Workflows for Performance Predictable and Reproducible HPC Applications

Keira Haskins
Department of Computer Science
University of New Mexico
Albuquerque, NM USA
keirahaskins@cs.unm.edu

Quincy Wofford
Department of Computer Science
University of New Mexico
Albuquerque, NM USA
qwofford@cs.unm.edu

Patrick G. Bridges
Department of Computer Science
University of New Mexico
Albuquerque, NM USA
patrickb@unm.edu

***Abstract*—This poster presents an HPC application workflow system whose goal is to provide verifiably-reproducible HPC application performance. This system combines existing container, experiment, and data management techniques with HPC performance models, allowing it to both maximize performance reproducibility *and* inform users when application performance deviates from what should be expected even when running at scales or for lengths of time at which the application had never run.**

***Index Terms*—container, HPC, experiment workflow, reproducibility**

I. Introduction

Performance predictability and reproducibility is important for high-performance computing (HPC) applications. Predictable performance allows the system to better allocate resources to the applications, and significant variation from expected performance can be stronger indication of potential application correctness problems. Predicting HPC application performance, however, can be difficult in the face of changing application inputs, scales, systems, and software configurations.

This poster presents our ongoing research on creating HPC application workflows that provide verifiably-reproducible HPC application performance. This system does so by combining performance-predictive HPC application models with existing container, experiment, and data management techniques. This allows the system to both maximize performance reproducibility *and* inform users when application performance deviates from what should be. Importantly, it can do so even when running at scales or for lengths of time at which the application had never run.

a) Overall Architecture:: Our general workflow, shown in Figure 1, relies on a variety of different tools and techniques in order to build a flexible architecture which may easily be adapted for use on many different systems. First, we integrate git and docker-based source code control, data storage, and container management systems into end-to-end experiment workflows based on the Popper convention [1]. These experiments are executed on standard HPC systems using container execution systems such as Singularity [2]. Performance monitoring data is collected and pre-processed on this infrastructure using performance monitoring tools such as Open SpeedShop [3] or LDMS [4]. Finally, data analysis and machine learning techniques then process the resulting performance data and generate a model of application performance which can then be used to guide and optimize future application execution. The resulting continuous integration pipeline can also be used to regularly validate application behavior and associated research results.

b) Modeling Approach:: We build on recent work using Extreme Value Theory (EVT) to model large-scale application performance [5]. This approach provides a foundation for using big-data analytics techniques to reason about HPC application performance, and focuses on estimating the distribution of the lengths of the application's bulk synchronous parallel (BSP) intervals. Our current approach is focusing on using traces of application performance to estimate BSP interval lengths, but we are also examining other approaches to allow lower overhead sampling-based approaches to construct application performance models. These analyses are performed in an iPython environment, providing both a convenient dashboard for application scientists to visualize application performance characteristics and access to state-of-the-art data analytics techniques.

The contributions of this research are:

- A *performance-predictability oriented workflow design* that integrates state-of-the-art HPC container, data and experiment management, and performance monitoring tools;
- A model-based approach to *HPC application performance prediction, regression testing, and anomaly detection* in these workflows; and
- A demonstration of the ability of the resulting system to provide performance predictability and reproducibility across varying systems, system scales, and applications.

II. Results

Our HPC application workflow consists of the following Popper experiment pipeline stages:

1) An LDMS aggregator launches on the head node, monitoring hardware and software samplers at 2 second intervals across all allocated nodes: /proc/stat, /proc/vmstat, /proc/meminfo, /proc/interrupts, OpenMPI_Send, OpenMPI_Recv.

978-1-7281-4734-5/19/$31.00 ©2019 IEEE

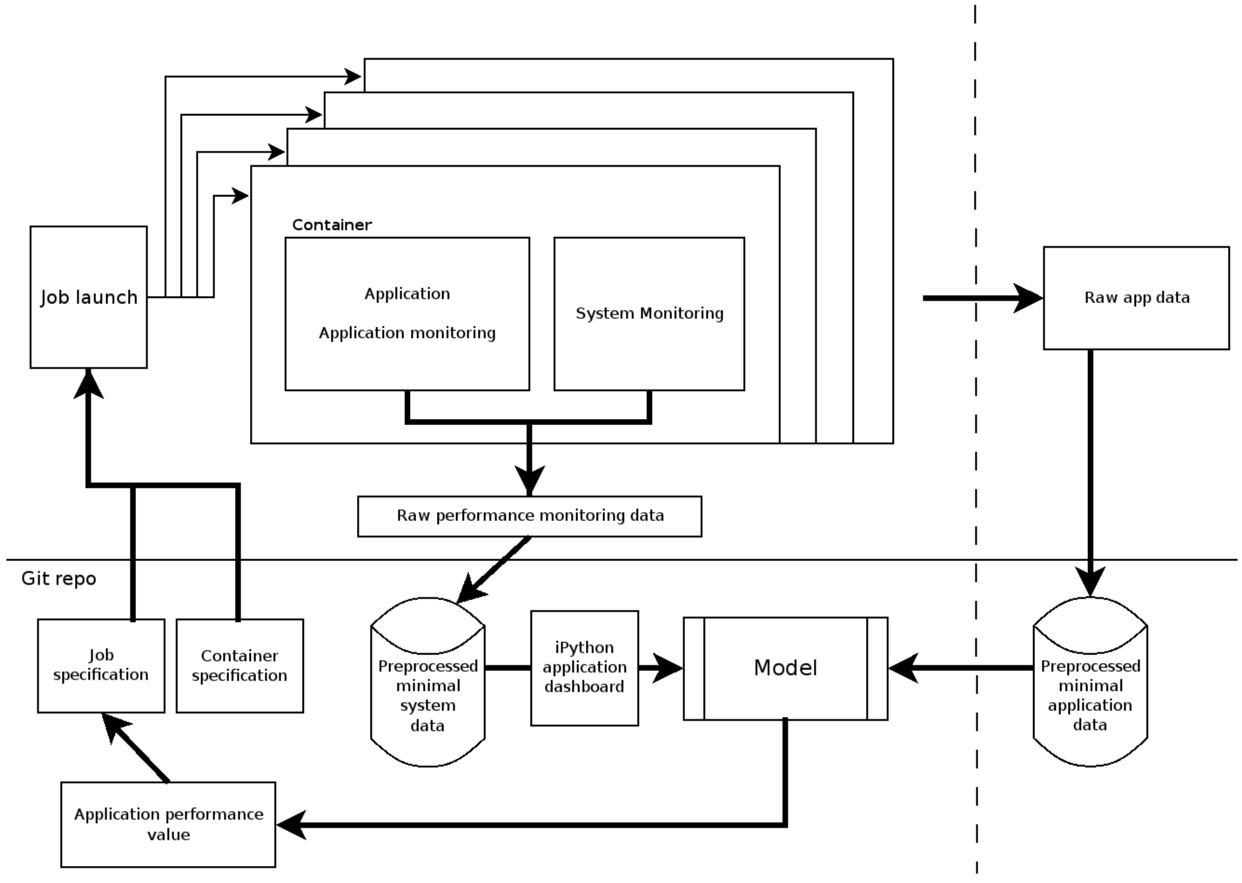
Job launch
Container
Application
Application monitoring
System Monitoring
Raw app data
Raw performance monitoring data
Git repo
Job specification
Container specification
Preprocessed minimal system data
iPython application dashboard
Model
Preprocessed minimal application data
Application performance value

Fig. 1. Overall Workflow Design

2) A pre-configured Docker image is pulled from Dockerhub into the UNM-CARC Wheeler cluster.
3) The **bsp_prototype** image is launched as a Singularity container on a Wheeler compute node
4) An LDMS sampler is launched within the container before running an OpenMPI workload.
5) Data are output by the aggregator in CSV format for post-hoc analysis.

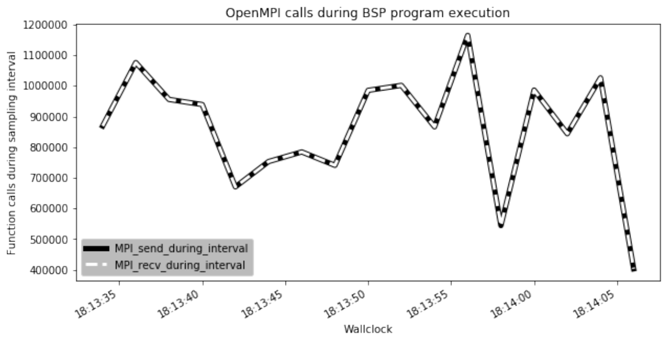

Fig. 2. LDMS OpenMPI Sampler data

Figure 2 shows output from the OpenMPI samplers described.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. OAC-1807563.

REFERENCES

[1] I. Jimenez, M. Sevilla, N. Watkins, C. Maltzahn, and J. Lofstead, "The popper convention: Making reproducible systems evaluation practical," in *2017 IEEE International Parallel and Distributed Processing Symposium Workshops, (IPDPSW)*, 2017.
[2] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute," *PLoS ONE*, vol. 12, no. 5, 2017.
[3] M. Schulz, J. Galarowicz, D. Maghrak, W. Hachfeld, D. Montoya, and S. Cranford, "Open speedshop: An open source infrastructure for parallel performance analysis," *Sci. Program.*, vol. 16, no. 2-3, pp. 105–121, Apr. 2008. [Online]. Available: http://dx.doi.org/10.1155/2008/713705
[4] A. Agelastos, B. Allan, J. Brandt, P. Cassella, J. Enos, J. Fullop, A. Gentile, S. Monk, N. Naksinehaboon, J. Ogden, M. Rajan, M. Showerman, J. Stevenson, N. Taerat, and T. Tucker, "The lightweight distributed metric service: A scalable infrastructure for continuous monitoring of large scale computing systems and applications," *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, vol. 2015, pp. 154–165, 01 2015.
[5] O. H. Mondragon, P. G. Bridges, S. Levy, K. B. Ferreira, and P. Widener, "Understanding performance interference in next-generation hpc systems," in *Conference: 2016 ACM/IEEE Conference on Supercomputing (SC'16)*, November 2016.