Benchmarking In Situ Triggers Via Reconstruction Error

Yuya Kawakami
Grinnell College
Grinnell, Iowa
kawakami@grinnell.edu

Nicole Marsaglia
University of Oregon
Eugene, Oregon
marsagli@uoregon.edu

Matthew Larsen
Lawrence Livermore
Livermore, California
larsen30@llnl.gov

Hank Childs
University of Oregon
Eugene, Oregon
hank@uoregon.edu

ABSTRACT

This work considers evaluating in situ triggers using reconstruction error. Our experiments use data from the Nyx and Cloverleaf simulation codes, and focus on two key topics. The first topic aims to increase understanding of total reconstruction error, both with respect to the impact of adding more time slices and with respect to the variation from different time slice selections. The second topic evaluates performance for two current approaches: entropy-based triggers and evenly spaced time slices. Finally, we use these study components to construct a benchmarking system that enables visualization scientists to reason about triggers.

CCS CONCEPTS

• **Human-centered computing → Scientific visualization**; **Visualization design and evaluation methods**.

KEYWORDS

in situ processing, scientific visulazation, triggers, benchmarking

ACM Reference Format:
Yuya Kawakami, Nicole Marsaglia, Matthew Larsen, and Hank Childs. 2020. Benchmarking In Situ Triggers Via Reconstruction Error. In *ISAV'20 In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV'20), November 12, 2020, Atlanta, GA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3426462.3426469

1 INTRODUCTION

Triggers are an increasingly important approach for enabling in situ processing. The idea behind triggers is to have an inspection routine that studies the current state of a simulation and decides whether or not to "fire." If a trigger does fire, then this cues additional action: visualization, analysis, or saving data to permanent storage ("dump"). If a trigger does not fire, then no further action is taken. The trigger-based paradigm provides potential opportunities for both cost savings and increased insight.

With respect to cost savings, triggers can minimize total execution time if the inspection routine executes quickly and is able to save on unneeded heavyweight visualization/analysis/dump tasks. For example, consider the scenario where a simulation runs for 1000 cycles, a trigger inspection routine takes 0.01s, and a visualization/analysis/dump routine takes 10s. In a non-trigger-based approach, visualization/analysis would occur at regular intervals, for example every 100 cycles. In this case, the overhead for visualization/analysis would be 100s (10 executions × 10s). On the other hand, if a trigger-based approach fires only seven times, then the overhead for visualization/analysis/dump would be 80s (1000 inspection routines × 0.01s + 7 executions × 10s). Further, these savings can be improved if the inspection routines are applied less often (e.g., every other cycle), if the fire rate decreases, or if the gap between inspection and visualization/analysis/dump routines widen.

With respect to increased insight, the main idea is that triggers can be used to perform visualization/analysis/dump at the most important cycles. Revisiting the previous example, it may be that the most interesting phenomenon occurs between cycles 520 and 580. A non-trigger approach that executes visualization/analysis every 100 cycles would execute at cycles 500 and 600, potentially missing key insights. A trigger-based approach, however, has the potential to capture the phenomenon. If the trigger fired seven times, then the simulation may be best conveyed by (for example) cycles 0, 250, 520, 550, 580, 750, and 999. However, it is harder to measure the increase in insight. This contrasts with the cost savings analysis, which could be placed into formulas that enabled direct comparisons. One mechanism for measuring insight is to incorporate perspectives from domain scientists. That said, while their perspectives are clearly paramount, relying on domain scientists to evaluate which time slices are most important would lead to domain-specific answers, could be somewhat subjective (in particular varying from domain scientist to domain scientist), and user studies can be quite time consuming. Instead, we feel that there is value in considering this question from a quantitative perspective and in a domain-agnostic manner. This belief is the first key premise (of two) that underlies our research.

Explicitly, our driving question is: "if a trigger can fire for K time slices, then which K should it choose?" With this work, our guiding principle to answer this question is: "the best K time slices are the ones that best reconstruct the remaining time slices." This guiding principle is the second key premise that underlies our research. From the perspective of this guiding principle, we consider how to calculate these time slices and evaluate error. We then use this method to explore reconstruction error, both from the perspective of how more time slices improve reconstruction error and from the perspective of how error varies across different selections of time slices. We conclude the paper by evaluating how a current trigger approach (entropy) compares to choices that minimize reconstruction error. In all, while we feel the results of our study are quite interesting, we feel the greatest utility from our efforts are in providing a benchmarking system that can evolve to consider more data sets, reconstruction methods, error metrics, and trigger techniques. This benchmark system, developed as part of this study, is available as open source software and is described in an appendix as part of ISAV's reproducibility initiative.

2 RELATED WORK

As several existing works survey in situ processing, and identify challenges with the approach [4, 6, 7, 16], this section focuses on research directly related to triggers and time slice selection. Specifically, the first two subsections are on triggers, and organized into two categories: domain-specific triggers and domain-agnostic triggers. These subsections serve to document the increasing prevalance of triggers for enabling in situ processing, and also — when possible — discuss how these works evaluated their trigger technology. The final subsection is on time slice selection outside of the trigger context.

2.1 Domain-Specific Triggers

There have been a number of recent works that have developed domain-specific triggers. Bennett et al. [5] developed a domain-specific trigger that detects ignition during combustion simulations. A premise of their work was that a large change in value indicated ignition, which eliminated the need for accuracy metrics. This work was continued by Salloum et al. [17] whom proposed an additional method to detect sudden heat release, and subsequently increasing the robustness of the trigger.

Liu et al. [14] developed a sea level anomaly (SLA)-based trigger that uses ocean satellite altimeter data to detect eddies. They compared their resulting eddy tracking with existing eddy tracking techniques. Sun et al. [18] built upon this work with a hybrid approach that utilizes both the physical and geometric properties of eddies to produce more accurate results over time compared to previous approaches.

Ullrich et al. [20] developed TempestExtremes, a framework for analyzing climate data sets that uses a number of climate-specific triggers, so as to support a wide-array of detection schemes for climate data. This work focused on frameworks, and did not consider evaluation of trigger efficacy. Zhao et al. [22] also uses climate-specific triggers in order to detect tropical storms within climate simulations. Similar to the work by Bennett and Salloum, the authors designed their triggers using domain knowledge, and did not make evaluating its efficacy a focal point of their study.

2.2 Domain-Agnostic Triggers

Ling et al. [13] developed a machine-learning trigger that detects and then saves local changes in the data by comparing local data to non-local data as well as previous time steps. This work established the relevance of machine learning in this space, but did not focus on evaluating the accuracy of the results.

Aditya et al. [1] proposed a method for detecting anomalies in distributed data. They evaluated their work using the "local outlier method" (LOF). A method like this could be incorporated into our benchmarking system in the future.

A number of works have found success using statistical methods as a trigger. Wendelberger et al. [21] applied change detection methods to images as a general approach to summarizing data, identifying important variables, selecting informative time steps and detecting events in the simulation. Banesh et al. [3] also used change point detection to detect events, to guide researchers to areas of interest, or to narrow a data set for further analysis. In both cases, the works proposed a method and demonstrated it on real-world data sets, but did not provide a framework for evaluating their selections. Finally, Myers et al. [15] made use of statistical methods to find time steps of global and local importance, respectively. They compare their technique to saving at regular intervals.

2.3 Time Slice Selection

Zhou and Chiang [23] considered the problem of automating storyboards for users, i.e., finding the best time slices to display to users. That said, our approaches for evaluating reconstruction error and tractable computation are similar to theirs. Dynamic time warping (DTW) [19] incorporates an information-theoretic approach to find salient time slices. Finally, recent machine learning-based approaches have also considered keyframe selection, as well as reconstructing from those time slices [9, 10]. For each of these works, our goals are different in that we are focused on in situ triggers and on establishing a benchmark suite.

3 METHOD OVERVIEW

This section described our method, and is broken into three parts. Section 3.1 formalizes a method for calculating reconstruction error. Section 3.2 then describes an algorithm for finding the time slices that minimize this reconstruction error. Finally, Section 3.3 describes the workflow that we used to calculate our results. This workflow depends on the reconstruction error from Section 3.1 and incorporates the algorithm for minimizing this error from Section 3.2.

3.1 Evaluating Reconstruction Error

While there are many ways to reconstruct temporal data and evaluate error, we consider a straightforward method with this work. In particular, although triggers are useful for visualization, analysis, and dumps, our evaluation focuses strictly on dumps. We assume that K time slices are saved, and then reconstruct values at the remaining time slices via linear interpolation. We then calculate the error as the difference between our reconstruction and the original field.

Formally, for a position x and a time t, let $F(x, t)$ be the original field from a simulation and let $F'(x, t)$ be our reconstructed field. Further, let $T_1, T_2, ... T_K$ be the times for the cycles where the trigger fired, causing these time slices to be saved to disk. For a specific time $t^\star$ and location $x^\star$, we calculate $F'(x^\star, t^\star)$ by first finding the i such that $T_i \leq t^\star$ and $t^\star \leq T_{i+1}$, i.e., the two time slices surrounding $t^\star$. We assume the trigger always fires for the first and last cycle of the simulation, so such an i always exists. Then we define F' as:

$$F'(x^\star, t^\star) = F(x, T_i) + \frac{t^\star - T_i}{T_{i+1} - T_i} \times (F(x, T_{i+1}) - F(x, T_i))$$

and we define the total error as:

$$Error = \int_t \int_x |F(x, t) - F'(x, t)|$$

In practice, the error is calculated for every mesh position and every time — if a mesh has P points and there are N time slices, then we sum the difference at $P \times N$ locations, i.e.,

$$Error = \sum_{j=1}^{N} \sum_{i=1}^{P} |F(x_i, t_j) - F'(x_i, t_j)|$$

3.2 Tractable Computation of Optimal Time Slices

We refer to the K time slices that minimize reconstruction error. as "optimal" time slices. Locating the optimal time slices is non-trivial, as the set of possible choices is quite large. In general, if there are N cycles in the simulation and the goal is to choose K time slices, then the number of choices is $\binom{N}{K}$, i.e., "N choose K," which is $\frac{N!}{K!(N-K)!}$. For $N = 200$ and $K = 10$, the number of choices is over 22 quadrillion.

Our first approach for improving computation time is to consider fewer cycles. Specifically, we consider only cycles that are multiples of 5 or 10. Revisiting the previous $\binom{200}{10}$ example, considering only multiples of 10 reduces the number of combinations to $\binom{20}{10}$, which is a much more tractable 184,756 combinations.

Our second approach for improving computation time is to incorporate dynamic programming. Dynamic programming depends on the ability to break a problem into simpler sub-problems where the optimal solution to the sub-problems can be used to form the optimal solution to the original problem. These sub-problems should then be able to decomposed again, and so on.

Optimal time slice calculation can be cast as a dynamic programming problem as follows. Let OPT(C_i, C_j, K) return the optimal solution that uses K time slices between C_i and C_j. Further, let COST(C_i, C_j) represent the reconstruction error for all time slices t, $C_i \leq t \leq C_j$, when using only C_i and C_j to reconstruct t. Then, for any C_i and C_j with $C_i < C_j$, OPT(C_i, C_j, K) can be calculated as:

$$OPT(C_i, C_j, K) = \min_{C_i \leq S \leq C_j} (OPT(C_i, S, K-1) + COST(S, C_j))$$

3.3 Workflow and Experiment Description

Our workflow operates as follows:

- Obtain data set at full temporal resolution.
- Run Ascent [11] to generate trigger information. This utilized Ascent's "replay" feature, which enabled Ascent to load data from disk and execute trigger code as if it was doing in situ analysis. In particular, this was used to calculate the time slices for trigger based on change in entropy [12].
- Run Python scripts that evaluated reconstruction error (3.1) for various time slice selections, and also calculated the optimal time slices (3.2).

For our experiments, we considered two data sets:

- Results from a Nyx [2] cosmology simulation, which produced 515 time slices on a rectilinear grid of size $69 \times 69 \times 69$. For this data set, we considered cycles that were multiples of 10.
- Results from a Cloverleaf [8] hydrodynamics simulation, which produced 200 time slices on a rectilinear grid of size $69 \times 69 \times 69$. For this data set, we considered cycles that were multiples of 5.

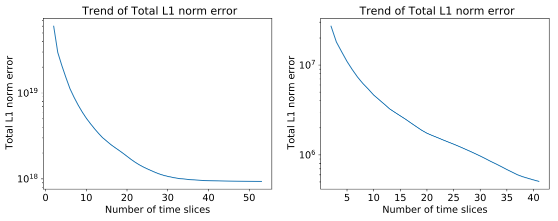


Figure 1: Reconstruction error as a function of budget for Nyx (left) and Cloverleaf (right) using optimal time slice choices. For example, when the budget allows for saving 20 time slices, we use the technique from Section 3.2 to calculate the 20 time slices that minimize reconstruction error, and the Y-value for X=20 corresponds to this error. Finally, note that the Y-axes in these two plots are very different because the error is proportional to the values in the scalar field, and the Nyx field has much higher values.

Our experiments can be calculated on a desktop computer, and take several hours to run. That said, our current implementation stores all time slices in memory (1.29GB for Nyx data), potentially limiting some computers from operating on larger data sets.

4 RESULTS

4.1 Understanding Reconstruction Error

4.1.1 How the Number of Time Slices Affects Reconstruction Error. This section explores how reconstruction error changes as the simulation code is able to dump more and more time slices. Clearly, when a simulation is able to dump more time slices, these additional time slices will reduce the total reconstruction error. That said, the magnitude of decrease is less clear. In all, we feel this analysis informs questions such as "how many time slices should I save?" and "will saving more time slices aid my analysis?"

Figure 1 plots the total reconstruction error as a function of budget (i.e., the allotted number of time slices to save) for Nyx and Cloverleaf. As the budget changes, the data reflects the error incurred when choosing the optimal time slices (via the technique described in Section 3.2). The figure shows clear diminishing returns as the budget increases for Nyx, with the error plateauing after 15 time slices. For Cloverleaf, the effect is not as severe. Of note, the actual time slices saved likely vary as the budget changes. For example, if a simulation can save two time slices, then the optimal choices may be to save cycles 115 and 260, but if it can save three time slices, then the optimal choices may be cycles 180, 220, and 240. Further, we assume the first and last cycle are always saved, so this example should be interpreted as saving *additional* time slices.

While Figure 1 shows behavior in the aggregate, it is not useful for understanding individual behavior. Figure 2 addresses this point, via four error histograms corresponding to four time slice budgets with Nyx data. As the budget increases, error becomes smaller and smaller. While this is consistent with the aggregate trend, we feel the persistence of high error regions is interesting, as is the widespread errors found in small budgets that become much less frequent in higher budgets.

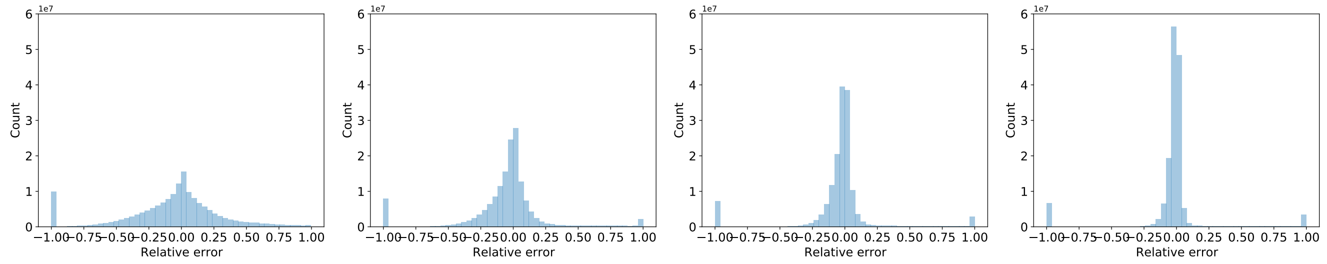


Figure 2: Four histograms of error for time slice budgets of (left-to-right) 3, 5, 7, and 10 time slices for Nyx data. Each histogram represents a range of errors, and the count for that bin is the number of cells (over all time slices and spatial location) that have error within that range. Finally, error is calculated in a relative sense, i.e., $\frac{\text{Reconstructed - Actual}}{\max |\text{Reconstructed}|, |\text{Actual}|}$. Using relative error for these histograms emphasizes significant errors with small-magnitude values and de-emphasizes insignificant errors with large-magnitude values.

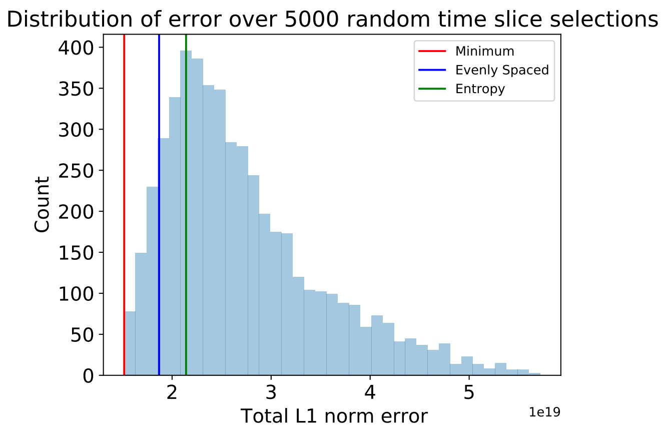


Figure 3: Histogram of errors for 5000 random selections of 5 time slices. Each bin cover a range of errors and counts how many of the 5000 random selections had that much error. Further, the error associated with the optimal choice (from Section 3.2) is indicated in red, with evenly spaced time slices in blue, and with an entropy trigger (that leads to five time slices) in green.

4.1.2 How Reconstruction Error Varies with Choice of Time Slice. This section explores the importance of time slice selection. In other words, is the "best" choice much better than other choices? Is seeking out "good" time slices worthwhile?

To provide insight into this topic, we generated 5000 random time slice selections for Nyx data. Each of these selections had exactly five time slices, i.e., a budget of size five. We then calculated total error for each of the selections. Figure 3 shows the results, plotted as a histogram. The minimum total error was $1.52e^{19}$, while the maximum total error was over $6e^{19}$, meaning that the worst choice was 4X worse than the best. That said, most random choices led to errors between $2e^{19}$ and $3e^{19}$, i.e., ranging from 1.33X to 2X error.

4.2 Evaluating a Current Trigger Approach

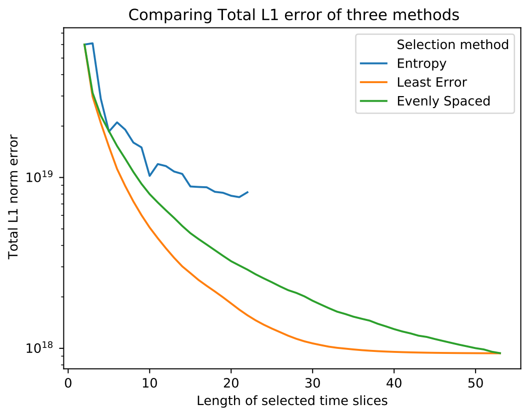
Figure 4 shows an evaluation of entropy triggers compared to evenly spaced time slices and the optimal selection from Section 3.2. Surprisingly, entropy has worse reconstruction error than either approach. While our benchmarking system does not illuminate the cause, its relatively poor behavior in this setting provides additional understanding of the trigger, and also reveals that additional research on efficacy would be useful. Of course, this finding does not mean that entropy is not an effective trigger, as it has been shown to capture certain phenomena well.

Further, we find the difference between evenly spaced and optimal to be noteworthy. For Nyx, the optimal selection is considerably better, while for Cloverleaf they are the same. In fact, evenly space outperforms "optimal" for some budgets, which reveals a limitation in our approach — our "optimal" approach considers only cycles that are multiples of five, while the evenly spaced selection accesses different cycles. Since evenly spaced is so suitable for this data set, its different choices occasionally are better.

5 CONCLUSION AND FUTURE WORK

The contribution of this work is in establishing a benchmarking system for evaluating in situ triggers. We feel it provides an important complement to the only current method of evaluating triggers, which is incorporating domain scientist feedback. In particular, this benchmarking can provide evidence as to whether one trigger scheme is better than another, which we feel fills a gap for our community. Finally, we feel the (non-trigger) results that focus on reconstruction error also inform important questions about necessary temporal resolution, and add to the value of our benchmarking system.

There are many areas of improvement for our system. Our choices in how to reconstruct the field, how to evaluate error, the data set we choose to evaluate, and how we analyze results can all be revisited and improved. In particular, many time-varying data sets have phenomena that sweep through a volume, making linear interpolation between time slices like a poor choice. Further, our current implementation assumes that all time slices can fit into memory, which could create prohibitive memory requirements for larger data sets. Therefore, our system could benefit from a



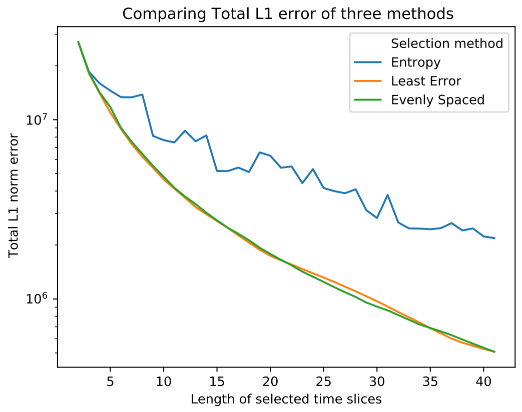


Figure 4: Total reconstruction error as a function of time slice budget for Nyx (top) and Cloverleaf (bottom). Both figures consider the optimal choices using the method from Section 3.2 (red), evenly-spaced time slices (green), and entropy triggers (blue). The entropy trigger for Nyx stops after a budget of 22 time slices due to details with its execution. In particular, entropy triggers fire when the change in entropy is greater than some threshold. For this simulation, it was possible to choose a threshold that dumped 22 time slices, but any smaller threshold caused hundreds of time slices to be saved. Finally, note that the Y-axes in these two plots are very different because the error is proportional to the values in the scalar field, and the Nyx field has much higher values.

new approach for data processing, whether it be out-of-core or parallelism.

Finally, this research relied on two key premises: (1) that a domain-agnostic, quantitative evaluation method would be useful and (2) that the time slices that minimize reconstruction error when dumping data are worthy time slices for triggers to "fire" when doing visualization and analysis. We feel our results section supports the first premise. We did not attempt to support the second premise, but we feel that this could make interesting future work, i.e., seeking domain scientist feedback that confirms or denies that the time slices that minimize reconstruction error are useful time slices for visualization and analysis.

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.

REFERENCES

[1] Konduri Aditya, Hemanth Kolla, W Philip Kegelmeyer, Timothy M Shead, Julia Ling, and Warren L Davis IV. 2019. Anomaly detection in scientific data using joint statistical moments. *J. Comput. Phys.* 387 (2019), 522–538.

[2] Ann S. Almgren, John B. Bell, Mike J. Lijewski, Zarija Lukić, and Ethan Van Andel. 2013. Nyx: A MASSIVELY PARALLEL AMR CODE FOR COMPUTATIONAL COSMOLOGY. *The Astrophysical Journal* 765, 1 (feb 2013), 39. https://doi.org/10.1088/0004-637x/765/1/39

[3] 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 Jänicke (Eds.). The Eurographics Association. https://doi.org/10.2312/envirvis.20181134

[4] Andrew C Bauer, Hasan Abbasi, James Ahrens, Hank Childs, Berk Geveci, Scott Klasky, Kenneth Moreland, Patrick O'Leary, Venkatram Vishwanath, Brad Whitlock, and E. Wes Bethel. 2016. In Situ Methods, Infrastructures, and Applications on High Performance Computing Platforms. *Computer Graphics Forum (CGF)* 35, 3 (June 2016), 577–597.

[5] 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

[6] Hank Childs et al. 2020. A Terminology for In Situ Visualization and Analysis Systems. *International Journal of High Performance Computing Applications (IJHPCA)* 34, 6 (Nov. 2020), 676–691.

[7] Hank Childs, Janine Bennett, Christoph Garth, and Bernd Hentschel. 2019. In Situ Visualization for Computational Science. *IEEE Computer Graphics and Applications (CG&A)* 39, 6 (Nov./Dec. 2019), 76–85.

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

[9] Jun Han and Chaoli Wang. 2019. TSR-TVD: Temporal super-resolution for time-varying data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2019), 205–215.

[10] Wenbin He, Junpeng Wang, Hanqi Guo, Ko-Chih Wang, Han-Wei Shen, Mukund Raj, Youssef SG Nashed, and Tom Peterka. 2019. InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 23–33.

[11] 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 Workshop of In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV).* Denver, CO, 42–46.

[12] Matthew Larsen, Amy Woods, Nicole Marsaglia, Ayan Biswas, Soumya Dutta, Cyrus Harrison, and Hank Childs. 2018. A Flexible System for In Situ Triggers. In *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV).* Dallas, TX, 1–6.

[13] 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

[14] Yingjie Liu, Ge Chen, Miao Sun, Shuai Liu, and Fenglin Tian. 2016. A parallel SLA-based algorithm for global mesoscale eddy identification. *Journal of Atmospheric and Oceanic Technology* 33, 12 (2016), 2743–2754.

[15] Kary Myers, Earl Lawrence, Michael Fugate, Claire McKay Bowen, Lawrence Ticknor, Jon Woodring, Joanne Wendelberger, and Jim Ahrens. 2016. Partitioning a large simulation as it runs. *Technometrics* 58, 3 (2016), 329–340.

[16] Tom Peterka, Deborah Bard, Janine C Bennett, E Wes Bethel, Ron A Oldfield, Line Pouchard, Christine Sweeney, and Matthew Wolf. 2020. Priority research directions for in situ data management: Enabling scientific discovery from diverse data sources. *The International Journal of High Performance Computing Applications* 4, 3 (March 2020), 409–427.

[17] 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* (Austin, TX, USA) *(ISAV2015).* ACM, New York, NY, USA, 41–45. https://doi.org/10.1145/2828612.2828619

[18] Miao Sun, Fenglin Tian, Yingjie Liu, and Ge Chen. 2017. An Improved Automatic Algorithm for Global Eddy Tracking Using Satellite Altimeter Data. *Remote Sensing* 9, 3 (Feb 2017), 206. https://doi.org/10.3390/rs9030206

[19] Xin Tong, Teng-Yok Lee, and Han-Wei Shen. 2012. Salient time steps selection from large scale time-varying data sets with dynamic time warping. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. 49–56.
[20] 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
[21] J. Wendelberger, D. Banesh, and J. Ahrens. 2017. Detecting Changes in Simulations. *Joint Statistical Meetings* (2017).
[22] 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
[23] 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

A ARTIFACTS DESCRIPTION

We divide our description into two parts:

- How to reproduce our results (Section A.1)
- How to extend our system (Section A.2)

A.1 Reproducing Our Results

The following files are needed to reproduce the Cloverleaf portion of our experiments:

- Cloverleaf data: http://cdux.cs.uoregon.edu/cloverleaf.tar
- Python scripts: http://cdux.cs.uoregon.edu/trigger_benchmark.tar

One of the Python scripts in `trigger_benchmark.tar` is `get_best_errors.py`. It calculates the optimal time slices with respect to reconstruction error. Depending on which data set is used, lines 16 and 18 should be adjusted.

Changing the error function is accomplished by changing the variables after the `if __name__ == __main__:` statement. The set of time slices that the optimal approaches considers can be changed in the same location. (For example, we only consider time slices multiples of 10 for the Nyx data, but obviously, we can change this to consider multiples of 5 instead.) As it stands, this script will output the least error selection of time slices and the respective costs to a CSV file. The script `get_entropy_trigger_time_slices.py` can be used to retrieve results of entropy-based triggers. Ascent's "replay" feature outputs a `ascent_sessions.yaml` containing entropy values at each time slice. We use this output when invoking the script, in order to return the results for entropy-based triggers. This script takes in one command line argument: the threshold value to use. As output, it prints the time slice selection based on that threshold.

A note on the readability of code: when the code was first written, we considered the problem from the perspective of graphs. As a result, many variables are named in the language of graphs.

A.2 Extending Our System

Given our goal of establishing a benchmark system, we believe it is important that the system be extensible in four ways: data set, trigger type, interpolation scheme, and error evaluation. In its current form, our benchmarking system makes it fairly easy to evaluate new data sets and error evaluation schemes. In particular, we considered two data sets and three error evaluation schemes (L1-norm, L2-norm, and sum of squared errors). Evaluating new interpolation schemes is a bigger effort, and we envision extending our system to abstract out the interpolation scheme in the future. Finally, triggers were arranged manually, i.e., selecting the time slices where a trigger would fire and adding that to the error evaluation. We would like for the triggers to be included automatically (i.e., no human intervention needed), and hope to add this soon.

How to change data sets:
Currently the `parse_hdf5` function takes care of handling the data, storing the data in a dictionary where the keys are the time slices and the value is the data at the given time slice. In the future, we plan to adapt this function to accommodate different data sets to enable support for more data sets.

How to change error evaluation scheme:
Currently, evaluation schemes are saved as a function. This function is passed to the `make_adjacency_matrix` function. Changing or adding evaluation schemes requires adding the new scheme as a function and passing this function to the `make_adjacency_matrix` function.

How to incorporate decisions from a trigger:
Entropy-based trigger decisions are found using Ascent's replay feature to retrieve the entropy value and by then parsing the `ascent_sessions.yaml` as described before. We hope to include this feature in our system by automating this process in the future.