# *In Situ* Visualization for Computational Science

**Hank Childs**
University of Oregon

**Janine Bennett**
Sandia National Laboratories

**Christoph Garth**
Technische Universität Kaiserslautern

**Bernd Hentschel**
RWTH Aachen University

***Abstract*—*In situ* visualization is an increasingly important approach for computational science, as it can address limitations on leading edge high-performance computers and also can provide an increased spatio-temporal resolution. However, there are many open research issues with effective *in situ* processing. This article describes the challenges identified by a recent Dagstuhl Seminar on the topic.**

■ **There are two** processing paradigms for visualizing data: *in situ* processing, i.e., processing data as it is generated, and post hoc processing, i.e., processing data well after it is generated (see Figure 1). Research results on *in situ* visualization started appearing approximately a quarter-century ago, albeit using terms such as coprocessing[1] and runtime visualization.[2] Despite promising findings, post hoc processing has retained its role as the dominant processing paradigm for scientific visualization. Post hoc processing enjoys advantages with respect to simpler interfacing (i.e., via files instead of code), increased ability for the end-user to explore data, and fewer consequences when encountering error conditions, among others. In contrast, *in situ* processing enjoys advantages with respect to access to more spatio-temporal data, decreased time to access data, and increased computational power. That said, it is difficult to make definitive statements comparing the merits of *in situ* and post hoc processing, since each can be implemented with variations and optimizations that mitigate their respective weaknesses. For example, post hoc processing can improve an access time via multiresolution methods, while *in situ* processing can improve on fault tolerance by performing concurrent visualizations on separate system resources.

In high-performance computing, the dominance of post hoc processing has faded over the last five years, and given way to a surge of interest around *in situ* processing.[3] The primary driver

*Digital Object Identifier 10.1109/MCG.2019.2936674*
*Date of current version 1 November 2019.*

## Processing paradigms for scientific visualization

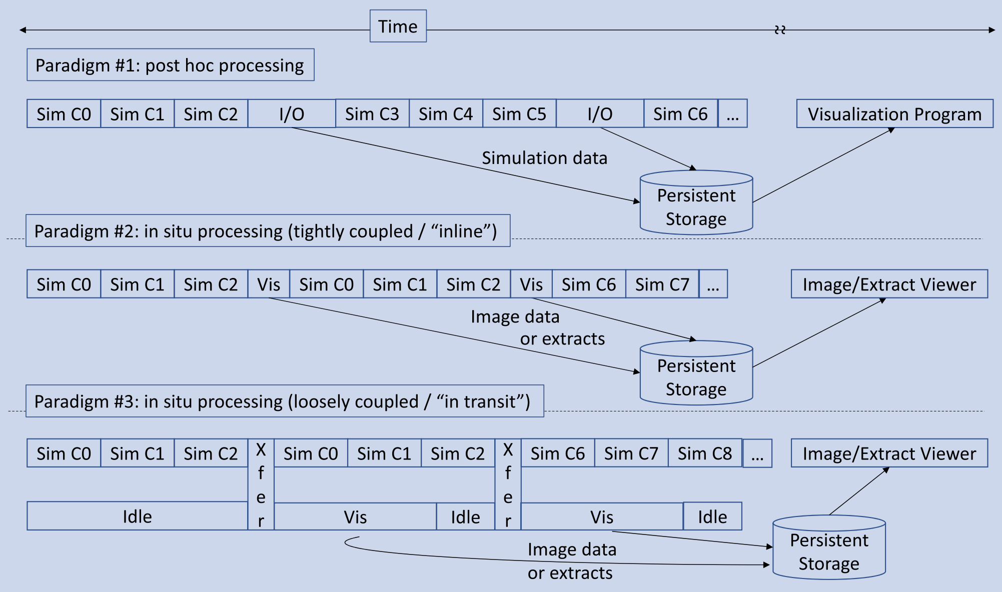

**Figure 1.** Contrasting post hoc and *in situ* processing. This diagram shows three processing paradigms, one for post hoc and two for *in situ*, running horizontally and divided by dashed lines. For each of the three workflows, time runs from left to right, with a break in the timeline indicating that a user views data sometime after it is stored. In each of the workflows, a simulation runs in so-called "cycles," where a simulation advances flow from one state to the next. These are denoted "Sim C1" for the first cycle, "Sim C2" for the second cycle, etc. In the top workflow, post hoc processing, the simulation stores its state to persistent storage every three cycles. In this notional example, the "I/O" rectangle is wider than any of the simulation cycle rectangles to indicate that I/O has become very expensive on supercomputers. For simulations with very quick cycle times, the I/O rectangle may be hundreds of times wider than a simulation cycle rectangle (i.e., take hundreds of times longer). Of course, I/O may also occur less frequently than every three cycles. The top workflow concludes when a user starts a separate visualization program to load the simulation data from persistent storage. This visualization program may run on the same resources as the simulation code, or on distinct resources. The middle workflow denotes a common variant of *in situ* processing that is sometimes referred to as "tightly coupled" or "inline" *in situ* visualization. In this mode, both visualization and simulation run on the same resources, in an alternating manner. The output of the visualizations are images or extracts, and these images or extracts can be viewed after (or during) the simulation via a viewer which likely runs on distinct resources. The final workflow denotes another common variant of *in situ* processing, "loosely coupled" or "in transit." With this workflow, additional resources for visualization are run concurrently to the simulation, and data is transferred from the simulation resources to the visualization resources. In this workflow, the resources for visualization are often smaller than those for simulation, making periods of idle time acceptable.

behind switching processing paradigms is to address I/O constraints on leading-edge supercomputers. On these machines, the ability to generate data is increasing much faster than the ability to store data to persistent storage; compute has gone up by a factor of approximately 100× over the last decade, with I/O performance typically only increasing by a factor of 10×. Consequently, visualization performance using the post hoc paradigm, already shown a decade ago to be limited on supercomputers by I/O performance,[4] becomes poorer and poorer with each new generation of hardware. Of course, *in situ* processing addresses this problem by avoiding the usage of disk altogether. Today, *in situ* processing is being used increasingly on supercomputers, including a

## State-of-the-art *in situ* visualization

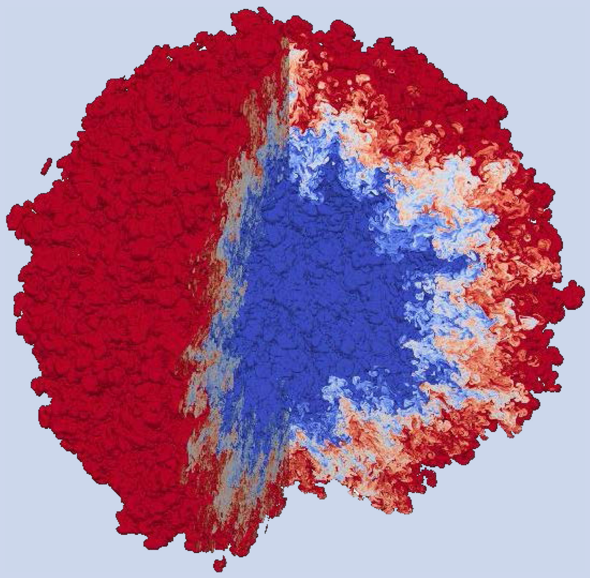

**Figure 2.** Example of a state-of-the-art *in situ* visualization, made by Lawrence Livermore National Laboratory on the Sierra supercomputer (courtesy Matthew Larsen and Cyrus Harrison). This image is of an idealized Inertial Confinement Fusion (ICF) simulation of a Rayleigh–Taylor instability with two fluids mixing in a spherical geometry. The image was created with the *in situ* visualization library Ascent[15], which performed its tasks using the same resources as the simulation code, in this case 4096 nodes and 16,384 GPUs. The mesh contained 98 billion hexahedrons, and rendering occurred in approximately 300 ms for each frame. By running *in situ*, all I/O costs were avoided; instead, time that could have been spent on I/O was spent making a movie flying around the mixing layer.

recent example of visualizing 98 billion cells on 16,384 GPUs (see Figure 2). Unfortunately, despite some successes, *in situ* processing is far from a solved problem, as there are multiple barriers impeding its widespread use.

The open research problems with *in situ* processing were the subject of a Dagstuhl Seminar in July 2018, titled "*In Situ* Visualization for Computational Science," which brought together practitioners from scientific visualization, computational science, and high-performance computing. The workshop participants identified ten *in situ* processing challenges that require significant research, and also "cross-cutting challenges" that do not require research. These challenges were documented in a workshop report[5] that serves as the source material for this article, although two pairs of challenges were combined for readability.

In all, the authors of this article believe that significant research is still needed to enable *in situ* processing to meet widespread production needs. We hope this article will inform our community about the challenges identified at the Dagstuhl Seminar and will also encourage additional research in this space.

## DATA QUALITY AND REDUCTION

An important consideration for *in situ* processing is whether the desired visualizations are known *a priori*. When they are, the visualizations can be specified ahead of time and carried out as the data are generated. When they are not known *a priori*, *in situ* processing is more complicated, since it is not clear which visualizations to carry out. One approach is to refuse *in situ* processing, i.e., to do post hoc processing instead, although saving data less frequently. The problem with this approach is that I/O constraints may cause the data to become so sparse temporally that important phenomena may be lost (i.e., a phenomenon begins after one time slice is saved and ends before the next one) or that features cannot be tracked over time.

Another approach when lacking *a priori* knowledge, and the overarching research challenge described in this section, is to use a combination of *in situ* and post hoc processing. Specifically, data are transformed and reduced *in situ*, and the resulting reduced data are saved to disk, to be explored later in the traditional post hoc manner. The benefit of this idea is that the reduced data could be small enough to store sufficient temporal frequency even in the face of I/O constraints.

The following topics were identified as specific research challenges for data quality and reduction.

- There are myriad of possible techniques for reducing scientific data.[6] Some techniques are general, while others are useful for very specific visualizations. Overall, it is unclear which techniques are the best match for given usage scenarios.
- This "*in situ* reduction + post hoc exploration" paradigm creates a tension between reduction and data integrity. If too much reduction is performed, then the result may be a loss of data integrity and, thus, meaningless visualizations. Similarly, if data integrity is held as paramount, then reduction may be minimal, and I/O problems will persist. The challenge, then, is to determine the acceptable levels of reduction and integrity and also to meet these levels.
- Temporal analysis is commonplace, and requires special attention, both on how to achieve better reduction-integrity tradeoffs and on how to predict which small scale features will be important and, thus, should not be reduced since they will expand over time.
- Reduction introduces error, and this error should be treated in a way that is acceptable to domain scientists, i.e., error bounds, verification, considerations for error propagation, etc.

Solving these challenges could help with both efficiency (e.g., reducing I/O costs or even reducing the need to repeat a simulation) and effectiveness (e.g., enabling scientific discoveries by ensuring that sufficient data are available for exploration).

## WORKFLOWS

There are a variety of instantiations that *in situ* processing can take. In one form, the *in situ* routines are provided as a library that is compiled into a simulation code, sharing memory and compute resources. In another form, the *in situ* routines are part of their own executable which runs on distinct resources. There are also many other possible instantiations, involving multiple modules, multiple data transfers, etc.

Workflows are a mechanism for encoding each of these instantiations with data moving from one task to another. The form of these tasks can vary, from subroutines within one binary to separate executables on distinct computing resources. Overall, the workflow methodology may seem heavyweight for the simplest instantiations of *in situ* processing, but it is widely viewed as necessary as more and more modules are incorporated to solve analysis problems.

The workshop participants differentiated between workflow specification and workflow execution since they each have their own challenges. Workflow specification refers to how tasks are defined, including possible inputs and outputs of a task, dependencies between tasks, and resource requirements. Workflow specification informs workflow execution, since it defines the set of possible actions to take but is distinct since it does not concern itself with how the workflow is executed.

The following were identified as specific research challenges for workflow specification.

- How do we specify desired workflow behavior as conditions change? These changes can come from the data (need more resources to explore a phenomenon) or from the system (faults or unresponsive nodes).
- How should the results of certain operations in the workflow drive future operations in the workflow? How do we specify this?
- How can resource priorities be embedded in the specification?
- How should tradeoffs between declarative and procedural approaches affect workflow specifications?

For workflow execution, the challenge is to develop systems that realize workflow specifications, and also to execute them efficiently. Realizing a workflow is involved: forming a set of tasks, scheduling those tasks on hardware, and handling error conditions. The tasks may have multiple granularities ranging from scheduling within a single program to scheduling across many programs. Further, the penalty for poor decisions can be high, for example, when stalls occur when waiting for resources to become available.

The following were identified as specific research challenges for workflow execution.

- What strategies balance flexibility and efficiency? How can abstractions avoid being too coarse (which prevents optimizations) or too fine (which becomes too complex to manage)?
- What should the data interfaces between modules be?
- How can workflow research outside the community be leveraged? What are the visualization community's unique requirements?

The benefits of solving this problem include improved resource utilization (via elasticity to fit resources optimally), by preventing wasted cycles when there is a fault, and potentially by getting algorithms to run on the types of hardware where they are most efficient. Other benefits include simplifying the job of workflow execution, increased user productivity, and particularly in code reuse and in collaboration. Finally, we note that the workshop discussion and this summary are particularly indebted to ideas from a U.S. Department of Energy workshop on workflows.[7]

## EXASCALE SYSTEMS

Exascale computing poses multiple fundamental changes with respect to visualization. First, exascale hardware architectures will be different than our previous generation of supercomputers because innovative approaches will be needed to achieve so much computing power within energy and cost constraints. The most important change for the visualization community is the relative decrease in the I/O bandwidth (i.e., the driver for *in situ* processing). Other notable changes include the pervasive use of accelerators, multiple accelerators per node, billion-way concurrency, an increased focus on power usage, and deep memory hierarchies. Second, exascale machines will be used in a different way than our previous generation of supercomputers. One example is a shift in science uses cases, including multiphysics solvers, ensembles, and the inclusion of machine learning. This diversity of use cases motivates another section in this article: exascale simulations will produce new, nontraditional types of data.

The following were identified as specific research challenges for exascale systems.

- Ensure that our algorithms can run scalably at exascale-level concurrencies, and/or developing production workflow capabilities to enable visualization on a subset of the compute allocation.
- Leverage exascale hardware features into our algorithms when they can improve performance, such as nonvolatile random access memory (NVRAM).
- As the gap between execution rate and RAM grows, *in situ* visualization will need to adapt, both in terms of efficient execution, and in terms of minimizing effects on simulation codes.

Solving these challenges will allow exascale hardware to be used efficiently. Ultimately, the solution may align with the principles of codesign—if we better understand the behavior of visualization software, then we can not only adjust our research regarding *in situ* approaches to perform better given supercomputer constraints but also affect the design of future supercomputers.

## ALGORITHMIC CHALLENGES

Some staple visualization algorithms are difficult to parallelize efficiently and/or apply to large data sets. In particular, some algorithms exhibit global access patterns that may not even be feasible due to prohibitive communication costs. Examples include particle advection (difficult to parallelize efficiently) and some topological techniques (sometimes global in nature). If we plan to run such algorithms *in situ*, then we will need to develop efficient parallel versions of these algorithms or to identify alternate techniques that are suitable for high-performance computer architectures. An example of the latter approach would be (perhaps) using convolution-type algorithms in the place of particle tracing flow visualization.

The following were identified as specific research challenges with respect to algorithms.

- Are there any features in future high-performance computer architectures that are helpful for difficult-to-parallelize algorithms? (e.g., deep memory hierarchies.)
- Can we use alternate forms, such as compressed data or higher order elements, to improve scalability? If so, how does that affect the accuracy of the results?
- If exploration-oriented use cases mandate that some of these algorithms be interactive, then how would that change our *in situ* design?
- What are the *in situ* configurations that promote efficiency? For example, if running at lower concurrency is more efficient, then this would inform how to carry out *in situ* processing. This topic is explored more in the section on Cost Models.

Solving these challenges is important; failing to do so will limit the techniques we can deliver to stakeholders, in turn limiting the potential for insight from simulations on the next generation of high-performance computers.

## EXASCALE ENABLES NEW USE CASES AND NEW TYPES OF DATA

The typical *in situ* visualization use case is a single simulation generating a high-resolution mesh. However, increased computational power is enabling new types of outputs that motivate new types of visualization. One very important, and increasingly prominent, use case is that of ensemble analysis, i.e., where simulation codes produce ensembles instead of single output. This change requires different processing paradigms, has different properties with respect to efficiently using hardware, and requires different visualization approaches to be effective. A notable example of success on this front is the Melissa project,[8] which was used to analyze an ensemble of 80,000 parallel simulations and avoided 288 TB of storage. Alternate data sources also include simulations that are run alongside experiments and that incorporate experimental data in their visualizations, such as the Xi-CAM effort.[9] Finally, this challenge is not limited to ensembles and including experimental data. Other examples include computational steering or, when mesh resolutions get sufficiently high, a multi-scale representation of data.

The following were identified as specific research challenges for new use cases/new types of data.

- What techniques will the visualization community need to incorporate to support these new types of data? In this case, it is important to note that new collaborations may be required to succeed, for example, bringing in mathematicians and statisticians.
- How should the *in situ* processing approach be adapted to include data wrangling for new types of data (ensembles and experimental)? What data models facilitate exchange and are acceptable to both visualization and simulation stakeholders?
- How should workflows and abstractions support multiple, simultaneous goals?

The challenges involved with these new usages of supercomputers are driven by the domain scientists. While some of the solutions are arguably outside visualization, our community is developing useful infrastructure to address some of these problems, and broadening these infrastructures to include more approaches will benefit all.

## COST MODELS

When running *in situ*, visualization routines often have to perform their tasks within a given time budget. Failure to complete their task in the allotted time typically means that either the visualization is aborted or that the simulation stalls. This situation can be avoided by assessing feasibility *a priori*: for a given set of visualization tasks and its parameters (e.g., isosurfacing task and parameters of specific isolevels), a given data set, and given resources, then can the task be completed within $T$ seconds? Currently, feasibility assessments are not rigorous, e.g., previous experiences, extrapolations from smaller data sets, etc. Cost models provide a more rigorous way to assess feasibility.

Cost models take an input workload (visualization tasks, parameters, data set, resources) and produce an estimate of how long it will take to complete the task. If the estimate is accurate, then these models can be used to answer feasibility questions. Unfortunately, cost models are often hard to generate, although our community has had some successes.[10–12] The following were among the specific research challenges identified for cost models.

- How can we design cost models without extensive studies sweeping many sets of parameters? Which parameters can be fixed or are not relevant?
- Can machine learning be used to solve this problem?
- What are the unique challenges specific to visualization?
- How accurate do the cost models need to be? How tolerant can they be of inaccuracies?

Solving this problem will enable more efficient use of resources. If a cost model reveals that tasks can be completed in under the time budget, then the time can be returned to the simulation or more visualization tasks can be performed.

## CONVERGENCE OF HPC AND BIG DATA

Developments happening outside visualization, high-performance computing (HPC), and computational science communities appear likely to affect each of these three fields. In particular, Big Data processing models produce ideas and implementations that may inform solutions to our research challenges. Similarly, recent machine learning/deep learning developments are poised to alter approaches in many fields, with the potential to benefit *in situ* visualization as well. Further, these activities impact hardware designs, and their components may well appear on supercomputers in the near future. The overarching challenge, then, is how to leverage the breakthroughs from these fields to our own.

The following were among the specific research challenges identified for the convergence of HPC and Big Data.

- How can machine learning be used to enable *in situ* data reduction or to optimize *in situ* workflows?
- How can the knowledge extracted from Big Data and/or machine learning help improve *in situ* visualization approaches?
- What elements of programming environments for Big Data (which are accessible to new programmers) can be incorporated for *in situ* frameworks?

Overall, the benefit of embracing Big Data/machine learning would be the harnessing of industry resources to do tasks more quickly and effectively than we could do otherwise.

## SOFTWARE COMPLEXITY, HETEROGENEITY, AND USER-FACING ISSUES

This research challenge focuses on making *in situ* visualization software accessible and usable to a large number of stakeholders. At the workshop, it was identified that there were three cross-cutting problems that fuel this challenge. The first is software complexity, i.e., *in situ* software is complex and hard to use and also exists in a difficult context—linking two complex software packages, each with their own constraints, data models, parallelization strategies, etc. The second is heterogeneity, i.e., heterogeneity limits adoption due to a cross product of options in hardware architecture, software tools, and usage. The third is user-facing issues, i.e., users are reluctant to adopt new technologies and in particular *in situ*, because it creates additional dependencies for their code, the dependent software they invest may have an uncertain lifetime, they inherit reliability issues from the software they adopt, etc. The following were among the specific research challenges identified for increasing *in situ* adoption.

- How can we minimize our intrusion into the simulation code?
- How can we deal with diverse hardware (i.e., performance portability) and software? That said, it was noted that the VTK-m project[13] is addressing part of the hardware heterogeneity problem, although it does not support deep memory hierarchies, networks, etc.

- How can we isolate faults so that reliability issues with *in situ* visualization software do not affect simulation codes?

Failing to make *in situ* techniques accessible to domain scientists will mean that some stake holders go forward without it, which could potentially result in lost discoveries. Further, solving the problem has a cost benefit—reduced integration times, reduced loss in computational time due to faults, etc.

## PRACTICAL ISSUES FACING *IN SITU*

The workshop participants felt that some challenges were worthy of documentation, even though they were not research challenges per se. The challenges were discussed in two panel sessions: "software engineering and deployment" and "programming and funding issues/interdisciplinary/pipeline." In all cases, there was a recognition that the problems discussed were not unique to *in situ* visualization, but it was also recognized that *in situ* visualization exacerbates these problems.

For the first panel, the primary theme was on the differences in priorities between *in situ* tools researchers/developers and their target user community, specifically for the following topics.

- Requirements, in that scientists and engineers, prefer specialized tools, tailored to their environment, while *in situ* tool developers prefer general-purpose solutions that can be deployed in many simulation codes.
- Adoption, with respect to multiple issues: effective communication between communities, committing to software with an uncertain lifetime, and simulation codes increasing their dependencies on external software packages.
- Accessibility, in particular, the tension between commercial software and open source.

For the second panel, a major focus was on the significant investment needed to develop *in situ* tools, and that acquiring funding for this investment can be difficult. An important aspect of this conversation was the role of industry collaboration and the unclear division of labor between research and production.

## CONCLUSION

This article follows 10 years after an article by Kwan-Liu Ma, also on the challenges for *in situ* visualization.[14] Comparing the predecessor article and the Dagstuhl Seminar shows the progress we have made over the last decade, as well as the areas where little has changed. The previous article asked several questions that are still highly relevant today: What are the optimal configurations for simulation and visualization to share the hardware? How should we reduce data and yet maintain its integrity? What are the best ways to exchange data between simulation and visualization codes? How do we run algorithms efficiently in parallel? Happily, some of the questions raised in Kwan-Liu's article have been answered in the last decade: Will simulation scientists accept part of their allocation being used for visualization? (Yes.) Can existing commercial and open-source visualization software tools be extended to support *in situ*? (In most cases, serious adaptation was needed, but that work has happened or is in progress.) Finally, some new questions have emerged: How to deal with new use cases and new types of data (experimental, ensembles, etc.)? How to incorporate workflow methodology? How to deal with the new architectural features on modern supercomputers? How can we use cost models to achieve efficiency? And how can we ride the wave of activity in Big Data and machine learning? As *in situ* processing is still in a nascent phase, it would not surprise the authors to have another article 10 years from now that has a similar breakdown, with some current problems well addressed, others unsolved, and new, unforeseen problems emerging.

## ACKNOWLEDGMENTS

This article is derived from a Dagstuhl report that had many contributors. The authors feel particularly indebted to participants that led the writing of individual research challenges, specifically E.W. Bethel, P.-T. Bremer, K. Isaacs, K. Moreland, B. Muite, J. Patchett, T. Peterka, D. Pleiter, D. Pugmire, B. Raffin, A. Ribes Cortes, H.-W. Shen, R. Sisneros, and M. Srinivasan. They are also very appreciative to the other seminar participants who greatly contributed to the general discussion: A. Bauer, T. Carrard, M. Dorier,

S. Frey, N. Gauger, M. Hadwiger, C. Hansen, K. Heitmann, I. Hotz, J. Kruger, M. Larsen, P. Messmer, K. Ono, M. Parashar, V. Pascucci, N. Rober, U. Rude, F. Sadlo, G. Weber, R. Westerman, and H. Yu. Sandia National Laboratories is a multimission laboratory managed and operated by the 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. Some of this work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract number DE-ASC52-07NA27344 (LLNL-JRNL-787609). This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

## REFERENCES

1. R. Haimes, "pV3: A distributed system for large-scale unsteady CFD visualization," in *Proc. 32nd Aerosp. Sci. Meeting Exhibit*, 1994, Paper 321.
2. K.-L. Ma, "Runtime volume visualization for parallel CFD," in *Parallel Computational Fluid Dynamics*. Amsterdam, The Netherlands: Elsevier, 1995, pp. 307–314.
3. A. C. Bauer *et al.*, "In situ methods, infrastructures, and applications on high performance computing platforms," *Comput. Graph. Forum*, vol. 35, no. 3, pp. 577–597, Jun. 2016.
4. H. Childs *et al.*, "Extreme scaling of production visualization software on diverse architectures," *IEEE Comput. Graph. Appl.*, vol. 30, no. 3, pp. 22–31, May/Jun. 2010.
5. J. C. Bennett, H. Childs, C. Garth, and B. Hentschel, "In situ visualization for computational science," *Dagstuhl Rep.*, vol. 8, no. 7, pp. 1–43, 2019.
6. S. Li, N. Marsaglia, C. Garth, J. Woodring, J. Clyne, and H. Childs, "Data reduction techniques for simulation, visualization and data analysis," *Comput. Graph. Forum*, vol. 37, no. 6, pp. 422–447, Sep. 2018.
7. E. Deelman *et al.*, "The future of scientific workflows," *Int. J. High Perform. Comput. Appl.*, vol. 32, no. 1, pp. 159–175, 2018.
8. T. Terraz *et al.*, "Melissa: Large scale in transit sensitivity analysis avoiding intermediate files," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2017, Paper 61.
9. R. J. Pandolfi *et al.*, "Xi-cam: A versatile interface for data visualization and analysis," *J. Synchrotron Radiat.*, vol. 25, no. 4, pp. 1261–1270, 2018.
10. M. Larsen, C. Harrison, J. Kress, D. Pugmire, J. S. Meredith, and H. Childs, "Performance modeling of in situ rendering," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, Salt Lake City, UT, USA, Nov. 2016, pp. 276–287.
11. V. Bruder, S. Frey, and T. Ertl, "Prediction-based load balancing and resolution tuning for interactive volume raycasting," *Vis. Inform.*, vol. 1, no. 2, pp. 106–117, 2017.
12. M. Dorier *et al.*, "Adaptive performance-constrained in situ visualization of atmospheric simulations," in *Proc. IEEE Int. Conf. Cluster Comput.*, Sep. 2016, pp. 269–278.
13. K. Moreland *et al.*, "VTK-m: Accelerating the visualization toolkit for massively threaded architectures," *IEEE Comput. Graph. Appl.*, vol. 36, no. 3, pp. 48–58, May/Jun. 2016.
14. K.-L. Ma, "In situ visualization at extreme scale: Challenges and opportunities," *IEEE Comput. Graph. Appl.*, vol. 29, no. 6, pp. 14–19, Nov./Dec. 2009.
15. M. Larsen *et al.*, "The ALPINE in situ infrastructure: Ascending from the ashes of strawman," in *Proc. 3rd Workshop In Situ Infrastructures Enabling Extreme Scale Anal. Vis.*, Denver, CO, USA, Nov. 12–17, 2017, pp. 42–46.

**Hank Childs** is currently an Associate Professor with the Department of Computer and Information Science, University of Oregon, Eugene, OR, USA. His research interests focus on scientific visualization, high-performance computing, and the intersection of the two. He received the Ph.D. degree in computer science from the University of California, Davis, CA, USA, in 2006. Contact him at hank@uoregon.edu.

**Janine Bennett** is a Principal Member of the Technical Staff with Sandia National Laboratories, Livermore, CA, USA. She began her career doing research on scientific visualization and topology, and has expanded her interest into issues relating to exascale computing and computational science. She received the Ph.D. degree in computer science from the University of California, Davis, CA, USA, in 2008. Contact her at jcbenne@sandia.gov.

**Christoph Garth** is a Professor of computer science with Technische Universität Kaiserslautern, Kaiserslautern, Germany. His research interests include large-scale data analysis and visualization, *in situ* visualization, topology-based methods in visualization, and interdisciplinary applications of visualization.

He received the Ph.D. degree from Technische Universität Kaiserslautern in 2007 and then spent four years as a Postdoctoral Researcher with the University of California, Davis, CA, USA. Contact him at garth@cs.uni-kl.de.

**Bernd Hentschel** is currently a Data Scientist with d.velop AG in Gescher, Germany, a major SME in the area of enterprise content management. Previously, from 2010 to 2018, he co-led the Virtual Reality Group with RWTH Aachen University, Aachen, Germany. His research interests include the analysis of domain-specific features in large simulation data, parallel visualization algorithms, and immersive visualization. He studied computer science with RWTH Aachen University, from where he received the Dipl.-Inform. degree in 2003 and the Dr. rer. nat. degree under the supervision of Prof. Dr. T. W. Kuhlen in 2009. Contact him at hentschel@vr.rwth-aachen.de.

Contact department editor Theresa-Marie Rhyne at theresamarierhyne@gmail.com.