A Machine Learning Approach to Understanding HPC Application Performance Variation

Burak Aksar[1], Benjamin Schwaller[2], Omar Aaziz[2], Emre Ates[1], Jim Brandt[2], Ayse K. Coskun[1], Manuel Egele[1], Vitus Leung[2]
[1]Electrical and Computer Engineering Department, Boston University
baksar|ates|acoskun|megele@bu.edu
[2]Sandia National Laboratories
bschwal|oaaziz|brandt|vjleung@sandia.gov

KEYWORDS
HPC, machine learning, performance variation, LSTM

1 INTRODUCTION

Performance anomalies in High-Performance Computing (HPC) systems can be defined as an issue which leads to sub-optimal execution time but does not affect the correctness of the running application. These anomalies are difficult to detect because often a "healthy system" is vaguely defined and the ground truth for how a system should be operating is evasive. As we move to exascale, however, detection of performance anomalies will become increasingly important with the increase in size and complexity of systems. Application performance can vary by 100% or more during runtime[1, 2]. These variations affect HPC system efficiency and therefore limit the amount of scientific work that can be achieved. However, understanding anomalies in an HPC system is not a straightforward task. There are very few accepted ways of detecting anomalies in the literature and there are no published and labeled sets of anomalous HPC behavior. HPC systems generate metadata that could be used to signal performance anomalies, however systems can generate thousands of metadata features per second and so data collection and analysis can be difficult. In this research, we develop a suite of applications that represent HPC workloads and use data from a lightweight metric collection service to train machine learning models to predict future behavior of metrics. In the future, this work will be used to predict anomalous runs in compute nodes and determine some root causes of performance issues to help improve the efficiency of HPC system administrators and users.

2 METHODOLOGY

2.1 Canary Suite

We create a "canary" suite of applications that, if performing strangely, could warn of system issues. These applications collectively exhibit behaviors similar to a wide-range of real HPC workloads and they are chosen by talking to system administrators and HPC users. The suite includes HACC, HPCG, HPL, LAMMPS, miniAMR, Nekbone, and QMCPACK. The suite is run on three different HPC systems: Sandia's Astra ARM Supercomputer, NERSC's Cori Cray supercomputer, and NSCA's Blue Waters Cray Supercomputer. Astra is a 2,592 node supercomputer with ARM-based Cavium Thunder-X2 processors. Astra is the world's fastest Arm-based supercomputer according to the TOP500 list, the supercomputer industry's standard. Cori is a Cray XC40 computer with 2,388 Intel Xeon"Haswell" processor nodes and 9,688 Intel Xeon Phi"Knight's Landing" nodes. We use only the Haswell nodes. Blue Waters is a Cray XE/XK hybrid machine composed of 26,864 AMD 6276"Interlagos" processors.

We are running the suite periodically over the course of several months. Runs occur on 64 nodes without multithreading with binaries which run for 15-20 minutes with a maximum time of 30 minutes. This size was chosen so that we could collect data from a sufficiently large portion of the system while also being able to easily schedule our jobs on these busy machines. Configuration parameters for each application are explained in the artifact description appendix. Our goal for this suite is not to have the most optimized applications, although we do try to use the best optimization flags and suggested compilers for each machine, but rather to have static binaries to run consistently over the course of our experiment.

Occasionally, we also inject synthetic anomalies which run alongside the application using Boston University's HPC Performance Anomaly Suite (HPAS)[3]. HPAS is designed to provide researchers with a foundation for reproducible anomaly injection and has several types of anomalies to stress different subsystems of the HPC architecture. These runs will be labeled anomalous for our machine learning models in future work along with applications which have runtime outliers.

2.2 System Data Collection

Application monitoring data is collected with a tool called Lightweight Distributed Metric Service (LDMS)[4]. LDMS is able to collect thousands of system features and hardware counters at sub-second intervals with negligible overhead. These features are collected in sets, called metric sets, and include memory usage, kernel/system statistics, and Performance Application Progress Interface (PAPI) counters. In our research we collect over 1200 metrics at 1 Hz for each application run.

2.3 Data Analysis & Model Training

We perform many steps in our data analysis routine to produce our models. Our main goal is to forecast performance metric behavior to characterize if the run is anomalous or not. First, metric sets from PAPI counters, meminfo, vmstat and procstat are concatenated and metrics with zero variance are dropped. We determine the absolute value of the cross-correlation for the metric set and remove metrics that are correlated more than 95% with other metrics. We take the

absolute value because if two metrics are highly inversely correlated, such as memory free and memory active, we only need one to encapsulate that information. Each metric is tested with"Dickey-Fuller Test" to understand whether metric data is stationary or not. For non-stationary metric data, we used differencing method to transform non-stationary data into stationary data. We use Long-Short Term Memory (LSTM) networks for our machine learning models. LTSM is used because it is powerful at extracting patterns from long time sequences even with multiple input features. Figure 1 illustrates an example of LSTM framework.

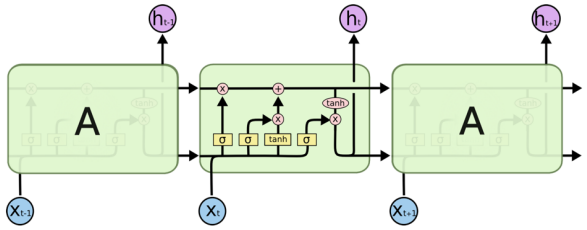

Figure 1: LSTM model for time series input[5]

In the training phase, we form a 3D input vector that LSTM networks accept. The vector's dimensions are samples, timesteps and features. We use rolling windows for each time point with a window size of 30 data points. 80% of our dataset is used for training and the remaining part as test set. We create a sequential model which has the following order: a LSTM layer with 100 units, a dropout layer with the dropout rate 0.1 and a densely-connected Neural Network layer with 1 unit. The model is trained with two different approaches. In the first approach, we train the model on specific metrics that explain more than 90% variance in the dataset and are determined by applying PCA. In the second approach, we combine all metrics to train a model. To combine all metrics in a format that LSTM accepts, for each time step, every metric and its historic values are concatenated in the same row.

3 RESULTS

Our application suite is still periodically running and gathering data on the supercomputers listed above and will continue to do so for several months. Initial results from our small datasets show that our all-metrics model predictions for the LAMMPS application closely match the actual results as seen in Figure 2.

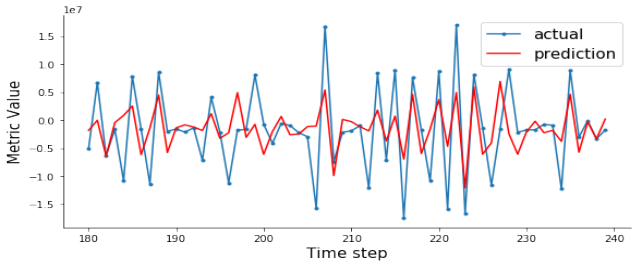

Figure 2: Metric forecasting with the all-metric model

As seen in Figure 3, individual-metric model follows the general curve of the metric but do not capture the spikes. Further data collection and analysis could improve these models and will reveal insights about intra-application and intra-architecture performance variation. We also showed that using our simple dimensionality techniques was effective at selecting relevant system metadata in Figure 4. We reduced our dataset from over 1200 features to about 150 by removing zero variance and highly correlated features. This technique could be used for selective data collection in the future.

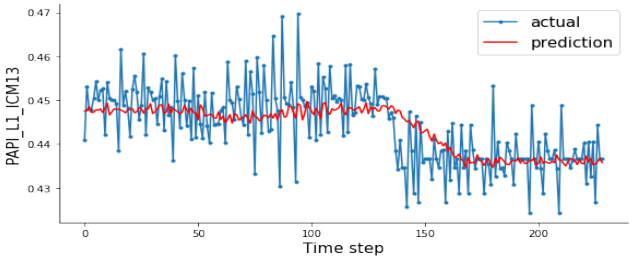

Figure 3: Instruction cache miss forecasting with the individual-metric model

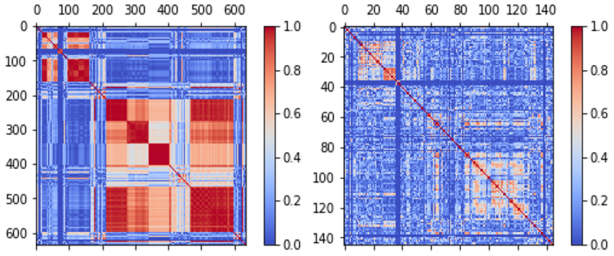

Figure 4: Metric set cross-correlation heatmap before and after removing highly correlated features; axes indicate the ID of the metric

4 CONCLUSION AND FUTURE WORK

In this research we run a large set of HPC applications with high-fidelity instrumentation that were run on a variety of high-end supercomputers. We trained machine learning models on the data to predict system behavior and saw success when training over all metrics. We have several tasks for our future work. We would like to improve our model considering scalability in different systems using more data. From this data we would like to examine intra-architecture and intra-application similarities of anomalies which might shed light onto the root causes of performance variation when combined with system logs. We could add new applications to our suite, such as SPARTA, to mimic more aspects of HPC workloads. Finally, we would like to generate a system to collect application input identifiers from users and get their direct feedback on job quality to create an additional labeling method.

5 ACKNOWLEDGMENTS

Part of this work was done under a joint-contact between Sandia National Laboratories and PeacLab at Boston University. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

REFERENCES

[1] Bhatele, A., Mohror, K., Langer, S.H., Isaacs, K.E.: There goes the neighborhood: Performance degradation due to nearby jobs. In: SC. pp. 41:1 - 41:12 (Nov 2013)

[2] Leung, V.J., Phillips, C.A., Bender, M.A., Bunde, D.P.: Algorithmic support for commodity-based parallel computing systems. Tech. Rep. SAND2003-3702, Sandia National Laboratories (2003)

[3] Ates, E., Zhang, Y., Aksar, B., Brandt, J., Leung, V. J., Egele, M., & Coskun, A. K. (2019). HPAS: An HPC Performance Anomaly Suite for Reproducing Performance Variations. 10. To appear in International Conference on Parallel Processing (ICPP), 2019. https://github.com/peaclab/HPAS.git

[4] Agelastos, Anthony, et al. "The lightweight distributed metric service: a scalable infrastructure for continuous monitoring of large scale computing systems and applications." SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2014.

[5] Understanding LSTM Networks. (n.d.). Retrieved from http://colah.github.io/posts/2015-08-Understanding-LSTMs