Trigger Happy: Assessing the Viability of Trigger-Based In Situ Analysis

Matthew Larsen*
Lawrence Livermore National Laboratory

Cyrus Harrison
Lawrence Livermore National Laboratory

Terece L. Turton
Los Alamos National Laboratory

Sudhanshu Sane
University of Utah

Stephanie Brink
Lawrence Livermore National Laboratory

Hank Childs
University of Oregon

ABSTRACT

Triggers are an emerging strategy for optimizing execution time for in situ analysis. However, their performance characteristics are complex, making it difficult to decide if a particular trigger-based approach is viable. With this study, we propose a cost model for trigger-based in situ analysis that can assess viability, and we also validate the model's efficacy. Then, once the cost model is established, we apply the model to inform the space of viable approaches, considering variation in simulation code, trigger techniques, and analyses, as well as trigger inspection and fire rates. Real-world values are needed both to validate the model and to use the model to inform the space of viable approaches. We obtain these values by surveying science application teams and by performing runs as large as 2,040 GPUs and 32 billion cells.

1 INTRODUCTION

There are two processing paradigms for analyzing the data from computational simulations: in situ (processing data as it is generated) and post hoc (processing data after it is generated). Until recently, post hoc processing has been the dominant processing paradigm for scientific visualization and analysis on supercomputers. With a typical post hoc model, a computational simulation does temporal subsampling, i.e., saves a subset of its cycles (time steps) to disk, and then a domain scientist explores that data afterward with an analysis tool. However, over the last half decade, supercomputers have seen a significant shift towards in situ processing [11, 13, 31]. The primary driver behind changing processing paradigms is to address I/O constraints on leading-edge supercomputers. On these machines, the ability to generate data is increasing much faster than the ability to store data to persistent storage — compute power has gone up by two orders of magnitude over the last decade, while I/O performance typically has only increased by one order of magnitude. As a result of these trends, post hoc processing is becoming increasingly problematic, as (1) the data takes too long for post hoc analysis programs to load and (2) simulation codes cannot store sufficient temporal resolution.

Compared to post hoc processing, in situ processing offers both advantages and disadvantages. One major advantage is increased access to temporal data, as in situ routines can potentially be invoked during any cycle. That said, a major disadvantage is that in situ routines are often limited in their time budgets. In particular, if the simulation and in situ routines share resources, then in situ routines that execute for too long will prevent the simulation from advancing quickly enough. At first glance, this disadvantage (limited execution time) would appear to negate the advantage (increased temporal access) — increased temporal data is not useful if there is not sufficient time budget to visualize or analyze this data. However, an emerging strategy, referred to as "triggers," can help alleviate this tension.

Triggers are lightweight analysis tasks that inspect simulation data and return *true* when interesting phenomena occur (*i.e.,* the trigger "fires") and *false* otherwise (*i.e.,* the trigger does not "fire"). Triggers augment the workflow for in situ processing. At the end of a cycle, a computational simulation starts by calling a trigger. If the trigger indicates analysis would be useful, then the computational simulation would schedule heavyweight analyses.

A trigger-based approach has the potential to provide more insight at reduced cost. Consider an example of a simulation running for N cycles. A non-trigger approach may execute heavyweight in situ routines every K cycles, for a total of $\frac{N}{K}$ executions. A trigger-based approach, however, may be able to perform fewer than $\frac{N}{K}$ heavyweight executions, but perform them during cycles that will lead to more insight overall. Of course, the trigger-based approach will only save on execution time if the lightweight executions are sufficiently fast. In all, triggers are about saving resources — without triggers, the same heavyweight analyses could still occur, but it would require running them at a high frequency and incurring a (potentially much) higher cost.

Despite the potential utility of triggers, domain scientists and trigger practitioners currently have no framework for reasoning about whether a given trigger-based approach is viable, *i.e.,* whether a given trigger inspection task applied at a given inspection rate with a given fire rate will exceed their time budget. This gap is the motivating challenge of this paper. In response, we consider two fundamental research questions: **RQ1** and **RQ2**.

- **RQ1** is *"can we predict whether a given trigger approach will exceed a given time budget?"* To address this question, we propose a solution based on cost modeling, expanding previous in situ cost models to include triggers. One term in the cost model is data dependent, and we pay special attention to this term to explore its effects.
- **RQ2** is *"can we inform the space of viable approaches for trigger tasks, inspection rates, fire rates, and analysis tasks?"* To address this question, we analyze a variety of trigger scenarios using our cost model. This research question is in particular helpful for visualization and analysis experts designing new triggers, as it can inform limits for execution time as well as tradeoffs.

Overall, the contribution of this paper is improved understanding about how to approach the usage of triggers with respect to staying within a time bound. That said, this topic is quite large. In order to make progress, our work limits scope via three key assumptions: (1) that the in situ model is "tightly coupled," i.e., that visualization/analysis routines execute on the simulation's compute resources, (2) that an analyst will have intuition about the consequences of running visualization/analysis too infrequently, and that they will use this intuition in combination with our approach to reason about tradeoffs between execution time and accuracy, and (3) that estimates for costs for various operations are useful surrogates for real runtimes.

*larsen30@llnl.gov

Each of these three key assumptions is revisited in our future work section.

2 Related Work

In situ approaches have rapidly gained traction since Bauer et al.'s in situ survey in 2016 [9]. In situ functionality can be implemented via APIs such as ParaView's Catalyst [2, 6], VisIt's LibSim [12, 33], or Ascent [21]. In transit APIs include ADIOS [24] and SENSEI [8]. With the availability and acceptability of robust approaches to perform in situ processing, in addition to development and deployment, research has been extended to include understanding in situ usage via a cost performance analysis.

2.1 Cost Models

A cost analysis requires a performance model with appropriate parameters to describe the in situ analysis and visualization workflow. In order to make a simulation workflow adaptive and able to meet the I/O bandwidth challenges, we need to consider the various costs involved in a simulation run. These may include any inline analysis tasks (*e.g.,* calculating a basic set of statistics for each time step), costs associated with trigger mechanisms, and any heavyweight data analysis and visualization tasks (DAV) or I/O tasks based on the triggers.

The most related to our work is the seminal work by Malakar et al. [26]. This paper identified the basic parameters needed to create an in situ cost analysis model: the types and frequencies of analyses, whether to perform tightly (in situ) or loosely coupled (in transit), and output frequencies. Their work included an evaluation of compute and memory constraints, assuming multiple analyses with associated weights based on importance. The goal was to optimize the scheduling of the in situ analyses within user constraints. A follow-up paper by Malakar et al. [25] extended these ideas to co-analysis (in transit) studies, for both local and remote off-loading of the data. Although scaling to exascale architectures with multiple cores per node and greater concurrency has rapidly changed the underlying problem, these two works provide a solid foundation for in situ cost analyses. That said, they do not consider trigger-based workflows.

Contemporaneous work by Ayachit et al. [7, 8] surveyed and presented in situ and in transit approaches, and studied performance characteristics to understand impacts on simulation codes. Kress et al. [19, 20] looked at execution time and cost, directly comparing in situ and in transit approaches and developing cost models for each method. Aupy et al. [5] also considered this topic, with an emphasis on memory constraints and their impacts on scheduling algorithms. Finally, work on developing performance cost models for in situ power and energy usage can be found in studies by Adhinarayanan et al. [1] and Haldeman et al. [16].

In this paper, we explore a cost model for the use of triggers during in situ processing. Although prior work has explored cost models for various aspects of in situ analysis, none of these efforts explicitly addressed the high frequency use of triggers in situ. We believe a cost model for the use of triggers in situ is novel, and particularly valuable given the increased interest for future in situ DAV. Specifically, our contribution enables scientists to make informed decisions regarding inspection frequencies, feasibility of types of triggers, and the corresponding analyses.

2.2 Triggers

Efforts by Larsen et al. [22] demonstrated trigger technology in Ascent [21], a flyweight, exascale-capable infrastructure. Recent work in trigger design and development spans from domain and code-specific approaches to generic approaches that could be adapted across domains. Bennett et al. [10] introduced the general concept of an *indicator* that can be frequently calculated and an associated *trigger* as a Boolean function to test the indicator properties against a predefined condition, thereby triggering DAV or I/O tasks. Bennett et al. [10] and Salloum et al. [32] apply this methodology to detect precursors to heat release in turbulent combustion simulations. They demonstrate an overhead of approximately 1% of the simulation time when computing the trigger every 10^{th} cycle.

Multiple works have focused on using triggers to save important time steps. Nouanesengsy et al. [30] proposed an algorithm that constructs a prioritization tree to enable saving highest priority information as a simulation runs. The algorithm measured temporal entropy of every simulation time step to form a collection of high entropy time steps that together capture phases of maximum simulation change. With a similar objective to reduce I/O costs, Myers et al. [29] computed a piecewise linear model to capture the simulation. Thus, triggers have been employed at varying frequencies, *i.e.,* every cycle and periodically, and for varying purposes, *i.e.,* DAV or I/O tasks at important time steps.

3 Trigger Cost Model and An Approach for Rightsizing In Situ Analysis

In this section, we introduce our cost model for trigger-based in situ analysis (3.1). To demonstrate how the cost model can be used, we then introduce an approach for "rightsizing" in situ analysis (3.2).

3.1 Cost Model

Our model is in the context of "tightly-coupled" *in situ*, *i.e.,* the analysis routines run on the same compute resources as the simulation and can directly access the simulation's memory (and thus its data). In this model, the simulation and analysis alternate — the simulation advances, then the analysis occurs, then the simulation again, etc. We also assume a model where the transition from simulation to analysis (and vice-versa) happens at the same time. That is, every compute node will stop doing simulation and start doing analysis at the same time. If one compute node finishes its task (simulation or analysis) before the other compute nodes, then the node sits idle, and its cost is considered part of the current activity. This assumption fits common practices today.

Consider the following terms:

- C_i: The time to complete cycle i of a simulation.
- P_i: The time for a simulation to publish data, *i.e.,* format or relocate data so it can be consumed by analysis routines. If no data is published at cycle i, then P_i is 0.
- T_i: The time to perform a trigger task at cycle i. If no trigger task is performed at cycle i, then T_i is 0.
- A_i: The time to perform an analysis task at cycle i. If no analysis task is performed at cycle i, then A_i is 0.

Then the total time, T_{total}, for N cycles is:

$$T_{total} = \sum_{1}^{N} C_i + \sum_{1}^{N} P_i + \sum_{1}^{N} T_i + \sum_{1}^{N} A_i \quad (1)$$

Equation 1 is not practical to use as a cost model, since it requires knowing the values C_i, P_i, T_i, and A_i for all cycles. In some cases, these terms could be replaced by averages. For example, many simulations perform the same number of computations each cycle, and so the only variance in execution time is due to hardware factors (*e.g.,* network contention, memory caching). In these cases, $C_i \approx C_j$ for all i and j, and so an average is appropriate. In other cases, the variance is much higher. For example, some analysis tasks become more expensive as a simulation advances and its state becomes more complex. In these cases, an average may not be appropriate.

To deal with variation, we introduce four terms: C_μ, P_μ, T_μ, and A_μ. The symbol "μ" is chosen because of its association with the mean behavior in statistics. In our model, μ can be chosen as the average value, *i.e.,* $C_\mu = \frac{\sum_1^N C_i}{N}$, or as the maximum value, *i.e.,*

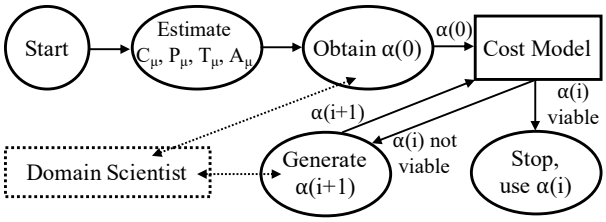

Figure 1: A flow chart describing our iterative process for rightsizing in situ analysis tasks before a simulation begins. The goal of this process is to make a configuration, $\alpha(i)$, for carrying out analysis tasks within time constraints.

$C_\mu = \max_N(C_i)$. The choice of average represents a typical behavior, while the choice of maximum represents a worst case behavior. Further, additional choices of μ are possible, such as a distribution of outcomes, although this direction is not explored in this paper.

Using these new terms, we define our cost model as:

$$T_{total} = N \times (C_\mu + R_I \times (P_\mu + T_\mu) + R_F \times A_\mu) \quad (2)$$

where R_I is the rate at which triggers inspect (*i.e.,* $R_I = 0.1$ means trigger inspections are performed every ten cycles), and R_F is the rate at which triggers fire and call analysis tasks. Note that the value of R_I provides an upper bound for R_F, *i.e.,* fire rate cannot exceed the inspection rate. Further, inspecting data requires publishing data, so the R_I term is used to cover both inspection and publishing (as opposed to introducing an R_P term).

Data analysis and visualization (DAV) overhead, *i.e.,* how much overhead the simulation incurs from doing *in situ* analysis, is a related concept. We define this as a percentage relative to the simulation time itself:

$$DAV_{Overhead} = \frac{\sum_1^N P_i + \sum_1^N T_i + \sum_1^N A_i}{\sum_1^N C_i} \times 100\% \quad (3)$$

With respect to our model, $DAV_{overhead}$ is:

$$DAV_{Overhead} = \frac{R_I * (P_\mu + T_\mu) + R_F * A_\mu}{C_\mu} \times 100\% \quad (4)$$

While we include publishing costs in our model and also consider them in the next section, we treat P_i as 0 for the remainder of the paper. In practice, the publishing cost can approach zero if the analysis routines can adapt to the memory layout of the simulation and if there is no movement of the data. These conditions are commonplace for custom analysis routines and for newer in situ libraries. However, if data has to be copied to new forms, then P_i can significantly affect tradeoffs.

3.2 An Approach for Rightsizing Trigger-Based In Situ Analysis

This subsection describes our approach for deciding how to perform trigger-based in situ analysis. This approach is one of many possible instantiations; this instantiation is useful for motivating the value of a cost model, but we recognize that other variations may also be useful.

The focus for our approach is "rightsizing," *i.e.,* making choices to efficiently meet overall campaign goals for a simulation. The workflow for our approach is displayed in Figure 1. We intend for this workflow to take place before the simulation begins, and the outcome of the approach is a *configuration* of decisions for how to carry out the simulation and analysis. We assume decisions specific to the simulation are fixed: what computational problem is being simulated, physics approaches, mesh resolution, etc. These decisions in turn fix the time per cycle (each C_i and thus C_μ, however it is determined) and the number of cycles (N), making them terms we cannot affect. We also assume the publishing cost (P_i or P_μ) is fixed, although possibly this process could be optimized if its cost was significant. For the remainder of the choices, we assume we can make changes. That is, we assume that the trigger inspection and fire rates (R_I and R_F) can be modified, as can the lightweight trigger task and the heavyweight analysis task(s) being performed, which would affect T_μ and A_μ, respectively. The final part of a configuration is the desired time to carry out the simulation campaign, sometimes referred to as the "makespan," which we denote as $T_{desired}$. In all, a configuration is made up of five choices: inspection rate, fire rate, trigger task, analysis task, and desired total time.

Our suggested approach is iterative. It begins with assessing values for the terms of the model — C_μ, P_μ, T_μ, A_μ. This process is the subject of Section 4.1, and is discussed further there. The next step is for a domain scientist to specify their desired configuration, which we denote as $\alpha(0)$. In Figure 1, this step is labeled "Obtain $\alpha(0)$." With C_μ and N already fixed, the choices behind $\alpha(0)$ inform the rest of the values from Equation 2: R_I, T_μ, R_F, and A_μ. The next step in our approach is to consult the cost model to calculate the predicted total time, T_{total}. We then compare the predicted time to the desired time, $T_{desired}$. If T_{total} is larger than $T_{desired}$, then the configuration is not viable, *i.e.,* it cannot complete in the desired amount of time. In this case, our approach would again consult with the domain scientist to make a new configuration, $\alpha(1)$, with the aim to reduce the T_{total} value to the point that it is less than $T_{desired}$ (or for $T_{desired}$ to be raised so that it is bigger than T_{total}). In Figure 1, this step is labeled "Generate $\alpha(i+1)$." Our iterative approach would then return to the cost model with $\alpha(1)$, again checking for viability. Assuming $\alpha(1)$ is not viable, then we would repeat the "Generate $\alpha(i+1)$" step, and then check the cost model again. This process would continue until it produces a configuration $\alpha(i)$ which is viable, meaning that its T_{total} predicted by our cost model is less than the $T_{desired}$. Once this occurs, we would stop, and use the decisions for that configuration for inspection rate, fire rate, trigger task, and analysis task. Another outcome is that no viable configuration is obtainable, in which case the domain scientist would restart the process with different trigger and analysis tasks. Finally, one of the assumptions of our work is that the domain scientist would have intuition about how frequently analysis would need to occur, and we assume that the domain scientist would incorporate this knowledge during "Generate $\alpha(i+1)$" — the value of our approach is enabling the domain scientist to reason about execution time when making tradeoffs.

A critical consideration for this workflow is the accuracy of the cost models. There are two ways that the model can be inaccurate. The first potential inaccuracy comes from the choice of T_μ or A_μ (and to a lesser extent C_μ and P_μ). For example, if the analysis task is modeled as the average cost (*i.e.,* $A_\mu = A_{ave}$) and if the actual execution takes more than average, then the workflow may use more cycles than allotted. The second potential inaccuracy is in estimating fire rate (R_F), which can be unpredictable (see Section 4.2.2 for more detail). For either type of inaccuracy, the mitigation approaches would be to expand the model for different μ functions (*i.e.,* go from T_{ave} to T_{max}) or to "pad" the desired time such that underestimates in cost will still fit within overall campaign goals. Further, we think that a framework for reasoning about cost is useful, even if it leads to estimates that are off by large amounts (for example 30%).

4 Results

Our results are organized into three parts:

- Section 4.1 establishes a corpus of real-world values for cycle time (C_μ), trigger time (T_μ), DAV time (A_μ), and acceptable overhead ($DAV_{Overhead}$), *i.e.,* the terms in the cost model relating to the simulation or DAV routines. These values are then used in the remaining two parts.
- Section 4.2 validates the cost model through two steps, first

considering artificial triggers that enable validation of the entire model and then exploring the effects of fire rate in real-world scenarios (since fire rate is data dependent). This validation, when combined with the cost model introduced in Section 3.1, answers **RQ1**.
- Section 4.3 applies the cost model to inform the space of viable trigger approaches. This analysis answers **RQ2**.

Most of the results in this section choose the μ function for the cost model to be the average, which we denote C_{ave}, T_{ave}, and A_{ave}. That said, some of the validation tests in Section 4.2 also consider maximums for μ. Finally, all evaluations of the cost model treat P_i as 0, for reasons discussed in Section 3.1.

4.1 Data Corpus

There is a broad spectrum of possible values for our model, and to our knowledge, there are no attempts to explore this space holistically. To define a set of reasonable values for this space, we use a two-pronged approach:

- **Survey of Science Application Teams**: This prong allows us to establish a set of constraints for our model parameters by directly asking simulation code teams what maximum total analysis overhead they are willing to accept, average cycle times, and at what rate applications currently perform analysis. It informs cycle time (C_μ) and acceptable overhead ($DAV_{Overhead}$).

- **Experiments**: This prong directly measures analysis cost and average cycle times with three simulation codes. It informs trigger time (T_μ) and data analysis and visualization time (A_μ).

4.1.1 Survey of Science Application Teams

We reached out to computational scientists across a range of simulation codes to survey their in situ usage. The goal was to establish sets of constraints for model parameters and total analysis cost that simulations are willing to pay. While we asked a wide variety of questions including target architectures, mesh types, and current analysis approaches, Table 1 summarizes the responses most relevant to this paper. Overall, total analysis overhead constraints ranged from 5%-20% of total runtime. Average cycle time ranged from 0.05 seconds to 180 seconds, and analysis frequency ranged from every cycle to every 10,000 cycles. Every application we surveyed is currently using some form of in situ analysis, and most applications want additional analysis capabilities.

Table 1: Table mapping simulation survey results to model parameters. Average simulation cycle times (C_{ave}) are represented as ranges (in seconds), since cycle times can vary depending on problem size and dimensionality (*e.g.,* 2D versus 3D). Analysis Frequency is the range of average analysis frequencies that application teams reported. DAV_{max} is the maximum overhead applications are willing to allow.

App	C_{ave}	Analysis Frequency	DAV_{max}
MFIX	(0.1, 10)	0.1% to 1%	20%
Castro	10	NA	5%
WarpX	1.0	1%	20%
Nyx	(60,180)	1% to 100%	20%
Truchas PBF	0.1	0.02%	20%
MARBL	(0.05,60)	0.01% to 1%	15%
GTS	(5,15)	2% to 50%	5%

4.1.2 Experiments

To gather parameters for our model, we ran experiments that evaluated three simulations in a variety of configurations, including many-core CPUs and GPUs.

In situ software: We used the Ascent in situ infrastructure [21]. Ascent is a lightweight library with existing integrations with simulation codes, and has been demonstrated running on 16,384 GPUs. For visualization, Ascent leverages VTK-m [28], a visualization library capable of running on many-core CPUs and GPUs. Ascent is tightly-coupled with the simulations, sharing the same resources and memory space. At some frequency, the simulation turns over control of all resources to Ascent. Once Ascent has completed the analysis operations, Ascent returns control of the resources back to the simulation.

Table 2: The costs of all operations for these experiments. The first row is the average cycle time (C_{ave}) and is in seconds. The remaining rows all correspond to analysis algorithms and are relative, i.e., $A_{ave}/C_{ave} * 100\%$. A value of 100% means that the analysis algorithm and the average cycle time are the same, resulting in a 50% overhead if the analysis algorithm is run every cycle. Finally, the time reported is for the operation to complete in parallel, i.e., if there are parallel inefficiencies with some operations, then they are captured in these timings.

Operation	MARBL	Clover	SW4
C_{ave}	18.31s	2.65s	0.20s
Histogram	0.05%	15%	80%
Statistics	0.16%	17%	105%
RayTracer	1.37%	15%	35%
Slice	0.38%	42%	180%
Contours	1.53%	92%	105%
IsoVolume	1.26%	235%	450%
Cinema	13%	185%	800%
IO_{SUB}	2.40%	501%	9435%
IO_{FULL}	4.97%	2943%	12290%

Trigger algorithms: We selected two current methods:

- **Histogram**: Calculating a distributed-memory histogram with 256 bins.
- **Statistics**: Calculating the 4^{th}-order moments including average, variance, skewness and kurtosis.

In each case, the method analyzes the current time step, and then the trigger fires (or not) based on the result.

DAV algorithms: We chose a variety of DAV visualization tasks that capture a spectrum of algorithmic and parallel communication complexity:

- **Contours**: Calculating isosurfaces of 10 evenly spaced isovalues.
- **RayTracer**: Rendering images of the output of the Contours results.
- **Slice**: Calculating three axis-aligned slice planes intersecting with the center of the data set.
- **IsoVolume**: Calculating a volumetric region of a scalar field between a minimum and maximum value.
- **Cinema**: Creating a 64 image Cinema [3] database of the Contours results.
- **IO_{SUB}**: Saving the mesh and a single field into HDF5 files (1 file per domain).
- **IO_{FULL}**: Saving the mesh and all published fields into HDF5 files (1 file per domain).

Simulations: We chose three diverse simulations with respect to in situ performance:

- **MARBL**: A high-order finite element Lagrangian hydrodynamics code [4].

Table 3: Table showing notional cases with our cost model to inform overhead with triggers for the MARBL simulation code. T_{ave} and A_{ave} are specified relative to C_{ave} (18.31s for MARBL), *i.e.,* a T_{ave} value of 0.01 refers to $0.01 \times C_{ave}$. Cases #1 through #6 show the cost of analysis with triggers, and with the assumption that the inspection task only triggers analysis 10% of the time ($R_F = 0.1$). Case #'s 7 through 10 (below the double line) have R_I of zero, meaning that no trigger tasks are called. In these cases, the time to perform the trigger task is moot.

Case	R_I	T_{ave}	R_F	Operation	A_{ave}	Overhead
#1	1.0	0.01	0.1	Cinema	0.89	9.9%
#2	1.0	0.01	0.1	IO_{FULL}	0.05	1.5%
#3	1.0	0.01	0.1	IsoVolume + RayTracer	0.025	1.25%
#4	1.0	0.1	0.1	Cinema	0.89	18.9%
#5	1.0	0.1	0.1	IO_{FULL}	0.05	10.5%
#6	1.0	0.1	0.1	IsoVolume + RayTracer	0.025	10.25%
#7	0.0	–	0.1	Cinema	0.89	8.9%
#8	0.0	–	0.1	IO_{FULL}	0.05	0.5%
#9	0.0	–	1.0	Cinema	0.89	89%
#10	0.0	–	1.0	IO_{FULL}	0.05	5%

- **Cloverleaf**: A hydrodynamics proxy-application [27] that solves the compressible Euler equations.
- **SW4**: A three-dimensional seismic modeling [17] code that uses a combination of rectilinear and curvilinear grids.

The experiments in this section were run on one of three machines: Quartz, Summit, or Topaz. Quartz and Topaz both have Intel Xeon E5-2695 architectures. Summit uses both NVIDIA V100 GPUs (6 per node) and IBM POWER9 CPUs. The configuration for each simulation was:

Simulation Code	MARBL	Cloverleaf	SW4
1 MPI Task Per	Core	Node	GPU
Total MPI Ranks	2304	256	2040
Total Nodes	64	256	340
# of Elements	22.M	14.9B	32.7B
Mesh Type	Unstruct.	Rectilinear	Rect.+Curv.
Machine	Topaz	Quartz	Summit

Each configuration evaluated all analysis algorithms every 100 cycles, and recorded the times of all algorithms and simulation cycles. In all cases, Ascent ran with the same configuration, *i.e.,* if the simulation executed on the GPU then so did the analysis.

Table 2 shows the results of our experiments. The difference in cycle time between the three simulations is nearly an order of magnitude, with MARBL having the longest cycle time and SW4 having the shortest. As a result, the visualization and I/O operations vary in burden. The most expensive operation, IO_{FULL} takes less than 5% of a cycle for MARBL (0.91s total) while it is more than 100X the cycle time of SW4 (24.5s total). This is because SW4 is producing more data per cycle despite its faster cycle time. Further, this example illustrates a relationship seen throughout the table, namely that the cost of the visualization, analysis, and I/O is more correlated with the number of elements produced than the cycle time. Finally, each simulation was run with a fixed configuration, and the timings of the operations could change as the configuration changes. For example, if the number of MPI ranks increases, then parallel inefficiencies could increase for some operations.

4.1.3 Notional Usage of Our Model with Values from the Data Corpus

Table 3 shows notional cases using the simplified model (*i.e.,* Equation 4) for the MARBL simulation. The table uses information gathered from domain scientists in Section 4.1.1 and results from the experiments in Section 4.1.2. The table considers three DAV tasks: generating a Cinema database, IO_{FULL}, and the combined IsoVolume plus RayTracer task. The costs for these tasks (A_{ave}), which can also be found in Table 2, are 89% (Cinema), 5% (IO_{FULL}), and 2.5% (IsoVolume and RayTracer combined) of the time of one simulation cycle (C_{ave}). Further, Table 3 considers two scenarios for the cost of running a trigger (T_{ave}): 1% and 10% of C_{ave}.

We consider Case #1 in Table 3 to be a typical example, as the trigger inspection is called every cycle ($R_I = 1$) and the trigger test is very lightweight (1% of C_{ave}). It is also reasonable for MARBL, given the costs of in situ Statistics and Histogram operations. Using Equation 2, the total time for Case #1 would be:

$$
\begin{aligned}
T_{total} = \\
& N \times (C_\mu + R_I \times (P_\mu + T_\mu) + R_F \times A_\mu) \\
& N \times (C_{ave} + 1.0 \times (0 + 0.01 \times C_{ave}) + 0.1 \times 0.89 \times C_{ave}) \\
& N \times (C_{ave} + 0.01 \times C_{ave} + 0.089 \times C_{ave}) \\
& N \times C_{ave} \times (1 + 0.01 + 0.089) \\
& N \times C_{ave} \times (1.099)
\end{aligned}
\tag{5}
$$

This aligns with the simplified cost model (Equation 4), which shows the cost to be 9.9% overhead for Cinema analysis.

From our simulation survey, we know that the DAV_{max} for MARBL is 15% (Table 1). Eight cases of our notional usage of the model for MARBL simulation are within this limit. Case #'s 7 through 10 reflect a static model for analysis tasks, *i.e.,* scheduling analysis tasks at regular intervals and not using triggers. While both the adaptive and static models can incur unacceptable overhead in some cases (case #4 has 18.9% overhead and case #9 has 89% overhead), the key point of the adaptive model is to choose the cycles where the analysis will be the most meaningful. Finally, cases #1, #2, #3, #5, and #6 illustrate ideal trigger setups — the trigger task is called frequently, but leads to acceptable overhead, and the analysis task is called infrequently enough to make the overall overhead be acceptable.

4.2 Validation

This section presents results for validating the cost model. This validation is complicated by the data-dependent nature of the fire rate (R_F). To see that fire rate is data dependent, consider the example of a simulation of a steel rod impacting a wall. The area of interest for analysis is the period of time when rod impacts the wall, and the impact has detectable characteristics, although we might not know exactly when that occurs. One possible way to detect the impact event is to monitor the entropy of density. At first, the two materials have constant density, but at the time of impact, the materials begin to deform and the entropy value increases, and when the impact event is over, the entropy values converges to a new state.

In response to the data-dependent nature of fire rate, we approach validation in two steps, first considering a scenario where the fire rate can be set through artificial means (Section 4.2.1) and then considering real-world scenarios where the fire rate varies based on simulation properties (Section 4.2.2). We feel this combined approach allows us to validate the cost model, and thus answer research question **RQ1**.

Table 4: Overhead error ranges for all validation experiments. Positive values mean that the prediction underestimated simulation overhead and negative values mean that the prediction overestimated simulation overhead.

	Test A	Test B	Test C	Test D	Test E
ave	(-21.0%, 14.4%)	(-7.3%, 4.2%)	(-0.46%, 0.11%)	(-2.3%, 0.20%)	(-1.5%, 0.29%)

Table 5: Overhead error ranges for validation experiments where the simulation overhead was less than 15%.

	Test A	Test B	Test C	Test D	Test E
ave	(-1.5%, 0.89%)	(-1.1%, 0.78%)	(-0.23%, 0.06%)	(-0.43%, 0.15%)	(-0.60%, 0.22%)
max	(-4.2%, 0.56%)	(-4.8%, -0.05%)	(-2.3%, -0.02%)	(-0.91%, 0%)	(-2.3%, -0.08%)

4.2.1 Validation with Artificial Fire Rates

The first step of our validation approach is to use artificial fire rates and compare estimated values to actual values.

Experiment Overview: Our validation runs used the Cloverleaf simulation code, running on 576 cores and operating on data sets of 544M cells. The cores were divided over 16 nodes, with one MPI rank per node and OpenMP parallelism within a node. The supercomputer, "Pascal," was composed of Intel Xeon E5-2695 nodes, each with 36 cores.

We ran five sets of tests using the following trigger-analysis combinations:

- Test A: Histogram-Cinema
- Test B: Statistics-IO_{SUB}
- Test C: Statistics-Slice
- Test D: Histogram-IsoVolume
- Test E: Histogram-Contour

For each of the five tests we ran the cross product of 10 inspection rates and four target fire rates:

- R_I: 1, $1/2$, $1/3$, ..., $1/10$
- Target R_F: $R_I * 0.25$, $R_I * 0.5$, $R_I * 0.75$, $R_I * 1.0$

In total, we ran 200 tests ($5 * 10 * 4 = 200$). To fire the artificial trigger, we used the Mersenne Twister algorithm to generate a random number each time the analysis filter was invoked and called the analysis when the random number was below the target R_F.

Validation Results: Across all of our runs, the Cloverleaf cycle time was 2.5s with very low variation, so, to simplify the validation, we use this value as C_μ. For each test, there were 40 experimental values for T_μ and A_μ to choose as inputs to the model. We cross-validated every possible choice of model inputs by using each of the 40 experimental values as inputs to the model and evaluated error against the remaining 39. We compared model accuracy in terms of actual simulation overhead versus predicted simulation overhead, and we summarized all model choices as ranges, which is the union of all error ranges. Saying it another way, the accuracy ranges represent the worst-case scenario.

Table 4 shows the results of all validation experiments. Tests C-D have accuracy ranges that are reasonable, *i.e.,* the predicted simulation overhead was never more than a few percent away from the actual overhead. In tests A and B, the prediction accuracy was lower. We chose to measure simulation overhead because it demonstrates the practical application of our cost model. That said, the relative costs of the analysis operations have a large impact on the overall error ranges. For example, the cost of Cinema (Test A) is approximately 2.5x more than a simulation cycle, so even a small prediction error is amplified when put in terms of simulation overhead. Additionally, for completeness, our validation experiments covered impractical choices of R_I and R_F, *e.g.,* performing Cinema every simulation cycle. Table 5 contains the error ranges for all tests where the estimated simulation overhead is less than 15%. When using the average values for T_μ and A_μ, total model accuracy in all tests is -1.51% to 0.89%. Overall, we believe this demonstrates that the model can be used to make informed decisions when considering triggers.

4.2.2 Studying Fire Rate in Practice

In this section, we extend the validation by considering fire rates in the context of the data collected from the three simulations. In general, trigger fire rates can be challenging to understand, since they depend on the specifics of the simulation advancements, i.e., they are data dependent. That said, domain scientists are well positioned to reason about fire rates, since they understand the underlying physical processes they are simulating. In many cases, this enables them to define trigger functions tailored to a simulation and effectively predict fire rates.

To study the effect of a data-dependent function, we consider a trigger that evaluates the change in entropy over time, calculated from the histogram operation, and examine how this trigger function behaves in the context of the entropy in each simulation. Our trigger function (Equation 6) evaluates the absolute value of the entropy change at an inspection rate of R_I:

$$Fire = abs(entropy(t_i) - entropy(t_{i-1})) > threshold \quad (6)$$

Ultimately, R_F is dependent on R_I, the threshold value, and the entropy values. Since the user sets up the simulation, we expect that the user will have general a priori knowledge about the expected simulation behavior or have data from previous simulation runs, which gives them an idea about realistic values of threshold and entropy and in turn R_F. To that end, we use the data gathered from our experiments as a stand-in for this a priori knowledge, and we use the entropy trigger as a proxy to demonstrate the usefulness of our model. Figure 2 shows the entropy values over time in different fields for each simulation. Using the entropy data, we can explore the costs from R_I and threshold values.

Due to limited compute time, we only performed inspection and analysis every 100 cycles, so as not to exhaust our compute resources performing mostly analysis and visualization. This constraint mirrors the cost-benefit choice we are exploring. Thus, we use values of R_I beginning at 0.01, and we subsample the data points (*i.e.,* every n^{th} sample point) to evaluate smaller values of R_I. Based on the entropy distribution from each simulation, we chose a range of potential threshold values to explore, and we chose analysis operation where the cost was relatively high compared to the simulation's C_{ave}. Figure 3 shows the results of applying the model to our experimental data and our trigger function.

Using the cost model combined with the range of acceptable DAV_{max} values in Table 2 informs the resulting costs given different choices of R_I and threshold values. Looking at MARBL, all choices of R_I and threshold result in a total analysis cost of less than 1%. If the user is satisfied with choices within this space, then all they have to do is pick the most appropriate parameters for their use case. Conversely, the cost space for Clover and SW4 is much more varied. Each of the plots contains regions where the total cost is approaching 20%, which is the upper limit of the analysis cost from Table 1 that simulation scientists indicated they find acceptable. As per the rightsizing approach, the user must decide what cost they are willing to pay and choose the appropriate threshold and R_I values. If none of the choices are viable, then they must re-evaluate trigger choice, analysis choice, or their trigger function, then re-apply the

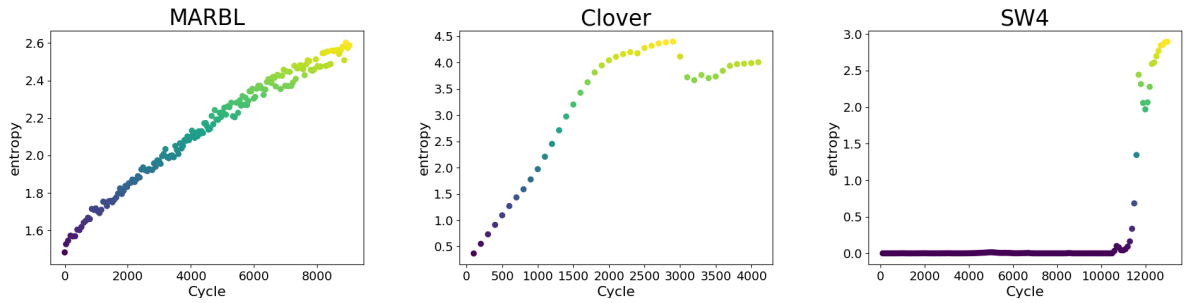

Figure 2: The entropy curves for all three simulations. From left to right, MARBL (density), Clover (energy), and SW4 (velocity magnitude).

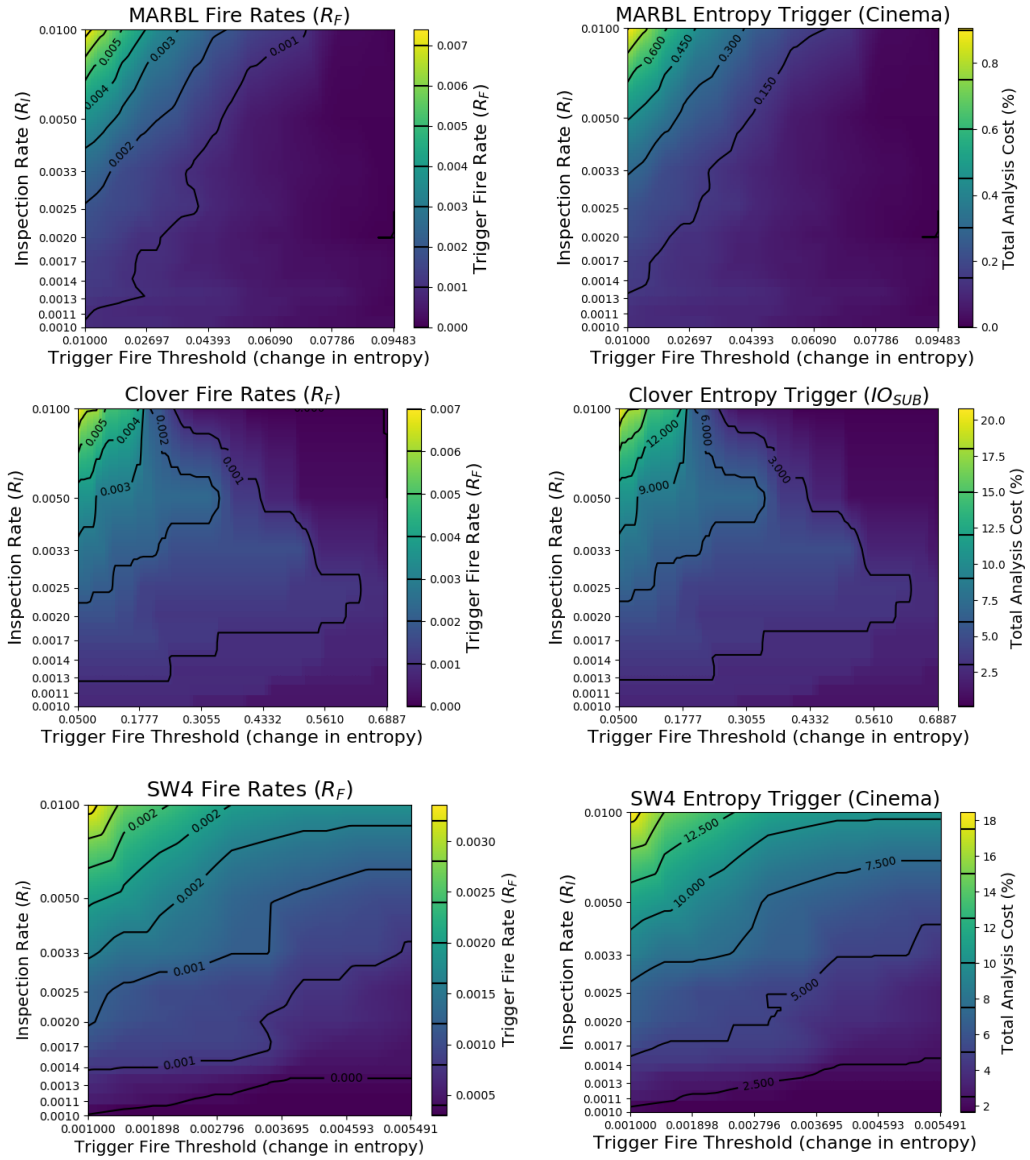

Figure 3: Plots showing the results of applying the model using the entropy based function to the experimental data. The x-axis is a range of potential threshold values to the trigger, and the y-axis is the log scale of R_I. For the left column, the color shows the R_F values for each trigger, and for the right column, the color shows the total cost as a percentage of simulation time. Each row corresponds to one of the three simulations. Finally, each plot has black contour lines to better illustrate the complexity of the underlying field. (The contour lines are of the same field as the pseudocolor plot underneath.)

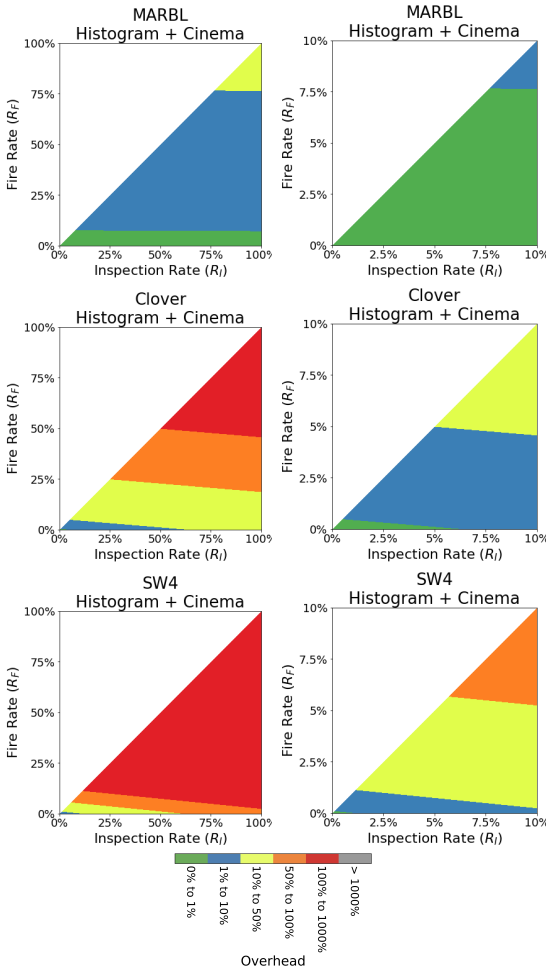

Figure 4: Cost contours for Cinema analysis triggered by Histogram inspection for the three simulation codes included in our experiments. These cost contours can help inform viable combinations of R_I and R_F given desired overhead costs. In all graphs, the X-axis is the inspection rate, the Y-axis is the fire rate, and the color represents the overhead incurred. These plots are generated from the cost models (not experimentally), so the boundaries between contours are straight lines. There is interesting behavior with both high inspection rates and low inspection rates, so we separate the two cases into separate graphs — each simulation has a pair of images, one with inspection rates varying from never ($R_I = 0\%$) to every cycle ($R_I = 100\%$), and the other with inspection rates from zero cycles ($R_I = 0\%$) to 10% of the cycles ($R_I = 10\%$). Since R_F cannot exceed R_I, the upper triangle of the plot is not valid. Further, costs on the diagonal line represent points where $R_F = R_I$ (*i.e.,* every inspection leads to a fire), and so values along that diagonal correspond to the maximum possible cost for a given inspection rate.

cost model until they have a choice that works.

The data in Figure 3 was generated by applying the cost model. We validated the accuracy of the cost model by re-running the SW4 experiment with a R_I of 1, meaning that we performed trigger inspection every cycle. This allowed us to compare 1000 interpolated values generated by the cost model with actual overheads. We found that the average error in the total analysis costs was 3%.

4.3 Using the Cost Model to Inform the Space of Viable Trigger Approaches

This section addresses **RQ2** by applying our model to determine the viability of different use cases, using the model values determined from the previous investigation. While our data corpus is not exhaustive, we feel its members are representative and analyzing viability questions with these members informs the entire space. We break this analysis into two parts. First, Section 4.3.1 considers how the model can be used to inform different trigger investigation and fire rates for a single pairing of trigger and DAV algorithm (specifically histograms and generating a Cinema database). Second, Section 4.3.2 repeats the analysis from Section 4.3.1 but with multiple pairings of trigger and DAV algorithms.

4.3.1 Varying Trigger Inspection and Fire Rate

For our rightsizing approach, we feel the most common settings to change will be inspection rate (R_I) and fire rate (R_F). In other words, we feel the other terms in the model will be less likely to change — certainly not cycle time (C_μ), and likely not the DAV task (A_μ) or an increase in the amount of overhead ($DAV_{Overhead}$). Further, if a domain scientist believes that a trigger approach will capture the right insight and is lightweight enough, then it is also not likely to change (T_μ). In all, our goal with this analysis is to inform the variation in overhead as trigger settings vary. To do this, we pick one combination of trigger and analysis — Histograms and Cinema output — and consider the effects over all three simulation codes.

Figure 4 shows cost contour plots for all three simulations and for all inspection rates and fire rates. A plot like this can allow a domain scientist to quickly infer whether DAV overhead will exceed their acceptable overhead (DAV_{MAX}). For example, the MARBL team is willing to allow up to 15% overhead. Figure 4 shows that this level of overhead almost never occurs in practice, and so less care is needed in arranging triggers and DAV. On the other hand, the costs for Clover and SW4 quickly get quite large. In these cases, domain scientists would likely want to scale back inspection rate and also adjust their trigger so the fire rate was low.

4.3.2 Varying Trigger Inspection and Fire Rate for Multiple Triggers and DAV Algorithms

Where Section 4.3.1 considered a single pair of trigger and DAV algorithm, this section considers all pairs from our experiments — the results in this section are the ones most responsive to **RQ2**. Figure 5 plots these results, enabling exploration across a wide range of cases. The value of this figure is that it enables comparisons across trigger and DAV algorithm. For example, while an assumption from Section 4.3.1 was that domain scientists were not likely to change their DAV algorithm, this plot enables them to at least consider the change in cost from switching using our rightsizing approach. In this particular case, producing a Cinema database (the example from Section 4.3.1) was problematic for Clover and SW4, but the RayTracer is more reasonable — a domain scientist may decide that DAV algorithm represents a worthy tradeoff given its reduced overhead. With respect to **RQ2**, DAV practitioners can reason about the costs of their algorithms. In particular, trigger evaluation time (T_μ) was not problematic for MARBL, but Figure 5 shows that it is more significant for Clover and SW4. This is clearly visible with the RayTracer plots, where the zero-cost trigger stays less than 50% overhead (blue or yellow colors) for many more configurations than Histogram and Statistics.

In terms of takeaways for DAV scientists developing triggers, there is clearly a large spectrum of outcomes, as the costs vary significantly from simulation to simulation. There are many reasons for this variation, including data size, mesh types, the types of solvers used, and architectures that the codes run on. For example, MARBL is simulating a Kelvin-Helmholtz instability while Clover is solving the incompressible Euler equations, *i.e.,* MARBL is doing far more work than Clover. That said, for a DAV scientist, the opportunity to do more frequent inspections and heavier weight triggers exists for MARBL, but does not for Clover. For SW4 (which is representative of a simulation code that generates lots of data and has a short cycle

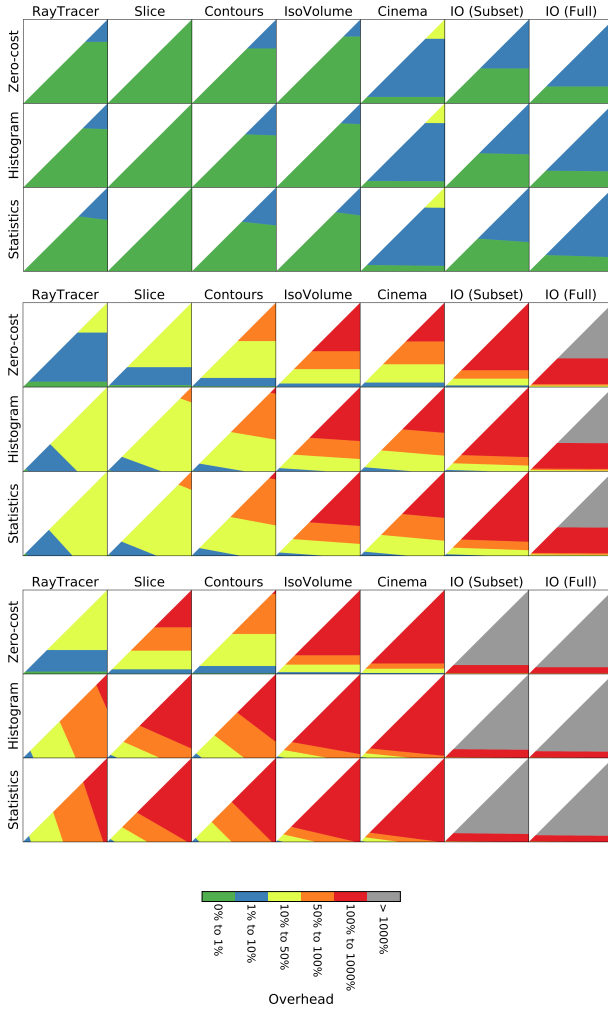

Figure 5: Cost contours for all DAV algorithms for MARBL (top grouping), Clover (middle grouping), and SW4 (bottom grouping). Further, a hypothetical trigger that has zero cost was included, to evaluate the contribution of the trigger operation to the contours. Each grouping has three rows, each corresponding to a trigger: the zero-cost trigger (top row within a grouping), Histogram (middle row within a grouping), and Statistics (bottom row within a grouping).

time), the situation is even worse — trigger-based approaches must employ some combination of being very fast, inspecting infrequently, and having a low fire rate. Once again, the main takeaway is that the appropriateness of a trigger is highly dependent on the simulation code.

5 Conclusion

This paper considers two important research questions with respect to execution time for trigger-based in situ analysis, with the first on whether a specific configuration is viable and the second on informing the space of viable approaches. We used a cost model for answering both questions, and we believe this is the first in situ cost model that incorporates triggers. Our approach provides a tool that enables trigger users and researchers to make decisions balancing between science goals, cost constraints, and data-dependent properties. It informs analysis overhead in a variety of scenarios, and how the design of a specific trigger can be adapted to increase or decrease overhead.

We believe the work is valuable to both computational scientists and in situ analysis researchers. For computational scientists, our model will help with reasoning about what tasks can be performed with triggers, and how modifying thresholds in triggers can affect overhead. For in situ analysis researchers, this work helps informs the boundaries of future research, *i.e.,* what are reasonable limits for execution time for both triggers and follow-on analysis tasks? Finally, we feel cost viability questions such as this one will become increasingly important as more application codes look to optimize in situ analysis.

Future work primarily involves revisiting the three key assumptions made in our study. First, while our work assumed a "tightly-coupled" environment, considering a flexible in transit setting — as used by ADIOS [24], Bredala [15], Damaris [14], FlowVR [23], GoldRush [34], SENSEI [8], and others — could lead to valuable propositions for improved analysis at reduced cost. With respect to our model, the analysis for costs would involve usage of dedicated resources, non-zero publishing costs, and (of course) questions of where triggers/analysis should be performed. Second, we assumed that an analyst would have intuition about when visualization/analysis was being performed too infrequently and would adapt choices to ensure this does not happen. One path forward to eliminating this assumption is to gain more holistic understanding of the error incurred based on trigger strategy and frequency. The benchmarking work by Kawakami et al. [18] is an initial attempt in this direction, and expanding this benchmark suite with more trigger techniques, simulations, analyses, etc., could add significant insight on this issue. Also, while the experiments in this paper do not include saving simulation state to disk at a regular interval, this practice certainly fits within our approach and cost model framework, and also limits concerns about missing important phenomena and potentially having to re-run the simulation. Third, our approach requires estimates of cycle times, inspection times, analysis times, etc., which we primarily obtained experimentally. Additional cost models would eliminate the need to run such experiments. Further, while our model accounted for a range of outcomes from cycle to cycle (*i.e.,* allowing for A_{μ} instead of A_{ave}), we feel that predictions can become increasingly accurate as the range of outcomes are better understood. In particular, we saw low variance among activities in our experiments, but other simulations may have higher variance, which would be challenging for our approach.

Finally, while this work considered binary triggers that consider a given time step, triggers can be more probabilistic in nature, for example attempting to quantify the value of analyzing the current time step given previous analyses. Our approach can support non-binary triggers (since they ultimately do decide to fire or not fire), but reasoning about their fire rates can be more difficult. We view non-binary triggers techniques as a worthy future research direction, as well as expanding our model to accommodate them.

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-799838). This work was released under LA-UR-21-25986. 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. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

REFERENCES

[1] V. Adhinarayanan, W. Feng, D. Rogers, J. Ahrens, and S. Pakin. Characterizing and modeling power and energy for extreme-scale in-situ visualization. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 978–987, 2017.

[2] J. Ahrens, B. Geveci, and C. Law. Paraview: An end-user tool for large data visualization. *The Visualization Handbook*, pp. 717–731, 2005.

[3] J. P. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D. H. Rogers, and M. Petersen. An Image-based Approach to Extreme Scale in Situ Visualization and Analysis. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 424–434, 2014.

[4] F. Alexander et al. Exascale Applications: Skin in the Game. *Philosophical Transactions of the Royal Society*, 378(2166), 2020.

[5] G. Aupy, B. Goglin, V. Honoré, and B. Raffin. Modeling high-throughput applications for in situ analytics. *The International Journal of High Performance Computing Applications*, 33(6):1185–1200, 2019.

[6] U. Ayachit, A. Bauer, B. Geveci, P. O'Leary, K. Moreland, N. Fabian, and J. Mauldin. 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 (ISAV)*, pp. 25–29, Nov. 2015.

[7] U. Ayachit et al. Performance Analysis, Design Considerations, and Applications of Extreme-Scale In Situ Infrastructures. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC16)*, pp. 921–932, Nov. 2016.

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

[9] A. C. Bauer, H. Abbasi, J. Ahrens, H. Childs, B. Geveci, S. Klasky, K. Moreland, P. O'Leary, V. Vishwanath, B. Whitlock, and E. W. Bethel. In Situ Methods, Infrastructures, and Applications on High Performance Computing Platforms. *Computer Graphics Forum (CGF)*, 35(3):577–597, June 2016.

[10] J. Bennett, A. Bhagatwala, J. Chen, A. Pinar, M. Salloum, and C. Seshadhri. Trigger detection for adaptive scientific workflows using percentile sampling. *SIAM Journal on Scientific Computing*, 38(5):S240–S263, 2016.

[11] H. Childs, J. Bennett, C. Garth, and B. Hentschel. In Situ Visualization for Computational Science. *IEEE Computer Graphics and Applications*, 39(6):76–85, 2019.

[12] H. Childs et al. VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization—Enabling Extreme-Scale Scientific Insight*, pp. 357–372. CRC Press/Francis–Taylor Group, Oct. 2012.

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

[14] M. Dorier, R. Sisneros, T. Peterka, G. Antoniu, and D. Semeraro. Damaris/viz: a nonintrusive, adaptable and user-friendly in situ visualization framework. In *2013 IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV)*, pp. 67–75, 2013.

[15] M. Dreher and T. Peterka. Bredala: Semantic data redistribution for in situ applications. In *2016 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 279–288, 2016.

[16] G. Haldeman, I. Rodero, M. Parashar, S. Ramos, E. Z. Zhang, and U. Kremer. Exploring energy-performance-quality tradeoffs for scientific workflows with in-situ data analyses. *Computer Science - Research and Development*, 30(2):207–218, May 2015.

[17] H. Johansen, A. Rodgers, N. A. Petersson, D. McCallen, B. Sjogreen, and M. Miah. Toward exascale earthquake ground motion simulations for near-fault engineering analysis. *Computing in Science & Engineering*, 19(5):27–37, 2017.

[18] Y. Kawakami, N. Marsaglia, M. Larsen, and H. Childs. Benchmarking In Situ Triggers Via Reconstruction Error. In *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV)*, Nov. 2020.

[19] J. Kress, M. Larsen, J. Choi, M. Kim, M. Wolf, N. Podhorszki, S. Klasky, H. Childs, and D. Pugmire. Comparing the Efficiency of In Situ Visualization Paradigms at Scale. In *ISC High Performance*, pp. 99–117, June 2019.

[20] J. Kress, M. Larsen, J. Choi, M. Kim, M. Wolf, N. Podhorszki, S. Klasky, H. Childs, and D. Pugmire. Opportunities for Cost Savings with In Transit Visualization. In *ISC High Performance*, pp. 146–165, June 2020.

[21] M. Larsen, J. Ahrens, U. Ayachit, E. Brugger, H. Childs, B. Geveci, and C. Harrison. The ALPINE In Situ Infrastructure: Ascending from the Ashes of Strawman. In *Proceedings of the Workshop of In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV)*, pp. 42–46, Nov. 2017.

[22] M. Larsen, A. Woods, N. Marsaglia, A. Biswas, S. Dutta, C. Harrison, and H. Childs. A Flexible System for in Situ Triggers. In *Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, pp. 1–6, 2018.

[23] J.-D. Lesage and B. Raffin. High performance interactive computing with flowvr. In *Proceedings of IEEE Virtual Reality SEARIS Workshop*, pp. 13–16, 2008.

[24] Q. Liu et al. Hello ADIOS: The Challenges and Lessons of Developing Leadership Class I/O Frameworks. *Concurrency and Computation: Practice and Experience*, 26(7):1453–1473, May 2014.

[25] P. Malakar, V. Vishwanath, C. Knight, T. Munson, and M. E. Papka. Optimal execution of co-analysis for large-scale molecular dynamics simulations. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 702–715, 2016.

[26] P. Malakar, V. Vishwanath, T. Munson, C. Knight, M. Hereld, S. Leyffer, and M. E. Papka. Optimal scheduling of in-situ analysis for large-scale scientific simulations. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC15)*, pp. 52:1–52:11, Nov. 2015.

[27] A. Mallinson, D. A. Beckingsale, W. Gaudin, J. Herdman, J. Levesque, and S. A. Jarvis. Cloverleaf: Preparing hydrodynamics codes for exascale. *The Cray User Group's Meeting*, May 2013.

[28] K. Moreland et al. VTK-m: Accelerating the Visualization Toolkit for Massively Threaded Architectures. *IEEE Computer Graphics and Applications (CG&A)*, 36(3):48–58, May/June 2016.

[29] K. Myers, E. Lawrence, M. Fugate, C. M. Bowen, L. Ticknor, J. Woodring, J. Wendelberger, and J. Ahrens. Partitioning a Large Simulation as It Runs. *Technometrics*, 58(3):329–340, 2016.

[30] B. Nouanesengsy, J. Woodring, J. Patchett, K. Myers, and J. Ahrens. ADR Visualization: A Generalized Framework for Ranking Large-Scale Scientific Data Using Analysis-Driven Refinement. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 43–50, Nov. 2014.

[31] T. Peterka, D. Bard, J. C. Bennett, E. W. Bethel, R. A. Oldfield, L. Pouchard, C. Sweeney, and M. Wolf. Priority research directions for in situ data management: Enabling scientific discovery from diverse data sources. *The International Journal of High Performance Computing Applications*, 34(4):409–427, Mar. 2020.

[32] M. Salloum, J. C. Bennett, A. Pinar, A. Bhagatwala, and J. H. Chen. Enabling adaptive scientific workflows via trigger detection. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV)*, pp. 41–45, Nov. 2015.

[33] B. Whitlock, J. M. Favre, and J. S. Meredith. Parallel In Situ Coupling of Simulation with a Fully Featured Visualization System. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)*, pp. 101–109, Apr. 2011.

[34] F. Zheng, H. Yu, C. Hantas, M. Wolf, G. Eisenhauer, K. Schwan, H. Abbasi, and S. Klasky. Goldrush: Resource efficient in situ scientific data analytics using fine-grained interference aware execution. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, 2013.