Missouri University of Science and Technology

Scholars' Mine

Computer Science Faculty Research & Creative Works

Computer Science

01 Jan 2021

Visualization as a Service for Scientific Data

David Pugmire

James Kress

Jieyang Chen

Hank Childs

et. al. For a complete list of authors, see https://scholarsmine.mst.edu/comsci_facwork/1106

Follow this and additional works at: https://scholarsmine.mst.edu/comsci_facwork

 Part of the Computer Sciences Commons

Recommended Citation

D. Pugmire and J. Kress and J. Chen and H. Childs and J. Choi and D. Ganyushin and B. Geveci and M. Kim and S. Klasky and X. Liang and For full list of authors, see publisher's website., "Visualization as a Service for Scientific Data," *Communications in Computer and Information Science*, vol. 1315 CCIS, pp. 157-174, Springer Verlag, Jan 2021.
The definitive version is available at https://doi.org/10.1007/978-3-030-63393-6_11

This work is licensed under a Creative Commons Attribution 4.0 License.

This Article - Conference proceedings is brought to you for free and open access by Scholars' Mine. It has been accepted for inclusion in Computer Science Faculty Research & Creative Works by an authorized administrator of Scholars' Mine. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

Visualization as a Service for Scientific Data

David Pugmire[1(✉)], James Kress[1], Jieyang Chen[1], Hank Childs[3], Jong Choi[1], Dmitry Ganyushin[1], Berk Geveci[2], Mark Kim[1], Scott Klasky[1], Xin Liang[1], Jeremy Logan[1], Nicole Marsaglia[3], Kshitij Mehta[1], Norbert Podhorszki[1], Caitlin Ross[2], Eric Suchyta[1], Nick Thompson[1], Steven Walton[3], Lipeng Wan[1], and Matthew Wolf[1]

[1] Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
pugmire@ornl.gov
[2] Kitware, Inc., Clifton Park, NY, USA
[3] University of Oregon, Eugene, OR 97403, USA

Abstract. One of the primary challenges facing scientists is extracting understanding from the large amounts of data produced by simulations, experiments, and observational facilities. The use of data across the entire lifetime ranging from real-time to post-hoc analysis is complex and varied, typically requiring a collaborative effort across multiple teams of scientists. Over time, three sets of tools have emerged: one set for analysis, another for visualization, and a final set for orchestrating the tasks. This trifurcated tool set often results in the manual assembly of analysis and visualization workflows, which are one-off solutions that are often fragile and difficult to generalize. To address these challenges, we propose a serviced-based paradigm and a set of abstractions to guide its design. These abstractions allow for the creation of services that can access and interpret data, and enable interoperability for intelligent scheduling of workflow systems. This work results from a codesign process over analysis, visualization, and workflow tools to provide the flexibility required for production use. Finally, this paper describes a forward-looking research and development plan that centers on the concept of visualization and analysis technology as reusable services, and also describes several real-world use cases that implement these concepts.

Keywords: Scientific visualization · High-performance computing · In situ analysis · Visualization

D. Pugmire et al.—Contributed Equally.

This manuscript has been co-authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

This is a U.S. government work and not under copyright protection in the U.S.; foreign copyright protection may apply 2020

J. Nichols et al. (Eds.): SMC 2020, CCIS 1315, pp. 157–174, 2020.
https://doi.org/10.1007/978-3-030-63393-6_11

1 Introduction

Gaining insight from large scientific data sets, while challenging, has traditionally been tractable because the process has generally been well understood. This tractability is the result of three key properties: low barrier to entry, collaboration, and standardization. These traditional approaches had a low barrier to entry as the data was written to permanent storage in a standardized way and could easily be shared with others. This in turn enabled rich collaboration among domain, computational and visualization scientists. Once data is stored on disk, each stakeholder can access the data at their convenience, and do so with dedicated visualization and analysis software, custom scripts, etc., which are easily shared. Exploration of data often takes place using GUI-based tools that are well supported and easy to learn. Further, the standardization is helpful on a variety of fronts, not only in how data is stored and represented, but also in how data is accessed and processed. The benefit of standardization is in code reuse, enabling the efforts of a community of software developers to increase their impact. This is particularly needed for visualization and analysis software, since such software often contains a large number of algorithms and data format readers.

The three beneficial properties of low barrier to entry, collaboration, and standardization are rapidly becoming infeasible because of two important trends in high-performance computing: Big Data and hardware complexity. With respect to Big Data, scientific data has been dramatically affected by the three V's—volume, velocity, and variety. With respect to hardware complexity, modern computers increasingly have heterogeneous hardware, deep memory hierarchies, and increased costs for data movement and access. As a result of the volume and velocity components of the Big Data trend, along with the increased costs of data movement and access, saving all data to disk is no longer possible. Instead, data will need to be visualized and analyzed while it is being generated, i.e., in situ processing. But in situ processing presents challenges to the three beneficial properties. In particular, standardization is more difficult since data is being delivered in a variety of ways and locations. Rather than files in known file formats stored to permanent storage, data may come from a computational simulation over a socket, from a remote experimental resource, or it may be located in the memory of a GPU accelerator, just to name a few. Further, the barrier to entry is often substantially higher, requiring highly-experienced, "ninja" programmers to incorporate visualization and analysis algorithms. This limits collaboration, since it is difficult to get visualization and analysis routines applied, leaving the task to only those that can wrangle complex software.

Scientific campaigns have dealt with these challenges by moving toward automated workflows to control the complexities with running simulations. These systems are enabled by middleware systems that provide efficient layers between applications and systems, and by emerging workflow systems that orchestrate executables and the movement of data. That said, visualization and analysis has struggled to adapt to this workflow approach. Despite recent support for in situ processing and heterogeneous architectures, the fundamental "glue" is lacking for bringing together the disparate tools and libraries for a scientific software

campaign. Best efforts often are targeted out of necessity at a narrow range of use cases and are often brittle and difficult to reuse at a later date or generalize for usage in other situations. These problems make the practical and widespread use of these tools difficult, further leading to fragmented approaches as every scientific team creates its own customized approach. Finally, while the results to date have been lacking, they have also taken great expertise to achieve. Fundamentally, we feel that this mismatch—great expertise to achieve poor results—indicates a failure in the underlying approach.

In this paper, we advocate for a new model for visualization and analysis of scientific data to address these challenges that is based on following the "aaS" paradigm—as a service. This model is focused on identifying abstractions for points of interaction between visualization, middleware, and workflow systems. The abstractions provide clear interfaces between these three sub-components in a scientific campaign and makes it easier for them to work together. These abstractions will make it much easier to move visualization computation *to the data*, which is a reversal from the previous model, in which it was easier to move the data. This in turn restores the possibility of low barrier to entry, collaboration, and standardization, by making visualization workflows more user-friendly and intuitive and enabling them to become more schedulable, lightweight, and pervasive. Overall, we feel the entire ecosystem will be more cost effective, portable, efficient, and intuitive—a return to the benefits our community has traditionally enjoyed.

An important benefit of an aaS approach is that it enables each participant to focus on their own area of expertise. For application scientists, visualization should be about declarative intentions. For example, isocontours of primary variables are needed in near-real-time (NRT) to track the progress of a simulation, and high-quality renderings of vorticity magnitude and particle traces around regions of interest are needed after the campaign is completed. Visualization experts should focus on algorithms that provide the necessary functionality, perform well on computing platforms, and operate on a variety of data types. Middleware experts should focus on providing efficient I/O and data movement capabilities between data producers and data consumers. Workflow experts should focus on taking scientific intentions and orchestrating the movement of data from producers among all the data consumers to provide the desired results. By providing clear interfaces (i.e., abstractions) between these pieces, it is possible to rethink how analysis and visualization at scale are performed.

The remainder of this paper is organized around the discussion of a set of abstractions (Fig. 3) we have identified that enable Visualization As A Service (VAAS). These abstractions are targeted at addressing the barriers to extracting insight from large scientific data by providing a service based paradigm, and provide a road map for research and development that can take full advantage of the immense power of modern computing systems. At the same time, these abstractions lower the barriers to entry for users giving them the flexibility to build and connect services together in arbitrary ways. In Sect. 2 we provide two motivating examples that helped guide our thinking in the identification of these abstractions, and Sect. 3 discusses related work and complementary efforts towards these

goals. Section 4 describes the two tiers of abstractions in detail. The base tier of abstractions provides the foundation necessary for creating visualization services. These abstractions include **data access**, **data interpretation**, and **service composition/workflow** abstractions. Together, these three abstractions allow for the creation of basic visualization services since there is a way to access the data, a way to interpret the data, and a workflow system that understands how to schedule the visualization services in conjunction with the simulation or experiment. The second tier of abstractions is built on top of the base tier and is concerned with making visualization services more powerful, easier to use and schedule, and more intelligent. Specifically, we identify **portable performance**, **performance modeling**, and **declarative invocation** as this higher tier. Section 5 discusses how our prior research and experience with application engagements have guided our thinking and the development of these abstractions. We show how these abstractions have proven useful and describe their impact on scientific applications. Finally, Sect. 6 concludes with a discussion on how further research and development in these abstractions can improve the process of analysis and visualization in scientific campaigns.

2 Motivating Workflows

Creating and successfully executing large, complex workflows is a challenging task. These workflows must be extensively vetted before execution to ensure that the necessary results can be captured in a timely manner that efficiently uses computing and/or experimental facilities. This vetting process often requires substantial time from teams of experts, including application scientists, computer scientists, mathematicians, and data analysts. The efforts of these individuals create unique and complicated workflows with a myriad of different analysis and visualization needs [23]. This section describes two different recent visualization and analysis workflows with which our group has been involved and highlights the interesting aspects and complexities of both efforts. The first use case involves work with a simulation, and the second is with an experiment.

2.1 Fusion Simulation Workflow

The simulation use case comes from the high-fidelity whole device modeling (WDM) of magnetically confined fusion plasmas. WDM is among the most computationally demanding and scientifically challenging simulation projects that exists within the US Department of Energy (DOE). The 10 year goal of WDM is to have a complete and comprehensive application that will include all the important physics components required to simulate a full toroidal discharge in a tokamak fusion reactor.

This workflow primarily comprises two different fusion codes, XGC and GENE, which must be coupled together. Coupling these codes enables the simulation to advance further in a shorter amount of time while retaining more accuracy than either code can achieve on its own. XGC is a particle-in-cell code

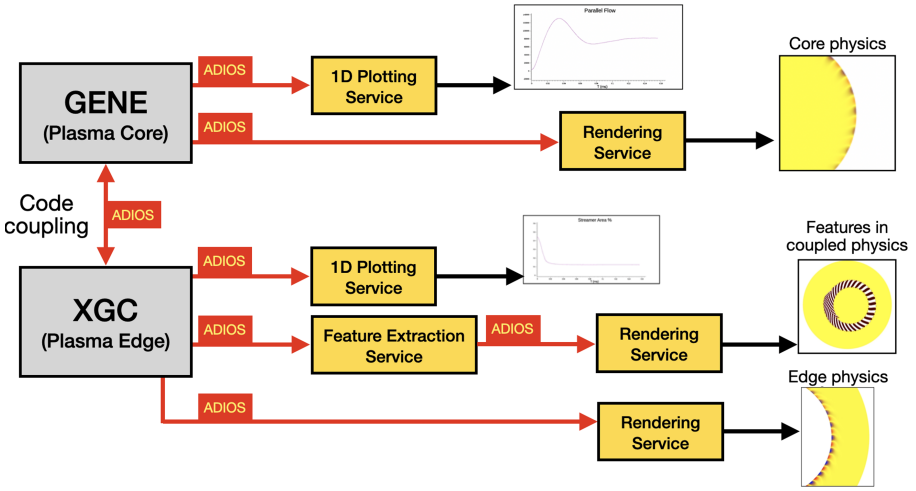

Fig. 1. Workflow for coupled physics simulation. Data from the core and edge coupled physics codes are sent to services to perform analysis and visualization. The resulting images from the rendering services are saved to disk.

optimized for treating the edge plasma, and GENE is a continuum code optimized for the core of the fusion reactor. In the WDM workflow, ADIOS is used to save checkpoint/restart files and offloads variables for in situ analysis and visualization [12]. For in-memory data exchange, ADIOS is used to couple the core and edge simulations [13]. Figure 1 shows the various components of the WDM workflow. The workflow is a complex process that requires sending data to and from multiple separate executables to advance the physics while also visualizing important variables.

2.2 KSTAR

The experiment analysis workflow that comes from fusion experiments is designed to validate and refine simulations that model complex physical processes in the fusion reactor and to test and validate hypotheses. Recent advances in sensors and imaging systems, such as sub-microsecond data acquisition capabilities and extremely fast 2D/3D imaging, allow researchers to capture very large volumes of data at high spatial and temporal resolution for monitoring and diagnostic purposes and post-experiment analyses. Alone, a 2D spatial imaging system, called Electron Cyclotron Emission Imaging, at the Korean Superconducting Tokamak Advanced Research (KSTAR) can capture 10 GB of image data per 10 second shot [51].

A system using ADIOS was developed for KSTAR to support various data challenges by executing remote experimental data processing workflows in fusion science. This system is one of the drivers for the development of the DataMan engine to support science workflows execution over the wide-area network for NRT streaming of experiment data in remote computing resource facilities.

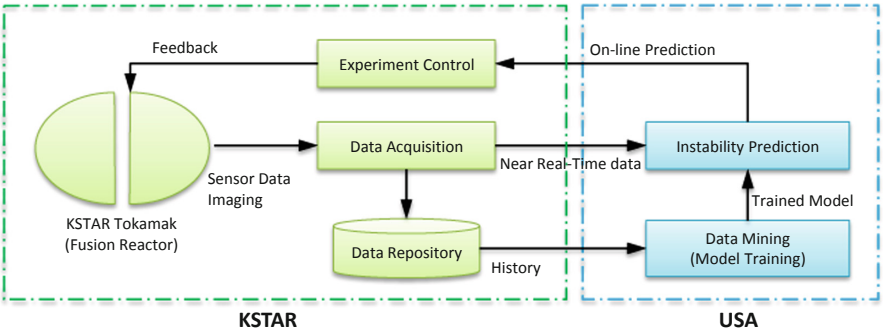

Fig. 2. The KSTAR worfklow showing the data traveling back and forth from KSTAR and the USA. Each box in the workflow is composed of multiple different visualization services.

An example of a KSTAR workflow is shown in Fig. 2. This workflow is a multilevel workflow in that each box comprises one or more sub-workflows. One main goal is to stream online fusion experiment data from KSTAR in Korea to a computing facility in the United States to perform various computationally intensive analyses, such as instability prediction and disruption simulation. Although our previous effort [11] focused on building remote workflows with data indexing, we are currently composing the KSTAR workflow with DataMan. In this workflow, ADIOS provides a remote coupling service to move raw observational data as streams from Korea to the USA. Once data streams arrives in a US computing facility, a set of analysis and visualization workflows will be launched to perform denoising, segmentation, feature detection, and selection to detect any instabilities. Visualization results can then be delivered back to Korea for designing the upcoming shots.

3 On the Shoulders of Giants

The abstractions introduced in Sect. 1 were identified through a careful analysis of our experiences working with application scientists and from the body of published literature. This section describes the systems and concepts that guide our thoughts.

3.1 Tier 1 Related Works

The tier 1 abstractions provide a foundation for data access, data interpretation, and the ability to compose and schedule visualization tasks.

Traditionally, visualization has been performed as a post-processing task, which worked well until the petascale era when it broke down due to the limited I/O bandwith in supercomputers [9,10,49]. In situ processing has been successfully used to avoid this I/O bottleneck, resulting in a rich body of research and production tools. Recent works [4,6] provide surveys of the state-of-the-art in

situ visualization. Middleware libraries have been developed to provide scalable I/O. Systems such as ADIOS [31] and HDF5 [47] provide a publish/subscribe model that enables flexible data access abstraction.

In situ processing is a rich space that consists of three predominant forms. In-line in situ is a synchronous method in which the data producer and visualization run concurrently on the same resource. Tools such as VisIt Libsim [48] and ParaView Catalyst [3,17] support this model. In-transit in situ is an asynchronous method in which the data producer and visualization run on separate resources. Tools such as EPIC [16], Freeprocessing [18], and ICARUS [45] support this model. Hybrid in situ methods provide the flexibility of supporting both synchronous and asynchronous processing. Tools such as Damaris/Viz [14] and SENSEI [4] provide interfaces to use VisIt Libsim and ParaView Catalyst to support a hybrid model. Ascent [28] is a lightweight in situ framework that also provides hybrid model support. Both SENSEI and Ascent use the ADIOS [39] middleware library, which provides a publish/subscribe view of data access using several different data transport mechanisms, including files, in-line, and in-transit.

Data interpretation has been largely focused on data models and schemas. Ascent uses the rich capabilities of BluePrint [29], whereas SENSEI, VisIt LibSim, and ParaView Catalyst rely on the Visualization Toolkit (VTK) data model, which is specifically targeted at the needs of visualization. VizSchema [46] provides an interpretation layer on top of ADIOS for streaming and file-based data. The Adaptable Data Interface for Services [2] is a follow-on work to VizSchema that provides more flexibility and better support for streaming data.

Many of the existing production in situ tools are monolithic and difficult to decompose for scheduling by workflow systems. Furthermore, they require instrumentation into application codes (e.g., VisIt Libsim, ParaView Catalyst, Ascent, SENSEI, Damaris, Freeprocessing) or a shared message passing interface communicator (e.g., EPIC), whereas other require coupling with files (e.g., ICARUS).

Using lightweight visualization tasks in addition to production tools has been explored in [21,43], as described in part in Sect. 2.

3.2 Tier 2 Related Works

The tier 2 abstractions are focused on providing flexibility, power, and intelligence in visualization tasks. These build on a substantial body of work by others as well as ourselves; we focus in the following discussion mostly on the connections of the abstractions to our previous work.

The importance of in situ processing highlighted the need for more flexible data models for in-memory layouts and portability across heterogeneous architectures. Early efforts such as the Extreme Scale Analysis and Visualization Library [36], Dax Toolkit [37], and Piston [32] looked at different aspects of these challenges and were combined into a single toolkit, VTK-m [38]. These efforts have demonstrated the benefits of flexible data models [35] and the portable algorithm performance across a wide variety of architectures [44,50].

A declarative view of visualization has been explored through understanding the performance of different algorithm implementations under different workloads, levels of concurrency, and architectures. Particle-tracing algorithms, which are useful methods for understanding flow, can be implemented in several different ways [42], and performance is dependent on factors such as workload, concurrency, and architecture [7,8,20,41]. Similar work was also done to understand the performance of different types of rendering algorithms for in situ settings [27], and the power-performance tradeoffs for visualization [26].

Models for performance and cost prediction can be useful to inform scheduling and placement by workflow systems. Performance and cost models for different in situ visualization methods are described in [24,25,33,34], analysis of costs for in situ analysis algorithms are described in [40], and a model for in situ rendering is provided in [27].

4 Visualization as a Service Abstractions

Moving away from monolithic or aggregated solutions would help address the challenges of visualization in an era of large streaming data and complex computing environments. The ability to break visualization and analysis tasks into pieces that can be deployed, managed, and automated by a workflow system is powerful and aligns well with the principles of service-oriented architectures (SOA) [30].

At a high level, SOA is characterized by a self-contained black box that provides a well-defined set of features for users. SOA takes several forms, including infrastructure as a service (IaaS)[1], software as a service (SaaS)[19], and microservices [15]. Cloud computing is the most common example of IaaS in which costs are controlled by dynamically allocating resources in response to changing user requirements. SaaS is characterized by the delivery of a capability using a thin client or ergonomic application programming interface. Scalability for SaaS is provided by different types of back-end implementations that are appropriately sized. Microservices are small, independently deployable executables with a distinct goal. Groups of microservices can be orchestrated to perform more complex tasks.

We envision that visualization as a service (VaaS) will apply the principles of the SOA paradigm to computational simulations and experiments. Importantly, we think that VaaS should provide a clear separation between the operations that scientists want to apply to data and the implementation details required to perform it. This will allow application scientists to concentrate on understanding their simulations. VaaS draws from several different aspects of SOA implementations.

– Similar to IaaS, visualization and analysis operations must be provisioned on an appropriate amount of resource. Too much or too little of the wrong kind of resource can result in inefficiency.

– Similar to SaaS, abstractions for access to data and execution must be provided so that application scientists can focus on the operations to be performed, and computer scientists can focus on implementation and scalability.
– Similar to microservices, VaaS would support a set of modular analysis and visualization operations that can be chained together to form complex scientific workflows.

4.1 Visualization as a Service Abstractions

Realization of an SOA to visualize large scientific data will require coordination and codesign with application scientists and disciplines within the computer science community. This section describes a set of abstractions that are targeted at guiding the framework design that follows an SOA philosophy. These abstractions serve as guiding principles for the design of visualization frameworks that can function in a service-based way. They have resulted from our work with application scientists to do visualization and from collaborations with other computer scientists in leveraging complimentary technologies.

From the perspective of an application scientist, our vision is that a service-based visualization framework would work as follows. A team of scientists plans a scientific campaign. They specify a set of visualization tasks in a declarative way. For example, isocontours of high vorticity around an inlet are required in NRT (e.g., every minute) to monitor the simulation. Volume renderings of pressure from three different views are necessary after the simulation has completed. These intentions would then be turned into a sequence of analysis and visualization tasks that would be input into an automated workflow system and run as services on the computing resources to provide the results. The abstractions and their relationships are shown in Fig. 3. These abstractions describe the points of interaction between the tasks and their sequencing that are needed to produce the results. The emphasis is on providing interfaces appropriate for the intended users. Declarative intentions separate the action from the particular algorithms selected and the resources used. Data models and schemas provide information to workflow systems about how tasks can be composed and connected. Performance models for algorithms can inform required resources and optimize the placement of tasks onto resources.

The remainder of this section describes the abstractions for VaaS in a bottom-up approach. We begin with a first tier of abstractions that provides a foundation for VaaS. These foundational abstractions address data access across memory hierarchies, service composition for workflow systems, and methods for interpretation of data between services. We then discuss a second tier of abstractions that builds on the first tier and provides improved flexibility, efficiency, and intelligence to services. These tier 2 abstractions help map visualization intentions onto efficiently executing service on the underlying computing resources.

4.2 Tier 1 Abstractions

The foundation required to support visualization requires three basic abstractions. First, a service must be able to access data from a variety of different sources.

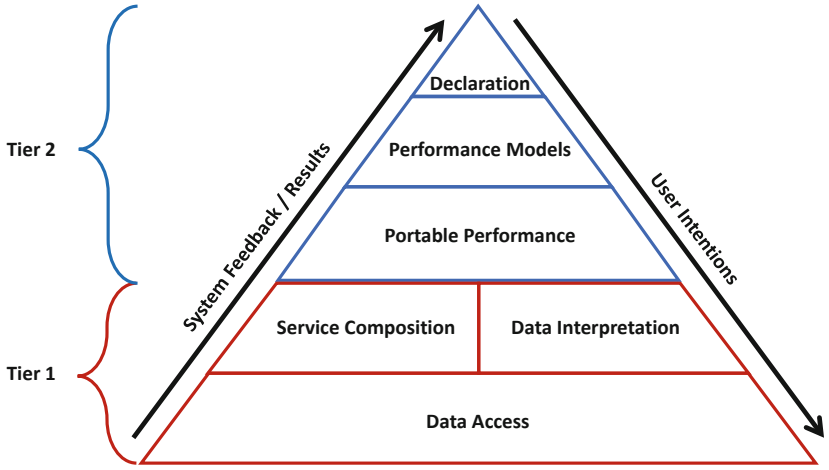

Fig. 3. Chart denoting the two tiers of abstractions that we have identified, their relationships to each other, and proximity to the user.

Second, automated workflow systems must be able to dynamically compose services into sequences and schedule and execute across a variety of resources. Finally, data models, schemas, and ontologies are needed so that workflow systems know how to connect and schedule services and so that services know how to operate on the incoming data.

Work in the first two abstractions has a heavy emphasis on disciplines outside the visualization community. The realization of VaaS will require codesign with these communities so that the pieces work together smoothly. The visualization and analysis community must create and codesign the third abstraction together with the other communities and application scientists so that things work well together. Each of the three abstractions are discussed in more detail in the following sections.

Data Access Abstraction: Visualization services need access to data that come from a variety of sources, including on-node RAM, NVRAM, different nodes in a system, nodes in a different system, and files. Furthermore, the same service might need to consume data from different sources under different circumstances (e.g., from shared memory for an in situ setting, or from disk in a post-processing setting). Supporting all of these data access modes directly in the visualization service is inefficient. Middleware systems such as ADIOS [31] and HDF5 [47] provide a publish/subscribe interface to data that hides the complexity of reading and writing data. The reliance on a data access abstraction allows the visualization community to focus on functionality and performance and the middleware community to focus on providing efficient data access. This also enables greater portability and reuse on different systems and the complex and evolving memory hierarchy.

Service Composition/Workflow Abstraction: Analysis and visualization tasks often consist of a sequence of composed subtasks. For example, rendering an isocontour might involve three steps: (1) recentering a cell-centered variable to the nodes, (2) computing the isocontour, and (3) rendering the geometry. These subtasks might have better performance if the variable recenter and isocontour are performed in situ, and the results are then sent to a set of visualization resources for rendering. In previous work, we have seen the utility of taking these "micro-workflows" and forming integrated in situ visualization libraries (e.g., Catalyst [3], libSim [48]) that can be hard-coded into an application code, as well as interface solutions such as SENSEI [4,5] that allow the workflow mechanics to be embedded into the code while leaving the choice of the in situ visualization or analytics to a run time configuration. However, to fully realize the VaaS design opportunities, we must go further in codesigning the size and scope of the visualization components with high-performance in situ workflow engines. When coupled with the other design abstractions in the VaaS system, this can enable an autonomously adapting visualization environment that can maximize efficiency, latency, or the constraint that is most relevant for that particular scientist's research campaign. One approach we have been exploring is to tie into the extended publish/subscribe semantics for ADIOS, as described in [22], so that VaaS provides context for "editing" and "managing" the data as it is published.

Data Interpretation Abstraction: Data interpretation is required for the workflow system to understand how services can be connected and for individual services to understand the data that is accessed. This information makes it possible for the workflow system to know what must be done and how an intention can be sequenced into a series of services that are chained together and placed onto resources. Data interpretation makes it possible to know which services can be connected together and ensures that inputs are paired with the appropriate outputs; in other literature this is often referred to schemas, data model matching, or ontologies. This includes information about the execution behavior of the service (e.g., the service requires collective communication and so it would run more efficiently on a smaller resource).

Once a service has access to a data stream, ontologies for interpretation and mapping to a data model are needed so that the ontologies can be used by the visualization routines. Ontologies provide the semantics for data, intentions, and operations. These provide information about a service (e.g., a service supports CPU and GPU execution, a service is compute bound or requires collective communication). Ontologies also map the intentions between different data sources (e.g., the variable "pressure" is the same as "press"). Data models include information about the types of meshes in the data (e.g., uniform grid, rectilinear grid, explicit), the fields that exist on the mesh and their associations (e.g., node, cell, edge), and other labels associated with the data. This allows a service to properly process the data. This information also enables the service to perform data conversions where needed or use optimized algorithms when possible (e.g., algorithms for structured data).

4.3 Tier 2 Abstractions

The abstractions in this section build on the aforementioned foundation and provide the ability to optimize functionality and performance and increase flexibility.

Portability Abstraction: Modern computing systems provide rich heterogeneous resources. Furthermore, executables in a workflow can be mapped onto these resources in several ways. A visualization service must be able to run on a variety of different hardware devices. For example, the same visualization service might need to run on all core types in a heterogeneous compute node or be restricted to use only a subset of a particular core type. Visualization services must run on computing systems that have differing architectures and hardware. These complications increase when considering edge computing. This relates to the aforementioned service composition abstraction by providing the workflow system with the flexibility to place services on available resources and across different types of systems. Service portability provides the workflow system with additional options to use for optimizing a scientific campaign.

Performance Models Abstraction: Models that provide performance and cost estimates for algorithms operating on a given type of data and set of resources can provide valuable information to a workflow system. Such models would help the workflow ensure that visualization results are provided in the required time on available resources. These models will inform the selection of cores (e.g., CPU, GPU), task placement on resources, and task dependencies that result from service execution time estimates. The way that a service is executed can have a dramatically different impact on a simulation or experiment. The synchronous in situ processing of expensive services can block the data producer, as can excessive data transfer to additional resources for asynchronous in transit processing.

This abstraction works in conjunction with user intentions, as well as the size and type of data and available resources. The service must be able to provide an estimate on the type and amount of resource required to perform the task or to report that it is impossible so that negotiations can occur with the scientists. For example, an expensive analysis task might be unfeasible to perform in situ for every simulation cycle. However, it might be possible to perform every tenth cycle or, if dedicated visualization resources can be allocated, the user intentions can be satisfied using in-transit processing.

Declarative Visualization Abstraction: An important distinction exists between the operation performed by a service and the algorithm used. Common visualization techniques—such as isocontouring, rendering, or particle tracing—can be accomplished using several different types of algorithms. Some algorithms are optimized for certain data types (e.g., structured grids, explicit grids) on certain hardware types (e.g., GPU or multicore CPU) and have a lower memory

footprint or minimize communication. A declarative abstraction provides a separation from the intentions of the scientists and the actual algorithm used by the service. Given the declarative intention from a scientist, separate from a specific algorithm, coordination with the workflow system is then possible to select the proper algorithm that will produce the desired result and optimize performance.

5 Connecting Abstractions to Applications

Both KSTAR and fusion whole device modeling benefits from a data access abstraction. Access to data is generally the first significant challenge in developing a visualization capability, especially for in situ environments. A simple implication of a data access abstraction is a service that can read data from anywhere in the memory hierarchy (i.e., file or in situ data access use the same interface). Generally, it is straightforward to obtain output files from previous runs or test runs from current scientific campaigns. Development, testing, validation, and scaling against files is generally much easier than trying to do live analysis in a running campaign. The data access abstraction makes it possible to easily switch between files and in situ. This was particularly useful for KSTAR where the data were being moved across the globe. The ability to develop services and then switch the access mode from file to streams without needing to change anything else made the development and testing more efficient. This abstraction enabled the codesign of these services between the visualization and middleware teams.

The composability and interpretation of data was used in fusion whole device modeling. This workflow consisted of several different feature extraction services. As each service extracted features from the simulation output, the data stream was annotated with VizSchema to describe the relationship among the underlying data. This allowed a single implementation of a rendering service to support several different use cases. The workflow system chained these service together and placed them for execution on the computing resources. The rendering service used the VizSchema provided in the stream to properly interpret the data and then rendered images. The portability abstraction was also used by the fusion example. The rendering service and the isocontouring service used the VTK-m library, which provides portable performance across multiple processor architectures.

6 Conclusion and Vision for the Future

Rapidly changing computer architectures, the increasing cost of data movement relative to compute, and the move to automated workflow systems is a significant challenge to extracting insight from scientific data. However, a move to service-oriented visualization allows decoupling the complexity of all these tasks. Our abstractions provide a road map for visualization services that can take full advantage of the immense power of modern computing systems, while affording the flexibility to be connected in arbitrary ways by application scientists.

We envision a future in which application scientists will make use of visualization services without depending on outside expertise for workflow composition. The ability to specify intentions for visualization and analysis on data, along with priorities and timelines for when results are necessary will become a mandatory feature of visualization packages. We envision that these declarative intentions will automatically be converted into a set of services via natural language processing. The statements of priorities and deadlines will form constraints that can be validated as satisfiable using performance models. Negotiations with the user might be necessary if there are conflicting requirements; deadlines might need adjusting, or additional resources might be required. The workflow system will then take this information and construct a graph of requisite services and orchestrate its execution. Services will use data access and interpretation schemas to understand and appropriately process in-flight data. The workflow system will use dynamic monitoring to update the performance models and make real-time modifications to service behavior and execution. As the data size and complexity increases and services require more time, the granularity of service execution can be adjusted (e.g., from every tenth cycle to every hundredth cycle) or the algorithm used by the service can be changed (e.g., use a faster but lower quality rendering algorithm).

In order to support the tier 1 abstractions, efforts must be made to agree on standard methods for data access (e.g., a publish/subscribe model). Several schemas and data models are actively being used and developed, but ontologies are needed to ensure flexibility and the interoperability of services. The access and interpretation of data greatly reduces the barriers to service composition by workflows systems. Research efforts addressing tier 2 abstractions have been significant, but these challenges have not all been resolved, and continued work is needed. Great strides have been made in performance portable algorithms, and these needs will continue into the foreseeable future. Declarative interfaces between the user and algorithm implementations will allow the users to specify requirements and the visualization service can select the correct algorithm for the type and amount of data, and the specified time frame. Performance models for a wide range of algorithm classes, workloads and data types are needed that provide time and cost estimates so that services can be scheduled and placed on resources.

Collectively, there are rich sets of capabilities for addressing these challenges. The work required to support the VaaS abstractions involves codesign and multidisciplinary collaboration to ensure that implementations for interfaces are available. Adoption of these abstractions, and the standardization of these interfaces will enable rich visualization ecosystems. This ecosystem will make it easier for application scientists to use visualization in their campaigns. It will also make it easier for visualization scientists to deploy methods and techniques into workflows and help extract understanding from the large amounts of scientific data.

Acknowledgment. This research was supported by the DOE SciDAC RAPIDS Institute and the Exascale Computing Project (17-SC-20-SC), a collaborative effort of DOE Office of Science and the National Nuclear Security Administration. This research

used resources of the Argonne and Oak Ridge Leadership Computing Facilities, DOE Office of Science User Facilities supported under Contracts DE-AC02-06CH11357 and DE-AC05-00OR22725, respectively, as well as the National Energy Research Scientific Computing Center (NERSC), a DOE Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.

References

1. Infrastructure as a service (2018). https://webobjects.cdw.com/webobjects/media/pdf/Solutions/cloud-computing/Cloud-IaaS.pdf
2. ADIS: Adaptive data interfaces and services. https://gitlab.kitware.com/vtk/adis. Accessed 10 June 2020
3. Ayachit, U., et al.: Paraview catalyst: Enabling in situ data analysis and visualization. In: Proceedings of the First Workshop on in Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization, pp. 25–29. ACM (2015)
4. Ayachit, U., et al.: Performance analysis, design considerations, and applications of extreme-scale in situ infrastructures. In: ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC16). Salt Lake City, UT, USA (2016). https://doi.org/10.1109/SC.2016.78. LBNL-1007264
5. Ayachit, U., et al.: The sensei generic in situ interface. In: 2016 Second Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV), pp. 40–44 (2016). https://doi.org/10.1109/ISAV.2016.013
6. Bauer, A., et al.: In situ methods, infrastructures, and applications on high performance computing platforms. In: Computer Graphics Forum, Vol. 35, pp. 577–597. Wiley Online Library (2016)
7. Binyahib, R., et al.: A lifeline-based approach for work requesting and parallel particle advection. In: 2019 IEEE 9th Symposium on Large Data Analysis and Visualization (LDAV), pp. 52–61 (2019)
8. Camp, D., et al.: Parallel stream surface computation for large data sets. In: IEEE Symposium on Large Data Analysis and Visualization (ldav), pp. 39–47. IEEE (2012)
9. Childs, H., et al.: Extreme scaling of production visualization software on diverse architectures. IEEE Comput. Graph. Appl. **30**(3), 22–31(2010). https://doi.org/10.1109/MCG.2010.51.
10. Childs, H., et al.: Visualization at extreme scale concurrency. In: Bethel, E.W., Childs, H., Hansen, C. (eds.) High Performance Visualization: Enabling Extreme-Scale Scientific Insight. CRC Press, Boca Raton, FL (2012)
11. Choi, J.Y., et al.: ICEE: Wide-area in transit data processing framework for near real-time scientific applications. In: 4th SC Workshop on Petascale (Big) Data Analytics: Challenges and Opportunities in conjunction with SC13, vol. 11 (2013)
12. Choi, J.Y., et al.: Coupling exascale multiphysics applications: methods and lessons learned. In: 2018 IEEE 14th International Conference on e-Science (e-Science), pp. 442–452 (2018). https://doi.org/10.1109/eScience.2018.00133
13. Dominski, J., et al.: A tight-coupling scheme sharing minimum information across a spatial interface between Gyrokinetic turbulence codes. Phys. Plasmas **25**(7), 072,308 (2018). https://doi.org/10.1063/1.5044707.
14. Dorier, M., et al.: Damaris/viz: A nonintrusive, adaptable and user-friendly in situ visualization framework. In: LDAV-IEEE Symposium on Large-Scale Data Analysis and Visualization (2013)

15. Dragoni, N., et al.: Microservices: yesterday, today, and tomorrow. CoRR abs/1606.04036 (2016). http://arxiv.org/abs/1606.04036
16. Duque, E.P., et al.: Epic-an extract plug-in components toolkit for in situ data extracts architecture. In: 22nd AIAA Computational Fluid Dynamics Conference, p. 3410 (2015)
17. Fabian, N., et al.: The paraview coprocessing library: a scalable, general purpose in situ visualization library. In: 2011 IEEE Symposium on Large Data Analysis and Visualization (LDAV), pp. 89–96. IEEE (2011)
18. Fogal, T., et al.: Freeprocessing: transparent in situ visualization via data interception. In: Eurographics Symposium on Parallel Graphics and Visualization: EG PGV:[proceedings]/Sponsored by Eurographics Association in Cooperation with ACM SIGGRAPH. Eurographics Symposium on Parallel Graphics and Visualization, vol. 2014, p. 49. NIH Public Access (2014)
19. Hang, D.: Software as a service. https://www.cs.colorado.edu/~kena/classes/5828/s12/presentation-materials/dibieogheneovohanghaojie.pdf
20. Joy, K.I., et al.: Streamline integration using MPI-hybrid parallelism on a large multicore architecture. IEEE Trans. Vis. Comput. Graph. **17**(11), 1702–1713 (2011). https://doi.org/10.1109/TVCG.2010.259
21. Kim, M., et al.: In situ analysis and visualization of fusion simulations: lessons learned. In: Yokota, R., Weiland, M., Shalf, J., Alam, S. (eds.) ISC High Performance 2018. LNCS, vol. 11203, pp. 230–242. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02465-9_16
22. Klasky, S., et al.: A view from ORNL: Scientific data research opportunities in the big data age. In: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), pp. 1357–1368. IEEE (2018)
23. Kress, J., et al.: Visualization and analysis requirements for in situ processing for a large-scale fusion simulation code. In: 2016 Second Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV), pp. 45–50. IEEE (2016)
24. Kress, J., et al.: Comparing the efficiency of in situ visualization paradigms at scale. In: Weiland, M., Juckeland, G., Trinitis, C., Sadayappan, P. (eds.) ISC High Performance 2019. LNCS, vol. 11501, pp. 99–117. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20656-7_6
25. Kress, J., et al.: Opportunities for cost savings with in-transit visualization. In: ISC High Performance 2020. ISC (2020)
26. Labasan, S., et al.: Power and performance tradeoffs for visualization algorithms. In: Proceedings of IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 325–334. Rio de Janeiro, Brazil (2019)
27. Larsen, M., et al.: Performance modeling of in situ rendering. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis SC 2016, pp. 276–287. IEEE (2016)
28. Larsen, M., et al.: The ALPINE In Situ Infrastructure: ascending from the Ashes of Strawman. In: Proceedings of the In Situ Infrastructures on Enabling Extreme-Scale Analysis and Visualization, pp. 42–46. ACM (2017)
29. Lawrence Livermore National Laboratory: Blueprint. https://llnl-conduit.readthedocs.io/en/latest/blueprint.html. Accessed 30 June 2020
30. Lian, M.: Introduction to service oriented architecture (2012). https://www.cs.colorado.edu/~kena/classes/5828/s12/presentation-materials/lianming.pdf
31. Liu, Q., et al.: Hello adios: the challenges and lessons of developing leadership class i/o frameworks. Concurr. Comput. Pract. Exp. **7**, 1453–1473. https://doi.org/10.1002/cpe.3125

32. Lo, L., et al.: Piston: a portable cross-platform framework for data-parallel visualization operators. In: EGPGV, pp. 11–20 (2012)
33. Malakar, P., et al.: Optimal scheduling of in-situ analysis for large-scale scientific simulations. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, p. 52. ACM (2015)
34. Malakar, P., et al.: Optimal execution of co-analysis for large-scale molecular dynamics simulations. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, p. 60. IEEE Press (2016)
35. Meredith, J., et al.: A distributed data-parallel framework for analysis and visualization algorithm development. ACM Int. Conf. Proc. Ser. (2012). https://doi.org/10.1145/2159430.2159432
36. Meredith, J., et al.: EAVL: the extreme-scale analysis and visualization library. In: Eurographics Symposium on Parallel Graphics and Visualization, pp. 21–30. The Eurographics Association (2012)
37. Moreland, K., et al.: Dax toolkit: A proposed framework for data analysis and visualization at extreme scale. In: 2011 IEEE Symposium on Large Data Analysis and Visualization (LDAV), pp. 97–104 (2011)
38. Moreland, K., et al.: VTK-M: accelerating the visualization toolkit for massively threaded architectures. IEEE Comput. Graph. Appl. **36**(3), 48–58 (2016)
39. Oak Ridge National Laboratory: ADIOS2: The ADaptable Input/Output System Version 2 (2018). https://adios2.readthedocs.io
40. Oldfield, R.A., et al.: Evaluation of methods to integrate analysis into a large-scale shock shock physics code. In: Proceedings of the 28th ACM international conference on Supercomputing, pp. 83–92. ACM (2014)
41. Pugmire, D., et al.: Scalable computation of streamlines on very large datasets. In: Proceedings of the ACM/IEEE Conference on High Performance Computing (SC 2009), Portland, OR (2009)
42. Pugmire, D., et al.: Parallel integral curves. In: High Performance Visualization-Enabling Extreme-Scale Scientific Insight. CRC Press/Francis-Taylor Group (2012). https://doi.org/10.1201/b12985-8
43. Pugmire, D., et al.: Towards scalable visualization plugins for data staging workflows. In: Big Data Analytics: Challenges and Opportunities (BDAC-2014) Workshop at Supercomputing Conference (2014)
44. Pugmire, D., et al.: Performance-Portable Particle Advection with VTK-m. In: Childs, H., Cucchietti, F., (eds.) Eurographics Symposium on Parallel Graphics and Visualization. The Eurographics Association (2018). https://doi.org/10.2312/pgv.20181094
45. Rivi, M., et al.: In-situ visualization: State-of-the-art and some use cases. Brussels, Belgium, PRACE White Paper; PRACE (2012)
46. Tchoua, R., et al.: Adios visualization schema: a first step towards improving interdisciplinary collaboration in high performance computing. In: 2013 IEEE 9th International Conference on eScience (eScience), pp. 27–34. IEEE (2013)
47. The HDF Group: Hdf5 users guide. https://www.hdfgroup.org/HDF5/doc/UG/. Accessed 20 June 2016
48. Whitlock, B., Favre, J.M., Meredith, J.S.: Parallel in situ coupling of simulation with a fully featured visualization system. In: Kuhlen, T., et al. (eds.) Eurographics Symposium on Parallel Graphics and Visualization. The Eurographics Association (2011). https://doi.org/10.2312/EGPGV/EGPGV11/101-109
49. Wong, P.C., et al.: The top 10 challenges in extreme-scale visual analytics. IEEE Comput. Graph. Appl. **32**(4), 63 (2012)

50. Yenpure, A., et al.: Efficient point merge using data parallel techniques. In: Eurographics Symposium on Parallel Graphics and Visualization (EGPGV), pp. 79–88. Porto, Portugal (2019)
51. Yun, G., et al.: Development of Kstar ECE imaging system for measurement of temperature fluctuations and edge density fluctuations. Rev. Sci. Instrum. **81**(10), 10D930 (2010)