A Flexible System For In Situ Triggers

Matthew Larsen[*]
Lawrence Livermore Nat'l Lab

Amy Woods
Lawrence Livermore Nat'l Lab

Nicole Marsaglia
University of Oregon

Ayan Biswas
Los Alamos National Lab

Soumya Dutta
Los Alamos National Lab

Cyrus Harrison
Lawrence Livermore Nat'l Lab

Hank Childs
University of Oregon

ABSTRACT

Triggers are an important mechanism for adapting visualization, analysis, and storage actions. With this work, we describe the Ascent in situ infrastructure's system for triggers. This system splits triggers into two components: when to perform an action and what actions to perform. The decision for when to perform an action can be based on different types of factors, such as mesh topology, scalar fields, or performance data. The actions to perform are also varied, ranging from the traditional action of saving simulation state to disk to performing arbitrary visualizations and analyses. We also include details on the implementation and short examples demonstrating how the system can be used.

ACM Reference Format:
Matthew Larsen, Amy Woods, Nicole Marsaglia, Ayan Biswas, Soumya Dutta, Cyrus Harrison, and Hank Childs. 2018. A Flexible System For In Situ Triggers. In *ISAV: In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV '18), November 12, 2018, Dallas, TX, USA.* Denver, CO, USA, 6 pages. https://doi.org/10.1145/3281464.3281468

1 INTRODUCTION

Computational simulations produce large amounts of data, and managing this data is an important part of these simulations' workflow. Common actions available to simulations for processing their data are to store the data to disk, to apply in situ visualization and analysis techniques, or a combination of the two (i.e., use in situ techniques to transform and/or subset data and then store the result to disk). That said, perhaps the most common action for processing simulation data is to discard it — the data produced at a given cycle is thrown away as the simulation advances.

A simulation's workflow for visualization, analysis, and/or storage (VAS) can be either non-adaptive or adaptive. With a non-adaptive workflow, the VAS actions occur based on some fixed policy that is established before the simulation begins. Examples of such fixed policies are to perform VAS every N cycles, to perform VAS every time a simulation has advanced T seconds of simulation time, or to perform VAS every M minutes of time the simulation runs on a computer. With these workflows, much of the simulation's data is not inspected for VAS purposes. For example, if the policy is to save to disk every 100 cycles, then 99% of the data is not subject to VAS, i.e., a temporal subsampling. Temporal subsampling is often acceptable, as simulation data from one cycle to the next is often very similar. However, if the proportion of the data inspected gets too low, then temporal subsamplings can be highly problematic, as key phenomena may occur during the uninspected cycles. In such cases, the phenomena may affect the data from subsequent cycles to a degree that scientists determine important information was lost, in which case the simulation could be re-run. Or, worse, the phenomena may go undiscovered altogether.

Adaptive workflows behave differently. With adaptive workflows, the data produced by a simulation is regularly inspected (often every cycle). The job of these inspections is to determine if interesting phenomena had occurred. If so, then the inspection process should prompt VAS actions to occur. This approach has the potential to prevent the loss of key phenomena, provided the right VAS actions are taken at the right time. This approach has been gaining momentum with many recent results (see the Related Work section), and is termed "in situ triggers," since the inspection routines are applied in situ to the simulation data and the routines can "trigger" VAS actions to take place.

With this work, we describe the Ascent in situ library's mechanism for in situ triggers. Our focus with this work is to provide a flexible system that enables a variety of trigger approaches. Our system defines a trigger as two pieces: (1) an inspection routine that determines when VAS should occur and (2) the specific VAS actions to apply. We feel this approach will encourage re-use, as Ascent developers can focus on developing inspection routines and/or VAS actions, and then pair them as appropriate. Further, while the two pieces that make up a trigger can be completely orthogonal, they also can have a dependency. Specifically, the inspection routines can dynamically alter the VAS operations used.

[*]email: larsen30@llnl.gov

ISA V '18, November 12, 2018, Dallas, TX, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6579-6/18/11.
https://doi.org/10.1145/3281464.3281468

In the remainder of this paper, we describe Ascent's trigger system, as well as the inspection routines and VAS actions we have developed to date. The inspection routines are based on simulation field data, simulation mesh topology, and simulation state data, and our VAS actions involve all of the capabilities within Ascent.

2 RELATED WORK

Recent research into triggers can be characterized into two categories: triggers that are domain-agnostic and triggers that are domain-specific, and the organization of this section mirrors this current division in research. Conceptually, each of the works surveyed below could be supported by our system, although in some cases new inspection routines or VAS actions would need to be added to Ascent.

2.1 Domain-Agnostic Triggers

Domain-agnostic triggers leverage algorithms that apply broadly to arrays of values (e.g., simple aggregation, summary statistics, many machine learning techniques). These can be used generically by any simulation application, so there is value to sharing implementations.

Work by Ling et al. [10] implements an autonomous, domain-agnostic trigger, but their method involves several machine learning algorithms that are computationally significant, which is likely a barrier for many in situ use cases. Malakar et al. [11] take into account the computational strain of in situ analysis and present a mathematical formula for choosing the optimal process-to-node mapping given the system's constraints.

Zhou and Chiang [18] and Myers et al. [12] utilized statistical methods to determine time steps of global and local significance, respectively, though the former work is not performed in situ. Banesh et al. [1] expanded on this work by utilizing change detection techniques for both time-based and specific parameter-based eddy identification in simulated ocean data. And while this research utilizes application-specific parameters for their detection, the concept of change point detection can be applied to other simulations [15].

2.2 Domain-Specific Triggers

Domain-specific triggers leverage algorithms that use special knowledge of either a simulation method or its scientific domain (e.g., triggering on a gradient calculated using specific finite-element method, or enstrophy for as a measure of dissipation). These are not as broadly applicable to any simulation, so it is important for an in situ infrastructure to support integrating custom algorithms to support these types of triggers.

A number of works have developed domain-specific triggers. Work by Bennett et al. [2] developed an application-specific trigger for ignition during combustion simulations. This work was then extended by Salloum et al. [13], who provided an alternative trigger metric that increased the robustness of the original work. Similarly, work by Zhao et al. [17] and Ullrich et al. [14] created domain-specific triggers for tropical cyclone trackers within climate simulations.

3 TRIGGER INFRASTRUCTURE OVERVIEW

Our trigger infrastructure is implemented in Ascent [9], a fly-weight in situ visualization and analysis infrastructure. Ascent's flexibility is underpinned by a generic data-flow network. Filters in a data-flow network have arbitrary inputs and outputs, and each component of Ascent is implemented as a filter. The Ascent runtime translates data and actions described in the high-level API into a graph that is executed by the data-flow network. Using this modular architecture, all of Ascent's components can be connected together. The trigger implementation in Ascent is simply another filter in a larger data-flow network that can conditionally modify the filters in the execution graph.

In this section, we discuss what types of simulation data are used to make decisions, how decisions are made, and what types of actions are supported.

3.1 Trigger Input

Ascent's trigger filters process Conduit [7] data that conforms to the Mesh Blueprint [8]. This is the same data representation used to publish data to Ascent. Since outputs from Ascent operations can also be converted to Mesh Blueprint data, this representation allows triggers to operate on Ascent results as well. The triggers have access to all of the mesh data, however it is useful to further categorize triggers by the parts of simulation mesh required by inspection routines. Categorizing triggers is useful for declaring a trigger's purpose (i.e., what data is used to make decisions) and for sharing common parameter verification code that checks pre-conditions for each category.

In Ascent, we define the following categories of triggers:

- **Field**: inspect a specific field on the mesh. Field triggers have access to all the values of a named field on the mesh. A simple example of a field trigger is one that fires when the maximum value exceeds a user-defined threshold.
- **Topology**: inspect a specific mesh topology. Topology triggers are passed a named mesh topology. An example topology trigger would inspect the mesh, firing when a tangle is detected.
- **Coordinate Set**: inspect a specific coordinate set from the mesh. Coordinate Set triggers are passed a named coordinate set. An example coordinate set trigger would inspect the coordinates, firing when a bounding box grows to a given size.
- **State**: inspect state meta-data. State triggers consume extra meta-data associated with the mesh. This includes both basic information such as the simulation time and cycle, and custom data, such as simulation performance metrics. Examples of state triggers are fire every N cycles or fire when a performance metric rises above some critical threshold.

3.2 Trigger Decisions

Triggers make a binary (yes/no) decision to execute a set of actions. In simple cases, a trigger can inspect a single value, e.g., firing every N cycles. Trigger decisions occur simultaneously on all MPI processes to determine a single collective decision. However, when using problem size data such as a mesh field, the data must usually be reduced before making a final decision to fire.

3.2.1 Data Reduction. Data reduction may leverage a pipeline of existing VAS actions, or it may be a simple data summarization algorithm. While triggers can leverage expensive algorithms, in practice they are often used to reduce overall computation by screening data before applying more expensive VAS actions. Because of this, inexpensive data reduction techniques are important.

The most basic technique is a simple aggregation of a field or a topological feature (minimum, maximum, mean, variance, integral, etc). These are inexpensive because they do not require much intermediate memory, are typically simple linear loops over values and their distributed-memory implementations are straightforward. In Ascent, we provide a small set of methods that serve as building blocks for trigger development and encourage code re-use. Current methods include retrieving the minimum and maximum values of a field, along with the element or vertex it originated from, and the element's location. As we continue to develop Ascent, the set of available methods will increase.

A more sophisticated way to reduce data is to calculate a coarse distribution of a field or a topological feature. This can be calculated inexpensively with a counting or weight-based histogram. Calculating exact cumulative distributions requires an expensive global sort, so we do not plan to implement these. Multi-dimensional binning can also be used to calculate coarse averages which can be overlaid back onto the mesh topology as input to create more complex derived quantities for summarization [4]. For distributions, you can further reduce them using another summary metric, such as identifying percentiles or calculating the shannon entropy.

3.2.2 Decision to fire. After data reduction, triggers apply a final test to determine if the trigger should fire.

The Ascent trigger infrastructure allows two trigger variations:

- **Stateless**: trigger execution is independent of any previous invocations
- **Stateful**: the trigger that maintains state information about previous invocations

Stateless decisions are made using only the simulation's current published data, and stateful decisions can use the time history of the aggregate values from previously published data. In the stateless case, the standard menu of numeric comparison operations (e.g. greater-than, less-than, equal, etc) are useful building blocks for deciding to fire using a single value. For the stateful case, trigger firing can be based on critical points of the curve (minimum, maximum, saddle points), or leverage a more complex property, such as the value changing more than some percentage of the current peak. In Ascent we provide methods for storing and retrieving arbitrary state data, and state data can range from a single value, a series of values from all previous time steps, or to the entire published data from a previous time step.

3.3 Trigger Actions

Ascent's interface supports execution of a set of actions described by the user. These actions direct Ascent to create pipelines (i.e., transform data), extracts (i.e., capture data), and scenes (i.e., make pictures). The new trigger system extends Ascent's interface to accept trigger declarations.

The declaration accepts a trigger name, any input parameters needed for the inspection routine, and a set of actions to execute if the trigger fires. The set of actions is declared using Ascent's existing action interface, thus exposing all of Ascent's capabilities including declaration of new triggers.

Examples of trigger actions are saving the entire mesh to disk, creating a pipeline to transform the data and saving the resulting extract, or rendering images. Additionally, trigger actions can include other triggers that create a set of conditions (contained in multiple triggers), which all must be satisfied before the final set of actions is performed. Since triggers have access to the input actions, it is possible for triggers to themselves modify the actions, creating dynamically adaptive visualization workflows.

3.4 Trigger API

Triggers are described using Conduit nodes, following Ascent's existing actions interface. Listing 1 shows an example of specifying a stateless threshold trigger of the field pressure that fires when the condition is true.

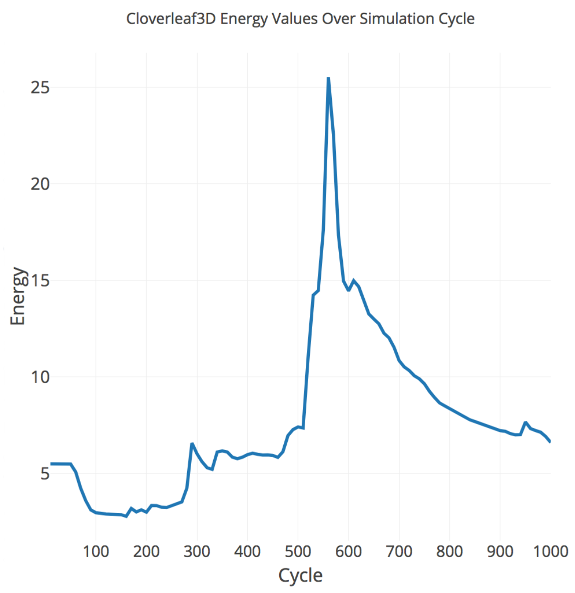
```
conduit::Node actions;
// trigger actions not shown
conduit::Node trigger;
trigger["type"] = "threshold";
trigger["params/field"] = "pressure";
trigger["params/actions"] = actions;
trigger["params/reduction"] = "max";
trigger["params/compare/type"] = "gte";
trigger["params/compare/value"] = 3.14;
```

Listing 1: An example of the user-facing trigger API. This trigger executes the actions when the maximum value of the pressure field is greater than or equal to 3.14.

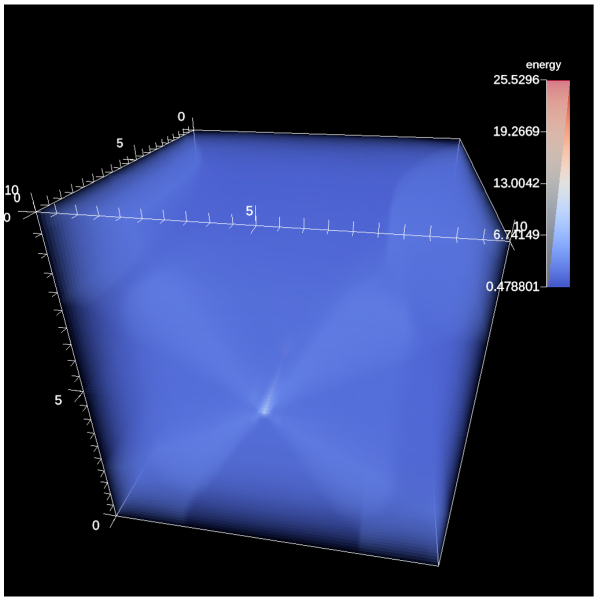
The threshold trigger supports several reductions including "min", "max", and "average." As we continue to develop the infrastructure, the menu of available reduction operations will increase. After the reduction, the trigger evaluates the condition "gte" (greater-than-or-equal-to) against the "value" and the result of the reduction.

3.5 Developing New Triggers in Ascent

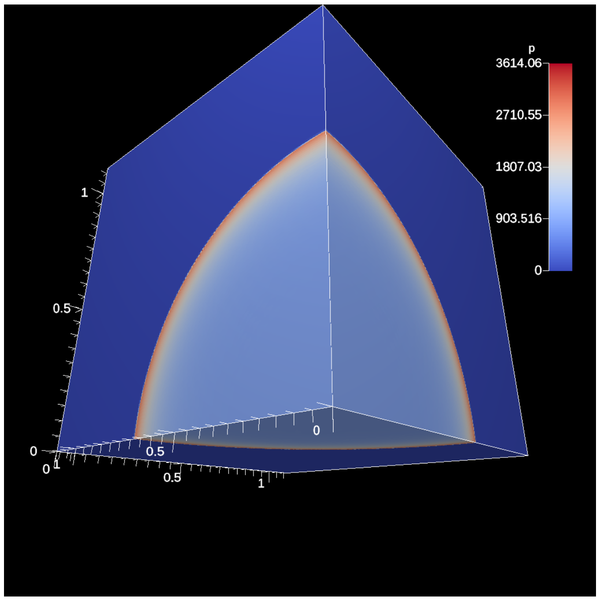
Our goal in Ascent is making trigger development as easy as possible. To that end, we provide a trigger base class that includes methods to verify parameters, to provide easy access to input data (e.g., state, field, coordinate sets, and topology



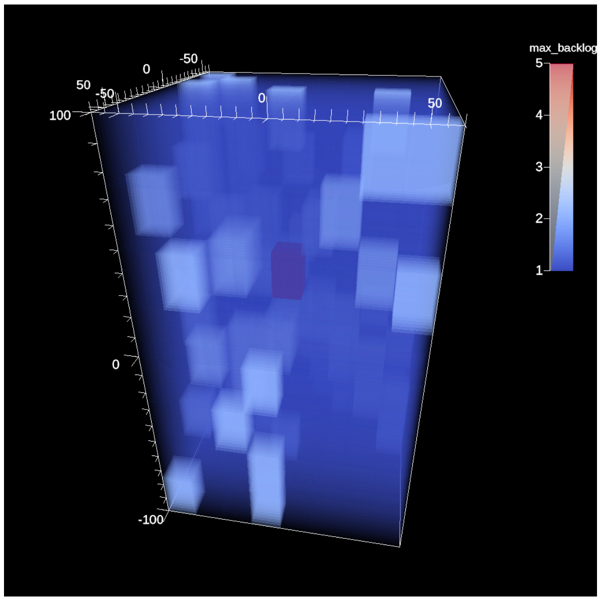
(a)



(b)



(c)



(d)

Figure 1: (a) a plot of the maximum energy value over time in CloverLeaf3D, (b) an image created by a threshold trigger in CloverLeaf3D, (c) an image created by a max radial distance trigger at cycle 3120 in Lulesh, and (d) an image created by a simulation state trigger in Kripke.

data), and to execute the trigger. The only methods a trigger must implement are the "trigger" function that returns true or false and the verification function of any trigger specific parameters. A trigger implementer only needs to manipulate data contained in Conduit nodes and does not need any knowledge about the internals of Ascent. Additionally, we are building a set of data reductions, such as the ones mentioned in Section 3.4 for trigger developers.

4 TRIGGER EXAMPLES

In this section, we demonstrate a few examples of using Ascent triggers that inspect different combinations of trigger inputs. These examples use Ascent's built-in proxy applications because they provide a simple path to motivate and show common use cases.

4.1 Threshold Trigger

The maximum value of a field value over time is basic quantity of interest for many physics simulations.

We developed threshold triggers to help examine maximum energy field values in CloverLeaf3D [3], a hydrodynamics proxy-application included with Ascent.

To track the maximum value, stateless triggers should fire whenever the maximum field value exceeds a predefined value as described in Listing 1. The image in Figure 1a plots the maximum energy value through 1000 cycles of a CloverLeaf3D simulation. The maximum value spikes as waves of energy reflect off the boundary and collide in the center of the data set, indicated by a 400% increase in the maximum value.

We used a stateless threshold trigger to capture the collision by setting the threshold at an energy value of 10, and image captured by this trigger is shown in Figure 1b. Additionally, we developed a stateful trigger that identified that peak of the energy spike by tracking the maximum over all timesteps. With the stateful trigger, the entire field is stored in memory each time a higher value is encountered, and at the end of the simulation the data from timestep identified as maximum is saved to disk for further analysis.

4.2 Maximum Radial Distance Example

There are several cases where expensive processing is only necessary after a simulation model has evolved to a certain regime that cannot easily be identified a priori based on simulation time.

To demonstrate this type of use case, we developed a trigger for Lulesh [5], a shock-hydrodynamics proxy-application that models the evolution of a shock-wave as it propagates from the origin of the simulation. The trigger executes actions after the main shock-wave has reached a specified radial distance from the origin of the simulation.

Knowing exactly when the shock-wave passes a point is problem dependent since it changes both with the input deck and problem size. In Ascent, the maximum radial distance trigger combines field, topology, and coordinate set information to track the distance of the element containing the maximum pressure value. When this distance exceeds the threshold, the trigger fires. Figure 1c shows an image of the first time the trigger fires.

4.3 Simulation State Trigger

Kripke [6] is a physics proxy-application included with Ascent that implements a parallel sweep used to solve equations for deterministic neutron transport.

Each sweep consists of wavefronts for each quadrature direction that propagate through the MPI tasks. The Ascent integration captures sweep solver performance metrics, including the maximum backlog of incoming requests from different quadrature directions.

In order to show an overall view of the maximum backlog for each MPI task, we created a performance trigger that fires when the maximum backlog exceeds four unanswered requests. Using a volume plot shown in Figure 1d, MPI tasks with higher backlogs appear more opaque than tasks with lower backlogs, similar to the visualizations in [16]. Using performance visualization techniques can both highlight performance bottlenecks and provide useful visual debugging information for simulation developers.

5 CONCLUSION

In this paper, we described a flexible system for in situ triggers inside of Ascent. Supporting trigger use cases with traditional visualization tools requires control flow logic be written outside of the tool to coordinate trigger decisions and VAS actions. In contrast, our work directly integrates trigger capabilities into an in situ visualization and analysis infrastructure to simplify these use cases. The trigger system uses modular components that provide different options for VAS actions, comparisons, and reduction methods (if needed). For trigger developers, we provide re-usable and interchangeable components that lower barriers to trigger development. We also demonstrated how different categories of triggers can be used to identify features of interest (e.g., threshold), filter out un-needed data (e.g., maximum radial distance). and highlight simulation performance bottlenecks (e.g., simulation state).

For future work, we intend to continue to refine Ascent's in situ trigger system and provide additional reduction and summarization building blocks. Additionally, we would like to explore more dynamic triggers workflows where trigger modify the input actions.

6 ACKNOWLEDGMENTS

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC (LLNL-CONF-757069). Accordingly, the United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this article or allow other to do so, for United States Government purposes.

REFERENCES

[1] Divya Banesh, Joanne Wendelberger, Mark Petersen, James Ahrens, and Bernd Hamann. 2018. Change Point Detection for Ocean Eddy Analysis. In *Workshop on Visualisation in Environmental Sciences (EnvirVis)*, Karsten Rink, Dirk Zeckzer, Roxana Bujack, and Stefan Jnicke (Eds.). The Eurographics Association. https://doi.org/10.2312/envirvis.20181134

[2] J. Bennett, A. Bhagatwala, J. Chen, A. Pinar, M. Salloum, and C. Seshadhri. 2016. Trigger Detection for Adaptive Scientific Workflows Using Percentile Sampling. *SIAM Journal on Scientific Computing* 38, 5 (2016), S240–S263. https://doi.org/10.1137/15M1027942

[3] UK Mini-App Consortium. 2018. CloverLeaf3D: A 3D Lagrangian-Eulerian hydrodynamics benchmark. http://uk-mac.github.io/CloverLeaf3D/

[4] Kenneth Joy, Mark Miller, Hank Childs, E. Wes Bethel, John Clyne, George Ostrouchov, and Sean Ahern. 2007. Frameworks for visualization at the extreme scale. 78 (08 2007), 012035.

[5] Ian Karlin, Jeff Keasler, and JR Neely. 2013. *Lulesh 2.0 updates and changes*. Technical Report. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States).

[6] Adam J Kunen, Teresa S Bailey, and Peter N Brown. 2015. *KRIPKE-a massively parallel transport mini-app*. Technical Report. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States).

[7] Lawrence Livermore National Laboratory. 2018. Conduit: Simplified Data Exchange for HPC Simulations. https://software.llnl.gov/conduit/

[8] Lawrence Livermore National Laboratory. 2018. Conduit: Simplified Data Exchange for HPC Simulations - Conduit Blueprint. https://software.llnl.gov/conduit/blueprint.html

[9] 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 (ISAV'17)*. ACM, New York, NY, USA, 42–46. https://doi.org/10.1145/3144769.3144778

[10] J. Ling, W. P. Kegelmeyer, K. Aditya, H. Kolla, K. A. Reed, T. M. Shead, and W. L. Davis. 2017. Using feature importance metrics to detect events of interest in scientific computing applications. In *2017 IEEE 7th Symposium on Large Data Analysis and Visualization (LDAV)*. 55–63. https://doi.org/10.1109/LDAV.2017.8231851

[11] P. Malakar, V. Vishwanath, C. Knight, T. Munson, and M. E. Papka. 2016. Optimal Execution of Co-analysis for Large-Scale Molecular Dynamics Simulations. In *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 702–715. https://doi.org/10.1109/SC.2016.59

[12] K. Myers, E. Lawrence, M. Fugate, C. McKay Bowen, L. Ticknor, J. Woodring, J. Wendelberger, and J. Ahrens. 2014. Partitioning a Large Simulation as It Runs. *ArXiv e-prints* (Sept. 2014). arXiv:stat.ME/1409.0909

[13] Maher Salloum, Janine C. Bennett, Ali Pinar, Ankit Bhagatwala, and Jacqueline H. Chen. 2015. Enabling Adaptive Scientific Workflows Via Trigger Detection. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV2015)*. ACM, New York, NY, USA, 41–45. https://doi.org/10.1145/2828612.2828619

[14] P. A. Ullrich and C. M. Zarzycki. 2017. TempestExtremes: a framework for scale-insensitive pointwise feature tracking on unstructured grids. *Geoscientific Model Development* 10, 3 (2017), 1069–1090. https://doi.org/10.5194/gmd-10-1069-2017

[15] J. Wendelberger, D. Banesh, and J. Ahrens. 2017. Detecting Changes in Simulations. *Joint Statistical Meetings* (2017).

[16] C Wood, M Larsen, A Gomez, C Harrison, T Gamblin, and A Malony. 2017. Projecting Performance Data Over Simulation Geometry Using SOSflow and ALPINE. In *In Proceedings of the 4th International Workshop on Visual Performance Analysis (VPA '17)*.

[17] Ming Zhao, Isaac M. Held, Shian-Jiann Lin, and Gabriel A. Vecchi. 2009. Simulations of Global Hurricane Climatology, Interannual Variability, and Response to Global Warming Using a 50-km Resolution GCM. *Journal of Climate* 22, 24 (2009), 6653–6678. https://doi.org/10.1175/2009JCLI3049.1

[18] Bo Zhou and Yi-Jen Chiang. 2018. Key Time Steps Selection for Large-Scale Time-Varying Volume Datasets Using an Information-Theoretic Storyboard. *Computer Graphics Forum* (2018). https://doi.org/10.1111/cgf.13399