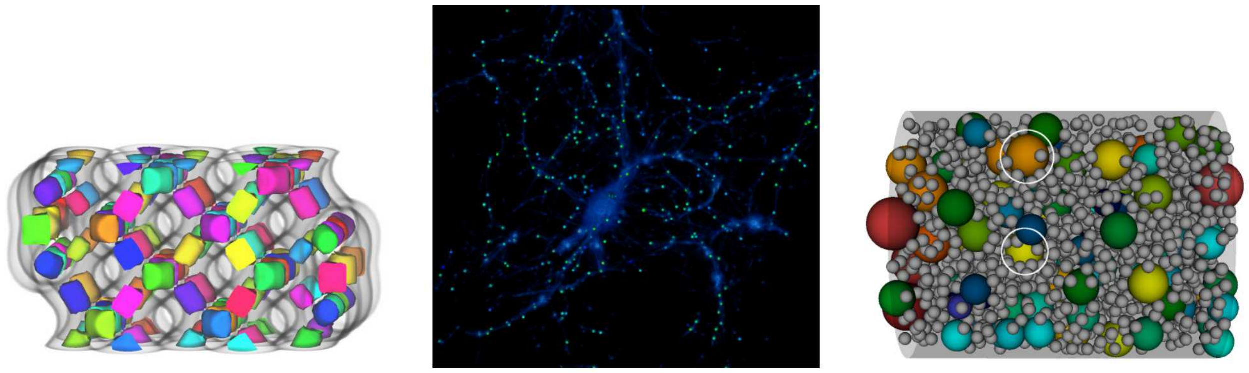


Figure 2.4. Examples use cases of parallel peak pruning to find contour trees: identifying individual atoms in a molecular simulation (left), finding halos in a cosmology simulation (center), and finding pores and pockets in a porous material (right).

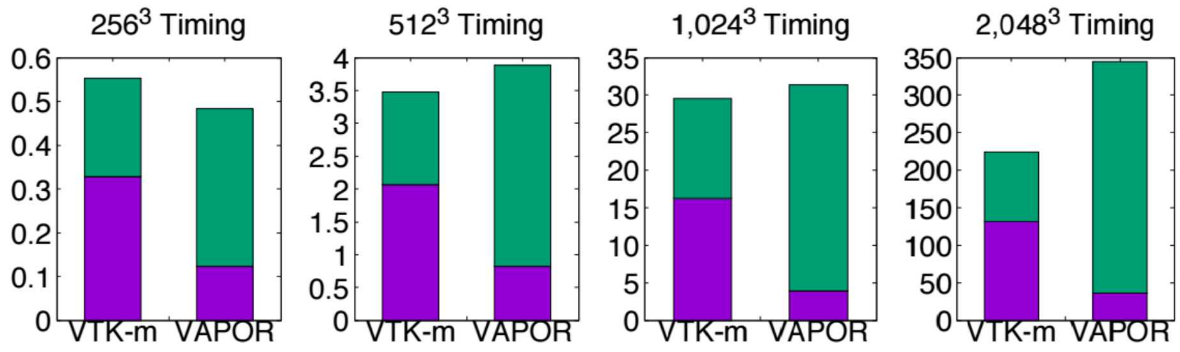


Figure 2.5. Performance of wavelet compression (in seconds) for VTK-m and for a CPU-specific implementation (VAPOR) for data sets ranging from 256^3 to 2048^3.

Samuel Li minimized the number of data transfers and copies required by the algorithm. He compared the performance and accuracy of the data-parallel VTK-m algorithm to domain decomposition parallelization strategies as used by the wavelet compression algorithms in VAPOR and found that the two implementations have similar overall performance characteristics even though that have slightly different approaches to the algorithm [19] as shown in Figure 2.5. The VTK-m implementation also allows these same worklets to run on multiple architectures, including GPUs.

Milestone 1.e Function Characterization Design methods to characterize how functions behave and leverage this information for heterogeneous architectures.

XVis investigated how different functional structures effect vectorization on Intel-based architecture. We found that for a surprising number of function calling parameters, a technique called loop tiling can have dramatic impact on the efficiency of the execution and the effectiveness of vectorization. In loop tiling for loops that iterate an operation over some set of arrays is broken into a nested inner loop of fixed size (say 1024 items). This minor change can have surprising effect on both the compiler and the hardware (caching and vectorization) units. Additionally, in some circumstances we found that while engaging loop tiling we could transform the structure of data (e.g. from AoS to SoA) without incurring a performance penalty.

This experiment was followed up by exploring the idea of adding this loop tiling to the scheduling within VTK-m. We verified that loop tiling within the worklet scheduling within VTK-m can indeed improve performance, and the optimal tiling includes an inner loop of 512 to 1024 items. We are currently working on using this loop tiling and block buffering to try to optimize he vectorization that compilers can achieve. We are integrating this work into VTK-m in such a way that none of the software that it affects needs to be changed and are in the process of widening the performance tests.

XVis also researched how to design efficient polymorphic runtime classes that work across numerous accelerator systems. This has been driven by the need to have algorithms whose components can be switched at runtime such as which integrator to use for streamlines or what function to clip by. The introduction of runtime polymorphism also offers VTK-m the ability to reduce binary size without compromising on performance. The result of this research has been the discovery and application of a pattern very similar to C function callbacks, that works on all accelerator devices, and doesn't compromise performance. This design has been used to improve the implicit function support in VTK-m, and as part of a redesign of the scheduling infrastructure in VTK-m, which resulted in significant binary size reduction.

2.2 *In Situ* Integration

Milestone 2.a Expand Data Models Expand visualization data models to encompass broader scope from new science domains.

The VTK-m data model is designed to include 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 with 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.

Milestone 2.b Post Hoc Interaction Implement three algorithms that use extreme-scale features such as non-volatile memory or knowledge of communication efficiencies.

The first algorithm XVis investigated involves using SSDs to do more effective compression by considering temporal compression. SSDs are appearing increasingly often on leading-edge supercomputers. Following the "*in situ* reduction+post hoc" paradigm in collaboration with Hank Childs' Early Career award, we wanted to explore the opportunities available from having significantly more memory for storing data. In particular, using that memory to store multiple time slices and then compressing the data to take advantage of temporal coherence. Our experiments specifically focused on wavelet compression. While wavelet compression typically operates on one time slice at a time (3D data), our study also included multiple time slices (4D data). Our findings showed that the 4D approach could take advantage of temporal coherency, and, for all metrics studied, the benefits were approximately a factor of two improvement [20].

The second algorithm XVis investigated 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. This work concluded that significant improvements can be made on GPU's when accessing GPU-specific memory (such as texture memory). This finding was a contributor in the expansions of VTK-m's memory management.

The third algorithm XVis investigated was particle advection. This algorithm particularly plays into the post hoc interaction paradigm via extraction of Lagrangian flows, which depends on particle advection. This in situ reduction operator was a key finding of Childs' Early Career work. Particle advection is particularly a challenging algorithm with respect to communication, since it requires hybrid parallelism, i.e., both paralelism within a node (via VTK-m) and parallelism across nodes (via MPI). Prior to XVis, Childs did significant work on considering hybrid parallel solutions for particle advection [4, 7]. The key outcome from this work was to consider how to do a portably performant algorithm. This occurred due to a collaboration between Pugmire, Childs, and others, which was published at EGPGV [31]. (This work is referenced again in Milestone 4.c.)

Milestone 2.c Flyweight *In Situ* Provide flyweight *in situ* visualization techniques into a feature-rich, general-purpose library.

XVis investigated 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 developed and integrated the next generation version of this framework including the classes AOSDataArrayTemplate and SOADataArrayTemplate into VTK. This design depends on template-based polymorphism, and gives us performance that is close to using raw pointers while attaining the objective of allowing tight coupling of VTK *in situ* with simulations. The updated design allows for things such as constant value arrays, implicit point arrays, and other efficient data model concepts that VTK-m also has.

XVis also demonstrated the integration of VTK-m into the SENSEI and ALPINE *in situ* frameworks. Both of these frameworks aim to provide general purpose lightweight *in situ* capability. SENSEI aims to facilitate the instrumentation of simulation codes for *in situ* processes while supporting a diverse set of frameworks including Catalyst [1], Libsim and ADIOS. In addition, it provides an interface that can be

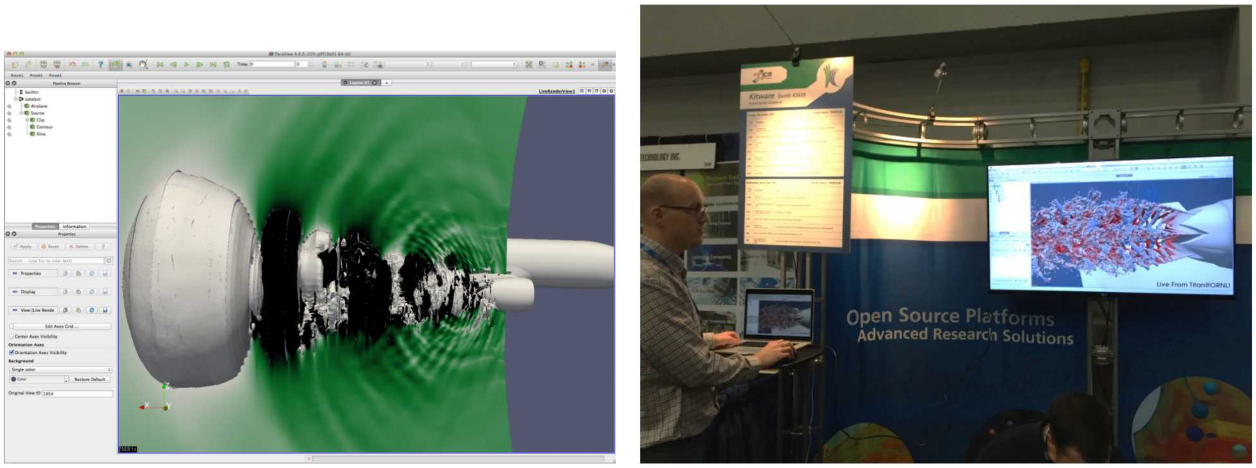


Figure 2.6. Demonstration of live visualization and control of a simulation running on Titan. At left is the output of the visualization on the client in ParaView. At right is a picture of the live demostration given on the show floor at SC15.

integrated with VTK-m for a lightweight *in situ* solution. It has been demonstrated to scale to thousands of MPI ranks.

As part of our collaboration with NVIDIA we developed a prototype that demonstrated VTK-m integration inside VTK and ParaView Catalyst for *in situ* analysis at Supercomputing 2015 shown in Figure 2.6. The prototype used the PyFr simulation running on 256 nodes of Titan. The entire operation from simulation computation, visualization algorithms such as IsoSurface, to rendering happened completely on the GPUs and required no memory copies of PyFr simulation data.

This work was later scaled up to a signficant portion of Titan using 5000 GPUs [38]. To aid the researchers in producing the most relevant images we designed a in situ pipeline that allowed for the simulation to dictate where to place any number of cameras, and what visualization algorithms should those cameras capture. The production run used multiple cameras tracking different areas of the simulation with each camera rotating through different visualization algorithms. Figure 2.7 shows some of the visualization captured from these large-scale runs.

Milestone 2.d Data Model Application Explore application of new data models to novel architectures appropriate to *in situ*.

XVis explored 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

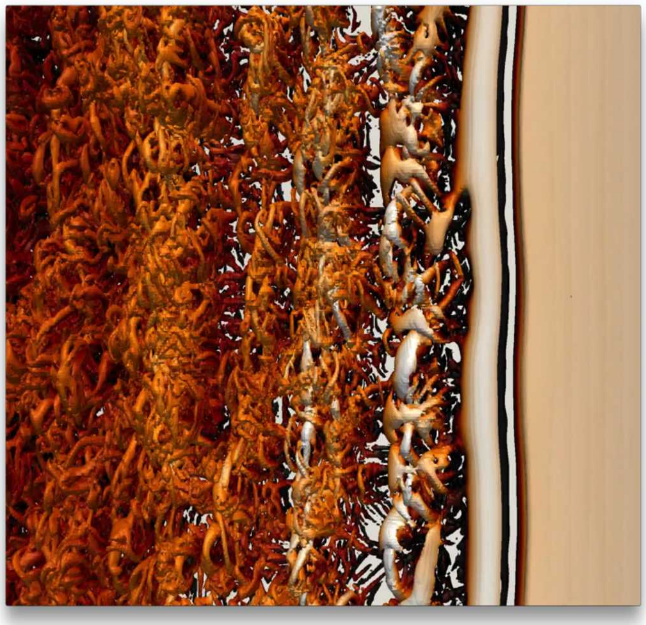


Figure 2.7. Flow over a Low Pressure Turbine Linear Cascade simulation by PyFR, visualized with VTK-m, Paraview, and Catalyst. The image shows the isosurface of the pressure gradient.

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. We have focused these efforts on two SciDAC fusion simulation codes: XGC1 and Xolotl. XGC1 is a particle in cell code used to study plasmas in fusion tokamak devices, particularly in the edge region. Xolotl is a new code that is being used to study the interaction with the tokamak wall.

VTK-m was used for test and production runs of Xolotl for *in situ* monitoring and visualization. We also explored the use of light-weight visualization services to do analysis and visualization for XGC1 in an in transit setting [11, 12]. We used the ADIOS middleware system along with the DataSpaces and DIMES transport methods for data movement. The VTK-m services include the calculation of blobby turbulence around the edge and bulk velocity derived from the particle data. In this later example, we are tracking individual particle paths to derive a time varying vector field. This vector field, which is a significantly reduced representation of the particles makes it possible for much more frequent output for post-hoc analysis and visualization.

Milestone 2.e Memory Hierarchy Streaming Develop streaming out-of-core versions of key visualizations and analysis algorithms to efficiently use deep memory hierarchies within *in situ* applications.

XVis experimented 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.

XVis then implemented basic support for streaming data through the GPU directly in VTK-m. A custom array handle class, ArrayHandleStreaming, wraps a standard array handle and provides an interface of one block at a time. A custom dispatcher, DispatcherStreamingMapField, operates similarly to the standard DispatcherMapField. It uses the function interface infrastructure to iterate through all input array handles to transfer data from the control to the execution environment. When it does this, it also wraps all input and output array handles with an ArrayHandleStreaming class and then loops through the desired number of blocks by appropriately setting the parameters of the ArrayHandleStreaming class and calling the scheduler. Since all the output data cannot reside in the execution environment memory simultaneously, it also loops through the output array handles at the end of the iteration for each block to synchronize the execution environment with the control environment. These custom array handle and dispatcher classes allow a VTK-m developer to easily run basic field map worklets when all the data will not fit on the GPU. We have also implemented a StreamingScanInclusive method in the DeviceAdapterAlgorithmGeneral class, which wraps the input and output array handles with an ArrayHandleStreaming and iterates through each block, applying the last result from the previous block to the first element of the next block, and performing the inclusive scan within each block by calling the ScanInclusive method for the active device adapter.

In follow-on work Chris Sewell and Li-Ta Lo at LANL in collaboration with Tom Fogel from NVIDIA have implemented Unified Memory support for NVIDIA's Pascal GPUs. This enables VTK-m to seamlessly process datasets larger than GPU memory. We explored several different memory cache policies for performance tuning. A preliminary performance benchmark shows a range of 1x–4x performance impact depending on the complexity of the visualization operation.

Milestone 2.f Interface for Post Hoc Interaction Define an abstract VTK-m interface for extreme-scale, implement a prototype for the interface, and exercise the prototype with the algorithms chosen for Milestone 2.b.

The algorithms chosen for Milestone 2.b were wavelet compression, rendering, and Lagrangian flow. In each case, it turned out that no post hoc interaction interface was needed. In general, the issue is that the in situ reduction phase is producing a much smaller data set, and so the extracts produced by VTK-m can be accessible without significant computational power (and thus is accessible from existing tools). This finding, while obvious in retrospect, came about by implementing VTK-m reduction operations, and then seeing the resulting extracts be used by existing projects. Consider the three algorithms. For wavelet compression, the result is a hierarchical data set on disk. These data sets are readily processed by existing wavelet-based software, such as VAPOR, who are careful to limit the amount of data being processed at any given time (and thus have less of a need to maximize computational power). For rendering, the result is an image. In particular, our efforts through this project led to VTK-m having a very performant renderer, which was subsequently incorporated into the Ascent in situ library (the deployment vehicle for ECP), and became the basis of Ascent's routines to generate Cinema databases. Cinema, the preferred technology for consuming image databases, is also not computationally bounded — the work is in generating the images in situ, but not in post hoc reconstruction. Finally, for Lagrangian flow, the work is again in producing the extract, since the post hoc interaction with Lagrangian basis flows is mostly a search problem (finding the right basis flows) as opposed to a computational problem (interpolating between a handful of basis flows is not computationally expensive).

Milestone 2.g Data Model Array Characterization Extend the data model to support array characterizations from Milestone 1.b for supporting heterogeneity.

We have completed this milestone by having the data model subsume the work done in Milestone 1.b. The array characterizations done in Milestone 1.b were implemented in device specific ways inside the ArrayHandle classes, which is the basis for the data model. These include texture memory support for CUDA and the Array of Structures / Structure of Arrays for the TBB execution environments.

Milestone 2.h Flyweight *In Situ*-VTK-m Integration Fully integrate VTK-m execution and data models (from Milestones 1.a, 2.a, and 2.d) with flyweight *in situ* (from Milestone 2.c) and demonstrate its application.

VTK-m has been integrated into SENSEI. We have developed a contour analysis module in SENSEI and have demonstrated its functionality with the mini apps included in SENSEI. Additional filters can now be exposed in SENSEI by developing simple analysis modules. All of our work has been included in the upstream SENSEI repository.

Milestone 2.i Fault Tolerant Primitive Functions Develop fault tolerant versions of key data-parallel primitives (such as scan, transform, reduce, etc.) used by VTK-m algorithms in order to transparently provide a level of resiliency within *in situ* applications.

The initial prototype was implemented to catch bad_alloc errors and retry with the streaming array handle. This code has been integrated into the Resilient branch of the VTK-m repository. In pursuing this initial prototype, we found that the scope of this milestone was much broader than originally described, and would require more extensive R&D than was originally scoped. Thus, we focused our research efforts on the remaining milestones.

2.3 Usability

Milestone 3.a Develop Techniques to be Studied Identify existing and new visualization techniques to be studied. Revise existing ones and implement new ones as needed.

XVis pursued two independent studies. In one study, we aim to evaluate the usability of VTK-m for realizing visualization operations. We initially implemented two volume rendering algorithms, ray-casting and cell projection, in Dax because VTK-m was not yet ready to support these features [36].

In the second study, XVis developed *in situ* visualization technologies that will be used in our proposed usability studies. 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 [41]. 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 [35]. We are presently studying the effectiveness of this framework.

Milestone 3.b Prepare Usability Studies 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.

With help from the XVis project, a University of Oregon student (James Kress) relocated to Oak Ridge lab to embed with the XGC team in order to complete this milestone. This allowed us to evaluate their entire visualization and analysis pipeline and determine which pieces can be done *in situ*, which can be done post hoc, and which can be either way. James Kress conducted over twenty interviews to understand the XGC team's data needs [12]. This study has included physicists, computational scientists, and computer scientists. Information obtained ranges from predicted data sizes to desired visualization and analysis operations. This information was then used to consider *in situ* feasibility. The desired visualization and analysis operations were grouped into approximately five equivalence classes, and we are doing performance modeling for these classes, grounded by real-world experiments on Oak Ridge's supercomputers.

Milestone 3.c Start Usability Studies Conduct pilot studies with algorithms and participants identified in Milestones 3.a and 3.b.

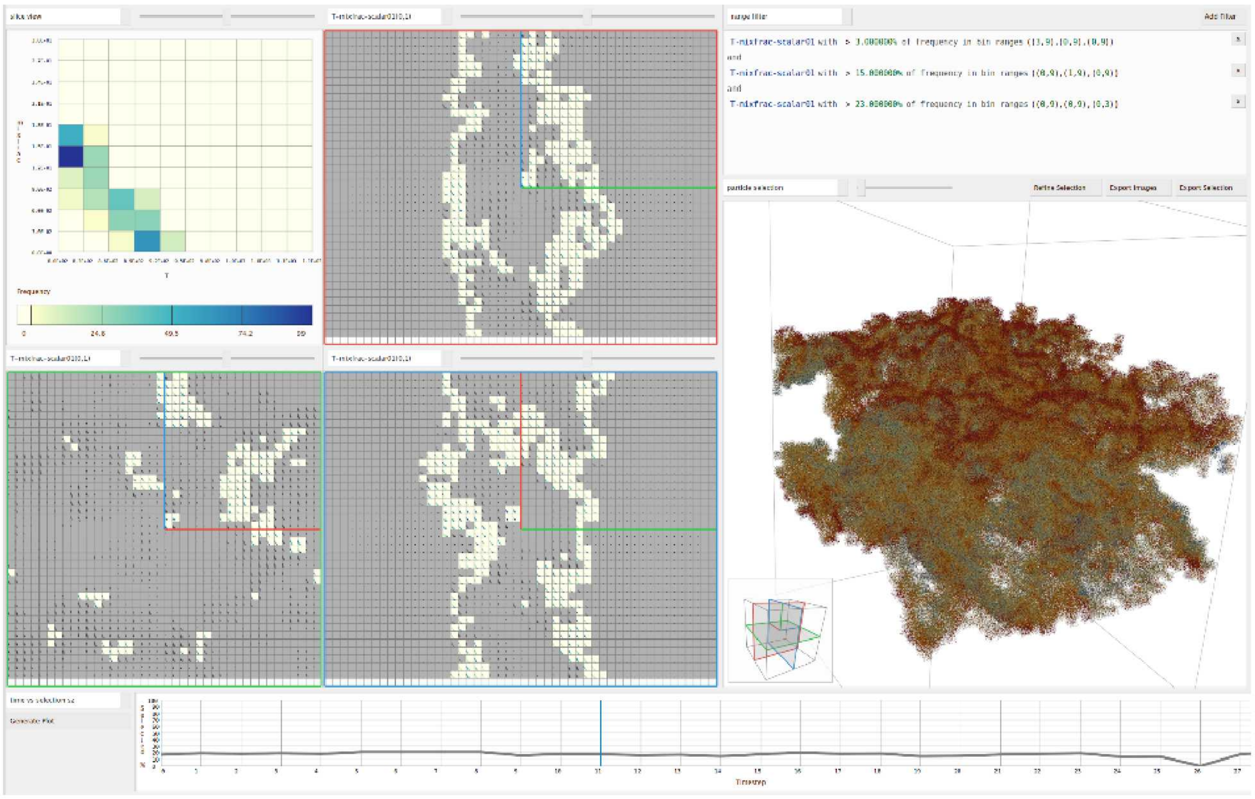
In the area of combustion, XVis worked with Dr. Jackie Chen's research group at Sandia National Labs. We developed analysis and visualization tools to streamline their current workflow and solve new analysis problems as identified by the domain scientists [42]. We completed the development of a scalable histogram-based particle selection scheme which couples both *in situ* and post processing analyses [28].

Spatially distributed histograms (probability distribution functions) are generated *in situ* using a set of modules developed at UC Davis. These probability distributions are generated using the full resolution simulation data and can be leveraged in post processing to aid certain analysis tasks. We have been working on investigating how we can use these modules to best interface with the S3D combustion simulation in an unobtrusive manner. Not only must these modules match the scalability of the simulation, scientists need to be able to easily make modifications to the code based on their most recent needs. We completed scalability tests in large scale production runs.

On the post processing side of things, XVis developed a comprehensive visualization software shown in Figure 2.8 that can utilize the *in situ* generated histograms to make real time particle selections from large scale datasets [40]. Users can select spatial regions based on desired distributions in the histograms which in turn extracts particle subsets (which were spatially sorted *in situ*) for later analysis. We have been making detailed design decisions based on a close collaboration with the domain scientists and their analysis needs. We have designed this software (including UI, data management, etc.) so that it can be easily be used by the scientists with little to no modification to their usual workflow.

In the area of cosmology, XVis worked with Dr. Salman Habib's research group at Argonne National Laboratory to develop and deploy an interactive visualization tool that can effectively use the parallel GPU clusters available to them [29, 30]. We have developed GPU accelerated, parallel rendering methods for interactive visualization of both the particle data and merger tree data [37]. We have also incorporated a few quantitative analysis functionalities [34].

Part of this work included improving the scalability of halo tag processing. Halo tags are critical for interactive analysis of halo structure, but we must first associate



```
T-mixfrac-scalar01 with > 3.000000% of frequency in bin ranges ([3,9],[0,9],(0,9])
and
T-mixfrac-scalar01 with > 15.000000% of frequency in bin ranges [(0,9),(1,9),[0,9)]
and
T-mixfrac-scalar01 with > 23.000000% of frequency in bin ranges [(0,9),(0,9),[0,3)]
```

Figure 2.8. A screenshot of the post hoc visualization tool for visualizing probability density functions (PDFs) generated *in situ*. On the left are three slice views of the spatially distributed PDFs as well as a zoomed in view of a PDF of interest. Selected PDFs are highlighted and are used to query simulation particles as shown on the right.

halo tags with their particles in the 3D spatial domain. We have also improved the handling of temporal data in order to facilitate both command line (server-side) and GUI (client-side) use and have integrated a phase-space mesh rendering method [32]. The first deployment of this interactive visualization facility at Argonne has been completed and is ready for subsequent usability studies to be carried out with full participation of the scientists.

Milestone 3.d Continue Usability Studies Continue studies of Milestone 3.c.

In supporting the combustion simulations, the XVis project has completed initial usability tests for the *in situ* modules. Although our own tests have been successful in showing the scalability of our tools, more evaluation was needed to test the ease at which the domain scientists can invoke or modify our routines as well as their ability to handle the large variety of chemical cases that the S3D simulation can produce.

Feedback from the scientists have identified certain key functionalities that would improve usefulness of the software, such as adding the ability to dynamically modify distribution generation parameters while the simulation is in progress. These new components have been added to the *in situ* modules, which are now fully deployed and integrated within the S3D simulation.

XVis also worked on improving the post processing visualization software based on feedback from the domain scientists. We have completed initial usability tests with our domain collaborators to evaluate the ease at which they can use the software and how successful it is in identifying desired trends in the data. We have identified and carefully documented requested improvements to the user interface, interaction techniques, visual depictions of the results, and analysis functionalities [33].

For the cosmology application, XVis worked on improving the functionality of the recently deployed interactive visualization facility. Recent trips to Argonne National Laboratory to meet with domain experts have provided additional insight into how scientists can effectively use these new tools. We have identified key areas of improvement, which focus on the interface design and the ways users can interact with each of the data views, and have been modifying our software accordingly.

Milestone 3.e Apply Usability Studies Conduct a review session with each scientist team on usability study results from Milestone 3.d. Refine the design and implementation of techniques and workflow according to lessons learned.

In our previous work for this milestone, we interviewed stakeholders from the XGC team about their data needs, and documented the findings in a workshop publication. One key finding from the study was how this team is forced to decide which data to keep and which data to throw away.

This has led XVis to participating in a new algorithm for this team. Specifically, XGC produces significant numbers of particles, each with velocity data. The number of particles is larger than could/should be stored to disk, and so we investigated a method for binning the particles onto a mesh. An important thrust to this research was determining tradeoffs between mesh size, number of particles considered, and the integrity of analyses using the binning (rather than the original data) [10]. Studies shows that the reduced data was able to represent visualizations like Poincarè plots with little error, as shown in Figure 2.9.

For the combustion application, we have applied additional improvements to the post processing visualization software based on feedback from the scientists and our usability tests from the previous milestones. The continuous use of our analysis and visualization tools have validated the effectiveness of using *in situ* generated distribution functions to the scientists. This has now motivated them to request new analysis tools to explore the statistical properties found within the distributions in more detail. While the current tools will continue to be improved in an iterative fashion, these

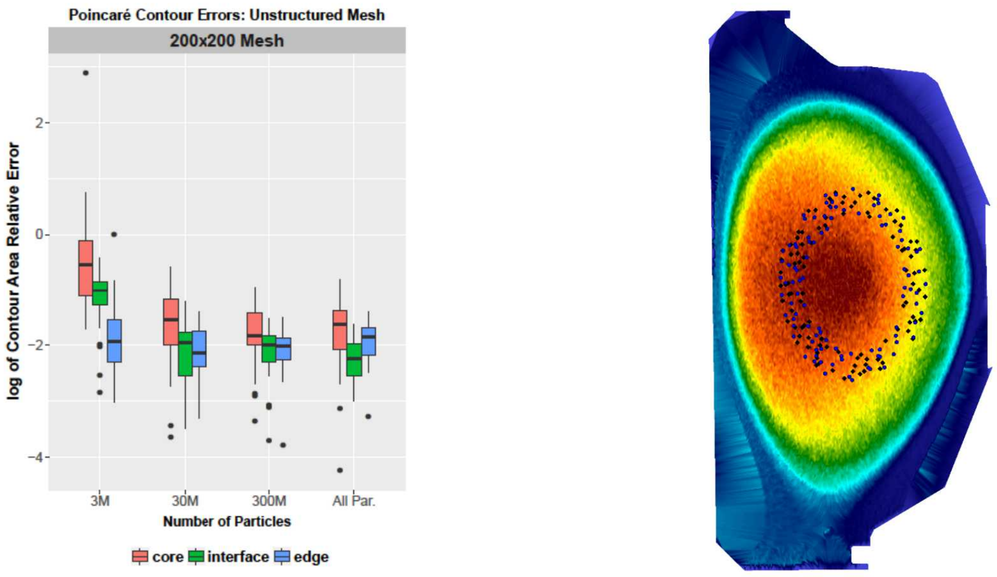


Figure 2.9. At left: Boxplots of the differences in area of a Poincarè plot generated from full resolution particle data and a Poincarè plot generated from the binned vector field data for one studied mesh size. At right: The corresponding Poincarè plot for the test using 300M particles. Reduced data are shown in black and the original resolution data are shown in blue. This test represents an error of approximately 1% represented in 89 MB, reduced from 500 GB of original particle data.

new tools with become a key part of our continued collaboration with this scientific team. We have also found ways to generalize the techniques of building probability density functions and visualizing them post hoc to other science domains [27].

For the cosmology application, we have also been working on additional improvements to the interactive visualization and analysis tools. Recent evaluations of the deployed system's performance, as well as discussions with domain experts, are allowing us to target specific end-user hardware and architecture types to improve the efficiency of our analysis tools. Furthermore, we have received new requests to facilitate communication and data flow between our tools and other software packages that are commonly used by the domain scientists. Our future collaboration with this scientific team will continue to improve the way our tools can seamlessly fit into their normal analysis workflow.

2.4 Proxy Analysis

Milestone 4.a Initial Mini-App Implementation An initial implementation of mini-applications based on visualization and *in situ* workloads.

Working with Intel engineers, XVis implemented two early mini-apps to study the ability to parallelize common visualization algorithms using multithreading and CPU SIMD vector extensions. We implemented multiple structured and unstructured grid contouring (Marching Cubes, and Marching Tetrahedra) algorithm variations to determine what approach performs better.

This original code was eventually reworked into a self-contained contouring mini-app named miniIsosurface [3]. The modifications include a reworking of this original implementation to improve clarity of the code and guarantee correctness while still utilizing full compute resources. miniIsosurface now includes additional parallel implementations of the marching cube algorithm and currently have one serial, three OpenMP, one MPI, and one Kokkos-based implementations. In additional to the Marching Cubes algorithm, we have implemented the Flying Edges algorithm, which tends to outperform the Marching Cubes algorithm. We have developed Flying Edges implementations that runs in serial, runs using OpenMP, and runs on the NVIDIA GPU (using the Thrust library).

Additionally, XVis produced a parallel rendering mini-app named miniGraphics. More specifically, miniGraphics demonstrates sort-last parallel rendering, which is the most common in HPC applications [21]. A sort-last parallel rendering process can be split into two parts: a rendering part that happens locally on a node using data available on that node, and an image compositing part that takes the locally generated images and blends them together. The framework for miniGraphics provides separate implementations for the rendering and the image compositing functions so that they may be mixed to measure different rendering effects. The initial work includes simple reference implementations for both. Future work includes making example rendering that employs OpenGL/GLFW and example compositing using IceT. These implementations provide more industry-standard implementations of rendering and compositing.

XVis also used the Ascent *in situ* visualization framework. Ascent is a lightweight framework which is part of the Alpine project, and is designed to help explore the *in situ* analysis and visualization needs of simulation code teams running on supercomputers. Ascent contains a thin data model that allows a simulation code to provide the semantics for data and then wrap the data in the VTK-m data model. Ascent also ships with several different simulation mini-apps. We have implemented an isosurface functionality in Ascent as well as rendering [13], and run several of the simulation mini-apps as test cases. We have also integrated the ADIOS middleware into the Ascent framework which allows us to experiment with both *in situ*, and in transit scenarios. Using the ADIOS middleware, the mini-apps in can generate self-describing data using the ADIOS Visualization Schema. These data, along with operations to be performed on the data, are written with the ADIOS middleware. We have created several simple visualization services that are based on VTK-m that can ingest the data from the ADIOS stream, and perform the specified operations on the data. These experiments have been performed for *in situ*, in transit and post-hoc scenarios.

Milestone 4.b Validate Mini-App Characteristics Validate behavior and resource usage of mini-applications against that of real applications and generate performance/resource models.

XVis explored the characteristics of the Marching Cubes mini-app (developed as part of Milestone 4.a) with the Oxbow suite. These early investigations show similarities and differences with the EAVL version of the algorithm. For example, in terms of instruction mix both have a high proportion of integer operations (about 40%) and memory operations (about 25%), but algorithmic choices resulted in different tradeoffs between branch operations and memory movement operations. In memory bandwidth behavior, both had similar read bandwidths, but a noticeable difference in write bandwidths. This preliminary data is shown in Table 2.1.

Table 2.1. Performance comparison between mini-app and EAVL (predecessor to VTK-m).

	Instruction Mix (%)				Memory Bandwidth			
	BrOps	IntOps	MemOps	Moves	Read B/Cycle	Read MS/s	Write B/Cycle	Write MS/s
Mini-app	20.35	40.06	27.60	11.01	0.07	240.76	0.02	63.18
EAVL	8.76	38.99	24.62	27.12	0.10	283.92	0.04	109.62

XVis then performed subsequent analysis using tools from the Oxbow suite to compare the marching cubes implementation in both VTK and VTK-m. As shown in Table 2.2, the instruction mix for both the VTK-m and VTK implementations of the marching cubes algorithms are roughly the same.

Table 2.2. Performance comparison between VTK-m and VTK.

Application	Instruction Mix (%)				
	BrOps	IntOps	FpOps	MemOps	Moves
VTK	15.0	35.5	0.3	35.8	13.2
VTK-m	12.9	33.9	.2	38.6	14.1

We have also engaged with the TAU performance tools group at the University of Oregon to study the usage of their suite of tools for performance analysis. The TAU suite of tools has the advantage of a lower-barrier of entry and the ability to obtain performance information from both CPU and GPU codes. We have created

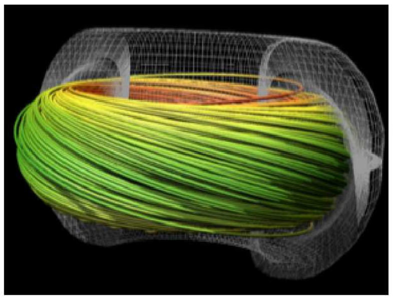


Figure 2.10. Particle advection in fusion simulation: Streamlines generated from the magnetic field in a fusion simulation. The magnetic field is one of the primary drivers in generating fusion in a plasma. Streamlines are a common way of understanding the nature of this magnetic field.

some test applications for performance analysis that include volume rendering, ray casting, marching cubes, streamlines, and external face extraction. Using TAU, we were able to locate a bottleneck in the marching cubes algorithm during a normal calculation step. The insights into the performance of the code using TAU have enabled the VTK-m team to identify a work around for this performance bottleneck.

Milestone 4.c Architectural Studies Use the mini-applications, simulation, and experiments to perform studies of behavior on future architectures, including performance, memory hierarchy usage, and heterogeneous component use.

We are currently studying the performance of the particle advection filter functionality on a variety of different architectures [31] such as that shown in Figure 2.10. The hardware we are targeting include the Titan supercomputer at ORNL, the Rhea analysis cluster at ORNL, and the test bed for Summit at ORNL. We are targeting both CPU and GPU implementations, as well as studying the scalability as a function of number of cores on CPU implementations. We are testing both a particle advection algorithm, which will be used for applications like FTLE, as well as a streamline algorithm where particle paths are recorded.

One focus of our studies is the portability of particle advection algorithms in VTK-m, and how they compare to hand-tuned implementations for both CPU and GPU. In order to study the portability, we have run a large series of tests on the VTK-m implementation (for both CPU and GPU), and a previously published hardware specific implementation (for both CPU and GPU). The tests that we have run include a wide variety of workloads, including both numbers of particles, duration of advection, and types of vector fields. Table 2.3 summarizes our initial findings. Each workload, denoted W_i, is run on a number of different GPUs. The File column denotes the type of vector field, the CUDA Code columns denote the timings for the hardware-specific particle advection implementation, and the VTK-m Comparison columns list the speed-up factors for the VTK-m implementation. A factor of $> 1\times$ indicates that VTK-m is faster.

Table 2.3. Speedup of particle advection in VTK-m on GPUs.

		CUDA Code			VTK-m Comparison		
	File	K20X	K80	P100	K20X	K80	P100
	Astro	0.8s	0.3s	0.8s	$1.4\times$	$0.6\times$	$2.2\times$
W_1	Fusion	0.8s	0.3s	0.8s	$1.4\times$	$0.6\times$	$2.2\times$
	Thermal	0.8s	0.3s	0.8s	$1.4\times$	$0.5\times$	$2.1\times$
	Astro	0.9s	0.3s	0.8s	$1.3\times$	$0.6\times$	$2.1\times$
W_2	Fusion	0.9s	0.3s	0.8s	$1.4\times$	$0.6\times$	$2.1\times$
	Thermal	0.9s	0.3s	0.8s	$1.3\times$	$0.6\times$	$2.1\times$
	Astro	3.4s	2.0s	2.4s	$2.3\times$	$2.1\times$	$4.1\times$
W_3	Fusion	3.4s	1.8s	2.2s	$2.2\times$	$1.9\times$	$3.8\times$
	Thermal	3.3s	1.8s	2.2s	$2.2\times$	$2.0\times$	$3.8\times$
	Astro	6.7s	5.1s	3.6s	$1.3\times$	$1.8\times$	$2.0\times$
W_4	Fusion	8.8s	6.8s	4.4s	$1.6\times$	$2.4\times$	$2.5\times$
	Thermal	8.8s	6.8s	4.5s	$1.7\times$	$2.4\times$	$2.5\times$
	Astro	14.4s	13.0s	6.4s	$0.4\times$	$0.6\times$	$0.5\times$
W_5	Fusion	64.0s	54.7s	25.7s	$1.6\times$	$2.2\times$	$1.9\times$
	Thermal	56.1s	49.2s	23.2s	$1.4\times$	$2.0\times$	$1.7\times$

Chapter 3

Professional Activities

During its 3 years, the XVis project was very active in the research community. This chapter lists the publications and other professional activities performed during this time.

3.1 Publications

The following citations are a comprehensive list of the publications made possible in total or in part because of the XVis project.

[1] **ParaView Catalyst: Enabling In Situ Data Analysis and Visualization**. Utkarsh Ayachit, Andrew Bauer, Berk Geveci, Patrick O'Leary, Kenneth Moreland, Nathan Fabian, and Jeffrey Mauldin. In: *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV 2015)*. Nov. 2015, pp. 25–29. DOI: 10.1145/2828612.2828624.

[2] **In Situ Methods, Infrastructures, and Applications on High Performance Computing Platforms**. Andrew C. Bauer, Hasan Abbasi, James Ahrens, Hank Childs, Berk Geveci, Scott Klasky, Kenneth Moreland, Patrick O'Leary, Venkatram Vishwanath, Brad Whitlock, and E. Wes Bethel. In: *Computer Graphics Forum* 35.3 (June 2016), pp. 577–597. DOI: 10.1111/cgf.12930.

[3] **Isosurface Visualization Miniapplication**. Daniel Bourgeois, Michael Wolf, and Kenneth Moreland. Tech. rep. SAND2018-2780O. Sandia National Laboratories, 2018. URL: https://cfwebprod.sandia.gov/cfdocs/CompResearch/docs/proceedings/ccr17.pdf#page=141.

[4] **GPU Acceleration of Particle Advection Workloads in a Parallel, Distributed Memory Setting**. David Camp, Hari Krishnan, David Pugmire, Christoph Garth, Ian Johnson, E. Wes Bethel, Kenneth I. Joy, and Hank Childs. In: *Proceedings of EuroGraphics Symposium on Parallel Graphics and Visualization (EGPGV)*. Girona, Spain, May 2013, pp. 1–8.

[5] **Hybrid Data-Parallel Contour Tree Computation**. Hamish Carr, Christopher Sewell, Li-Ta Lo, and James Ahrens. In: *Proceedings of the Computer Graphics and Visual Computing Conference*. Sept. 2016. DOI: 10.2312/cgvc.20161299.

[6] **Parallel Peak Pruning for Scalable SMP Contour Tree Computation**. Hamish Carr, Gunther Weber, Christopher Sewell, and James Ahrens. In: *Proceedings of the IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. Oct. 2016. DOI: 10.1109/LDAV.2016.7874312.

[7] **Particle Advection Performance over Varied Architectures and Workloads**. Hank Childs, Scott Biersdorff, David Poliakoff, David Camp, and Allen D. Malony. In: *IEEE International Conference on High Performance Computing (HiPC)*. Goa, India, Dec. 2014, pp. 1–10. DOI: 10.1109/HiPC.2014.7116900.

[8] **The future of scientific workflows**. Ewa Deelman, Tom Peterka, Ilkay Altintas, Christopher D Carothers, Kerstin Kleese van Dam, Kenneth Moreland, Manish Parashar, Lavanya Ramakrishnan, Michela Taufer, and Jeffrey Vetter. In: *International Journal of High Performance Computing Applications* 32.1 (Jan. 2018), pp. 159–175. DOI: 10.1177/1094342017704893.

[9] **In Situ Visualization of Radiation Transport Geometry**. Mark Kim, Tom Evans, Scott Klasky, and David Pugmire. In: *Proceedings of the In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV)*. Nov. 2017, pp. 7–11. DOI: 10.1145/3144769.3144770.

[10] **Binning Based Data Reduction for Vector Field Data of a Particle-In-Cell Fusion Simulation**. James Kress, Jong Choi, Scott Klasky, Michael Churchill, Hank Childs, and David Pugmire. In: *ISC Workshop on In Situ Visualization (WOIV)*. June 2018.

[11] **Preparing for In Situ Processing on Upcoming Leading-edge Supercomputers**. James Kress, Randy Michael Churchill, Scott Klasky, Mark Kim, Hank Childs, and David Pugmire. In: *Supercomputing Frontiers and Innovations* 3.4 (2016). DOI: 10.14529/jsfi160404.

[12] **Visualization and Analysis Requirements for In Situ Processing for a Large-Scale Fusion Simulation Code**. James Kress, David Pugmire, Scott Klasky, and Hank Childs. In: *Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV)*. 2016.

[13] **Optimizing Multi-Image Sort-Last Parallel Rendering**. Matthew Larsen, Kenneth Moreland, Chris Johnson, and Hank Childs. In: *Proceedings of the IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. Oct. 2016. DOI: 10.1109/LDAV.2016.7874308.

[14] **External Facelist Calculation with Data-Parallel Primitives**. Brenton Lessley, Roba Binyahib, Robert Maynard, and Hank Childs. In: *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)*. June 2016. DOI: 10.2312/pgv.20161178.

[15] **Techniques for Data-Parallel Searching for Duplicate Elements**. Brenton Lessley, Kenneth Moreland, Matthew Larsen, and Hank Childs. In: *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. Oct. 2017. DOI: `10.1109/LDAV.2017.8231845`.

[16] **DPP-PMRF: Rethinking Optimization for a Probabilistic Graphical Model Using Data-Parallel Primitives**. Brenton Lessley, Talita Perciano, Colleen Heinemann, David Camp, Hank Childs, and E. Wes Bethel. In: *Proceedings of IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. Berlin, Germany, Oct. 2018, pp. 34–44. DOI: `10.1109/LDAV.2018.8739239`.

[17] **Maximal Clique Enumeration with Data-Parallel Primitives**. Brenton Lessley, Talita Perciano, Manish Mathai, Hank Childs, and E. Wes Bethel. In: *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. Oct. 2017. DOI: `10.1109/LDAV.2017.8231847`.

[18] **High Performance Heterogeneous Computing for Collaborative Visual Analysis**. Jianping Li, Jia-Kai Chou, and Kwan-Liu Ma. In: *SIGGRAPH Asia Visualization in High Performance Computing*. Nov. 2015. DOI: `10.1145/2818517.2818534`.

[19] **Achieving Portable Performance For Wavelet Compression Using Data Parallel Primitives**. Shaomeng Li, Nicole Marsaglia, Vincent Chen, Christopher Sewell, John Clyne, and Hank Childs. In: *Proceedings of EuroGraphics Symposium on Parallel Graphics and Visualization (EGPGV)*. June 2017, pp. 73–81. DOI: `10.2312/pgv.20171095`.

[20] **Spatiotemporal Wavelet Compression for Visualization of Scientific Simulation Data**. Shaomeng Li, Sudhanshu Sane, Leigh Orf, Pablo Mininni, John Clyne, and Hank Childs. In: *IEEE International Conference on Cluster Computing (CLUSTER)*. Honolulu, HI, Sept. 2017, pp. 216–227. DOI: `10.1109/CLUSTER.2017.15`.

[21] **Comparing Binary-Swap Algorithms for Odd Factors of Processes**. Kenneth Moreland. In: *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. To appear in. Oct. 2018.

[22] **The Tensions of In Situ Visualization**. Kenneth Moreland. In: *IEEE Computer Graphics and Applications* 36.2 (Mar. 2016), pp. 5–9. DOI: `10.1109/MCG.2016.35`.

[23] **VTK-m User's Guide (Version 1.1)**. Kenneth Moreland. Tech. rep. SAND 2018-0475 B. Sandia National Laboratories, 2018.

[24] **Visualization for Exascale: Portable Performance is Critical**. Kenneth Moreland, Matthew Larsen, and Hank Childs. In: *Supercomputing Frontiers and Innovations* 2.3 (2015). DOI: `10.14529/jsfi150306`.

[25] **Formal Metrics for Large-Scale Parallel Performance**. Kenneth Moreland and Ron Oldfield. In: *ISC High Performance*. June 2015, pp. 488–496. DOI: `10.1007/978-3-319-20119-1_34`.

[26] **VTK-m: Accelerating the Visualization Toolkit for Massively Threaded Architectures**. Kenneth Moreland, Christopher Sewell, William Usher, Li-Ta Lo, Jeremy Meredith, David Pugmire, James Kress, Hendrik Schroots, Kwan-Liu Ma, Hank Childs, Matthew Larsen, Chun-Ming Chen, Robert Maynard, and Berk Geveci. In: *IEEE Computer Graphics and Applications* 36.3 (May 2016), pp. 48–58. DOI: `10.1109/MCG.2016.48`.

[27] **Scalable Visualization of Time-varying Multi-parameter Distributions Using Spatially Organized Histograms**. Tyson Neuroth, Franz Sauer, Weixing Wang, Stéphane Ethier, Choong-Seock Chang, and Kwan-Liu Ma. In: *IEEE Transactions on Visualization and Computer Graphics* 23.12 (Dec. 2017), pp. 2599–2612. DOI: `10.1109/TVCG.2016.2642103`.

[28] **Scalable Visualization of Discrete Velocity Decompositions Using Spatially Organized Histograms**. Tyson Neuroth, Franz Sauer, Weixing Wang, Stéphane Ethier, and Kwan-Liu Ma. In: *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. Oct. 2015. DOI: `10.1109/LDAV.2015.7348073`.

[29] **An Integrated Visualization System for Interactive Analysis of Large, Heterogeneous Cosmology Data**. Annie Preston, Ramyar Ghods, Jinrong Xie, Franz Sauer, Nick Leaf, Kwan-Liu Ma, Esteban Rangel, Eve Kovacs, Katrin Heitmann, and Salman Habib. In: *Proceedings of IEEE PacificVis*. Apr. 2016. DOI: `10.1109/PACIFICVIS.2016.7465250`.

[30] **Integrated explorer for cosmological evolution**. Annie Preston, Franz Sauer, Ramyar Ghods, Nick Leaf, Jinrong Xie, and Kwan-Liu Ma. In: *IEEE Scientific Visualization Conference (SciVis)*. Oct. 2015. DOI: `10.1109/SciVis.2015.7429499`.

[31] **Performance-Portable Particle Advection with VTK-m**. David Pugmire, Abhishek Yenpure, Mark Kim, James Kress, Robert Maynard, Hank Childs, and Bernd Hentschel. In: *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)*. June 2018, pp. 45–55. DOI: `10.2312/pgv.20181094`.

[32] **Spatio-Temporal Feature Exploration in Combined Particle/Volume Reference Frames**. Franz Sauer and Kwan-Liu Ma. In: *IEEE Transactions on Visualization and Computer Graphics* 23.6 (Feb. 2017), pp. 1624–1635. DOI: `10.1109/TVCG.2017.2674918`.

[33] **Audience-Targeted Design Considerations for Effective Scientific Storytelling**. Franz Sauer, Tyson Neuroth, Jacqueline Chu, and Kwan-Liu Ma. In: *IEEE Computing in Science and Engineering* 18.6 (2016), pp. 68–76. DOI: `10.1109/MCSE.2016.100`.

[34] **A Combined Eulerian-Lagrangian Data Representation for Large-Scale Applications**. Franz Sauer, Jinrong Xie, and Kwan-Liu Ma. In: *IEEE Transactions on Visualization and Computer Graphics* 23.10 (Oct. 2016), pp. 2248–2261. DOI: `10.1109/TVCG.2016.2620975`.

[35] **Visualization Techniques for Studying Large-Scale Flow Fields from Fusion Simulations**. Franz Sauer, Yubo Zhang, Weixing Wang, Stéphane Ethier, and Kwan-Liu Ma. In: *Computing in Science and Engineering* 18.2 (Mar. 2016), pp. 68–77. DOI: `10.1109/MCSE.2015.107`.

[36] **Volume rendering with data parallel visualization frameworks for emerging high performance computing architectures**. Hendrik A. Schroots and Kwan-Liu Ma. In: *SIGGRAPH Asia Visualization in High Performance Computing*. Nov. 2015, 3:1–3:4. DOI: `10.1145/2818517.2818546`.

[37] **Parallel Distributed, GPU-Accelerated, Advanced Lighting Calculations for Large-Scale Volume Visualization**. Min Shih, Silvio Rizzi, Joseph Insley, Thomas Uram, Venkatram Vishwanath, Mark Hereld, Michael E. Papka, and Kwan-Liu Ma. In: *Proceedings of the IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. Oct. 2016. DOI: `10.1109/LDAV.2016.7874309`.

[38] **Towards Green Aviation with Python at Petascale**. Peter Vincent, Freddie Witherden, Brian Vermeire, Jin Seok Park, and Arvind Iyer. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. Nov. 2016. DOI: `10.1109/SC.2016.1`.

[39] **Fast Uncertainty-driven Large-scale Volume Feature Extraction on Desktop PCs**. Jinrong Xie, Franz Sauer, and Kwan-Liu Ma. In: *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. Oct. 2015. DOI: `10.1109/LDAV.2015.7348067`.

[40] **In Situ Generated Probability Distribution Functions for Interactive Post Hoc Visualization and Analysis**. Yucong Chris Ye, Tyson Neuroth, Franz Sauer, Kwan-Liu Ma, Giulio Borghesi, Aditya Konduri, Hemanth Kolla, and Jacqueline Chen. In: *Proceedings of the IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. Oct. 2016, pp. 65–74. DOI: `10.1109/LDAV.2016.7874311`.

[41] **In Situ Depth Maps Based Feature Extraction and Tracking**. Yucong Ye, Yang Wang, Robert Miller, Kwan-Liu Ma, and Kenji Ono. In: *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. Oct. 2015. DOI: `10.1109/LDAV.2015.7348065`.

[42] **Scalable Parallel Distance Field Construction for Large-Scale Applications**. Hongfeng Yu, Jinrong Xie, Kwan-Liu Ma, Hemanth Kolla, and Jacqueline H. Chen. In: *IEEE Transactions on Visualization and Computer Graphics* 21.10 (Oct. 2015), pp. 1187–1200. DOI: `10.1109/TVCG.2015.2417572`.

3.2 Chairs

1. Tutorials Chair, **International Conference for High Performance Computing, Networking, Storage and Analysis (SC)**, Hank Childs, November 12–17, 2017.

2. Papers Co-Chair, **In Situ Infrastructures for Enabling Extreme-scale Analysis and Visualization (ISAV)**, Kenneth Moreland, November 12, 2017.
3. Papers Co-Chair, **Large Scale Data Analysis and Visualization (LDAV)**, Kenneth Moreland, October 2, 2017.
4. Symposium Co-Chair, **Large Scale Data Analysis and Visualization (LDAV)**, Hank Childs, October 23, 2016.
5. Papers Chair, **International Symposium on Graph Drawing & Network Visualization (GD)**, Kwan-Liu Ma, September 25–27, 2017.
6. Papers Co-Chair, **IEEE Information Visualization (InfoVis)**, Kwan-Liu Ma, October 23–28, 2016.
7. Papers Co-Chair, **Large Scale Data Analysis and Visualization (LDAV)**, Kenneth Moreland, October 23, 2016.
8. Papers Co-Chair, **EuroVis**, Kwan-Liu Ma, June 6–10, 2016.
9. Workshop Co-Chair, **The 10th Workshop on Ultrascale Visualization**, Kwan-Liu Ma, SC15, November 16, 2015.
10. Papers Co-Chair, **Large Scale Data Analysis and Visualization (LDAV)**, Hank Childs, October 25–26.

3.3 Committees

1. Program Committee, **IEEE Big Data**, Kwan-Liu Ma, December 11–14, 2017.
2. Program Committee, **ACM SIGGRAPH ASIA Symposium on Visualization (SOV17)**, Kenneth Moreland, November 27–30, 2017.
3. Best Papers Committee, **ACM SIGGRAPH ASIA Symposium on Visualization (SOV17)**, Kwan-Liu Ma, November 27–30, 2017.
4. Program Committee, **Visualization Showcase SC17**, Kenneth Moreland, November 12–17, 2017.
5. Papers Co-Chair, **In Situ Infrastructures for Enabling Extreme-scale Analysis and Visualization (ISAV)**, Kenneth Moreland, November 12, 2017.
6. Program Committee, **Pacific Graphics**, Kwan-Liu Ma, October 16-19, 2017.
7. Program Committee, **IEEE Scientific Visualization (SciVis)**, David Pugmire, October 1-6, 2017.

8. Program Committee, **IEEE Scientific Visualization (SciVis)**, Hank Childs, October 1-6, 2017.

9. Best Papers Committee, **IEEE Scientific Visualization (SciVis)**, Kwan-Liu Ma, October 1-6, 2017.

10. Program Committee, **IEEE Conference on Visual Analytics Science and Technology (VAST)**, Kwan-Liu Ma, October 1-6, 2017.

11. Best Papers Committee, **IEEE Conference on Visual Analytics Science and Technology (VAST)**, Kwan-Liu Ma, October 1-6, 2017.

12. Steering Committee, **Large Scale Data Analysis and Visualization (LDAV)**, Kwan-Liu Ma, October 2, 2017.

13. Program Committee, **Large Scale Data Analysis and Visualization (LDAV)**, Hank Childs, October 2, 2017.

14. Steering Committee, **IEEE Symposium on Visualization for Cyber Security (VizSec)**, Kwan-Liu Ma, October 2, 2017.

15. Program Committee, **IEEE Conference on Software Visualization (VISSOFT)**, Kwan-Liu Ma, September 18–19, 2017.

16. Program Committee, **IEEE Cluster 2017**, David Pugmire, September 5-8, 2017.

17. Program Committee, **IEEE Cluster 2017**, Hank Childs, September 5-8, 2017.

18. Program Committee, **EG/VGTC Conference on Visualization (EuroVis)**, Kenneth Moreland, June 12–16, 2017.

19. Program Committee, **Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)**, Kenneth Moreland, June 12–13.

20. Steering Committee, **IEEE PacificVis**, Kwan-Liu Ma, April 18–21, 2017.

21. Program Committee, **SPIE Visualization and Data Analysis (VDA)**, Hank Childs, January 29–February 2, 2017.

22. Program Committee, **12th International Symposium on Visual Computing (ISVC)**, Kenneth Moreland, December 12–14, 2016.

23. Program Committee, **ACM SIGGRAPH ASIA 2016 Symposium on Visualization (SA16VIS)**, Kwan-Liu Ma, December 5–8, 2016.

24. Program Committee, **In Situ Infrastructures for Enabling Extreme-scale Analysis and Visualization (ISAV)**, Kenneth Moreland, November 13, 2016.

25. Program Committee, **In Situ Infrastructures for Enabling Extreme-scale Analysis and Visualization (ISAV)**, Hank Childs, November 13, 2016.

26. Program Committee, **Visualization Showcase SC16**, David Pugmire, November 2016.

27. Program Committee, **Cooperative Design, Visualization, and Engineering (CDVE)**, Kwan-Liu Ma, October 24–27.

28. Program Committee, **IEEE Scientific Visualization (SciVis)**, Kenneth Moreland, October 23–28, 2016.

29. Program Committee, **IEEE Symposium on Visualization for Cybersecurity (VizSec)**, Kwan-Liu Ma, October 24, 2016.

30. Steering Committee, **IEEE Symposium on Visualization for Cyber Security (VizSec)**, Kwan-Liu Ma, October 24, 2016.

31. Program Committee, **IEEE Large Data Analysis and Visualization (LDAV)**, Christopher Sewell, October 23, 2016.

32. Program Committee, **IEEE Large Data Analysis and Visualization (LDAV)**, Kwan-Liu Ma, October 23, 2016.

33. Program Committee, **Graph Drawing & Network Visualization**, Kwan-Liu Ma, September 19–21, 2016.

34. Program Committee, **IEEE Cluster**, Kenneth Moreland, September 12–16, 2016.

35. Program Committee, **EuroVis**, Kenneth Moreland, June 6–10, 2016.

36. Program Committee, **Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)**, Kenneth Moreland, June 6–7, 2016.

37. Program Committee, **Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)**, Hank Childs, June 6–7, 2016.

38. Program Committee, **IEEE Pacific Visualization (PacificVis)**, Hank Childs, April 20–23, 2016.

39. Program Committee, **IEEE Conference on Multimedia and Big Data (BigMM)**, Kwan-Liu Ma, April 20–22, 2016.

40. Program Committee, **IEEE BigDataService**, Kwan-Liu Ma, March 29–April 1, 2016.

41. Program Committee, **Workshop on Emotion and Visualization (EmoVis)**, Kwan-Liu Ma, March 10, 2016.

42. Program Committee, **SPIE Visualization and Data Analysis**, Hank Childs, February 16–18, 2016.

43. **NSF III 2016 Review Panel**, Kenneth Moreland.

44. Program Committee, **ACM/IEEE Supercomputing**, Hank Childs, November 15–20, 2015.
45. Program Committee, **Visual Performance Analysis (SC workshop)**, Hank Childs, November 20, 2015.
46. Program Committee, **IEEE Scientific Visualization (SciVis)**, Kenneth Moreland, October 25–30, 2015.
47. Program Committee, **IEEE Scientific Visualization (SciVis)**, Hank Childs, October 25–30, 2015.
48. Steering Committee, **IEEE Symposium on Visualization for Cyber Security (VizSec)**, Kwan-Liu Ma, October 26, 2017.
49. Program Committee, **IEEE Cluster**, Kenneth Moreland, September 8–11, 2015.
50. Program Committee, **Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)**, Kenneth Moreland, May 25–26, 2015.

3.4 Presentations and Other Outreach

1. **Big Data Visualization**, Keynote Speaker, Kwan-Liu Ma, Taiwan Data Science Conference, Taipei, Taiwan, November 11, 2017.
2. **Visualization for Scientific Discovery and Storytelling**, Invited talk, Kwan-Liu Ma, Oak Ridge National Laboratory, Tennessee, August 24, 2017.
3. **In Situ Processing: Opportunities, Challenges, and Instantiations**, Hank Childs, ISC High Performance Conference, Frankfurt, Germany, June 2017.
4. **State of the Art for In Situ Visualization**, Keynote, Hank Childs, IXPUG Workshop on Software-Defined Visualization, Austin, TX, May 2017.
5. **Audience Targeted Exploratory and Explanatory Visualization**, Keynote Speaker, Kwan-Liu Ma, Spring Conference on Computer Graphics (SCCG), Czech Republic, May 15-17, 2017.
6. **In Transit Visualization of Full and Reduced Simulation Data on HPC Platforms**, Invited Talk, David Pugmire, Parallel CFD Conference, Glasgow Scotland, May 2017.
7. **Emerging Topics in Visual Analytics**, Keynote, Kwan-Liu Ma, PacificVAST, Seoul, South Korea, April 18, 2017.
8. **Data Visualization**, Invited lecture, Kwan-Liu Ma, UC Davis Medical School, Sacramento, CA, February 8, 2017.

9. **Visualization for the Public**, Invited talk, Kwan-Liu Ma, University of Tokyo, Tokyo, Japan, December 19, 2016.
10. **Big Data Visualization**, Invited Talk, Kwan-Liu Ma, Fuji Xerox, Yokohama, Japan, December 15, 2016.
11. **Introduction to Data Visualization**, Invited Lecture, Kwan-Liu Ma, Keio University, Yokohama, Japan, December 13, 2016.
12. **Audience-Targeted Visualization Designs for Effective Storytelling**, Invited talk, Kwan-Liu Ma, Keio University, Yokohama, Japan, December 8, 2016.
13. **Visualization for You**, Keynote Speech, Kwan-Liu Ma, Australian Conference on Human-Computer Interaction (OzCHI), Tasmania, Australia, November 30, 2016.
14. **Exascale Visualization and In Situ Processing**, Invited lecture, Hank Childs, University of Tennessee, Knoxville, TN, September 2016.
15. **In Situ Processing: Opportunities, Challenges, and Instantiations**, Invited talk, Hank Childs, Smoky Mountains Computational Sciences and Engineering Conference, September 1, 2016.
16. **In Situ Processing: Opportunities, Challenges, and Instantiations**, Invited talk, Oak Ridge National Laboratory, Oak Ridge, TN, August 2016.
17. **Visualization: A Tool for Data Exploration and Storytelling**, Invited lecture, Kwan-Liu Ma, Hokudai University, Hokkaido, Japan, August 9, 2016.
18. **Visualization and Analysis Services for Extreme Scale Computing**, Invited lecture, David Pugmire, Rutherford Appleton Laboratory, UK, July 2016
19. **Visualization and Analysis Services for Extreme Scale Computing**, Invited lecture, David Pugmire, Oxford University, UK, July 2016
20. **Visualization and Analysis Services for Extreme Scale Computing**, Invited lecture, David Pugmire, Swansea University UK, July 2016
21. **Visualization and Analysis Services for Extreme Scale Computing**, Invited lecture David Pugmire, Daresbury Laboratory, UK, July 2016
22. **Visualization and Analysis Services for Extreme Scale Computing**, Invited lecture David Pugmire, University of Leeds, UK July 2016
23. **Scientific Visualization for the Public**, Invited talk, Kwan-Liu Ma, Hanzhou Low Carbon Science & Technology Museum, Hangzhou, China, July 8, 2016.
24. **Visualization: An Exploratory and Explanatory Tool**, Invited talk, Kwan-Liu Ma. International Symposium on Visual Computing, Hangzhou, China, July 7, 2016.

25. **Recent Advances in Visualization Research**, Invited seminar, Kwan-Liu Ma, National Chiao Tung University, Hsinchu, Taiwan, June 29, 2016.

26. **Big-Data Visualization Techniques for Studying Behaviors, Connections, and Evolution**, Invited Talk, Kwan-Liu Ma, Institute of Statistical Science, Academia Sinica, Taipei, Taiwan, June 17, 2016.

27. **Visualization: An Essential Tool for Scientific Discovery and Storytelling**, Invited talk, Kwan-Liu Ma, Pacific Science Congress, Academia Sinica, Taipei, Taiwan, June 16, 2016.

28. **Visualization: A Tool for Exploration and Storytelling**, Invited talk, Kwan-Liu Ma, Biophotonics Seminar, UC Davis, June 2, 2016.

29. **Visualizing Extreme Scale CFD Simulations**, Plenary speech, Kwan-Liu Ma, Parallel CFD 2016 Conference, May 11, 2016.

30. **Visualization: A Tool for Data Exploration and Storytelling**, Invited talk, Kwan-Liu Ma, Taipei Medical University, Taiwan, April 28, 2016.

31. **The In Situ Terminology Project**, Hank Childs, Department of Energy Computer Graphics Forum (DOECGF), April 28, 2016.

32. **Recent Advances in Visualization Research**, Invited talk, Kwan-Liu Ma, Institute of Sociology, Academia Sinica, Taiwan, April 27, 2016.

33. **Big Data Visualization**, Invited talk, Kwan-Liu Ma, Summit Forum on Big Data Visualization, Fudan University, Shanghai, China, April 14, 2016.

34. **Visualization Toolkit: Improving Rendering and Compute on GPUs**, Presentation, Robert Maynard, GPU Technology Conference, April 2016.

35. **Adapting the Visualization Toolkit for Many-Core Processors with the VTK-m Library.**, Presentation, Christopher Sewell and Robert Maynard, GPU Technology Conference, April 2016.

36. **Exascale Visualization: What Will Change**, Invited talk, Hank Childs, National Center for Atmospheric Research, Boulder, CO, March 2016.

37. **Topics in Visualization**, Invited talk, Institute for Visualization and Interactive Systems, Kwan-Liu Ma, University of Stuttgart, Germany, March 11, 2016.

38. **Exploratory and Explanatory Visualization**, Keynote speech, Kwan-Liu Ma, 3rd EMBO Conference on Visualizing Biological Data (VIZBI), March 9, 2016.

39. **Exascale Visualization: What Will Change.**, Invitied talk, National Center for Atmospheric Research, Boulder, CO, March 2016.

40. **XVis, VTK-m, and the ECP** Kenneth Moreland, Data/Vis Panel for the Exascale Computing Initiative Project, February 19, 2016.

41. **Data Visualization**, Invited talk, Kwan-Liu Ma, Medical Health Informatics, UCDMC, Sacramento, CA, November 25, 2015.

42. **VTK-m: Building a Visualization Toolkit for Massively Threaded Architectures**, Invited presentation, Ultrascale Visualization Workshop, November 2015.

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

44. **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.

45. **Exascale Visualization: Get Ready for a Whole New World**, Hank Childs, Invited talk, International Computing for the Atmospheric Sciences Symposium (iCAS2015), Annecy, France, September 2015.

46. **VTK-m**, Jeremy Meredith, FASTMath PI Meeting, September 2015.

47. **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.

48. VTK-m Code Sprint, LLNL, September 1-2, 2015.

49. **VTK-m Overview**, Kenneth Moreland, VTK-m Code Sprint, September 1, 2015.

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

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

52. **VTK-m**, Kenneth Moreland, DOECGF, April 2015.

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

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

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

DISTRIBUTION:

1 Laura Biven
U.S. Department of Energy
SC-21/Germantown Building
1000 Independence Ave SW
Washington, DC 20585-1290

1 Teresa Beachley
U.S. Department of Energy
SC-21/Germantown Building
1000 Independence Ave SW
Washington, DC 20585-1290

1	MS 1326	Kenneth Moreland, 1461
1	MS 0899	Technical Library, 9536 (electronic copy)