SOSflow and a Semantic Workflow Performance Model for Understanding Variability in Scientific Workflows at Scale

Chad Wood
Kevin Huck
Allen Malony
Department of Computer and Information Science
University of Oregon
Eugene, OR United States
Email: cdw@cs.uoregon.edu

***Abstract*—SOSflow provides a run-time system and performance model designed to enable the characterization and analysis of complex scientific workflow performance at scale.**

***Keywords*-hpc; exascale; in situ; performance; monitoring; introspection; scientific workflow;**

I. Introduction

Here we set up the problem space we are addressing. It will be a good time!

II. Motivations for Tracking Variability

[NOTE: More than simply demonstrating that there are different and possibly incompatible motivations for variability research, this section is setting up the necessity for a new performance model.]

Due to the novelty of this research area, there will be opportunities to further refine the way in which research is taxonomized, and to establish a set of definitions that provide adequate coverage of the conceptually distinct types of variability studies. Though such preliminaries are not our primary purpose here, it is a useful exercise to consider some of the different purposes behind current studies of performance variability. The design choices made for the SOSflow system and its inherent workflow performance model emerged from reflecting on the general nature of variability studies, the divergent purposes that are create tensions between the efforts being undertaken in this new field, and the anticipated realities of writing infrastructure codes for exascale HPC clusters.

A. Why Variability is Tracked

Many factors have contributed to the emergence of variability studies as an important research topic within the HPC community. At a low-level, HPC node engineering has grown in complexity and sophistication, many on-core processor behaviors that used to be isolated, synchronous, and predictable, are now interrelated, data-driven, asynchronous, and impossible to predict a priori. As core density increases on the nodes of a cluster (TODO: Cite paper showing this is the necessary direction of exascale) the unpredictability and inconsistency of the hardware itself becomes an increasingly significant contributor to observed variability. Hardware is an important source to consider when it comes to variability because there is almost nothing that can be done to control for it, the noise has to be admitted in the results, and therefore is an important part of the output of any performance-related experiment. At higher-levels of description, variability in workflow performance can be introduced by many different sources:

- Versions of software libraries across can differ across clusters, or even the same cluster across time.
- Tuning factors to extract maximum performance from a code can vary across clusters even if the code and the data do not change. Shared filesystem performance, data transport methods, available per-process memory, co-processor presence and architecture, ...and more, all can be responsible for influencing sensitivity to novel tuning factors.
- Concurrent activity elsewhere on the same cluster, activity that will necessarily vary between every iteration of the workflow, may be having a significant impact on observed performance.
- At extreme scales, parts of a simulation are almost guaranteed to fail due to the marginal failure rates of hardware components approaching absolute certainty as the number of involved components increases. These failures cannot be accurately predicted a priori, and the design constraints that account for and respond to them introduce performance perturbation and further complexity.
- ...
- ...

1) Awareness of Fine-Grained Performance Jitter: [NOTE: This section is purposed with showing that even episodic close analysis of individual components at small scale will have issues with significant variability. There are tractible concerns (demonstrating sensitivity to var.)

and inherently intractible concerns (controlling var. across runs).]

Even in cases where nothing can be done to influence the variations in performance, it is important to recognize that performance variability is present and to attempt to both quantify it and render reasonable attributions of its source[s]. Significant variability between runs is seen (SCHULTZ paper/graph) when tracking a single simple experiment, even if the input data, job queue parameters, and hardware allocation is held constant. At extreme scales, on line detection and attribution of variability of a scientific workflow will require well-annotated metadata to facilitate "apples to apples" comparisons driven by unsupervised machine learning rather than a priori developer knowledge or offline centralized analysis. It is important to characterize a code's sensitivity to variability, as well as a cluster's propensity for creating performance variability.

2) Accuracy in Performance Research: [NOTE: This section is intended to motivate a new model for performance when considering scientific workflows on exascale systems. Nail the coffin shut on existing performance research validation tools and techniques. They are intractible, per the above notes, and irrelevant, per this discussion]

Performance research is principally concerned with decreasing the resource consumption and compute time required by low-level components and libraries that are used when constructing higher-level scientific workflows. For example: In order to validate a 5 percent increase in some code's performance, it will be necessary to show that performance increase was observed across a vast array of runs and hardware allocations, especially if it is the case that a particular HPC cluster (when in an overall state similar to the one it was in during those workflow runs) has a history of performance variability with any statistical significance relative to the observed performance gain. When it comes to the behavior of codes at extreme scales, accuracy validation using traditional models of component-based performance analysis will become cost prohibitive in both allocation consumption and developer time.

3) Reproduction of Experimental Results: Scientific workflows attempt to yield results that have truth-coorespondence with the physical world with some overt degree of significance.

B. How Variability is Tracked

1) Concepts and Methods:

2) Existing Models and Tools:

C. The Utility a Comprehensive Workflow Performance Model

1) Attribution:

2) Resource Requirement Prediction:

3) Automated Component Performance Tuning: Don't want conflicting optimizer purposes. Need to know where the hotspots *really* are and not

4) Intelligent Compiler Hints:

5) Intelligent Job Scheduling:

III. A Performance Model for Scientific Workflows

Traditionally, HPC performance monitoring is focused on low-level efficiency of an application binary on some particular iron. Higher-level systems (TACC Stats) allow the tracking and exploration of execution wall-time for various library versions, integration of multiple modalities of information (LDMS) like program invocation or work allocation across a cluster as informed by network congestion statistics, and other hybridized or meta-execution data points. Low-level metrics are more naturally suited for off-line episodic performance analysis of individual workflow components, but cannot yield insight into the run-time performance of a complex workflow. Characterizing and understanding the emergent properties of a workflow comprised of many components that are interacting asynchronously across a distributed HPC cluster requires taking a new approach to the problem, especially when considering the extreme scales of parallelism to which scientific workflows are being driven.

A. Workflow Variability is Revealed by Invariant Meaning

B. Levels of Description and "The View from Anywhere"

Not knowing *a priori* what component or layer of the workflow will be responsible for the introduction of variability, the workflow performance model needs to be populated by a diversity of information sources that provide metrics, metadata, and events from many different layers of an execution environment:

- Application
- Libraries
- Environment
- Operating System
- Node Hardware
- Network
- Enclave
- Cluster
- Workflow
- Epoch

C. Semantics

All information that is gathered by the monitoring system should be annotated as richly as possible to maximize its usefulness when performing analysis to

IV. SOSflow

Applying a semantic workflow performance model to actual run-time environments necessitated the development of new infrastructure software, and so the second contribution to the general challenges of monitoring scientific workflows is the SOSflow software artifact.

A. Behavior

B. Implementation

C. Limitations and Concerns

D. Results of Initial Benchmarking

Tool development is drawing to a close and research related to SOSflow and the cooresponding performance model is now shifting into results gathering mode. Early benchmarks are being gathered by instrumenting

V. Conclusion

Given a diverse set of motivations for variability studies, and a seemingly intractable problem space when classical performance models are applied to scientific workflows at extreme scale, we argue that a new performance model is required. SOSflow was developed to enable the exploration and validation of new performance models, especially those built for reasoning over high-level and human-understandable semantic annotation that is affixed to all captured data and events.

A. Recommendations

Our research initiative is oriented towards on line and in situ monitoring and analytics. A system should be designed that is efficient enough that it need not be disabled for full-scale production runs of scientific workflows. This creates the important case that the lessons learned through the use of such a monitoring system are in fact applicable to future full-scale production runs, and do not only apply to smaller sample test runs where the workflow's behavior has been heavily perturbed by invasive monitoring technologies. The conclusion goes here. this is more of the conclusion

B. Future Work

Acknowledgment

Research has been conducted under the following grants:
TODO: INSERT GRANT INFORMATION HERE

The authors would like to thank:
NAME of Georgia Tech University
Hasan Abassi of Argonne National Lab
Todd Gamblin of Lawrence Livermore National Lab

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.