Report from Dagstuhl Seminar 18271

In Situ Visualization for Computational Science

Edited by

Janine C. Bennett[1], Hank Childs[2], Christoph Garth[3], and Bernd Hentschel[4]

1 **Sandia National Laboratories* – Livermore, US**
2 **University of Oregon – Eugene, US**
3 **Technische Universität Kaiserslautern, DE**
4 **RWTH Aachen University, DE**

Abstract

In situ visualization, i.e., visualizing simulation data as it is generated, is an emerging processing paradigm in response to trends in the area of high-performance computing. This paradigm holds great promise in its ability to access increased spatio-temporal resolution and leverage extensive computational power. However, the paradigm is also widely viewed as limiting when it comes to exploration-oriented use cases and further will require visualization systems to become more and more complicated and constrained. Additionally, there are many open research topics with in situ visualization. The Dagstuhl seminar 18271 "In Situ Visualization for Computational Science" brought together researchers and practitioners from three communities (computational science, high-performance computing, and scientific visualization) to share interesting findings, to identify lines of open research, and to determine a medium-term research agenda that addresses the most pressing problems. This report summarizes the outcomes and findings of the seminar.

Seminar July 1–6, 2018 – http://www.dagstuhl.de/18271

2012 ACM Subject Classification Human-centered computing → Visualization, Computing methodologies → Simulation environmentsComputing methodologies → Simulation evaluation

Keywords and phrases In situ processing, scientific visualization, high-performance computing, computational science, Dagstuhl Seminar

Digital Object Identifier 10.4230/DagRep.8.7.1

1 Executive Summary

Janine C. Bennett (Sandia National Laboratories, Livermore, jcbenne@sandia.gov)
Hank Childs (University of Oregon, hank@uoregon.edu)
Christoph Garth (Technische Universität Kaiserslautern, garth@cs.uni-kl.de)
Bernd Hentschel (RWTH Aachen University, hentschel@vr.rwth-aachen.de)

License Creative Commons BY 3.0 Unported license
© Janine C. Bennett, Hank Childs, Christoph Garth, Bernd Hentschel

The workshop identified ten challenges for in situ processing that require significant research. These challenges were identified by spending the first day of the workshop with participants

* Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & 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.

Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license
In Situ Visualization for Computational Science, *Dagstuhl Reports*, Vol. 8, Issue 07, pp. 1–43
Editors: Janine C. Bennett and Hank Childs and Christoph Garth and Bernd Hentschel
Dagstuhl Reports
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

giving short presentations on their experiences with in situ processing, with a special focus on unsolved problems. The participant perspectives were then organized into the ten research challenges. Over the following days, sub-groups discussed each of the ten challenges and then presented the key points of their discussions to the group and received feedback. Shortly after the workshop, the leaders of each sub-group wrote summaries for its associated research challenge; these summaries are the basis of this report.

The ten challenges identified by our participants were:

- Data quality and reduction, i.e., reducing data in situ and then exploring it post hoc, which is likely the form that will enable exploration of large data sets on future supercomputers.
- Workflow specification, i.e., how to specify the composition of different tools and applications to facilitate the in situ discovery process.
- Workflow execution, i.e., how to efficiently execute specified workflows, including workflows that are very complex.
- Exascale systems, which will have billion-way concurrency and disks that are slow relative to their ability to generate data.
- Algorithmic challenges, i.e., algorithms will need to integrate into in situ ecosystems and still perform efficiently.
- Use cases beyond exploratory analysis, i.e., ensembles for uncertainty quantification and decision optimization, computational steering, incorporation of other data sources, etc.
- Exascale data, i.e., the data produced by simulations on exascale machines will, in many cases, be fundamentally different than that of previous machines.
- Cost models, which can be used to predict performance before executing an algorithm and thus be used to optimize performance overall.
- The convergence of HPC and Big Data for visualization and analysis, i.e., how can developments in one field, such as machine learning for Big Data, be used to accelerate techniques in the other?
- Software complexity, heterogeneity, and user-facing issues, i.e., the challenges that prevent user adoption of in situ techniques because in situ software is complex, computational resources are complex, etc.

From group discussion, two other important topics emerged that do not directly lead to open research questions, but rather are concerned with effective organization of the often highly interdisciplinary research into in situ techniques. To address these, two panels were held to facilitate effective discussion. Finally, the workshop featured technical presentations by participants on recent results related to in situ visualization.

2 Table of Contents

Executive Summary
Janine C. Bennett, Hank Childs, Christoph Garth, Bernd Hentschel 1

Overview of Talks

A Simple (?) Computational Monitoring Workflow
Andrew Bauer 5

The In Situ Terminology Project
Hank Childs 5

Reduced Representation Tradeoffs, Dynamic Prediction and Adjustment
Steffen Frey 5

Ascent: A Fly-Weight In Situ Visualization Framework
Matthew Larsen 6

Performance Modeling of In Situ Rendering
Matthew Larsen 6

Using VTK-m
Kenneth Moreland 7

Slycat VideoSwarm
Kenneth Moreland 7

Fast Fourier Transform in Solving Partial Differential Equations
Benson Muite 7

Design of In Situ Framework for Time-Varying Data
Kenji Ono 8

Optimizing Scientist Time
Through In Situ Visualization and Analysis
John Patchett 8

EC's Strategy towards Exascale
Dirk Pleiter 9

Visualization Services in the ADIOS Framework
David Pugmire 9

Melissa: Large Scale In Transit Sensitivity Analysis
Bruno Raffin 9

Challenges for the visualization and analysis
of high-resolution simulation data
Niklas Röber 10

Toward Tuned to Terrific:
Parallel Particle Advection, I/O Optimization, and Deep Learning
Robert Sisneros 10

Supporting the Virtual Red Sea Project
Madhusudhanan Srinivasan 10

ECP ALPINE Algorithm Overview
Gunther H. Weber 11

Working Groups

Data Quality and Reduction
Peer-Timo Bremer and Han-Wei Shen 11

Workflow Specification
Tom Peterka and Madhusudhanan Srinivasan 13

Workflow Execution
Hank Childs and John Patchett 16

Exascale Systems
Kenneth Moreland and Dirk Pleiter 18

Algorithmic Challenges
Christoph Garth and David Pugmire 22

Use Cases Beyond Exploratory Analysis
Katherine Isaacs and Alejandro Ribes Cortes 25

Exascale Enables / Requires Different Data
Bernd Hentschel and Dave Pugmire 27

Cost Models
Robert Sisneros and Christoph Garth 29

Convergence of HPC and Big Data
Bruno Raffin and Benson Muite 31

Software Complexity, Heterogeneity, and User-facing Issues
John Patchett and E. Wes Bethel 34

Panel Discussions

Panel Discussion on “Software Engineering & Deployment” 37

Panel Discussion on “Programmatic & Funding Issues / Interdisciplinary / Pipeline” 39

Participants 43

3 Overview of Talks

3.1 A Simple (?) Computational Monitoring Workflow

Andrew Bauer (Kitware – Clifton Park, US, andy.bauer@kitware.com)

License Creative Commons BY 3.0 Unported license
© Andrew Bauer

The complexities of in situ processing can be daunting initially for both users and developers. In this presentation we demonstrate that much of the complexity of using ParaView for computational monitoring can be hidden from the user such that the user experience is very similar to using ParaView in a post hoc fashion. This is done by using the tools for remote connections that many scientists that work on HPC-sized problems will likely be familiar with. This remote connection mechanism available through the ParaView GUI launches both the ParaView Catalyst linked simulation and ParaView's pvserver executable and lets them know how to communicate with each other. Additionally, the tools automatically connect the ParaView GUI to pvserver for a fully connected in situ system.

3.2 The In Situ Terminology Project

Hank Childs (University of Oregon – Eugene, US, hank@uoregon.edu)

License Creative Commons BY 3.0 Unported license
© Hank Childs

The term "in situ processing" has evolved over the last decade to mean both a specific strategy for processing data and an umbrella term for a processing paradigm. The resulting confusion makes it difficult for visualization and analysis scientists to communicate with each other and with their stakeholders. To address this problem, a group of approximately fifty experts convened with the goal of standardizing terminology. This presentation summarizes their findings and proposes a new terminology for describing in situ systems. An important finding from this group was that in situ systems can be described via multiple, distinct axes: integration type, proximity, access, division of execution, operation controls, and output type. This paper discusses these axes, evaluates existing systems within the axes, and explores how currently used terms relate to the axes.

3.3 Reduced Representation Tradeoffs, Dynamic Prediction and Adjustment

Steffen Frey (Universität Stuttgart, DE, steffen.frey@visus.uni-stuttgart.de)

License Creative Commons BY 3.0 Unported license
© Steffen Frey

Reduced representations for in situ visualization often exhibit tradeoffs between different quantities of interest, including data size, generation time, quality/accuracy, etc. These tradeoffs are steered via parameters, and can be adjusted dynamically during the execution to account for the variability in the data and the simulation process. The basis for informed adjustments are prediction models that estimate the impact of parameters on quantities of

interest. In the talk, first Volumetric Depth Images (a reduced representation for volume data) and the involved tradeoffs are presented. Second, different approaches for online prediction and balancing from related visualization scenarios are outlined, and finally open research questions to bring these two parts together for in situ visualized are discussed.

3.4 Ascent: A Fly-Weight In Situ Visualization Framework

Matthew Larsen (Lawrence Livermore National Laboratory, US, larsen30@llnl.gov)

License Creative Commons BY 3.0 Unported license
© Matthew Larsen

A key trend in contemporary HPC hardware development is an increasing spread between compute performance on the one hand and I/O bandwidth and available system memory on the other. In this setting, classical post hoc analysis quickly become infeasible. With respect to these constraints, in situ strategies, which process data as it is being generated – thereby avoiding the I/O constraints – are widely regarded a necessity. However, in situ visualization presents a multitude of challenges and questions, including efficient execution on a variety of HPC infrastructures; dealing with unconventional data structures, particularly higher order elements; and making state of the art in situ methods readily accessible. To this end, Ascent is an in situ scientific visualization infrastructure for HPC physics simulation codes being developed as part of the US Department of Engergy's Exascale Computing Project (ECP). The infrastructure is designed for leading-edge supercomputers, and has support for both distributed- memory and shared-memory parallelism. It can take advantage of computing power on both conventional CPU architectures and on many-core architectures (e.g., NVIDIA GPUs or the Intel Xeon Phi). In addition to scientific visualization, Ascent also serves as vehicle to deploy custom analysis. We are interested in ideas related to using Ascent as a gateway to couple our HPC simulation codes to the large ecosystem of data science of tools. In support of this goal, we have demonstrated using Ascent for two powerful use cases that leverage data science capabilities on simulation data: 1) We used Ascent to couple data in situ from a simulation code to Python-based machine learning algorithms 2) We used Ascent to provide in situ access to simulation data for general consumption in a Python-based Jupyter Notebook There are many software engineering and data modeling challenges related to bridging HPC simulations and data science capabilities. From these successful demonstrations, we feel Ascent infrastructure's is uniquely prepared to help connect these worlds.

3.5 Performance Modeling of In Situ Rendering

Matthew Larsen (Lawrence Livermore National Laboratory, US, larsen30@llnl.gov)

License Creative Commons BY 3.0 Unported license
© Matthew Larsen

With the push to exascale, in situ visualization and analysis will continue to play an important role in high performance computing. Tightly coupling in situ visualization with simulations constrains resources for both, and these constraints force a complex balance of trade-offs. A performance model that provides an a priori answer for the cost of using an in situ approach for a given task would assist in managing the trade-offs between simulation and visualization

resources. In this work, we present new statistical performance models, based on algorithmic complexity, that accurately predict the run-time cost of a set of representative rendering algorithms, an essential in situ visualization task. To train and validate the models, we conduct a performance study of an MPI+X rendering infrastructure used in situ with three HPC simulation applications. We then explore feasibility issues using the model for selected in situ rendering questions.

3.6 Using VTK-m

Kenneth Moreland (Sandia National Labs – Albuquerque, US, kmorel@sandia.gov)

License Creative Commons BY 3.0 Unported license
© Kenneth Moreland

One of the most critical challenges for high-performance computing (HPC) scientific visualization is execution on massively threaded processors. Of the many fundamental changes we are seeing in HPC systems, one of the most profound is a reliance on new processor types optimized for execution bandwidth over latency hiding. Our current production scientific visualization software is not designed for these new types of architectures. To address this issue, the VTK-m framework serves as a container for algorithms, provides flexible data representation, and simplifies the design of visualization algorithms on new and future computer architecture.

3.7 Slycat VideoSwarm

Kenneth Moreland (Sandia National Labs – Albuquerque, US, kmorel@sandia.gov)

License Creative Commons BY 3.0 Unported license
© Kenneth Moreland

Slycat is a web-based system for analysis of large, high-dimensional data such as that produced by High Performance Computing (HPC) platforms. The Slycat server integrates data ingestion, scalable analysis, data management, and visualization with commodity web clients using a multi-tiered hierarchy of data and model storage. In this talk we discuss a project, VideoSwarm in which images created in situ during an ensemble run of simulations was used to identify common and diverging parameters being studied.

3.8 Fast Fourier Transform in Solving Partial Differential Equations

Benson Muite (University of Tartu, EE, benson.muite@ut.ee)

License Creative Commons BY 3.0 Unported license
© Benson Muite

The Fast Fourier Transform (FFT) is an algorithm used in the solution of many partial differential equations, primarily as a linear system solver. When the objective is the investigation of the properties of the partial differential equations, visualization is extremely important. In situ visualization helps avoid the io bottleneck. In situ frameworks make it easy to use supercomputers for scientific investigation of the properties of partial differential equations.

In cases where the FFT is not scalable, other methods for solving linear systems of equations can be used. Most of these methods require a sparse matrix vector multiply. Benchmarks such as HPCG and Graph 500 are highly correlated because the primary kernel "is a sparse matrix vector multiply". A question of interest is what are the primary kernels/patterns of interest for in situ visualization which will become more important on next generation supercomputers.

3.9 Design of In Situ Framework for Time-Varying Data

Kenji Ono (Kyushu University, JP, keno@cc.kyushu-u.ac.jp)

License Creative Commons BY 3.0 Unported license
© Kenji Ono

This talk gives an in situ framework designed to provide flexible data processing and visualization for time-varying data on an HPC system. The data staging approach used in the proposed framework utilizes the capabilities of the OpAS (Open Address Space) library, which enables asynchronous access, from outside processes, to any exposed memory region in the simulation side. A concept of temporal buffer is used to store the time-varying simulation results to execute interactive data processing and visualization. These features are expected to facilitate the decoupling of the simulation running time from the data processing and visualization execution time, and consequently to improve the flexibility for the in situ processing and visualization. As examples of this framework, we discuss a usage, ability, and code integration briefly using a CFD application for wind prediction on terrain.

3.10 Optimizing Scientist Time Through In Situ Visualization and Analysis

John Patchett (Los Alamos National Laboratory, US, patchett@lanl.gov)

License Creative Commons BY 3.0 Unported license
© John Patchett
Main reference John Patchett, James P. Ahrens: "Optimizing Scientist Time through In Situ Visualization and Analysis", IEEE Computer Graphics and Applications, Vol. 38(1), pp. 119–127, 2018.
URL http://dx.doi.org/10.1109/MCG.2018.011461533

Simulation scientist time should be considered when designing simulation runs. The decision to produce artifacts, in particular what kind of artifacts, representing the simulation has implications on the time a scientist must spend to understand the simulation state. This talk relates observations of four techniques used by a simulation scientist during a set of simulations runs to study asteroid generated tsunami: lossless compression, resampling to image data, feature extraction using simple threshold operation, and maintaining a small amount of provenance as metadata. All of these have the potential of saving the scientist time while doing analysis at the expense of a small amount of computational time.

3.11 EC's Strategy towards Exascale

Dirk Pleiter (Jülich Supercomputing Centre, DE d.pleiter@fz-juelich.de)

License Creative Commons BY 3.0 Unported license
© Dirk Pleiter

The European Commission is currently implementing its plans for realising 2 European pre-exascale and later 2 exascale systems. In this talk we give an overview about the general strategy to grow a European HPC ecosystem and provide details about a Joint Undertaking, which will execute the procurement of the (pre-)exascale systems. Finanly, we summarise opportunities for promoting research and innovation on in situ challenges in this context.

3.12 Visualization Services in the ADIOS Framework

David Pugmire (Oak Ridge National Laboratory, US, pugmire@ornl.gov)

License Creative Commons BY 3.0 Unported license
© David Pugmire

In situ paradigms allow for the analysis and visualization of unprecedented amounts of spatio-temporal data from simulations running on supercomputers. A variety of in situ methods are possible, including tightly coupled, loosely coupled, and hybrid methods. We use these paradigms, and analysis and visualization operations using a Service Oriented Approach to provide analysis and monitoring of a running simulation for a plasma fusion simulation.

3.13 Melissa: Large Scale In Transit Sensitivity Analysis

Bruno Raffin (INRIA – Grenoble, FR, bruno.raffin@inria.fr)

License Creative Commons BY 3.0 Unported license
© Bruno Raffin

Global sensitivity analysis is an important step for analyzing and validating numerical simulations. One classical approach consists in computing statistics on the outputs from well-chosen multiple simulation runs. Simulation results are stored to disk and statistics are computed postmortem. Even if supercomputers enable to run large studies, scientists are constrained to run low resolution simulations with a limited number of probes to keep the amount of intermediate storage manageable. In this talk we propose a file avoiding, adaptive, fault tolerant and elastic framework that enables high resolution global sensitivity analysis at large scale. Our approach combines iterative statistics and in transit processing to compute Sobol' indices without any intermediate storage. Statistics are updated on-the-fly as soon as the in transit parallel server receives results from one of the running simulations. For one experiment, we computed the Sobol' indices on 10M hexahedra and 100 timesteps, running 8000 parallel simulations executed in 1h27 on up to 28672 cores, avoiding 48TB of storage.

3.14 Challenges for the visualization and analysis of high-resolution simulation data

Niklas Röber (DKRZ Hamburg, DE, roeber@dkrz.de)

License Creative Commons BY 3.0 Unported license
© Niklas Röber

With growing data sizes and simulation complexity, the processes of visualization and analysis become technically more difficult, but at the same time also more important. This particularly holds for weather and climate simulations that already produce tera- and petabytes of data for high-resolution simulation runs. This presentation shows examples of large data visualizations, devises workflows to handle massive amounts of data, as well as discusses the benefits and drawbacks for an in-situ visualization of the simulation data.

3.15 Toward Tuned to Terrific: Parallel Particle Advection, I/O Optimization, and Deep Learning

Robert Sisneros (University of Illinois at Urbana Champaign, US sisneros@illinois.edu)

License Creative Commons BY 3.0 Unported license
© Robert Sisneros

Parallel particle advection refers to a class of particularly challenging data analysis and visualization algorithms. This is due to load balancing sensitivities, strong data dependencies, and computational requirements. For this reason, it also represents a particularly challenging prospect for in situ visualization. In this talk I will first outline efforts in understanding the effect of the tuneable parameters of particle advection algorithms on performance. I will then follow up with recent work in the application of deep learning to identifying optimal I/O configurations for simulations running at scale. To conclude I will show how the latter may be directly applied to further understand parallel particle advection algorithms.

3.16 Supporting the Virtual Red Sea Project

Madhusudhanan Srinivasan (King Abdullah University of Science and Technology – Thuwal, SA, madhu.srinivasan@kaust.edu.sa)

License Creative Commons BY 3.0 Unported license
© Madhusudhanan Srinivasan

The Virtual Red Sea project at King Abdullah University of Science and Technology (KAUST) aims to build an integrated data-driven modeling system to study and predict the circulation and the climate of the Red Sea. The scientific goal is to understand atmospheric and oceanic circulations and dynamics, atmosphere-ocean-biology interactions, transport and dispersion phenomena, and better reconstruction and forecasting. The Visualization Core Lab at KAUST supports this effort by researching and building in situ tools for analysis, verification, diagnosis, risk assessment and decision support. In this talk, we present our tools and workflows to support visualization and analysis of high-fidelity coupled atmospheric, oceanographic and ecological simulations over the Red Sea in Saudi Arabia.

3.17 ECP ALPINE Algorithm Overview

Gunther H. Weber (Lawrence Berkeley National Laboratory, US, ghweber@lbl.gov)

License Creative Commons BY 3.0 Unported license
© Gunther H. Weber

The Exascale Computing Project (ECP) ALPINE project develops algorithms for visualization and analysis that will be critical for ECP applications as the dominant analysis paradigm shifts from post hoc (post processing) to in situ (processing data in a code as it is generated). In this talk I provide an overview over the algorithms currently developed in the project: (1) topological analysis; (2) feature-centric analysis; (3) adaptive sampling; and (4) Lagrangian analysis.

4 Working Groups

4.1 Data Quality and Reduction

Peer-Timo Bremer (Lawrence Livermore National Laboratory, US, bremer5@lnll.gov)
Han-Wei Shen (Ohio State University – Columbus, US, shen.94@osu.edu)

License Creative Commons BY 3.0 Unported license
© Peer-Timo Bremer and Han-Wei Shen

Contributors: R. Westermann, T. Carrard, I. Hotz, C. Hansen, D. Pugmire, K. Heitmann, H. Yu, M. Hadwiger, J. Krüger, H.-W. Shen , S. Frey, C. Garth, V. Pascucci

Background and Motivation. A fundamental problem for large-scale computational experiments is that is infeasible to store all data generated during a run. The traditional solution is to subset data in time for permanent storage. This has worked well in most applications. However, as simulations become larger the ratio between what is computed and what can be saved keeps increasing and especially for the bleeding edge simulations there now is a significant risk in losing crucial information. For example, it is often no longer possible to reliably track features [1] through time.

Research Questions and Challenges. Consequently, new techniques of data reduction are needed which leads to new research questions and challenges:

- First, there is a large gamut of different reduction approaches with different characteristics. While there are many ways of classifying these, one convenient axis is to consider the trade-off between generality and specificity. On one end of the spectrum are traditional data compression techniques that aim to preserve as much information of the original "signal" as possible given a hard limit on the acceptable data volume. These techniques are considered general as they by and large do not restrict and/or favor any particular downstream post-processing. A slightly more specific variant would be to compress and preserve only some of the data fields (i.e. only some primary variables) or to allocate more space to data considered more sensitive. Generically, the more specific a technique becomes, meaning the more a priori information one has about what is considered important, the more aggressively one is able to reduce the data. For example, knowing only a certain data range is important or even what particular post-processing will be used will allow more data to be discarded without impacting the results. At the other extreme of highly specific techniques are feature extraction approaches which directly

target (a) specific research question(s). In the limit, a scientist might only be interested in a single characteristic, i.e. global average temperature. Note that this specificity axes also loosely corresponds to the move from what is considered data compression, i.e. wavelets, to what is considered data analysis, i.e. feature extraction though there are may intermediate approaches that may not easily fit into either category.

- The second challenge is how to determine – ideally a priory – what amount of data reduction is acceptable and how to guarantee that this constraint is observed. Clearly, this question is highly application specific and it may not be possible to decide a priori which information might be necessary to produce new scientific insight. Furthermore, there exist a second trade-off between techniques aimed to answer specific scientific questions defined in advanced vs. the ability to explore data post-hoc to gain new insights and find the unexpected. This is especially challenging at the largest scales since there the need for data reduction is greatest, yet the chance of observing previously unknown phenomena is also greatest and thus the risk of missing a breakthrough by an overly prescriptive reduction scheme is high.
- Third, there exists a wide variety of "basis functions" in which one may express information. These range from compression basis, i.e. wavelets, to statistical quantities, to trained models, or abstract feature descriptions. Each will have their own advantages and disadvantages in terms of ease of use, information density, computational costs, etc.
- Another generic challenge is to analyze temporal information since in virtually all cases this corresponds to simultaneously analyzing multiple timesteps concurrently. Generally, this is not possible in an in situ setting, as we rarely have enough memory available to maintain two let alone more time steps in the system. Therefore, time dependent analysis will virtually always start from reduced data and face the challenge that one must predict what might be interesting in the next step, i.e. whether a certain small scale undulation is noise to be ignored or the birth of a feature that will grow.
- Finally, any data reduction will have to come with error bounds, verification, and guarantees on error propagation to be trusted by the scientists. It will be especially crucial to integrate with the necessary uncertainty quantification as well as to consider the effects of data reduction on downstream processing, i.e. statistics or machine learning.

Conclusion. In all these cases, the challenge is that there exist a vast number of use cases all with different constraints and characteristics and many of the crucial questions, i.e. what information must be preserved, are either not known at all or cannot be easily computed on-the-fly. Furthermore, many large scale codes actually support an entire community of scientist who would like to exploit the data to answer questions unrelated or even orthogonal to those of the team actually running the simulation. In these cases, a direction of inquiry may not even be known when one has to make the decision to permanently delete some information. In general, the data reduction must support the entire range of scientific endeavors from open ended exploration to checking explicit hypotheses. Yet both conditions may apply for different data consumers leading to opposing constraints.

Progress in any of these challenges could greatly increase the value of any particular simulation and significantly reduce the number of repeats of the same (or similar) simulations to recover and/or explore previously unknown phenomena or ones not sufficiently captured on earlier attempts.

The existing body of work on in situ data reduction and compression is prevalently tailored to specific application use cases. It would be desirable to identify generic techniques that can support a large variety of use cases.

References

1 W. Widanagamaachchi, J. Chen, P. Klacansky, V. Pascucci, H. Kolla, A. Bhagatwala and P. Bremer, "Tracking features in embedded surfaces: Understanding extinction in turbulent combustion", in 5th IEEE Symposium on Large Data Analysis and Visualization, LDAV 2015, Chicago, IL, USA, October 25-26, 2015, J. Bennett, H. Childs and M. Hadwiger, Eds., IEEE Computer Society, 2015, pp. 9–16. 10.1109/LDAV.2015.7348066.

4.2 Workflow Specification

Tom Peterka (Argonne National Laboratory, US, tpeterka@mcs.anl.gov)
Madhusudhanan Srinivasan (KAUST – Thuwal, SA, madhu.srinivasan@kaust.edu.sa)

License Creative Commons BY 3.0 Unported license
© Tom Peterka and Madhusudhanan Srinivasan

Contributors: M. Parashar, P.-T. Bremer, K. Heitmann, M. Larsen, J. Patchett, T. Peterka, M. Srinivasan, N. Röber, S. Frey, B. Raffin

Background & Motivation. The problem of workflow specification is to specify the in situ discovery process as a composition of different tools and applications (tasks): What are the inputs and outputs of each task? When and where should a task execute? What resources are needed by each task?

The style of the specification (procedural, declarative), level of detail required (detailed, abstract), degree of problem domain specificity (domain-specific, generic), and the modification of the individual tasks (level of intrusion) are aspects of all of the above questions.

Workflow specification is closely tied to workflow execution because the specification also implies the functions needed to be carried out by the workflow runtime engine to execute the workflow. Here, we focus on the specification from the end-user's and developer's perspective: how humans describe the workflow to the runtime and instrument (modify) tasks to operate within the workflow.

Challenges. An in situ workflow specification must address several factors. Some are common to other workflows such as distributed-area ones, but other factors are particularly relevant when tasks are combined in situ in an HPC system. The challenges that are particular to in situ workflows are as follows:

- Dynamic, data dependent behavior
- Space, time, resource, and priority constraints
- Contingencies for unexpected behavior such as faults or resource unavailability
- Balance between generic and domain-specific abstractions
- Minimizing the level of intrusion when modifying tasks from standalone to in situ mode

Some post hoc distributed-area workflow systems such as Fireworks [1] and Galaxy [2] are limited to a specific domain such as material science or biology. Others such as Pegasus [3] are generic. For in situ workflows, systems such as ADIOS [4] are derived from storage systems, while SENSEI [5] is derived from visualization. Decaf [6] and FlowVR [7] are dataflow layers for workflows whose data model is generic and not tied to storage or visualization, with the workflow graph being statically described in a Python script. Swift [8], in comparison, is a programming language that derives a workflow graph implicitly from the data dependencies in the program. COSS [9] is a contractual system primarily for storage models, however a similar idea may be applied to declarative workflow definitions in the future. Data-driven in-situ [10] and wide-area [11] workflow services have also been built using the DataSpaces [12] framework.

Research Questions. The following research questions require further study:

1. Defining dynamic, data dependent, or system dependent behavior. How do we specify how the workflow should adapt when conditions change? Examples of workflow changes range from the amount of resources allocated to existing tasks to topological changes in the workflow graph when tasks begin and end. Sources of workflow changes are twofold: application/data behaviors may warrant a workflow change (e.g., a feature appears), or system behaviors may force a change (e.g., a compute node becomes unresponsive).
2. Defining data behaviors that drive workflow dynamics. How are features, patterns, or statistics of the data described, and how are the resultant workflow actions described?
3. Specifying space, time, resource, and priority constraints. In situ workflows are constrained by having to share resources among more than one task. This means that space and time constraints on resources need to specified in some way. Moreover, it is unlikely that all constraints can be satisfied, meaning that constraints need to be prioritized. Such priorities also need to be defined in the workflow specification.
4. Contingencies for unexpected behavior such as faults or resource unavailability. Even after data and system behaviors have been defined, constrained, and prioritized, execution may not proceed as planned. Resources may become temporarily unavailable, fail altogether, or results may be corrupted. Contingency plans may need to be specified ahead of time (as part of workflow specification) if faults should be handled in custom ways.
5. Balance between generic and domain-specific abstractions. The level of abstraction at which all of the above definitions are specified is an open question. Some workflow systems are limited to a specific domain while others are quite generic. However, no systems define all of the above behaviors, and going forward, the appropriate level of abstraction remains to be studied. Multiple levels of abstraction (domain-specific data behaviors, system-specific machine behaviors, generic constraints and priorities) have not been studied to the best of our knowledge.
6. Balance between declarative and procedural workflow specification. As in single applications, traditionally procedural methods have been used in the past, defining workflows in terms of tasks and their connections. Going forward, more concise declarative methods may be appropriate and easier to use, with the workflow system converting the user's declarations into workflow procedures. One example is the specification of data contracts defining input and output data models, from which the system builds a workflow automatically (similar to SQL database language).

Conclusion. A concise and easy to use workflow specification interface has many advantages. The first is workflow execution. Because of its close relationship with the runtime execution engine, a well-defined, concise method of description facilitates efficient execution. Conversely, poorly designed abstractions can lead to unreliable implementations. User productivity is another benefit. We postulate that end users (i.e., application scientists) can make better use of their own time as well as their resources when a workflow automates in situ tasks. However, this hypothesis only holds when the workflow specification is intuitive, easy to use, and affords a clear mapping between the scientist's mental model of the workflow and its formal description. The third gain comes from reuse of the workflow. A formal workflow specification enables replication of computational experiments, facilitating verification, validation, and scientific reproducibility. Moreover, similar workflows can be defined by editing the specification of an existing workflow rather than starting over. The fourth asset is collaboration and communication. The workflow specification can be shared among researchers and used for training the next generation of data and domain scientists.

References

1. A. Jain, S. P. Ong, W. Chen, B. Medasani, X. Qu, M. Kocher, M. Brafman, G. Petretto, G.-M. Rignanese, G. Hautier, D. Gunter and K. A. Persson, "Fireworks: A dynamic workflow system designed for high-throughput applications", Concurrency and Computation: Practice and Experience, vol. 27, no. 17, pp. 5037–5059, 2015. 10.1002/cpe.3505.
2. E. Afgan, J. Goecks, D. Baker, N. Coraor, A. Nekrutenko and J. Taylor, "Galaxy: A gateway to tools in e-science", in Guide to e-Science: Next Generation Scientific Research and Discovery, X. Yang, L. Wang and W. Jie, Eds. London: Springer London, 2011, pp. 145–177. 10.1007/978-0-85729-439-5_6.
3. E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. F. da Silva, M. Livny and K. Wenger, "Pegasus, a workflow management system for science automation", Future Generation Computer Systems, vol. 46, pp. 17–35, 2015. 10.1016/j.future.2014.10.008.
4. D. A. Boyuka, S. Lakshminarasimham, X. Zou, Z. Gong, J. Jenkins, E. R. Schendel, N. Podhorszki, Q. Liu, S. Klasky and N. F. Samatova, "Transparent in situ data transformations in adios", in 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, May 2014, pp. 256–266. 10.1109/CCGrid.2014.73.
5. U. Ayachit, B. Whitlock, M. Wolf, B. Loring, B. Geveci, D. Lonie and E. W. Bethel, "The sensei generic in situ interface", in 2016 Second Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV), Nov. 2016, pp. 40–44. 10.1109/ISAV.2016.013.
6. M. Dreher and T. Peterka, "Decaf: Decoupled dataflows for in situ high-performance workflows", Jul. 2017. 10.2172/1372113.
7. M. Dreher and B. Raffin, "A flexible framework for asynchronous in situ and in transit analytics for scientific simulations", in 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, May 2014, pp. 277–286. 10.1109/CCGrid.2014.92.
8. J. M. Wozniak, T. G. Armstrong, M. Wilde, D. S. Katz, E. Lusk and I. T. Foster, "Swift/t: Large-scale application composition via distributed-memory dataflow processing", in 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, May 2013, pp. 95–102. 10.1109/CCGrid.2013.99.
9. M. Dorier, M. Dreher, T. Peterka and R. Ross, "Coss: Proposing a contract-based storage system for hpc", in Proceedings of the 2Nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems, ser. PDSW-DISCS '17, Denver, Colorado: ACM, 2017, pp. 13– 18. 10.1145/3149393.3149396.
10. Z. Wang, A. Simonet, P. Subedi, P. E. Davis and M. Parashar, "Leveraging scalable event distribution to enable data-driven in-situ scientific workflows", in RDI2 Techical Report, 2018.
11. M. F. Aktas, J. Diaz-Montes, I. Rodero and M. Parashar, "Wa-dataspaces: Exploring the data staging abstractions for wide-area distributed scientific workflows", in Parallel Processing (ICPP), 2017 46th International Conference on, IEEE, 2017, pp. 251–260.
12. C. Docan, M. Parashar and S. Klasky, "Dataspaces: An interaction and coordination framework for coupled simulation workflows", Cluster Computing, vol. 15, no. 2, pp. 163–181, Jun. 2012. 10.1007/s10586-011-0162-y.

4.3 Workflow Execution

Hank Childs (University of Oregon – Eugene, US, hank@uoregon.edu)
John Patchett (Los Alamos National Laboratories, US, patchett@lanl.gov)

License Creative Commons BY 3.0 Unported license
© Hank Childs and John Patchett

Contributors: J. Patchett, H. Childs, A. Bauer, P.-T. Bremer, T. Carrard, M. Dorier, C. Garth, K. Heitmann, K. Moreland, T. Peterka, D. Pleiter, D. Pugmire, N. Röber

Background and Motivation. The fundamental challenge for workflow execution is to be able to execute a workflow specification. This challenge involves deriving the set of tasks to execute, scheduling those tasks including the hardware resources the tasks will use, managing invocation and execution of the software that carries out the tasks, and handling error conditions from small to large. This problem spans multiple granularities. A coarse example could involve scheduling distinct binaries that exchange data via the file system or a network connection. A finer example could involve coordinating modules within a single binary.

This topic is challenging for many reasons. One challenge is that the workflow specifications may be quite complex. For example, they may include hierarchical graphs, graphs that evolve dynamically over time, or event-driven workflows. Workflow specifications also must consider a rich set of use cases, including multiple types of programs, data exchanges, etc., and it can be quite difficult to support all of these use cases. A second challenge involves scheduling workflow tasks onto resources. This is due to several reasons. For one, the available resources are often variable, which complicates workflow scheduling with dynamic allocation. It is not always the case that new resources can be allocated when needed, and so workflow execution needs to be capable of dealing with a lag in obtaining new resources. Additionally, the resources needed may be variable. It is important that workflows can be elastic in their usage of the resources, i.e., adapt resource usage across tasks dynamically to speed up the overall workflow. Finally, the penalty for allocating sub-optimally can be quite large. If one task is given insufficient resources, then it can become a bottleneck for the entire workflow, leading to significant delays. A third challenge focuses around portability. Supercomputers increasingly have heterogeneous hardware, and workflow execution may need to schedule different types of hardware, and be composed of software that can run on different types of hardware. Execution environments differ as well, ranging from the high-level paradigm (e.g., task-based) to programming environment (MPI, OpenMP, etc). Workflow execution must be flexible with respect to this issue. A final type of portability involves data models, ranging from linking between distinct physics codes (i.e., remapping in a volume preserving way) to the ability to map from the data model of one program to another.

Most of the community successes have come as smaller contributions to a specific domain. The scientific visualization tools ParaView, VisIt, and EnSight all employ client-server models. With these tools, the visualization algorithms execute in parallel on the server and then transfer the resulting geometric primitives to a client for interactive rendering. To do this, they needed aspects of a workflow, including co-allocation of resources and data exchange. Through their in situ interfaces, these tools also can do computational steering. An example of this comes with the Uintah code, which has interfaced with LibSim and allows LibSim to set parameters that affect Uintah's execution. ADIOS [1] approached this by interfacing with simulation codes through the I/O layer. When simulation codes attempt to write through the ADIOS API calls, data is sometimes transferred to distinct resources where additional analysis can occur. ADIOS also supports running distinct binaries on the same

resources as the simulation. Damaris, a middleware for efficient I/O on HPC simulations, also can facilitate visualization, again in a variety of configurations. Henson has focused on workflows within the same binary. Their main abstractions are position-independent executables and co-routines. Through these abstractions, Henson allows arbitrary analysis code to run alongside a simulation. Decaf is a dataflow system for the parallel communication of coupled tasks in an HPC workflow. It can be used to forward data from distinct processes, as well as redistributing data. It also can be used within a program to form a task-based computing environment. Decaf has been demonstrated in situ in several settings on HPC systems. Other notable examples of workflow systems include Fireworks [2] and Galaxy [3], which are limited to a specific domain such as material science or biology, Melissa, for the analysis of ensembles, and Pegasus, which is generic. Workflows also can be specified as programming languages, such as Swift, which creates workflow graphs from data dependencies in a program.

Research Questions. There are multiple open research questions in workflow execution:

1. A primary question deals with finding the right strategies that balance flexibility and efficiency. A related question involve finding the right abstractions. In particular, the tasks that the workflow manages should likely be neither too coarse (not enough opportunity to optimize) or too fine (too much to manage). Hierarchical graphs may allow for the best of both worlds. On the efficiency side, it is important to understand how to dynamically (elastically) adapt resource usage on the fly. On the flexibility side, reliability and effective fault tolerance are still open questions.
2. One question revolves around data interfaces. While it is often possible to link the data format from one program to another, an interface that captures the data formats of many programs is much harder. One approach is to create an interface that describes a data model, and have each program convert their internal structures to the data model. Another approach is to focus on passing arrays, and then separately pass a scheme that conveys the meaning of each array.
3. Finally, excellent research in workflows is happening outside the visualization community. In some cases, the questions above are about understanding the unique requirements for visualization and analysis and adapting existing solutions to deal with these requirements.

Conclusion. Workflows can help in multiple ways. One outcome is improved resource utilization. This may occur via adapting execution to make sure all resources are used or via fault tolerance and ensuring work that has already occurred is not lost. Another possible benefit for resource utilization is that the modularization provided by workflows can enable algorithms to run on the hardware where they are most effective. Another outcome is optimizing people effort. Workflows can spare end users from having to master nuances of HPC systems (by encoding these nuances in the workflow execution) and can also spare development time via reduced complexity (although there is an overhead to incorporate a workflow in the first place).

References

1 J. F. Lofstead, S. Klasky, K. Schwan, N. Podhorszki and C. Jin, "Flexible io and integration for scientific codes through the adaptable io system (adios)", in Proceedings of the 6th International Workshop on Challenges of Large Applications in Distributed Environments, ser. CLADE '08, Boston, MA, USA: ACM, 2008, pp. 15–24. 10.1145/1383529.1383533.

2 A. Jain, S. P. Ong, W. Chen, B. Medasani, X. Qu, M. Kocher, M. Brafman, G. Petretto, G.-M. Rignanese, G. Hautier, D. Gunter and K. A. Persson, "Fireworks: A dynamic workflow system designed for high-throughput applications", Concurrency and Computation: Practice and Experience, vol. 27, no. 17, pp. 5037–5059, 2015. 10.1002/cpe.3505.
3 E. Afgan, J. Goecks, D. Baker, N. Coraor, A. Nekrutenko and J. Taylor, "Galaxy: A gateway to tools in e-science", in Guide to e-Science: Next Generation Scientific Research and Discovery, X. Yang, L. Wang and W. Jie, Eds. London: Springer London, 2011, pp. 145–177. 10.1007/978-0-85729- 439-5_6.

4.4 Exascale Systems

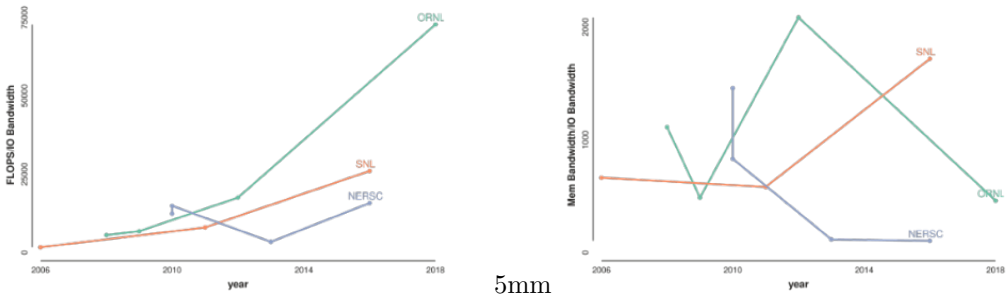
Kenneth Moreland (Sandia National Laboratories – Albuquerque, US, kmorel@sandia.gov)
Dirk Pleiter (Jülich Supercomputing Centre, DE, d.pleiter@fz-juelich.de)

License Creative Commons BY 3.0 Unported license
© Kenneth Moreland and Dirk Pleiter

Contributors: T. Peterka, K. Moreland, K. Isaacs, B. Muite, D. Pleiter, B. Raffin, U. Rüde, R. Sisneros, G. H. Weber

Background and Motivation The exascale system represents many paradigm shifts that absolutely necessitate the shift to in situ analysis and visualization. Already well established by the visualization community, and backed up by many applications scientists [1], is the need of in situ visualization as a response to the growing gap between compute rate and disk storage bandwidth. We expect other relevant fundamental changes to HPC processing at the exascale. HPC workload is shifting from a traditional solver PDE workload to a mixture of multiple physics, analytics, visualization, learning, and other complex processing units. For in situ visualization to be viable in this complex software ecosystem, we must better understand exascale system requirements for in situ applications; we must map application needs to system and analysis features, and we must identify gaps in current HPC systems for a set of in situ workloads viable for the exascale.

Challenges. HPC design and its use are a shifting target. HPC design, both known and unknown, present both interesting challenges and opportunities. It is generally observed that the throughput of floating-point operations is increasing at a much higher speed compared to I/O bandwidth, such that their ratio seems to increase exponentially (see Fig. 1, left panel). That is, I/O to an external storage system becomes ever more challenging. Newly presented to the community is the observation that maintaining a high aggregate memory bandwidth became challenging, too, resulting in a decline of the ratio of throughput of floating-point operations and memory bandwidth (see Fig. 1, right panel). This may be in part to a (possibly temporary) trend in using fewer nodes per computer (thus indirectly reducing the total number of memory buses). Regardless, it is worthwhile to closely monitor this trend as its behavior may drastically affect the consequential research of in situ visualization. For example, perhaps integrating visualization too tightly with simulation may divide an already constrained bandwidth. One option to mitigate the memory bandwidth challenge is to use new high-bandwidth memory technologies like HBM, which are currently, for example, used for high-end GPUs. For cost reasons this may, however, result in a degradation of the available memory capacity.



5mm

Figure 1 At left is a plot of the FLOPS to disk I/O bandwidth for the last three generations of computers at ORNL, NERSC and SNL/LANL respectively. At right is the ratio of the memory bandwidth to disk I/O bandwidth for the same computers.

A possible strategy for mitigating the capacity problem is the integration of persistent memory, which appears to be a common theme for exascale systems in a variety of forms. One such well documented application for persistent memory is the concept of burst buffers [2], which provides storage areas typically based on non-volatile memory (NVRAM) like NAND Flash as a staging area for slower external storage. Such buffers may provide an opportunity to intercept simulation data at a lower cost than that of a disk-based storage system. Another opportunity includes the integration of NVRAM into the compute nodes, where it could either be attached to the processors as a storage device or as memory. Although the non-volatile aspect of NVRAM may be secondary to high density (and thus high storage capacity) and low storage power for the base applications [3], the introduction of such memory may open up many opportunities such as the injection of in transit visualization capabilities while memory is staged.

As the scale of supercomputers increase, the feasibility of building dedicated visualization clusters decreases [4]. However the possibility of providing heterogeneous nodes in new supercomputers specialized for analysis and visualization processing still remains. Such dedicated nodes integrated into the rest of the supercomputer fabric enables much in situ processing, particularly of the in transit variety. As the complexities of simulations advance and in situ visualization is incorporated into the workflow, understanding the performance behavior of visualization algorithms becomes more critical. Currently, the performance behavior of in situ systems is not well understood, and in fact the number of potential in situ approaches and implementations are too vast to effectively study. Although research in new in situ approaches should continue to remain a priority, we need to downselect the potential space of probable in situ possibilities to the most probable (and effectively implemented) solutions to really measure performance. We also recognize that exascale and post-exascale machine definitions are not set. This gives the visualization community an opportunity to provide input to decision makers on what features should be provided by upcoming systems. Such recommendations can only be given if the performance of (the aforementioned downselected) in situ visualization capabilities are well understood.

Successes. There has been some introductory work on performance modeling of visualization algorithms, but much more remains to be done. Tools like Ascent [5] and SENSEI [6] provide small sample applications that can be used for targeted studies of performance. For a formalized study, we first need an ontology of in situ techniques to guide the experiments performed. The In Situ Terminology Project [7] provides a classification system that breaks down current in situ visualization techniques. See Section 4.8 on cost models for

more information. We endeavor to make in situ visualization capabilities an integral part of the design of an overall exascale system. Generally, co-design is used to ensure that future generations of supercomputers are well-suited to the applications to be run on them. Although visualization needs have previously been poorly represented, the model of co-design with science applications is an established model we can apply to our needs. As in situ visualization becomes more critical for scientific computation, we expect to be in better position to be included in co-design efforts. Co-design has become an established methodology towards exascale computing (see, e.g., [8]). In a co-design process, application and system software developers make their needs and requirements available to system architects. On this basis these will be able to identify the most efficient solutions and perform trade-off decisions. On the other hand, system architects and system technology experts should provide application and system software developers with expertise on both limitations of the underlying architecture and technologies as well as opportunities, which could be exploited. One example where co-design could be beneficial is the aforementioned NVRAM integration. Know-how on key performance characteristics, realisable memory capacities and possible access interfaces will empower developers of in situ visualisation software about design solutions, which could allow to efficiently exploit this NVRAM. Defining the needs of workflows involving in situ visualisation components will allow system architects to choose between different NVRAM solutions as well as APIs for accessing this memory.

Research Questions. To be successful in a co-design for future computing platforms, it is important to express needs and desires at an appropriate level with metrics that are meaningful to vendors. For example, in the context of co-design it is a poor idea to express a requirement in the form of a need for a burst buffer because doing so is unnecessarily prescriptive to a particular technology when another technology may satisfy the needs as well or better. Instead, the focus should be on defining needs and requirements, e.g. in terms of needing N amount of memory capacity with M amount of bandwidth for a duration of T time. Research might also identify potentially new elements, such as containers, that expand the capabilities of in situ visualization and ease the integration of such components. The current state of the art in visualization is quite poor at predicting the system needs of our own systems in isolation let alone understanding how they perform in a larger in situ workflow or when mapped to differentsystem components.

Our first task in understanding visualization performance better is to come up with an ontology of classes of general in situ processing. The In Situ Terminology Project has already identified 6 separate (but interdependent) axes of in situ features with each axis containing many potential values, and for the most part the terminology does not detail the actual analysis and visualization algorithms to be run. This expansive space is likely too vast to densely measure, so a certain amount of downselecting is important to identify the classes of in situ visualization most likely to be employed in practice. That is not to say that research of possible in situ solutions should similarly be constrained, and such unconstrained research may in turn adjust the priorities considered for supercomputer co-design. However, when building cost models this downselection is important to make the problem tractable.

There are numerous expected features of exascale computing that will prove challenging. As ever, the next generations of supercomputers will feature more parallelism: potentially many thousands of nodes and an aggregate of billions of computing threads. In situ visualization exacerbates the challenge by requiring the visualization to run on more computing threads than strictly necessary. This is because even large visualizations are traditionally run at smaller scales than the simulation (rule of thumb dictates 5-10% the size). Consequently, visualization run in situ either needs to take a leap in the scalability of the software or employ in situ techniques that separates visualization processing to smaller computing groups.

New memory structures, types, and hierarchies are also expected to provide numerous challenges and opportunities. As NVRAM continues to become more cost effective, their use in supercomputers will likely expand. Using NVRAM in SSD form for storage components like burst buffers remains a potential use case, but NVRAM may be considered in different ways as well. For example, their high data density and ability to retain data at low power make them attractive as a potential component of the addressable memory system, which could allow them to double as a communication mechanism between workflow components (such as simulation and visualization).

Also of note about the memory system is the observation that, although in situ research has primarily focused on the growing gap between throughput of floating-point operations and storage bandwidth, a similar trend can be seen between execution rate and the RAM memory bandwidth. As we use in situ visualization to move analysis closer to simulation, we may be exacerbating problems with limited memory bandwidth in favor of relieving storage bandwidth. It is an open question of how much, if at all, an in situ visualization affects the available memory bandwidth to other workflow functions and, if so, how the work may be divided across different memory busses.

Conclusion. Addressing these issues will enable better support of in situ visualization on exascale supercomputers and beyond. A better understanding of the behavior of our own visualization systems will allow us to both steer future HPC design to better enable and, conversely, adjust in situ visualization R&D to better perform on available platforms. Ultimately, such research and co-design will lead to more effective application of in situ visualization.

References

1. R. Gerber, J. Hack, K. Riley, K. Antypas, R. Coffey, E. Dart, T. Straatsma, J. Wells, D. Bard, S. Dosanjh, I. Monga, M. E. Papka and L. Rotman, "Crosscut report: Exascale requirements reviews, march 9–10, 2017 – tysons corner, virginia. an office of science review sponsored by: Advanced scientific computing research, basic energy sciences, biological and environmental research, fusion energy sciences, high energy physics, nuclear physics", US Department of Energy Office of Science and Technical Information, Tech. Rep., Jan. 2018. 10.2172/1417653.
2. J. Bent, G. Grider, B. Kettering, A. Manzanares, M. McClelland, A. Torres and A. Torrez, "Storage challenges at los alamos national lab", in 012 IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST), Apr. 2012, pp. 1–5. 10.1109/MSST.2012.6232376.
3. A. M. Caulfield, J. Coburn, T. Mollov, A. De, A. Akel, J. He, A. Jagatheesan, R. K. Gupta, A. Snavely and S. Swanson, "Understanding the impact of emerging non-volatile memories on high-performance, io-intensive computing", in SC '10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, Nov. 2010, pp. 1–11. 10.1109/SC.2010.56.
4. H. Childs, "Architectural challenges and solutions for petascale postprocessing", Journal of Physics: Conference Series, vol. 78, no. 1, p. 012012, 2007. 10.1088/1742-6596/78/1/012012.
5. M. Larsen, J. Ahrens, U. Ayachit, E. Brugger, H. Childs, B. Geveci and C. Harrison, "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, ser. ISAV'17, Denver, CO, USA: ACM, 2017, pp. 42–46. 10.1145/3144769.3144778.

6 U. Ayachit, B. Whitlock, M. Wolf, B. Loring, B. Geveci, D. Lonie and E. W. Bethel, "The sensei generic in situ interface", in 2016 Second Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV), Nov. 2016, pp. 40–44. 10.1109/ISAV.2016.013.
7 The in situ terminology project.
8 R. F. Barrett, S. Borkar, S. S. Dosanjh, S. D. Hammond, M. A. Heroux, X. S. Hu, J. Luitjens, S. G. Parker, J. Shalf and L. Tang, "On the role of co-design in high performance computing", in, J. J. D. E. H. D'Hollander, I. T. Foster, L. Grandinetti and G. R. Joubert, Eds. IOS Press, 2013, vol. 24. 10.3233/978-1-61499-324-7-141.

4.5 Algorithmic Challenges

Christoph Garth (Technische Universität Kaiserslautern, DE)
David Pugmire (Oak Ridge National Laboratory, US)

License Creative Commons BY 3.0 Unported license
© Christoph Garth and David Pugmire

Contributors: C. Garth, D. Pugmire, T. Carrard, B. Muite, K. Moreland, U. Rüde, R. Sisneros, D. Pugmire, H. Yu, H.-W. Shen, G. Weber, I. Hotz, R. Westermann, J. Krüger, M. Hadwiger, P. Messmer

Background and Motivation. The utility of in situ visualization and analysis for state-of-the-art problems fundamentally hinges on the ability to respond to both the size and nature of simulation data. In place methods will need to efficiently utilize similar levels of concurrency as the mechanism for generation of the data (i.e. simulation), whereas in transit methods will need to dedicate enough computational power to process the transferred data. Further, algorithms must be able to operate on disparate data representations (e.g. higher order elements) in a memory efficient manner, and for in place methods particularly, must respect the sharing of resources.

Challenges. Designing and engineering in situ visualization and analysis algorithms for extremely concurrent architectures, where the need for such analysis is most pressing, is challenging on a variety of levels.

Algorithms need to integrate into a complex ecosystem formed by the combination of simulation codes and heterogeneous architectures. This integration requires sharing of data between the simulation and the algorithms. The implications of this include working with foreign data structures and partitioning schemes, operating within memory requirements, and handling the complexity of data locality inherent in increasingly deeper memory hierarchies. These complexities will make it difficult for algorithms to perform efficiently.

The coupled nature of in situ processing also enforces specific access patterns that may be unsuitable in an analysis algorithm; using memory to store needed data is typically not possible. For the typical case of e.g. sequential-in-time coupling, commonly used algorithms like feature tracking appear difficult or even impossible to realize. Moreover, in complex analyses, several algorithms may require entirely different data layouts to function with acceptable efficiency.

In many applications, analysis workloads can be more diverse than the primary (i.e. simulation) workload, yet have to be adapted to the environment provided by the simulation. It is unclear how algorithm design can account for such heterogeneous runtime environments effectively.

Most fundamentally, looking ahead to future architectures, where the need for in situ analysis will be most pressing, algorithms with strongly sublinear scaling and/or global access patterns (e.g. topological analysis that is global in nature) may not even be feasible to use due to prohibitive cost of communication.

While some paradigms such as in transit processing can alleviate some of these problems, the fundamental problem of designing scalable algorithms of a strongly adaptive nature remains. As a consequence, some forms of analysis that are in widespread use today may not even be feasible for future problems.

Consider for example the case of particle tracing to produce streamlines or pathlines for the visualization of vector fields, e.g. in the study of turbulence. The generally imbalanced nature of the computation that results from the data itself induces strong limits on scalability in straightforward algorithms [1]. While load-balancing through data redistribution can address this issue [2], this may be very costly in practice. Another example includes the class of topological analysis algorithms for feature detection used in combustion, or halo identification in cosmology. These algorithms are used to compute connected components in the simplest case, up to full computation of the Morse-Smale complex [3], and essentially require global exchange of information for correctness. These types of methods appear to be difficult if current trends continue. In both cases, it would appear possible to address these issues through e.g. reduced-concurrency in transit processing or careful data reduction strategies.

Taking a broader view, exploratory analysis, which in addition to the above requirements must provide results in seconds, can in general can be made to fit these criteria.

Successes. It is difficult to define a general criterion for declaring a general visualization technique or specific algorithm applicable to in situ. One measure of success in this regard is precedent for the use of such an algorithm to address a science question at scale. (Other, much more difficult to satisfy criteria could rely on theoretical algorithm analysis or empirical performance models that predict acceptable properties.)

Considering this criterion, there are several noteworthy successes with adapting visualization techniques to in situ scenarios at scale. For example, volume rendering (and other ray casting-based and rasterization-based rendering techniques) has been successfully scaled to millions of cores in an in situ scenario. The communication patterns underlying rendering algorithms are well understood, local, and can flexibly accommodate data layouts in many cases [4]. The same holds true for isosurface extraction and strongly localized feature extraction techniques [5].

Furthermore, in some instances it is possible to decompose a particular analysis scenario into a component that is conceivably easily adapted to an in situ mode that produces an intermediate representation plus a post hoc component that facilitates analysis; for example, using a Lagrangian basis to represent vector fields [6].

Research Questions. Based on the above observation, we identify the following general research questions that are central to engineering and designing algorithm for an in situ ecosystem:

1. How can in situ ecosystems be characterized sufficiently to inform algorithm design? Which parameters (such as concurrency, memory requirements, heterogeneity), modalities (e.g. data type, data locality, partitioning and layout schemes, communication layout, in transit facilities, memory hierarchy), and available resources (compute, memory, I/O bandwidth) are important in this context?

2. Are there specific resources (e.g. half-precision hardware operations) or data types (e.g. higher-order elements) that future in situ analysis algorithms can leverage to substantial benefit? I.e. how can an analysis algorithm be optimally accommodated by the environment?
3. How can requirements regarding the above parameters, modalities, and resources be formulated in a general enough manner that it is possible to understand how to accommodate them in an existing ecosystem?
4. How can algorithms be designed to optimally leverage different aspects of an in situ system (in situ, in transit, post hoc)? What are general strategies to this effect?
5. Which alternative formulations of typical visualization algorithms are better suited for future in situ ecosystems? (For example, convolution-type algorithms are expected to work well on exascale systems – can these replace particle-tracing flow visualization?)
6. Can compression or data reduction (e.g. distribution-based descriptions) be used to increase the scalability and efficiency of in situ analysis algorithms? In the case of lossy representation, how can the trade-offs with respect to accuracy be quantified.
7. How do the above considerations change if in situ interactive exploration (mandating short response times) is considered, e.g. for computational steering applications?
8. Looking beyond classical use cases for field data, how can analysis other data types (such as e.g. streaming data) be incorporated into this characterization?

We note that Question 6 strongly interacts with the research questions raised in Topic 1: Data Quality and Compression.

Conclusion. Addressing these questions will ensure that future computational science pipelines can make use of a rich toolbox of appropriate analysis methodologies. Failure to achieve this stands to strongly limit the potential for insight into the complex underlying scientific problems.

Furthermore, a general understanding of the principles underlying these considerations would substantially ease the adaptation of techniques to new in situ scenarios as well as enable the development of new analysis strategies, thus increasing the productivity and robustness of computational science workflows.

References

1 D. Pugmire, H. Childs, C. Garth, S. Ahern and G. H. Weber, "Scalable computation of streamlines on very large datasets", in Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, Nov. 2009, pp. 1–12. 10.1145/1654059.1654076.

2 B. Nouanesengsy, T. Lee and H. Shen, "Load-balanced parallel streamline generation on large scale vector fields", IEEE Transactions on Visualization and Computer Graphics, vol. 17, no. 12, pp. 1785–1794, Dec. 2011. 10.1109/TVCG.2011.219.

3 A. Gyulassy, P.-T. Bremer, B. Hamann and V. Pascucci, "A practical approach to morse-smale complex computation: Scalability and generality", IEEE Transactions on Visualization and Computer Graphics, vol. 14, no. 6, 2008.

4 M. Larsen, C. Harrison, J. Kress, D. Pugmire, J. S. Meredith and H. Childs, "Performance modeling of in situ rendering", in SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Nov. 2016, pp. 276–287. 10.1109/SC.2016.23.

5 G. H. Weber, H. Childs and J. S. Meredith, "Efficient parallel extraction of crack-free isosurfaces from adaptive mesh refinement (amr) data", in IEEE Symposium on Large Data Analysis and Visualization (LDAV), Oct. 2012, pp. 31–38. 10.1109/LDAV.2012.6378973.

6 A. Agranovsky, D. Camp, C. Garth, E. W. Bethel, K. I. Joy and H. Childs, "Improved post hoc flow analysis via lagrangian representations", in 2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV), Nov. 2014, pp. 67–75. 10.1109/LDAV.2014.7013206.

4.6 Use Cases Beyond Exploratory Analysis

Katherine Isaacs (University of Arizona – Tucso, US, kisaacs@cs.arizona.edu)
Alejandro Ribes Cortes (EDF – Paris, FR, alejandro.ribes@edf.fr)

License Creative Commons BY 3.0 Unported license
© Katherine Isaacs and Alejandro Ribes Cortes

Contributors: K. Isaacs, J. C. Bennett, A. Bauer, E. W. Bethel, C. Hansen, A. Ribes Cortes, K. Ono, N. Gauger, F. Sadlo

Background and Motivation. The focus of in situ analysis has largely been exploratory analysis of the output of a single large-scale simulation. However, we see other use cases that will benefit from or even be critically enabled by in situ technology. We identified several such use cases, including

1. support for ensembles, e.g., for uncertainty quantification and decision optimization;
2. computational steering;
3. incorporation of other data sources, e.g., for data assimilation, data fusion, and model calibration; and
4. analysis of other data associated with the application such as algorithm convergence and machine performance.

These important topics have only received minimal attention so far.

Successes. Melissa and Sandia's Slycat have done in situ processing of ensemble runs for uncertainty quantification. Melissa is a framework to manage simulation runs of ensembles [1]. At present, the largest experiment treated by Melissa comprised an ensemble of 80.000 parallel simulations; it avoided 288 TB of storage. Techniques from one-shot optimization, may help determine what parameters to explore next in ensemble runs [2].

LBNL CAMERA's Xi-CAM [3] targets distributed workflows for visualization and analysis with a focus on experimental data as an additional data source, e.g., from the Advanced Light Source at LBNL or the Advanced Photon Source at ANL.

Performance analysis tools such as Vampir and Scalasca have handled execution traces (performance data) in an in situ manner for large scale simulation runs. Plasma fusion simulations have visualized performance data of run time, memory usage and other hardware usage, as well as data reductions to provide situational awareness of in situ performance analysis. Such additional data sources, especially for the input and assimilation of experimental data, may require new mathematical methods and guarantees. Existing research on data assimilation techniques, such as using Kalman filters [4], currently provide solutions using traditional post-hoc and file-based approaches. Adapting these techniques to an in-situ context remains a challenge

Coupling the Uintah Framework [5] with the VisIt toolkit allows scientists to perform parallel in situ visualization of runtime performance data and other ephemeral data. The coupling also provides the concept of a "simulation dashboard" which allows in situ computational steering and visual debugging.

Research Questions. In addition to common in situ challenges like scaling and platform heterogenity, these use cases add questions with regard to the complexity of performing multiple use cases simultaneously, the added data wrangling of multiple use cases, the number of runs (ensemble use cases), the increased temporal constraints (coordination of ensembles, interactivity for human-in-the-loop steering, decision making time bounds), the added data movement (additional data sources), as well as the increased contention for the same resources among those multiple use cases. Specifically, the we need to find answers to the following questions

1. How can we understand the constraints inside of which a solution must operate?
2. What mathematical guarantees can we put on the resulting analyses? This will critically hinge on successful collaborations between mathematicians and in situ specialists.
3. How can we bound its time and space complexities? This includes measures and models for how much complexity an in situ analysis adds to the overall process.
4. How can we develop workflows and abstractions that allow users to handle multiple simultaneous goals? This covers the aspects of both the technical integration of multiple simultaneous workflows as well as the user interface implications regarding the control of such a complex system.
5. How can we scale support for ensembles of 1 billion representatives and beyond and enable an analysis of the resulting high-dimensional data space?
6. How can we develop robust multi-scale and multi-physics models that are suitable for reliable, reproducible science? Among other aspects, this requires effective data integration techniques, which may or may not be transferred from existing approaches designed for classical post hoc workflows.

References

1 T. Terraz, A. Ribes, Y. Fournier, B. Iooss and B. Raffin, "Melissa: Large Scale In Transit Sensitivity Analysis Avoiding Intermediate Files", in The International Conference for High Performance Computing, Networking, Storage and Analysis (Supercomputing), Denver, United States, Nov. 2017, pp. 1–14. 10.1145/3126908.3126922.

2 S. Günther, N. R. Gauger and Q. Wang, "Simultaneous Single-Step One-Shot Optimization with Unsteady PDEs", ArXiv e-prints, Mar. 2015.

3 R. J. Pandolfi, D. B. Allan, E. Arenholz, L. Barroso-Luque, S. I. Campbell, T. A. Caswell, A. Blair, F. De Carlo, S. Fackler, A. P. Fournier, G. Freychet, M. Fukuto, D. Gürsoy, Z. Jiang, H. Krishnan, D. Kumar, R. J. Kline, R. Li, C. Liman, S. Marchesini, A. Mehta, A. T. N'Diaye, D. Y. Parkinson, H. Parks, L. A. Pellouchoud, T. Perciano, F. Ren, S. Sahoo, J. Strzalka, D. Sunday, C. J. Tassone, D. Ushizima, S. Venkatakrishnan, K. G. Yager, P. Zwart, J. A. Sethian and A. Hexemer, "Xi-cam: a versatile interface for data visualization and analysis", Journal of Synchrotron Radiation, vol. 25, no. 4, pp. 1261–1270, Jul. 2018. 10.1107/S1600577518005787.

4 G. Evensen, Data Assimilation – The Ensemble Kalman Filter. Springer, 2009. 10.1007/978-3-642-03711-5.

5 Q. Meng, A. Humphrey and M. Berzins, "The uintah framework: A unified heterogeneous task scheduling and runtime system", in 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, Nov. 2012, pp. 2441–2448. 10.1109/SCC.2012.6674233.

4.7 Exascale Enables / Requires Different Data

Bernd Hentschel (RWTH Aachen University, DE)
Dave Pugmire (Oak Ridge National Laboratory, US)

License Creative Commons BY 3.0 Unported license
© Bernd Hentschel and Dave Pugmire

Contributors: A. Bauer, E. W. Bethel, P.-T. Bremer, M. Dorier, M. Hadwiger, C. Hansen, K. Heitmann, I. Hotz, J. Krueger, J. Patchett, K. Moreland, K. Ono, T. Peterka, D. Pleiter, B. Raffin, A. Ribes Cortes, N. Rober, U. Rüde, F. Sadlo, H-W Shen, M. Srinivasan, G. Weber, R. Westermann, H. Yu

Background and Motivation. Approaching the exascale will profoundly change the way we handle data within integrated, in situ systems. Specific problems arise with respect to data models for both input and output data, such as increased resolution and/or the creation of large-scale ensembles, the inclusion of (streaming) experimental data, and the reliable execution of computational experiments.

First, driven by energy constraints, data movement will become a key bottleneck. Addressing this problem will require the development of shared data models that can be used for both simulation and visualization/analysis alike. This is problematic in the sense that today's formats are typically optimized for either the former or the latter.

Second, scientists will use next generation machines in different ways. Increasing spatio-temporal resolution will make future simulation results inherently multi-scale. Yet, the resolution of output displays is bounded by human perceptual limits. Beyond increased resolution, it is possible to use the system to generate large-scale ensembles by running hundreds or thousands of smaller-scale models in parallel. The effective summarization of ensembles including uncertainty quantification is yet unsolved.

Third, handling data from large-scale experiments will spawn additional requirements. This data might be streamed in real-time, e.g., from high-energy physics experiments thus requiring real-time processing.

Finally, we anticipate that effectively executing computational experiments in the exascale regime will require a constant monitoring of system status and performance data in order to cope with aspects of fault tolerance and optimal resource usage.

While all these problems do have an in situ perspective, we are aware that several aspects are not genuinely driven by the in situ paradigm; in fact, topics like ensemble visualization and uncertainty quantification have been worked on outside the context of large-scale data visualization for years.

Successes. To date, several efforts target the exchange of data between simulation codes and visualization exchange. From a conceptual point of view, theses efforts have in common that they try to provide a generic data model or an interface for the transformation from one model into another one. Conduit [1], ADIOS [2], and SENSEI [3] broadly fall into the first category, while the latter comprises, e.g., LibSim [4] and Catalyst [5].

A specific approach to address the need for common data structures are block-structured Adaptive Mesh Refinement (AMR) data [6]. AMR uses a hierarchy of axis-aligned rectilinear grids, which are interchangeably called either boxes, patches, or subgrids. These form the building blocks that represent the domain. These grids are ordered in a hierarchy of levels having increasing resolution, where data in finer levels replaces those in coarser levels. In turn, this can enable multi-resolution simulations such as mixed atomistic-continuum simulation of materials and multiscale cosmology simulations. Initial work on AMR visualization focused

on converting AMR data to suitable conventional representations and then visualizing these, e.g., [7]. While these first attempts converted the data into an unstructured mesh, later work focused on maintaining the AMR structure for direct and indirect volume rendering [8, 9, 10] with a focus on defining smooth interpolation schemes, extracting isosurfaces without discontinuities at hierarchy level boundaries. and efficient parallel implementations.

With respect to workflow systems, current implementations like Pegasus and Flux take a higher-level view of the data and provide a graph-based representation of the operations to be performed. These systems manage the dependencies, execution and resilience of executing the sequence of specified operations.

Research Questions. The aforementioned challenges directly translate into the following set of research questions.

1. With regard to data models, we need to analyze and understand what data models are available and which access patterns they facilitate. A profound understanding of predominant access patterns will help us develop new data models and implementations. Eventually, these should substitute existing models wherever feasible in order to increase data re-use and minimize energy-intensive data copies and/or data movement.
2. Effectively reducing data will require a way to transfer the successes of feature-based visualization from the post hoc world into an in situ setting. When dealing with multi-scale data, we need to research representations that allow for a fluid exploration of data at different scales.
3. Analogously, there are many open questions regarding good abstractions and/or summarizations of ensemble data. One can ask in how far good ensemble abstractions help optimize the setup of follow-up computational experiments, e.g., by providing information on parts of the parameters space that need to be sampled more densely?
4. The execution of large-scale computational experiments begs the question how much of the process can actually be described in a (semi-)formal fashion. Such a description could be used in order to reduce the burden on users; they would only describe what they need, leaving the questions of how, where, and when to a runtime system. The question then is how such descriptions should look like?

We note that most of these questions are not specific to the exascale regime. Yet, while pre-exascale setups will still allow some leeway, e.g., in terms of the level of integration, the arrival of exascale machines will put in situ techniques to their ultimate test.

References

1 M. Larsen, J. Ahrens, U. Ayachit, E. Brugger, H. Childs, B. Geveci and C. Harrison, "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, ser. ISAV'17, Denver, CO, USA: ACM, 2017, pp. 42–46. 10.1145/3144769.3144778.

2 J. F. Lofstead, S. Klasky, K. Schwan, N. Podhorszki and C. Jin, "Flexible io and integration for scientific codes through the adaptable io system (adios)", in Proceedings of the 6th International Workshop on Challenges of Large Applications in Distributed Environments, ser. CLADE '08, Boston, MA, USA: ACM, 2008, pp. 15–24. 10.1145/1383529.1383533.

3 U. Ayachit, B. Whitlock, M. Wolf, B. Loring, B. Geveci, D. Lonie and E. W. Bethel, "The sensei generic in situ interface", in 2016 Second Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV), Nov. 2016, pp. 40–44. 10.1109/ISAV.2016.013.

4 B. Whitlock, J. M. Favre and J. S. Meredith, "Parallel In Situ Coupling of Simulation with a Fully Featured Visualization System", in Eurographics Symposium on Parallel Graphics and Visualization, T. Kuhlen, R. Pajarola and K. Zhou, Eds., The Eurographics Association, 2011. 10.2312/ EGPGV/EGPGV11/101-109.
5 N. Fabian, K. Moreland, D. Thompson, A. C. Bauer, P. Marion, B. Gevecik, M. Rasquin and K. E. Jansen, "The paraview coprocessing library: A scalable, general purpose in situ visualization library", in 2011 IEEE Symposium on Large Data Analysis and Visualization, Oct. 2011, pp. 89–96. 10.1109/LDAV.2011.6092322.
6 M. J. Berger and P. Colella, "Local adaptive mesh refinement for shock hydrodynamics", J. Comput. Phys., vol. 82, no. 1, pp. 64–84, May 1989. 10.1016/0021-9991(89)90035-1.
7 M. L. Norman, J. Shalf, S. Levy and G. Daues, "Diving deep: Data-management and visualization strategies for adaptive mesh refinement simulations", Computing in Science Engineering, vol. 1, no. 4, pp. 36–47, Jul. 1999. 10.1109/5992.774839.
8 F. Wang, I. Wald, Q. Wu, W. Usher and C. R. Johnson, "CPU Isosurface Ray Tracing of Adaptive Mesh Refinement Data", IEEE Transactions on Visualization and Computer Graphics, Jan. 2019.
9 I. Wald, C. Brownlee, W. Usher and A. Knoll, "Cpu volume rendering of adaptive mesh refinement data", in SIGGRAPH Asia 2017 Symposium on Visualization, ser. SA '17, Bangkok, Thailand: ACM, 2017, 9:1–9:8. 10.1145/3139295.3139305.
10 G. H. Weber, H. Childs and J. S. Meredith, "Efficient parallel extraction of crack-free isosurfaces from adaptive mesh refinement (amr) data", in IEEE Symposium on Large Data Analysis and Visualization (LDAV), Oct. 2012, pp. 31–38. 10.1109/LDAV.2012.6378973.

4.8 Cost Models

Robert Sisneros (University of Illinois at Urbana-Champaign, US, sisneros@illinois.edu)
Christoph Garth (Technische Universität Kaiserslautern, DE, garth@cs.uni-kl.de)

License Creative Commons BY 3.0 Unported license
© Robert Sisneros and Christoph Garth

Contributors: D. Pugmire, R. Sisneros, T. Carrard, S. Frey, M. Larsen, K. Isaacs, N. Gauger, B. Muite, C. Garth

Background and Motivation. Cost models play a role for several topics fundamental to both simulations as well as in situ frameworks, including data reduction and quality, workflows, and algorithms. For many domains, the ability to estimate costs is even prerequisite to the approval or initiation of efforts. Creating a cost model, through performance modeling or alternative methods, facilitates the efficient implementation and application of in situ frameworks in that it serves to place bounds on the uncertainty introduced by performing in situ analysis and visualization. The simulations for which we deploy in situ data analysis and visualization approaches commonly have associated cost models, i.e. we expect application developers to understand the resource requirements of their applications. However, extending a particular cost model to incorporate even the deployment of a specific in situ method is likely to present significant challenges. That is, an in situ method presents many potential impacts to a simulation: increased run time, additional memory requirements, higher power consumption, etc.

Challenges. The typical challenges associated with creating a cost model for any application apply here as well. While we may readily describe a cost model's parameter space the process of sampling and benchmarking this is onerous. This is compounded by the existence of multiple relevant hardware configurations and the fact that in many cases architectural factors may have non-linear effects, which are hard to quantify, or be entirely provisional. These issues are magnified for in situ approaches as a cost model parameter space must incorporate hardware considerations as well as two independent, diverse applications (the visualization/analysis method to be run in situ and the simulation itself). The problem is further exacerbated in that an in situ framework may be composed of multiple algorithms, and visualization algorithms exhibit a strong data dependency.

Successes. As stated above, cost models are common across many domains. Here, we focus on recent in situ visualization work. Larsen et al. employ a half analytical, half empirical approach to model performance of raycast volume rendering [1]. The authors leverage a priori knowledge of acceleration structures and fixed costs (e.g. shading) and use stratified sampling to adequately cover parameter space. That work also models performance across hardware configurations, mainly CPU vs. GPU performance.

There is also work in workstation-specific performance modeling leveraging both online learning and offline models [2], as well as a general study toward understanding all factors, including cost models, contributing to an overall simulation campaign [3]. Damaris, a middleware with in situ capabilities proposes methods to fit in situ algorithms within time [4] or performance [5] constraints. Finally, we believe the body of work dedicated to dynamic load balancing over clusters may help determine where to start both in terms of successes as well as current inadequacies for our problem. For instance, we may leverage the concepts of work stealing schemes that utilize simple cost models to determine the bases upon which transfer of work takes place. Conversely, many assumptions in such work are likely insufficient for our purposes such as work items being constant-time or elements being constant-order (transition from low to higher order elements may profoundly impact in situ algorithms).

Research Questions. The following general research questions are those we have identified as critical in progressing toward future incorporation of robust cost models for in situ data analysis and visualization.

1. What are goals for cost models? What are the rules of thumb for selection vs. granularity vs. resolution?
2. What is the required accuracy of cost models? Is there a precedent for requiring very accurate models and in particular how important are the upper and lower bounds when using cost models?
3. How should a cost model interact with an in situ taxonomy, specifically that put forth in the in situ terminology project? Should cost models be constrained to the taxonomy and/or should the taxonomy be simplified to help ensure this?
4. What is the full set of parameters across hardware, simulation, in situ routine, data movement, load imbalance, data dependencies, etc. that we should consider? What are ways in which we may reduce this set?
5. What are the useful abstractions for applications vs. analysis and should we consider multiple cost models in the event of differing sets of abstractions?
6. What role do "substitute" models play? Are reduced-order models appropriate or are there benefits of a multi-resolution approach?
7. Methodology: what is the best way to conduct this research program? Specifically, do the benefits of a ground-up effort outweigh associated development costs and how does this change given the capabilities of existing tools?

8. What is the role of "learning" (in a broad sense) of models? In the case of machine learning, should we target community sourcing of the generation and curation of training data?
9. What are the additional difficulties inherent to scalable algorithms on parallel architectures and are there non-empirical methods for collecting knowledge?
10. How do we validate cost models?

At a high level, in situ cost models contribute to a scientist's ability to plan full science campaigns. Predicting how much in situ analysis will impact the sim code, e.g. during on-node co-processing, allows for balancing requirements, prioritizing results, or dialing in appropriate timeliness or accuracy. At the operational level, prediction capabilities help ensure that an in situ method will run within a pre-determined resource envelope.

References

1 M. Larsen, C. Harrison, J. Kress, D. Pugmire, J. S. Meredith and H. Childs, "Performance modeling of in situ rendering", in SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Nov. 2016, pp. 276–287. 10.1109/SC.2016.23.

2 V. Bruder, S. Frey and T. Ertl, "Prediction-based load balancing and resolution tuning for interactive volume raycasting", Visual Informatics, 2017. 10.1016/j.visinf.2017.09.001.

3 J. Kress, D. Pugmire, S. Klasky and H. Childs, "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), Nov. 2016, pp. 45–50. 10.1109/ISAV.2016.014.

4 M. Dorier, R. Sisneros, T. Peterka, G. Antoniu and D. Semeraro, "Damaris/viz: A nonintrusive, adaptable and user-friendly in situ visualization framework", in 2013 IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV), Oct. 2013, pp. 67–75. 10.1109/LDAV.2013.6675160.

5 M. Dorier, R. Sisneros, L. B. Gomez, T. Peterka, L. Orf, L. Rahmani, G. Antoniu and L. Bougé, "Adaptive performance-constrained in situ visualization of atmospheric simulations", in 2016 IEEE International Conference on Cluster Computing (CLUSTER), Sep. 2016, pp. 269–278. 10.1109/CLUSTER.2016.25.

4.9 Convergence of HPC and Big Data

Bruno Raffin (INRIA – Grenoble, FR bruno.raffin@inria.fr)
Benson Muite (University of Tartu, EE, benson.muite@ut.ee)

License Creative Commons BY 3.0 Unported license
© Bruno Raffin and Benson Muite

Contributors: B. Raffin, W. Bethel, C. Hansen, I. Hotz, N. Gauger, H.-W. Shen, M. Srinivasan, B. Muite, K. Moreland, K. Ono, D. Pleiter, A. Ribes Cortes, R. Sisneros, G. Weber, R. Westerman, K. E. Isaacs, H. Yu

Background and Motivation. Due to the fact that this topic is based on two rapidly evolving fields, an appropriate definition and scoping are still in development [1]. Big Data as a domain has developed its own solutions for machine architectures, software frameworks, and also specific methodologies for data processing, e.g., the map/reduce model or stream processing, descriptive statistics and predictive statistics such as machine learning (ML) algorithms and in particular the subset of ML, deep learning (DL). We believe that it is too

narrow to only list popular Big Data frameworks such as Spark, Hadoop, and Flink, and to compare them to MPI or PGAS languages used in high performance computing. Instead, identifying a set of candidate use cases and applications to experiment with different aspects of a future convergence will enable better evaluation of the potential and limitations of each approach. A key problem is to understand how Big Data processing approaches can be used for in situ visualization and vice versa, i.e. explore whether in situ solutions can find relevant applications in a Big Data context. Big Data is associated with machine learning, in particular neural networks. ML techniques have been successful for addressing a number of classification/optimization problems. How these techniques could be used efficiently in an in situ context still needs to be explored, with an important question being the identification of the data that needs to be extracted to enable learning. Many machine learning approaches can take a long time to be trained, usually with offline data. Thus, other optimization techniques should also be considered to enable real time in situ processing by coupling fast algorithms with fast implementations.

Another area of interest is co-design for in situ big data analysis. In cases of key interest (such as in transit analysis of telecommunication network data [2]), dedicated hardware is often used, and experiences from the in situ case can be used to aid this co design. Many high performance computing centers, under user requests, now need to provision frameworks such as Spark, Hadoop, and Flink on high performance computing hardware. In Situ processing can leverage this trend to experiment and integrate Big Data solutions in workflows. At present, in situ visualization has primarily focused on visualizing results of continuum mechanics simulations. Links to other communities with different types of streaming data that cannot be stored to disk may help in the development of new techniques and workflows [1, 2].

The diversity, size, and complexity of data in HPC is growing. In situ processing needs to consider not only data produced by large scale applications, but also data provided online by other scientific instruments or to combine observation data stored on disk (data assimilation) [1]. This type of workflow may require the combination of different tools from HPC and Big Data, making for a very heterogeneous software stack. Modular supercomputers may be a good place for initial experimentation with workflows that have in transit processing since the deployed software stacks on each modular component are under the control of one center. Big Data frameworks are not designed to run efficiently on current large high performance computing architectures, limiting their usability at scale for in situ analytics. Ease of use has typically been prioritized over efficiency. They do however inspire adaptation of HPC frameworks to allow for both ease of use and efficiency [3]. ML and more particularly DL are black box solutions. Given enough high quality data for learning, they can automatically produce good solutions without requiring the hand development of a complex model. However, the computed surrogate model is often difficult to understand and the guarantees on its qualities are limited. In many cases this may not be acceptable. ML may be used in an in situ workflow, for instance for feature tracking, but also for designing and optimizing in situ workflows. Such a task is often difficult, with very limited tools to assist the developer, often left to rely only on his experience and expertise.

Many big data workflows have been ported to HPC hardware and are available for users, in particular on medium and small scale high performance computing platforms. In the context of scientific computing today, they are primarily used for post-hoc analysis. Efforts are made to improve Big Data framework performance on supercomputers [3]. High performance computers are more reliable than traditional Big Data platforms, thus softening the need for strong fault tolerance protocols that often limit Big Data frameworks performance and

scalability. Data parallel and functional programming techniques underlying the map/reduce model are also used in some modern visualization frameworks like VTM-m. The map/reduce model has also been experimented with in the in situ context [4]. Success has been reported in using ML for optimizing HPC platforms, for instance in job scheduling [5].

Research Questions. Against this backdrop, we see the following immediate questions.

1. How can one use ML for in situ data reduction?
2. How can ML techniques help users to formulate and/or optimize in situ workflows?
3. How can in situ visualization be used to monitor and analyze deep learning algorithms [6]?
4. What guarantees can be provided on ML techniques when used in safety critical situations?
5. How can Big Data frameworks and their relatively easy to master programming environnements, in particular for stream processing, support or complement in situ processing frameworks?
6. More generally how can the knowledge that can be extracted through machine learning / big data techniques help improve in situ analytics, visualization, and – eventually – HPC at large?

Leveraging the additional tools and methodologies of Big Data for in situ processing may eventually enable researchers to do better science faster. The push to use these tools may very likely come from application scientists. But how they will be integrated and used in situ is still unclear.

References

1. M. Asch, T. Moore, R. Badia, M. Beck, P. Beckman, T. Bidot, F. Bodin, F. Cappello, A. Choudhary, B. de Supinski, E. Deelman, J. Dongarra, A. Dubey, G. Fox, H. Fu, S. Girona, W. Gropp, M. Heroux, Y. Ishikawa, K. Keahey, D. Keyes, W. Kramer, J.-F. Lavignon, Y. Lu, S. Matsuoka, B. Mohr, D. Reed, S. Requena, J. Saltz, T. Schulthess, R. Stevens, M. Swany, A. Szalay, W. Tang, G. Varoquaux, J.-P. Vilotte, R. Wisniewski, Z. Xu and I. Zacharov, "Big data and extreme-scale computing: Pathways to convergence-toward a shaping strategy for a future software and data ecosystem for scientific inquiry", The International Journal of High Performance Computing Applications, vol. 32, no. 4, pp. 435–479, 2018. 10.1177/1094342018778123.
2. R. Grosman, Y. Chen and P. Chun, "Opencl meets open source streaming analytics", in Proceedings of the 4th International Workshop on OpenCL, ser. IWOCL '16, Vienna, Austria: ACM, 2016, 18:1–18:3. 10.1145/2909437.2909457.
3. S. J. Plimpton and K. D. Devine, "Mapreduce in mpi for large-scale graph algorithms", Parallel Comput., vol. 37, no. 9, pp. 610–632, Sep. 2011. 10.1016/j.parco.2011.02.004.
4. Y. Wang, G. Agrawal, T. Bicer and W. Jiang, "Smart: A mapreduce-like framework for in-situ scientific analytics", in SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Nov. 2015, pp. 1–12. 10.1145/2807591.2807650.
5. D. Carastan-Santos and R. Y. de Camargo, "Obtaining dynamic scheduling policies with simulation and machine learning", in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ser. SC '17, Denver, Colorado: ACM, 2017, 32:1–32:13. 10.1145/3126908.3126955.
6. H. Strobelt, S. Gehrmann, H. Pfister and A. M. Rush, Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks, 2016.

4.10 Software Complexity, Heterogeneity, and User-facing Issues

John Patchett (Los Alamos National Laboratory, US, patchett@lanl.gov)
E. Wes Bethel (Lawrence Berkeley National Laboratory, US, ewbethel@lbl.gov)

License Creative Commons BY 3.0 Unported license
© John Patchett and E. Wes Bethel

Contributors: E. W. Bethel, J. Patchett, P. Messmer, M. Parashar, A. Bauer, M. Srinivasan, K. Ono, N. Röber, N. Gauger, J. Bennett, F. Sadlo, M. Dorier, M. Larsen, A. Ribes Cortes

Background and Motivation. The group identified three core problems crosscutting software complexity, heterogeneity and user-facing issues: in situ software is complex and hard to use; users are reluctant to adopt new technologies and in particular in situ; and increasing heterogeneity in computational resources, software tools, and use scenarios exacerbates the problem.

Challenges. In situ software is complex and hard to use. Coupling in situ software entails coupling two or more codes together, both of which are likely complex software applications in their own rights. This coupling is more complex than simply making a library call, because of the consumer-side factors: the in situ method must be correctly configured to produce the desired results and it may itself be a complex, parallel application. In all but the simplest of cases, the data models of the simulation code and in situ method will not be identical. This necessitates a potentially complex data model transformation.

Additional challenges arise from the complexity of the underlying computational platforms. These challenges are present for all computational endeavors, not just in situ methods. There is an increasing complexity and depth to memory hierarchies, and increasing number of cores per processor. With this added complexity arise challenges in determining the optimal configuration and placement of processing components in an in situ pipeline: some methods and configurations are best run truly in situ, where data does not move, while in other configurations, moving data to a different node may produce a lower time to solution. This set of challenges is compounded when considering portability: a given configuration that may perform well on one HPC platform is unlikely to be the best for a different platform.

Another challenge area is the fact that many users may be unwilling, or reluctant, to adopt and use in situ methods/infrastructure. They are concerned about uncertainties around software lifespan and support, the software complexity, coupled with uncertainty about whether the software will solve their problem, whether it computes correct results, and its reliability and potential adverse impact on their high-concurrency, long-running simulation. Users are unwilling to invest time and energy into adopting a new technology unless there are clear benefits to them, and they have a well-founded belief that the new technology they adopt will be around in the future for them.

Successes. A Eurographics 2016 State-of-the-Art report STAR report [1] provides a comprehensive discourse of in situ R&D that goes back over 20 years. Notably, there are a number of established APIs that target the coupling of simulation and in situ visualization codes [2, 3, 4, 5, 6, 7]. One of the long-standing weaknesses of in situ approaches is the idea they produce results, i.e., images, that reflect a given combination of visualization and rendering parameters, and the collection of output is then not amenable to exploratory analysis. CINEMA[8] is an approach where in situ visualization tools create a "CINEMA image database", such that the images may span some range of visualization and/or rendering parameters, and then a post-hoc viewer permits a user to interact with and browse the

collection of pre-computed images. Other work in this space includes generating topological summaries for higher-level, feature-based analysis and exploration [9].

Reasons why users might be more motivated to engage with the in situ community is if such engagement provides more value to their project. One example is that it may save them user time in examining results and performing planning for future runs [10]. Another may be that it enables them to see and study new science hidden in their data, features that had previously gone undiscovered due to limitations of the I/O bottleneck [11], or to perform complex simulation/analysis couplings where the analysis methods consist of combinations of lower-resolution surrogate model codes and advanced geometric/topological analysis methods [12]. Helping users to become better aware of the technologies and how they might be used is an important part of the landscape, as well.

Research Questions. From a resource standpoint, an open and ongoing challenge is coping with increasing heterogeneity of both software infrastructure and the underlying computational platforms. On the one hand, users can benefit from a growing collection of tools that provide new, advanced analytics methods. On the other hand, finding ways to effectively leverage them within familiar toolchains (integration) and on evolving computational platforms is a challenge. Users are also concerned with the reliability of these methods, in terms of correctness of results, along with the robustness of their execution.

A longer-term research question concerns how to minimize the intrusion into the simulation code of the interface code necessary to couple the data producer and consumer. While there are efforts that aim at providing portability across numerous in situ backends, the fact that even minimal instrumentation code is required in the simulation impedes adoption. It has been suggested that finding ways to eliminate the need for any explicit instrumentation code would represent a significant step forward in this regard. Yet, whether this is feasible in production codes remains an open question, chiefly because both data producers (simulations) and consumers (in situ analysis/visualization) are highly complex, massively parallel applications.

A difficult problem is determining the optimal ratio of simulation to analysis resources, since they are performing different types of computations, along with taking into account the cost of data movement between ranks. Helping users to determine optimal configurations of concurrency and placement of these computational components is an open research question, and entails interactions with researchers in other, related fields, such as operating systems/runtime.

Scientists are faced with an increasingly complex problem: computational platforms grow more powerful, but the I/O fabric grows in capacity much more slowly, which results in a debilitating imbalance between their ability to compute data and their ability to store it for analysis. One result of this imbalance is the loss of science, where important features in data are not discovered. The primary benefit of in situ methods in computational science is the ability to perform better science, and to do so more productively. Better science results from having access to full spatio-temporal resolution data for the purpose of knowledge discovery. Productivity increases when there is faster turnaround between hypothesis formation, experiment, and analysis of results. Reducing complexity, including lowering barriers to use, will result in increased user demand and adoption.

References

1 A. C. Bauer, H. Abbasi, J. Ahrens, H. Childs, B. Geveci, S. Klasky, K. Moreland, P. O'Leary, V. Vishwanath, B. Whitlock and E. W. Bethel, "In situ methods, infrastructures, and applications on high performance computing platforms", Computer Graphics Forum, vol. 35, no. 3, pp. 577–597, 2016. 10.1111/cgf.12930.

2 N. Fabian, K. Moreland, D. Thompson, A. C. Bauer, P. Marion, B. Gevecik, M. Rasquin and K. E. Jansen, "The paraview coprocessing library: A scalable, general purpose in situ visualization library", in 2011 IEEE Symposium on Large Data Analysis and Visualization, Oct. 2011, pp. 89–96. 10.1109/LDAV.2011.6092322.

3 B. Whitlock, J. M. Favre and J. S. Meredith, "Parallel In Situ Coupling of Simulation with a Fully Featured Visualization System", in Eurographics Symposium on Parallel Graphics and Visualization, T. Kuhlen, R. Pajarola and K. Zhou, Eds., The Eurographics Association, 2011. 10.2312/EGPGV/EGPGV11/101-109.

4 U. Ayachit, B. Whitlock, M. Wolf, B. Loring, B. Geveci, D. Lonie and E. W. Bethel, "The sensei generic in situ interface", in 2016 Second Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV), Nov. 2016, pp. 40–44. 10.1109/ISAV.2016.013.

5 U. Ayachit, A. Bauer, E. P. N. Duque, G. Eisenhauer, N. Ferrier, J. Gu, K. E. Jansen, B. Loring, Z. Lukić, S. Menon, D. Morozov, P. O'Leary, R. Ranjan, M. Rasquin, C. P. Stone, V. Vishwanath, G. H. Weber, B. Whitlock, M. Wolf, K. J. Wu and E. W. Bethel, "Performance analysis, design considerations, and applications of extreme-scale in situ infrastructures", in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ser. SC '16, Salt Lake City, Utah: IEEE Press, 2016, 79:1–79:12. 10.1109/SC.2016.78.

6 C. Docan, M. Parashar and S. Klasky, "Dataspaces: An interaction and coordination framework for coupled simulation workflows", Cluster Computing, vol. 15, no. 2, pp. 163–181, Jun. 2012. 10.1007/s10586-011-0162-y.

7 M. Larsen, J. Ahrens, U. Ayachit, E. Brugger, H. Childs, B. Geveci and C. Harrison, "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, ser. ISAV'17, Denver, CO, USA: ACM, 2017, pp. 42–46. 10.1145/3144769.3144778.

8 J. Ahrens, S. Jourdain, P. OLeary, J. Patchett, D. H. Rogers and M. Petersen, "An image-based approach to extreme scale in situ visualization and analysis", in SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Nov. 2014, pp. 424–434. 10.1109/SC.2014.40.

9 T. Biedert and C. Garth, "Contour tree depth images for large data visualization", in Proceedings of the 15th Eurographics Symposium on Parallel Graphics and Visualization, ser. PGV '15, Cagliari, Sardinia, Italy: Eurographics Association, 2015, pp. 77–86. 10.2312/pgv.20151158.

10 J. Patchett and J. Ahrens, "Optimizing scientist time through in situ visualization and analysis", IEEE Computer Graphics and Applications, vol. 38, no. 1, pp. 119–127, Jan. 2018. 10.1109/ MCG.2018.011461533.

11 O. R"übel, B. Loring, J. Vay, D. P. Grote, R. Lehe, S. Bulanov, H. Vincenti and E. W. Bethel, "Warpiv: In situ visualization and analysis of ion accelerator simulations", IEEE Computer Graphics and Applications, vol. 36, no. 3, pp. 22–35, May 2016. 10.1109/MCG.2016.62.

12 D. Morozov and Z. Lukic, "Master of puppets: Cooperative multitasking for in situ processing", in Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing, ser. HPDC '16, Kyoto, Japan: ACM, 2016, pp. 285–288. 10.1145/2907294.2907301.

5 Panel Discussions

In addition to identifying ten research topics, workshop attendees agreed upon the importance of establishing a pipeline to bring fundamental, theoretical in situ research into a production environment (i.e., sustainable, user-friendly, robust, high-quality software tools that are actively used by science and engineering application teams). Attendees also noted, however, that many pervasive, cross-cutting issues impede community progress in establishing an in situ research-to-production pipeline. These issues are due, in large-part, to the interdisciplinary nature of the pipeline. Specifically, each individual researcher within an interdisciplinary team has a different set of priorities or "definitions of success" that shapes the focus of their work. Individual priorities are defined by many factors, including research community, personal interests, funding sources, and/or performance measures at their institution (academia, industry, research laboratory). In order to build a successful in situ research-to-production pipeline, interdisciplinary team members need to resolve or "bridge" these very different priorities, some of which are outside of the team's direct control. Workshop attendees identified a number of bridging issues and classified them into three categories: 1) Software Engineering and Deployment, 2) Funding, and 3) Teaming and Pipeline. The first issue was discussed in the first panel, and the latter two were discussed in the second panel.

5.1 Panel Discussion on "Software Engineering & Deployment"

Panelists

- Andrew Bauer (Kitware – Clifton Park, US, andy.bauer@kitware.com)
- Jens Krüger (Universität Duisburg-Essen, jens.krueger@uni-due.de)
- Matthew Larsen (Lawrence Livermore National Laboratory, larsen30@llnl.gov)
- Kenneth Moreland (Sandia National Laboratories – Albuquerque, kmorel@sandia.gov)

Summary of Discussion

Background & Motivation. Software is the means by which theoretical and methodological techniques are made practical and useful to a user community. Depending on a researcher's priorities, their target user-community can vary dramatically. For example, an academic researcher can have a target user-community of a) application scientists/engineers or b) their students. Each of these user communities places different requirements on the quality, maturity, and support model of the resulting software product. In the first scenario, robust, user-friendly, production-quality software is often required for adoption. In the second scenario, the maturity required of the software product is typically significantly lower (e.g., the software is a learning tool, that the students are meant to build off of in their studies).

Software can be made accessible to users via a number of mechanisms: commercial, open-source, and via direct hand-off (but with no formal licensing/use agreements). Once software is deployed, user-support model requirements will vary by user-type.

Software also serves as a means for a researcher to establish an identity (analogous to publications).

Challenges. Many of the software challenges highlighted by the workshop attendees are not necessarily unique to the in situ research-to-development pipeline, but are exacerbated by the interdisciplinary nature of the work. Challenges are often due to discrepancies in priorities (and resulting expectations) between an in situ tools researcher and their target

user-community. These discrepancies can be minor (e.g., missing a feature that might be easily be added), to significant (researcher has a prototype implementation with a minimal user-support model, whereas the user-community may want a production-ready tool). However, challenges can also be due to language barriers between tool and user communities (e.g., unclarified assumptions, use of terms that mean different things to the two communities, jargon).

The following examples illustrate common discrepancies in priorities between in situ tools developers and their target user communities:

- **Requirements.** Scientists and engineers often prefer specialized tools, tailored to their application, over general tools. This is due, in large part, to their ease of use. In an in situ workflow, these preferences become strict requirements due to the increase in overall code-complexity and the implications of a tool's downstream dependencies. Technical user community concerns include stability of application programming interfaces (API), code and/or compile-time bloat, and code reliability. In situ tool developers, on the other hand, prefer to create general purpose tools, which promote sustainable code management and simplify (from the point of view of the tool developer) the process of supporting multiple target user-communities.
- **Advertising and Adoption.** Communication barriers can arise between user and tool communities in several ways that inhibit advertising. For example, different communities use of similar terms to mean different things. Jargon and acronyms can also serve as a barrier to adoption. Communication issues can be further exacerbated when competing tools exist that provide overlapping functionality. Not only can this cause friction and competition within a particular tool community, it can make it difficult for user communities to discern which of the competing tools to use. Aspects that may further inhibit adoption include, e.g., concerns over which tool will have greater longevity, lack of clarity regarding tradeoffs between the tools, and uncertainty about downstream dependencies.
- **Accessibility.** Researchers have different mechanisms to make their tools available to their target user-community. Commercial tools come at a financial cost to users, which provides a higher barrier-to-entry. For tool developers a primary benefit of commercialization is the financial means to fund code development and user-support. In contrast, releasing an open source tool imposes a lower barrier-to-entry for users. However, tool developers relinquish some control of ownership/identity. Furthermore, open source release of a code does not provide a direct funding mechanism for further development and user-support. We note that some scientists and engineering user communities have accessibility requirements on tools used within their workflows (e.g., must be open source).

Potential solutions. In spite of the numerous challenges, workshop attendees identified several suggestions for the path forward. Several of these focus on addressing concerns raised by the user community regarding in situ tool APIs, their stability, and their intrusion into user application code. The first suggestion is longer-term, nebulous, and strategic: think radically "outside the box" to facilitate "application unawareness." An analogy was made to cell phone chargers. Every time a cell phone company changes the form factor of its charging plug, it is disruptive to their user community. Recently, innovative wireless phone charging equipment has provided a revolutionary solution to this problem. Is there a similar revolutionary technology that can address the "tower of Babel" API complexity issue?

Other suggestions to address API concerns are nearer-term, specific, and more tactical in nature:

- Identify the lowest common denominator that is general, but still allows for specialization for specific applications (e.g., something analogous to the POSIX file system for post-hoc analysis use case).
- Engage external tools communities, for example, those developing containers (e.g., Docker, Kubernetes) and package managers (e.g., SPACK, LMOD).

Many of the advertising and adoption concerns raised were due to communication concerns. Potential solutions for these issues were raised but are discussed in greater detail in the Teaming and Pipeline subsection.

5.2 Panel Discussion on "Programmatic & Funding Issues / Interdisciplinary / Pipeline"

Panelists

- Peer-Timo Bremer (Lawrence Livermore NationalLaboratory, bremer5@llnl.gov)
- Charles D. Hansen (University of Utah, hansen@cs.utah.edu)
- Ingrid Hotz (Linköping University, ingrid.hotz@liu.se)
- Bruno Raffin (INRIA – Grenoble, bruno.raffin@inria.fr)
- Alejandro Ribes Cortes (EDF – Paris, alejandro.ribes@edf.fr)
- Han-Wei Shen (Ohio State University – Columbus, shen.94@osu.edu)

Summary Discussion on Programmatic & Funding Issues

Background and Motivation. In a research-to-development pipeline, funding for the various stages of software development is often provided by multiple different funding agencies, each with very different "success criteria." For example, many of the funding sources that promote in situ tools research are focused on theory and may relegate software development activities to prototypes only. However, user communities often require mature and reliable software, and their associated funding sources typically do not support software development and user support. Commercialization and collaborations with large corporations are other means of funding an in situ research-to-production pipeline.

Challenges. One of the predominant challenges identified by attendees is the dearth of sustained funding sources for software development, testing, and deployment activities aimed at bringing fundamental research tools into a production-quality state. As noted above, there are distinct funding agencies that support both the in situ tools and user communities. However, each of these funding agencies' success criteria are narrowly focused within their respective communities. Those funding sources aimed at bridging communities are few, and often transient in nature (limited duration). Those that provide more sustained funding are focused on very narrow user communities (e.g., DOE/NNSA/ASC funding).

Workshop attendees also noted some disconnects between funding program manager priorities and the priorities of in situ tools researchers. Some funding agencies have moved away from a focused emphasis on in situ processing, and are now looking to invest in newer research trends, such as machine learning and big data. This is in large part due to the hype and public appeal surrounding these trends in industry. However, this also highlights a potential communication disconnect between the in situ tools community and

funding agency program managers on the importance of continued investment in an in situ research-to-production pipeline.

While commercialization and collaboration with industry are another source of funding, many workshop participants noted that non-disclosure agreement (NDA) requirements are often in direct conflict with fundamental tools researcher goals of publishing their research.

Lastly, some in situ tools researchers noted difficulty in gaining access to high performance computing resources on HPC systems for user support build and test development operations efforts.

Potential Solutions. We begin this section by highlighting some early success stories in addressing funding issues:

- **HPC resource allocation.** In the United States, HPC resource allocation may be easier than it appears at some HPC facilities. Many in situ tools researchers do not pursue resource allocation because acquisition processes appear to be targeted solely at application developers. Although not well-advertised, there are compute cycles available to the tools communities and tools researchers have been successful in acquiring compute cycles by directly contacting facilities support staff to better understand request protocol options.
- **Institution-specific success stories.**
  - KAUST University has a support model for software development and deployment. Research faculty can work with software developers who are paid by the University to mature research tools into production capabilities.
  - EDF was highlighted as a company who funds work that spans the in situ research-to-production pipeline.
- **Recent changes to funding agency reporting criteria.** Some academic and fundamental research funding agencies have made changes to their reporting/success criteria. Specifically, the agencies are beginning to ask about data artifacts and software sustainability plans. While additional funding is not being provided to support these activities, this is an important first step in that direction.

In addition to early successes, suggestions for path forward highlight the need for increased alignment between in situ pipeline researchers and potential funding sources:

- **Academic and/or government funding agencies.** Academic researchers do not always have a clear understanding of a funding agency's program manager roles and responsibilities (which may differ significantly across agencies). Researchers could benefit from an understanding of funding agency terminology (e.g., technology readiness level (TRL) [1] to describe software maturity in a standardized way). A deeper, understanding of a funding agency's own success criteria and requirements would enable researchers to help shape a program manager's research focus, as well as to help program managers advocate for additional funding for their programs.
- **Industry collaborations.** As we establish a research-to-production pipeline, it is clear that the community has not yet identified which universal, baseline services industry partners should develop and maintain. For example, BLAS was highlighted as a former fundamental research activity that HPC industry providers now maintain across HPC system procurements. What is the analogous BLAS for the in situ tools community? How should the in situ tools community identify this division of labor with industry partners?

References

1 Technology Readiness Level, https://en.wikipedia.org/wiki/Technology_readiness_level (Accessed Sept. 16, 2018).

Summary Discussion on Teaming and Staffing Pipeline

Background and Motivation. The establishment of an in situ research-to-production pipeline is interdisciplinary, involving three different research communities (application scientists and engineers, high-performance computing experts, and visualization scientists) across academia, industry, and research laboratories. Universities are the staffing pipeline source, providing the next generation of researchers who will be joining these three communities.

Challenges. This subsection summarizes attraction, retention, and teaming challenges that negatively impact the establishment of an in situ research-to-production pipeline.

- **Staffing Pipeline.** While science-based research activities draw some students, the competition from industry is strong. This is due to salary potential in industry, and the hype of data science and machine learning, which compete directly with the in situ tools community. Consequently, students at universities are showing more interest in curriculum that prepares them for industry careers in data science, diminishing the in situ staffing pipeline. These challenges are exacerbated by losses in funding for universities, making it difficult, if not impossible, for professors and their students to collaborate on interdisciplinary research projects with laboratories and industry partners. The influx of data science and machine learning career options (and their associated salaries), pose a retention challenge as well, which particularly impacts research laboratory staffing.
- **Teaming.** Many teaming challenges have been alluded to already throughout this report because communication and trust are central to effective interdisciplinary work. Specifically, when a team does not effectively communicate, this exacerbates any technical and logistical issues. From a communication perspective, a number of issues impede effective collaboration. For example, different research communities may use similar terminology, but mean different things. Furthermore, the use of technical jargon and/or acronyms can cause additional confusion. Different communication styles and different priorities, can lead to misunderstandings that undermine trust, further hampering collaboration.

Potential Solutions. Workshop attendees identified suggestions, which mostly comprise teams learning and employing best practices in communication and psychology.

- **Staffing Pipeline.** suggestions on staffing pipeline center around understanding what motivates people to draw and maintain a healthy pool of interdisciplinary team members. These include:
  - Ensure internship mentors are educated in best practices
    - Understand that the experience of the internship is equally as important as the output from an attraction/retention perspective.
    - Meet regularly with both mentees and their professors.
    - Provide students with leadership opportunities.
    - Labs, industry, and funding agencies need to financially support recurring internships to establish long-standing relationships with potential employees.
  - Communicate with funding agencies the importance of sustained funding for in situ for University partners to maintain a healthy staffing pipeline (see 5.2).
  - Work with human resource staffing partners at Universities and Laboratories to make salaries as competitive as possible with industry.
  - Formalize recruiting "sales pitch": why should students want to pursue a career pursuing in situ pipeline work? Some of this may be technical, but other recruiting factors could be a draw: e.g., impact, team, work-life balance, etc.

  - Figure out how to capitalize on data science and machine learning to draw students towards science and/or mission-impact application domains, which overlaps with thrust from Section 4.9.
- **Teaming.** Interdisciplinary are likely to be diverse along several axes, including technical field, institutional, and social/communication style. Suggestions for teaming centered around embracing diversity and inclusion best practices to build and sustain effective teams:
  - Communicate clearly and without jargon.
  - Avoid assumptions: understand your own as well as your team members'.
  - Understand your team member's priorities and success criteria.
  - Find win-win situations: identify and agree upon mutually agreeable successful outcomes. Do this early in the teaming process and revisit on regular intervals.
  - Education and tools:
    * Provide team leads education on coaching.
    * Educate all team members on social styles (e.g., DISCS, Myers-Briggs) and communication (e.g., convergent vs divergent thinking).
    * Embrace associated tools chains.
  - Choose teams wisely: avoid working with people who do not abide by communication and teaming best practices.
  - Communicate about norms. This includes what data is private and what is public. Mailing lists, social media and some cloud services may be unwelcoming environments for some potential team members.
  - Understand and manage the costs (time, financial investment, education) of collaboration.

Participants

- Andrew Bauer
Kitware – Clifton Park, US
- Janine C. Bennett
Sandia National Labs – Albuquerque, US
- E. Wes Bethel
Lawrence Berkeley National Laboratory, US
- Peer-Timo Bremer
Lawrence Livermore National Laboratory, US
- Thierry Carrard
CEA – Bruyères-le-Châtel, FR
- Hank Childs
University of Oregon – Eugene, US
- Matthieu Dorier
Argonne National Laboratory – Lemont, US
- Steffen Frey
Universität Stuttgart, DE
- Christoph Garth
TU Kaiserslautern, DE
- Nicolas R. Gauger
TU Kaiserslautern, DE
- Markus Hadwiger
King Abdullah University of Science and Technology – Thuwal, SA
- Charles D. Hansen
University of Utah – Salt Lake City, US
- Katrin Heitmann
Argonne National Laboratory, US
- Bernd Hentschel
RWTH Aachen, DE
- Ingrid Hotz
Linköping University, SE
- Katherine E. Isaacs
University of Arizona – Tucson, US
- Jens Krüger
Universität Duisburg-Essen, DE
- Matthew Larsen
Lawrence Livermore National Laboratory, US
- Peter Messmer
NVIDIA – Zürich, CH
- Kenneth Moreland
Sandia National Labs – Albuquerque, US
- Benson Muite
University of Tartu, EE
- Kenji Ono
Kyushu University, JP
- Manish Parashar
Rutgers University – Piscataway, US
- Valerio Pascucci
University of Utah – Salt Lake City, US
- John Patchett
Los Alamos National Laboratory, US
- Tom Peterka
Argonne National Laboratory, US
- Dirk Pleiter
Jülich Supercomputing Centre, DE
- David Pugmire
Oak Ridge National Laboratory, US
- Bruno Raffin
INRIA – Grenoble, FR
- Alejandro Ribes Cortes
EDF – Paris, FR
- Niklas Röber
DKRZ Hamburg, DE
- Ulrich Rüde
Universität Erlangen-Nürnberg, DE
- Filip Sadlo
Universität Heidelberg, DE
- Han-Wei Shen
Ohio State University – Columbus, US
- Robert Sisneros
University of Illinois at Urbana Champaign, US
- Madhusudhanan Srinivasan
King Abdullah University of Science and Technology – Thuwal, SA
- Gunther H. Weber
Lawrence Berkeley National Laboratory, US
- Rüdiger Westermann
TU München, DE
- Hongfeng Yu
University of Nebraska – Lincoln, US