The Challenges of In Situ Analysis for Multiple Simulations

Alejandro Ribés, Bruno Raffin

The Challenges of In Situ Analysis for Multiple Simulations

Alejandro Ribés
EDF Lab Paris-Saclay
Palaiseau, France
alejandro.ribes@edf.fr

Bruno Raffin
Univ. Grenoble Alpes, Inria, CNRS, LIG
Grenoble, France
bruno.raffin@inria.fr

ABSTRACT

In situ analysis and visualization have mainly been applied to the output of a single large-scale simulation. However, topics involving the execution of multiple simulations in supercomputers have only received minimal attention so far. Some important examples are uncertainty quantification, data assimilation, and complex optimization. In this position article, beyond highlighting the strengths and limitations of the tools that we have developed over the past few years, we share lessons learned from using them on large-scale platforms and from interacting with end users. We then discuss the forthcoming challenges, which future in situ analysis and visualization frameworks will face when dealing with the exascale execution of multiple simulations.

KEYWORDS

In Situ Analysis, Uncertainty Quantification, Data Assimilation, Exascale Computing

1 INTRODUCTION

The focus of in situ analysis has largely been exploratory analysis of the output of a single large-scale simulation [9]. This is clearly observed when looking at the applications of the most popular in situ visualization libraries, Catalyst [4] and Libsim [20]. The technical literature typically presents the case where a single large simulation struggles with the I/O bottleneck. Leveraging in situ approaches, like Catalyst or Libsim, enables to alleviate this I/O problem by processing and reducing data close to their production source, before being written to disk, so as to reduce the need for storage.

In section 2 we summarize important developments that have been performed by the in situ community. However, it is clear that, in all the years of evolution of in situ technologies, an underlying assumption exists: a single large-scale simulation is run, this generates several problems, either technical or methodological, and the community tackles these problems. This position paper analyzes the challenges and opportunities for extending in situ processing beyond a single large-scale simulation, considering use cases where the analysis needs to combine data from multiple simulation runs (also commonly called ensemble run). Such use-cases are becoming more common with the need to sample the simulation behavior within some parameter ranges for extracting knowledge using statistical or machine learning based methods, combined with the availability of large supercomputers capable today of running thousands of large simulation instances. Each simulation being potentially large, the amount of data generated by multiple runs is huge, leading to a pressing need for frugal I/O solutions like in situ processing. In this paper, we analyze the challenges and opportunities that multi-simulation analytics represents for the in situ community. This category of applications calls for the development of novel in situ solutions.

In the following: Section 2 presents related work; Section 3 introduces several important examples where multi-run simulations are of fundamental importance; Section 4 describes lessons learned from the development of the open-source software Melissa (Modular External Library for In Situ Statistical Analysis)[42], which already deals with multi-run simulations in the context of large scale uncertainty quantification; Section 5 discusses the challenges associated to the in-situ treatment of multi-run simulations; A short conclusion and a bibliography section end up the chapter.

2 RELATED WORK

Before dedicated visualization libraries existed, in situ visualization was possible but very cumbersome, examples are the visualization of large-scale combustion simulations in 2010 [43] or the visualization of a trillion particle simulation in 2012 [10]. This is the reason why finding standardized solutions for the struggling case of large simulations have historically been driving the in situ community.

Catalyst [4] and Libsim [20] are examples of these standardized solutions. These libraries are compiled and linked to the solver and thus access the same computer resources and memory addresses than the simulations (see for instance [36] or [11] for examples concerning computational fluid dynamics). They indeed alleviate the I/O bottleneck but this tightly-coupled paradigm also presents some disadvantages, for instance an error in the visualisation library could propagate and block the solver. Thus the in situ community developed the so-called loosely-coupled paradigm, also commonly called in transit processing, which can be defined (from [19]) as when the simulation transfers data over the network to a separate set of processing or visualization nodes. Reference [29] gives some examples of in transit visualizations. Again, in transit techniques were studied for the single large simulation case.

Another aspect of in situ visualization that has received attention is the visualization pipeline, usually defined in a python script, that needs to be defined prior to the simulation run. This introduces

rigidity in the methodology, in the sense that the visualization operations must be a priori defined, preventing the post-hoc exploratory analysis of the simulations results. Several strategies have been proposed to alleviate this problem. One possible solution is the use of steering for interacting with scientific simulations while they are executing [18]. This solution introduces flexibility because the visualization operations can be changed while the simulation is being performed. This approach is implemented in Catalyst and Libsim. Another popular solution is Cinema [2], where in situ visualization tools create an "image database" as a way to save a highly compressed sample of the simulation results. Then, a post-hoc viewer allows the user to browse and interact with the collection of precomputed images, possibly changing some rendering parameters.

In the recent past years, in situ visualization became more and more mature. Then authors started to be interested in how to use the in situ paradigm to perform not only visualization but other kind of analysis on the simulation outputs. This requires the development of shared data models that can be used for both simulation and visualization/analysis alike [9]. Efforts to standardize the data models have been performed, important examples are Alpine [22], ADIOS [25], and SENSEI [5]. This naturally leads to the interaction of in situ techniques with computation workflows. Decaf [15], Damaris [13], FlowVR [16] or DataSpace [12] are examples of frameworks designed to support flexible workflows that can combine data processing and visualization, being in situ, in transit or a mix of both.

But so far the in situ community has clearly focused on processing on-line the data produced by a single simulation run. A few papers have stressed uses-cases requiring efficient, I/O savy, multi simulations data analysis that we discuss with a general perspective in the following section.

3 MULTI SIMULATION DATA PROCESSING USE CASES

We identify three main categories of applications relying on multiple simulation runs, namely uncertainty quantification, data assimilation and complex optimization. The two first ones are identified as important techniques by Jim Gray in The Fourth Paradigm of Scientific Discovery [17]. Complex optimization can take different forms that we also discuss.

3.1 Uncertainty Quantification

Uncertainty quantification (UQ) is the science of quantitative characterization and reduction of uncertainties. In the context of in situ techniques, we refer to computational uncertainties of numerical simulations. Several ways of defining and classifying uncertainties exist, a quite general one being the difference between epistemic and aleatoric uncertainty. A more practical classification refers to the element where the uncertainty is considered. For instance a mathematical model can be considered uncertain, or the input parameters of a solver of differential equations can be considered uncertain. This uncertainty is what is commonly found when dealing with numerical simulations.

The ability to fully quantify uncertainty in high performance computational simulations provides new capabilities for verification and validation of simulation codes [17]. Moreover, reducing uncertainty is needed for regulatory processes or for allowing simulation based decision making.

Why UQ needs the execution of multiple simulations? The UQ methodology [41] involves preparing a design of experiments, which explains how to sample the variables of interest of the simulation to extract the most information with the lowest computational costs. The variables of interest are usually values of initial or boundary conditions, or parameters of the solver. Next, multiple simulations are run, each instance with a different variable set drawn as defined by the design of experiments. Simulation outputs are combined through statistical estimators to support the understanding of how the outputs are influenced by the variability of sampled variables. Let consider the use-case presented in [21] to understand the challenge it represents from an I/O perspective. For a study with 9 varying paramaters they ran 60 000 simulations on the Trinity supercomputer, accounting for 34 million core hours and producing 5 Petabytes of intermediate files.

3.2 Data Assimilation

Data Assimilation (DA) can be viewed as a method for combining observations with a model state with the objective of improving the latter [3]. DA is about integrating observations, typically acquired by on-the-field sensors, with the simulation codes. The general approach consists in periodically correcting the progress of the simulation by minimizing the global error obtained from the combination of the observation and simulation errors. Data assimilation is particularly used in domains like weather forecast and climate simulation where numerical models are highly sensible to parameter values.

Why DA needs the execution of multiple simulations? Two main approaches are used for DA, variational and statistical. Statistical DA relies on multiple concurrent simulations to compute an estimate of the numerical model error. Several methods exist depending on the context for combining the simulation and observation data and derive steering decisions on the simulations. At the moment, the most popular statistical method is the Ensemble Kalman Filter (EnKF).

3.3 Complex Optimization

Complex optimization problems require large computational resources and often the use of multiple simulations. We cite three examples:

- **Shape Optimization** consists in finding an optimal shape, i.e. the shape that minimizes a certain cost functional under some given constraints. In many cases, the functional being solved depends on the solution of a partial differential equation defined on the variable domain.
  Why shape optimization needs the execution of multiple simulations? The domain of possible solutions can be explored by repeating simulations, each one considering a different candidate shape that is tested under the given constraints. This kind of problem is often not only complex but very time consuming. An example could be finding the shape of a boat under specific conditions of sea currents and weather

conditions. Computational Fluid Dynamics (CFD) simulations could be repeated to find the optimal shape of the boat, for instance for a minimal use of fuel or for a maximal boat stability.

- **Reinforcement Learning (RL)** consists in learning how to choose a sequence of actions in a given environment (based on a simulation code) to maximize a cumulative reward criteria. A typical example is to self-learn to play chess. Learning takes place through an iterative process, where the learning algorithm intends to progressively build a wining strategy by submitting actions to an environment and correcting its strategy according to the environment feedback.
  Why reinforcement learning needs the execution of multiple simulations? To speed-up the learning process multiple environments are running in parallel, enabling to explore faster different scenarios. The most visible success of RL is probably AlphaGo Zero that outperformed the best human go-players. The learning process required to run 4.9 million go games during 70 hours using 64 GPU workers and 19 CPU parameter servers. Today, in most cases the environment requires limited compute resources satisfied by a single node and/or one GPU. But RL is also emerging as an optimization technique for scientific applications, such as in [44]. We can expect that training with advanced parallel simulations will thus require supercomputers.
- **Hyper-parameter Tuning** for machine learning. A hyper-parameter is a parameter whose value is used to control a learning process. Tuning hyper-parameters consists in choosing a set of optimal hyper-parameters for a learning algorithm. Automatic approaches for hyper-parameter tuning are today a major research challenge, for instance, for deep learning.
  Why hyper-parameter tuning needs the execution of multiple simulations? Learning a model over a dataset with fixed hyper-parameters is a computational process that can be performed in a supercomputer. Finding the best set of hyper-parameters consists in repeating multiple times the learning process and compare its final quality via a established criteria.

4 LESSONS LEARNED

Over the last 5 years, we have developed the Melissa framework for enabling large scale sensitivity analysis from multiple simulation runs[1]. Please refer to [42] for a description of its architecture. Combining in transit methods and iterative statistics enabled to develop an elastic and file-avoiding approach. We share here the lessons learned through the development and use of Melissa.

4.1 Probes and Subsampling

An ensemble (the results from multiple simulation runs) is multivariate, its members (the result from one run) multidimensional (both in space and time) and multivalued (several quantities such as temperature, pressure, or velocity are considered). In our interaction with users, we observed that the usual way to deal with ensembles is to limit their size to reduce I/O pressure and facilitate the post hoc data analysis. Two main techniques are used for this purpose:

- Subsample the simulation outputs. In general the numerical simulation is performed at full resolution but the outputs are subsampled in the temporal/spatial domain.
- Using probes. The analysis is limited to several spatial locations or probes. This leads to smaller and more tractable ensembles of functional outputs, typically an ensemble of curves (see [38] for an example of this strategy).

Tightly-coupled in situ solutions are not directly applicable in this context because the data from different simulations need to be combined to compute statistics. On the other side, in transit solutions, where the intermediate data produced by members are not saved to disk but rather sent to dedicated staging nodes in charge of computing these statistics, would enable to bypass the file system and thus compute results at significantly higher resolution. However, the memory capabilities of these in transit nodes are several order of magnitude smaller than the file system. Specific strategies need to be developed to be able to gather and process the results from members, while staying within the node memory capabilities.

4.2 Iterative Statistics Unlock In Transit Analysis

Computing statistics from N samples classically requires $O(N)$ memory space to store these samples. But if the statistics can be *computed in one-pass* (also called iterative, on-line or even parallel), i.e. if the current value can be updated as soon as a new sample is available, the memory requirement goes down to $O(1)$ space. With this approach, not only simulation results do not need to be saved, but they can be consumed in any order, loosening synchronization constraints on the simulation executions. This is the key that enables an in transit approach.

4.3 Design of Experiments and Elasticity

The *design of experiments* defines how to sample the parameters of a UQ study. It can go from classical monte carlo sampling to more advanced approaches. The motivation is to try to keep a good coverage of the parameter space while reducing the sample size, i.e. the number of runs. The selected approach can influence the elasticity, or in other words the capability for asynchronism. Monte carlo sampling is extremely flexible as drawing a new set of parameters is independent from previous samples. The order of member execution is not constrained. The number of members can be adjusted online, according to convergence criteria for instance. Faulty parameter sets can be replaced with new ones (see subsection 4.4). In case of a run failure linked to a given parameter choice, the fault tolerance protocol can replace the faulty run with a new one, drawing a new set of parameters.

Other designs of experiments can require to increase the granularity of jobs. For instance, Melissa computes Sobol's indices using the pick-freeze method, requiring to execute simulations by groups of P+2 members (P being the number of varying parameters)[42].

[1] https://melissa-sa.github.io/

4.4 Fault Tolerance and Statistical Sampling

Melissa asynchronous client/server architecture and the iterative statistics computation enable to support a simple yet robust fault tolerance mechanism [42]. The protocol fully leverages the fact that data processing relies on iterative statistics, which enables to process data produced by members in any order. Thus, a failing simulation can be restarted independently from the others. Additionally, faults mainly come from numerical errors related to a specific set of input parameters. Restarting the same simulation usually leads to the same error. However, it is often statistically valid (in accordance with the design of experiments) to replace the failing simulation by another one running with a new generated parameter set.

5 CHALLENGES

In this section, we identify four different challenges to be addresses to enable high performance multi-simulations data processing: parallelism, iterative mathematics, software infrastructure, visualisation and monitoring. These encompass and extend the challenges usually identified for single simulation in situ data processing [14, 34].

5.1 Parallelism

Based on our experience, an important question concerning parallelism appears straightforward: is it possible to keep the level of massive parallelism achieved by Melissa in other cases such us Data Assimilation or Complex Optimization? For this to be performed, some specific aspects should be studied:

- Elasticity. Can the proposed solutions enable the dynamic adaptation of compute resource usage according to availability? Data Assimilation needs synchronization points each time observation data are available for assimilation. Complex optimization needs feedback on how the objective function is being reduced. Thus, these two cases present a lower level of massive parallelism and elasticity than UQ. An important challenge would consist in finding methods that will utilize the highest level of concurrent execution respecting the constraints that impose synchronizations between simulations.
- Fault Tolerance. Can a failed simulation be substituted by a different one in the design of experiences without modifying the result? This is possible for UQ studies. However, is that also valid, for instance, for optimization problems? It should be possible for a category of optimization algorithms using statistical sampling but other methods may not be compatible with such assumptions. Studying how optimization and DA algorithms allow for fault tolerance is an open question worth to be studied.

5.2 Iterative Mathematics

The combination of iterative mathematics and in situ techniques is a game changer because it allows the on-line data aggregation of high-resolution ensemble runs [42]. Indeed, as soon as each available simulation provides its results, the current value of the desired statistic can be iteratively updated. This in transit processing mode enables to fully avoid storage of intermediate data on disks.

However, this approach implies that the problem to solve can be expressed in iterative terms, which is not necessarily easy or even doable. For instance, Melissa currently implements the on-line computation of quantiles [37]. The iterative computation of quantiles presents mathematical challenges and it is currently an active research subject. This illustrates that collaboration with mathematicians is often needed to unlock this kind of problems.

Finding iterative optimization or DA algorithms that could be efficient in an in situ context is a challenge worth to be studied. Questions such as "What mathematical guarantees can we put on the resulting analyses?" or "What are the convergence and precision of the results?" should be answered if we want users to be confident in the use of in situ techniques in this context.

5.3 Software Infrastructure

A question of practical importance arises for the treatment of multi-run simulations: should we develop specific platforms for these new problems or can we re-use/extend existing tools?

We first cite some open questions formulated in [9] concerning future platforms in this context:

- 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.
- How can we support ensembles scaling to a billion members and beyond and enable an analysis of the resulting high-dimensional data space?
- 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.

Frameworks are available for the different use-cases identified. Dakota [1], UQLab [27] and OpenTurn[8] for UQ, EnTK [7] and PDAF [31] for DA, RLlib [23] for Reinforcement Learning, Scikit-learn [33] or Tune [24]for Hyperparameter search. Some, like RLlib, Tune, Sciki-learn and EnTK rely on a emerging generation of Python based distributed task based programming frameworks, like Dask [39], Ray [30], Parsl [6] or Radical-Pilot [28], to provide some level of flexible parallelization. But often scalability is limited, coupling with large-scale parallel simulations is not supported or data exchange is file based.

5.4 Visualisation

Uncertainty visualization has been long advocated as one of the top challenges in visualization [32]. Another challenge of the visualization community has been how to deal with the multivariate nature of the ensembles [26]. This leaded to the construction of tools for the post hoc visualization of ensembles (see for instance [40] or [35]). With the emergence of multi simulation studies (UQ, DA and complex optimization) at exascale, visualization techniques for ensembles will certainly need to be revisited and extended.

Furthermore, visualization may also be important in answering the following question: how can we monitor that these massive multiple simulations runs are being correctly executed? When, for

instance, executing 10,000 simulations, using log files or debugging in order to understand its execution becomes impossible. The monitoring of these complex exascale processes becomes a challenge for itself.

6 CONCLUSIONS

In the past years, we have developed the open-source software Melissa targeting in transit analysis for UQ studies. Melissa proposes a new approach to compute statistics at large scale by avoiding to store the intermediate results produced by multiple simulation runs.

We have identified other use cases relying on multiple simulation runs, Data Assimilation (DA) and Complex Optimization (CO), showing that data analysis for multiple simulations has a growing importance. Based on the Melissa experience, we have analyzed and extracted lesson from the treatment of UQ studies. Thus, we have identified similarities and challenges specific to other scenarios. Taking an in situ perspective on problems such as DA or CO could certainly bring innovative solutions.

ACKNOWLEDGEMENT

This work has been partly funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 824158 (EoCoE-2).

REFERENCES

[1] Brian M. Adams, Mohamed S. Ebeida, Michael S. Eldred, Gianluca Geraci, John D. Jakeman, Kathryn A. Maupin, Jason A. Monschke, Laura P. Swiler, J. Adam Stepheans, Dena M. Vigil, Timothy M. Wildey, William J. Bohnhoff, Keith R. Dalbey, John P. Eddy, Joseph R. Frye, Russell W. Hooper, Kenneth T. Hu, Patricia D. Hough, Mohammad Khalil, Elliott M. Ridgway, and Ahmad Rushdi. [n.d.]. *Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.6 User's Manual.* Technical Report. Sandia Technical Report SAND2014-4633, July 2014. Updated May 2017.

[2] J. Ahrens, S. Jourdain, P. OLeary, J. Patchett, D. H. Rogers, and M. Petersen. 2014. 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*. 424–434.

[3] Mark Asch, Marc Bocquet, and Maëlle Nodet. 2016. *Data assimilation: methods, algorithms, and applications*. Vol. 11. SIAM.

[4] Utkarsh Ayachit, Andrew Bauer, Berk Geveci, Patrick O'Leary, Kenneth Moreland, Nathan Fabian, and Jeffrey Mauldin. 2015. Paraview catalyst: Enabling in situ data analysis and visualization. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*. 25–29.

[5] U. Ayachit, B. Whitlock, M. Wolf, B. Loring, B. Geveci, D. Lonie, and E. W. Bethel. 2016. The SENSEI Generic In Situ Interface. In *2016 Second Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV)*. 40–44.

[6] Yadu Babuji, Anna Woodard, Zhuozhao Li, Daniel S Katz, Ben Clifford, Ian Foster, Michael Wilde, and Kyle Chard. 2019. Scalable Parallel Programming in Python with Parsl. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*. 1–8.

[7] Vivek Balasubramanian, Matteo Turilli, Weiming Hu, Matthieu Lefebvre, Wenjie Lei, Guido Cervone, Jeroen Tromp, and Shantenu Jha. 2018. Harnessing the Power of Many: Extensible Toolkit for Scalable Ensemble Applications. In *IPDPS 2018*.

[8] M. Baudin, A. Dutfoy, B. Iooss, and A-L. Popelin. 2017. Open TURNS: An industrial software for uncertainty quantification in simulation. In *Springer Handbook on Uncertainty Quantification*, R. Ghanem, D. Higdon, and H. Owhadi (Eds.). Springer, 2001–2038.

[9] Janine C. Bennett, Hank Childs, Christoph Garth, and Bernd Hentschel. 2019. In Situ Visualization for Computational Science (Dagstuhl Seminar 18271). *Dagstuhl Reports* 8, 7 (2019), 1–43. http://drops.dagstuhl.de/opus/volltexte/2019/10171

[10] Surendra Byna, Jerry Chou, Oliver Rubel, Homa Karimabadi, William S Daughter, Vadim Roytershteyn, E Wes Bethel, Mark Howison, Ke-Jou Hsu, Kuan-Wu Lin, et al. 2012. Parallel I/O, analysis, and visualization of a trillion particle simulation. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–12.

[11] Jose J Camata, Vitor Silva, Patrick Valduriez, Marta Mattoso, and Alvaro LGA Coutinho. 2018. In situ visualization and data analysis for turbidity currents simulation. *Computers & Geosciences* 110 (2018), 23–31.

[12] Ciprian Docan, Manish Parashar, and Scott Klasky. 2012. DataSpaces: an Interaction and Coordination Framework for Coupled Simulation Workflows. *Cluster Computing* 15, 2 (2012), 163–181.

[13] Matthieu Dorier, Gabriel Antoniu, Franck Cappello, Marc Snir, and Leigh Orf. 2012. Damaris: How to efficiently leverage multicore parallelism to achieve scalable, jitter-free I/O. In *2012 IEEE International Conference on Cluster Computing*. IEEE, 155–163.

[14] Matthieu Dorier, Matthieu Dreher, Tom Peterka, Justin M Wozniak, Gabriel Antoniu, and Bruno Raffin. 2015. Lessons learned from building in situ coupling frameworks. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*. 19–24.

[15] Matthieu Dreher and Tom Peterka. 2017. *Decaf: Decoupled dataflows for in situ high-performance workflows*. Technical Report. Argonne National Lab.(ANL), Argonne, IL (United States).

[16] Matthieu Dreher and Bruno Raffin. 2014. 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*. IEEE, 277–286.

[17] Tony Hey, Stewart Tansley, Kristin Tolle, et al. 2009. *The fourth paradigm: data-intensive scientific discovery*. Vol. 1. Microsoft research Redmond, WA.

[18] James A Kohl, Torsten Wilde, and David E Bernholdt. 2006. Cumulvs: Interacting with high-performance scientific simulations, for visualization, steering and fault tolerance. *The International Journal of High Performance Computing Applications* 20, 2 (2006), 255–285.

[19] James Kress, Scott Klasky, Norbert Podhorszki, Jong Choi, Hank Childs, and David Pugmire. 2015. Loosely coupled in situ visualization: A perspective on why it's here to stay. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*. 1–6.

[20] T Kuhlen, R Pajarola, and K Zhou. 2011. Parallel in situ coupling of simulation with a fully featured visualization system. In *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization (EGPGV)*, Vol. 10. Eurographics Association Aire-la-Ville, Switzerland, 101–109.

[21] Steven H. Langer, Brian Spears, J. Luc Peterson, John E. Field, Ryan Nora, and Scott Brandon. 2016. A HYDRA UQ Workflow for NIF Ignition Experiments. In *Proceedings of the 2Nd Workshop on In Situ Infrastructures for Enabling Extreme-scale Analysis and Visualization* (Salt Lake City, Utah) *(ISAV '16)*. IEEE Press, Piscataway, NJ, USA, 1–6.

[22] Matthew Larsen, James Ahrens, Utkarsh Ayachit, Eric Brugger, Hank Childs, Berk Geveci, and Cyrus Harrison. 2017. 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* (Denver, CO, USA) *(ISAV'17)*. Association for Computing Machinery, New York, NY, USA, 42–46.

[23] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. 2018. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*. 3053–3062.

[24] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. 2018. Tune: A Research Platform for Distributed Model Selection and Training. *arXiv preprint arXiv:1807.05118* (2018).

[25] Jay F. Lofstead, Scott Klasky, Karsten Schwan, Norbert Podhorszki, and Chen Jin. 2008. 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* (Boston, MA, USA) *(CLADE '08)*. Association for Computing Machinery, New York, NY, USA, 15–24.

[26] Alison L Love, Alex Pang, and David L Kao. 2005. Visualizing spatial multivalue data. *IEEE Computer Graphics and Applications* 25, 3 (2005), 69–79.

[27] Stefano Marelli and Bruno Sudret. 2014. UQLab: A framework for uncertainty quantification in Matlab. In *Vulnerability, uncertainty, and risk: quantification, mitigation, and management*. 2554–2563.

[28] Andre Merzky, Mark Santcroos, Matteo Turilli, and Shantenu Jha. 2015. RADICAL-Pilot: Scalable execution of heterogeneous and dynamic workloads on supercomputers. *CoRR, abs/1512.08194* (2015).

[29] Kenneth Moreland, Ron Oldfield, Pat Marion, Sebastien Jourdain, Norbert Podhorszki, Venkatram Vishwanath, Nathan Fabian, Ciprian Docan, Manish Parashar, Mark Hereld, et al. 2011. Examples of in transit visualization. In *Proceedings of the 2nd international workshop on Petascal data analytics: challenges and opportunities*. 1–6.

[30] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. 2017. Ray: A Distributed Framework for Emerging AI Applications. arXiv:cs.DC/1712.05889

[31] Lars Nerger and Wolfgang Hiller. 2013. Software for ensemble-based data assimilation systems—Implementation strategies and scalability. *Computers &*

Geosciences 55 (2013), 110–118.

[32] Alex Pang, Craig Wittenbrink, and Suresh Lodha. 1997. Approaches to Uncertainty Visualization. *The Visual Computer* 13, 8 (Nov 1997), 370–390.

[33] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[34] Tom Peterka, Deborah Bard, Janine Bennett, E. Wes Bethel, Ron Oldfield, Line Pouchard, Christine Sweeney, and Matthew Wolf. 2019. ASCR Workshop on In Situ Data Management: Enabling Scientific Discovery from Diverse Data Sources. (2 2019). https://doi.org/10.2172/1493245

[35] Kristin Potter, Andrew Wilson, Peer-Timo Bremer, Dean Williams, Charles Doutriaux, Valerio Pascucci, and Chris R Johnson. 2009. Ensemble-vis: A framework for the statistical visualization of ensemble data. In *2009 IEEE International Conference on Data Mining Workshops*. IEEE, 233–240.

[36] Alejandro Ribés, Benjamin Lorendeau, Julien Jomier, and Yvan Fournier. 2015. In-situ visualization in computational fluid dynamics using open-source tools: integration of catalyst into Code_Saturne. In *Topological and Statistical Methods for Complex Data*. Springer, 21–37.

[37] Alejandro Ribes, Théophile Terraz, Bertrand Iooss, Yvan Fournier, and Bruno Raffin. 2019. Large scale in transit computation of quantiles for ensemble runs. arXiv:math.ST/1905.04180

[38] Alejandro Ribés, Joachim Pouderoux, and Bertrand Iooss. 2020. A Visual Sensitivity Analysis for Parameter-Augmented Ensembles of Curves. *Journal of Verification, Validation and Uncertainty Quantification* 4, 4 (02 2020). 041007.

[39] Matthew Rocklin. 2015. Dask: Parallel Computation with Blocked algorithms and Task Scheduling. In *Proceedings of the 14th Python in Science Conference*, Kathryn Huff and James Bergstra (Eds.). 130 – 136.

[40] Jibonananda Sanyal, Song Zhang, Jamie Dyer, Andrew Mercer, Philip Amburn, and Robert Moorhead. 2010. Noodles: A tool for visualization of numerical weather model ensemble uncertainty. *IEEE Transactions on Visualization and Computer Graphics* 16, 6 (2010), 1421–1430.

[41] R.C. Smith. 2014. *Uncertainty quantification*. SIAM.

[42] Théophile Terraz, Alejandro Ribés, Yvan Fournier, Bertrand Iooss, and Bruno Raffin. 2017. Melissa: Large Scale In Transit Sensitivity Analysis Avoiding Intermediate Files. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'17)*. Denver.

[43] H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K. Ma. 2010. In Situ Visualization for Large-Scale Combustion Simulations. *IEEE Computer Graphics and Applications* 30, 3 (2010), 45–57.

[44] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. 2019. Optimization of molecules via deep reinforcement learning. *Scientific reports* 9, 1 (2019), 1–10.