SOSflow: A Scalable Observation System for Introspection and In Situ Analytics

Chad Wood
University of Oregon
Eugene, OR, United States
cdw@cs.uoregon.edu

ABSTRACT

The performance of HPC simulation codes is often tied to their simulated domains; e.g., properties of the input decks, boundaries of the underlying meshes, and parallel decomposition of the simulation space. A variety of research efforts have demonstrated the utility of projecting performance data onto the simulation geometry to enable analysis of these kinds of performance problems. However, current methods to do so are largely ad-hoc and limited in terms of extensibility and scalability. Furthermore, few methods enable this projection online, resulting in large storage and processing requirements for offline analysis. We present a general, extensible, and scalable solution for in-situ (online) visualization of performance data projected onto the underlying geometry of simulation codes. Our solution employs the scalable observation system SOSflow with the in-situ visualization framework ALPINE to automatically extract simulation geometry and stream aggregated performance metrics to respective locations within the geometry at runtime. Our system decouples the resources and mechanisms to collect, aggregate, project, and visualize the resulting data, thus mitigating overhead and enabling online analysis at large scales. Furthermore, our method requires minimal user input and modification of existing code, enabling general and widespread adoption.

KEYWORDS

sos, sosflow, hpc, performance, visualization, in situ

1 INTRODUCTION

Modern clusters for parallel computing are complex environments. High-performance applications that run on modern clusters do so often with little insight about their or the system's behavior. This is not to say that information is unavailable. After all, sophisticated parallel measurement systems can capture performance and power data for characterization, analysis, and tuning purposes, but the infrastructure for observation of these systems is not intended for general use. Rather, it is specialized for certain types of performance information and typically does not allow online processing. Other information sources of interest might include the operating system (OS), network hardware, runtime services, or the parallel application itself. Our general interest is in parallel application monitoring: the observation, introspection, and possible adaptation of an application during its execution.

Application monitoring has several requirements. It is important to have a flexible means to gather information from different sources on each node — primarily the application and system environment. Additionally, for the gathered information to be processed online, analysis will need to be enabled in situ with the application. Query and control interfaces are required to facilitate an active application feedback process. The analysis performed can be used to give feedback to both the application, the operating environment, and performance tools.

This research demonstrates the *Scalable Observation System (SOS)* for integrated application monitoring. The SOS design employs a data model with distributed information management and structured query and access. A dynamic database architecture is used in SOS to support aggregation of streaming observations from multiple sources. Interfaces are provided for in situ analytics to acquire information and then send back results to application actuators and performance tools. SOS launches with the application, runs along side it, and can acquire its own resources for scalable data collection and processing. A working implementation of SOS is contributed as a part of this research effort, *SOSflow* [4]. The SOSflow platform demonstrates all of the essential characteristics of the SOS model, showing the scalability and flexibility inherent to SOS with its support for observation, introspection, feedback, and control of scientific workflows.

1.1 Scientific Workflows

Scientific workflows feature two or more components that are coupled together, operating over shared information to produce a cumulative result. These components can be instantiated as lightweight threads belonging to a single process, or they may execute concurrently as independent processes. Components of workflows can be functionally isolated from each other or synchronously coupled and co-dependent. Some workflows can be run on a single node, while others are typically distributed across thousands of nodes. Additionally, parts of workflows may even be dynamically instantiated and terminated. The computational profile of a workflow can change between invocations or even during the course of one execution.

1.2 Multiple Perspectives

Application state and events can be sent to SOS from within the application at any point during its execution. Developers can instrument their programs to be efficiently self-reporting the data that is relevant to their overall performance, such as progress through specific phases of a simulation.

Application performance can be dramatically impacted by changes in *the state of the operating environment* that is hosting it. The effects of contention for shared resources by multiple concurrent tasks can be discovered when the events of concurrent tasks are fixed into a common context for reasoning about their individual and combined performance. SOS's distributed in situ design is

well-suited for capturing perspective of the global state of a machine. By co-locating the observation system with the workflow components that are observed, SOS improves the fidelity of system performance data without requiring the costly delays of synchronization or congesting the shared network and filesystem resources in use by applications.

Many existing *performance tools can provide useful observations* at runtime of applications, libraries, and the system context. The low-level timers, counters, and machine-level data points provided by specialized performance tools can be a valuable addition to the higher-level application and system data.

2 PERFORMANCE UNDERSTANDING

Projecting application and performance data onto the scientific domain allows for the behavior of a code to be perceived in terms of the organization of the work it is doing, rather than only the organization of its source code. This perspective can be especially helpful [3] for domain scientists developing aspects of a simulation primarily for its scientific utility, though it can also be useful for any HPC developer engaged with the general maintenance requirements of a large and complicated codebase [2].

There have been practical challenges to providing these opportunities for insight. Extracting the spatial descriptions from an application traditionally has relied on hand-instrumenting codes to couple a simulation's geometry with some explicitly defined performance metrics. Performance tool wrappers and direct source-instrumentation need to be configurable so that users can disable their invasive presence during large production runs. Because it involves changes to the source code of an application, enabling or disabling the manual instrumentation of a code often involves full recompilation of a software stack. Insights gained by the domain projection are limited to what was selected a priori for contextualization with geometry.

Without an efficient runtime service providing an integrated context for multiple sources of performance information, it is difficult to combine performance observations across several components during a run. Further limiting the value of the entire exercise, performance data collected outside of a runtime service must wait to be correlated and projected over a simulation's geometry during post-mortem analysis. Projections that are produced offline cannot be used for application steering, online parameter tuning, or other runtime interactions that include a human in the feedback loop. Scalability for offline projections also becomes a concern, as the potentially large amount of performance data and simulation geometry produced and operated over in a massively parallel cluster now must be integrated and rendered either from a single point or within an entirely different allocation.

The overhead of manually instrumenting large complex codes to extract meaningful geometries for use in performance analysis, combined with the limited value of offline correlation of a fixed number of metrics, naturally limited the usage of scientific domain projections for gaining HPC workflow performance insights.

2.1 Research Contributions

This work makes use of SOSflow and ALPINE Ascent [1] to overcome many prior limitations to projecting performance into the scientific domain. This research effort achieved the following:

- Eliminate the need to manually capture geometry for performance data projections of ALPINE-enabled workflows
- Provide online observation of performance data projected over evolving geometries and metrics
- Facilitate interactive selection of one or many performance metrics and rendering parameters, adding dynamism to projections
- Enable simultaneous online projections from a common data source
- In situ performance visualization architecture supporting both current and future-scale systems

REFERENCES

[1] Matthew Larsen, James Aherns, Utkarsh Ayachit, Eric Brugger, Hank Childs, Berk Geveci, and Cyrus Harrison. 2017. The ALPINE In Situ Infrastructure: Ascending from the Ashes of Strawman. In *Proceedings of the In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization Workshop (ISAV2017)*. ACM, New York, NY, USA.

[2] Martin Schulz, Abhinav Bhatele, David Böhme, Peer-Timo Bremer, Todd Gamblin, Alfredo Gimenez, and Kate Isaacs. 2015. A Flexible Data Model to Support Multi-domain Performance Analysis. In *Tools for High Performance Computing 2014*. Springer, 211–229.

[3] Martin Schulz, Joshua A Levine, Peer-Timo Bremer, Todd Gamblin, and Valerio Pascucci. 2011. Interpreting performance data across intuitive domains. In *Parallel Processing (ICPP), 2011 International Conference on*. IEEE, 206–215.

[4] Chad Wood, Sudhanshu Sane, Daniel Ellsworth, Alfredo Gimenez, Kevin Huck, Todd Gamblin, and Allen Malony. 2016. A scalable observation system for introspection and in situ analytics. In *Proceedings of the 5th Workshop on Extreme-Scale Programming Tools*. IEEE Press, 42–49.