Detecting and Identifying Applications by Job Signatures in HPC Systems

Jie Li
jie.li@ttu.edu
Texas Tech University
Lubbock, Texas, USA

Brandon Cook
Lawrence Berkeley National Laboratory
Berkeley, California, USA
bgcook@lbl.gov

Yong Chen
Texas Tech University
Lubbock, Texas, USA
yong.chen@ttu.edu

ABSTRACT

Knowing the applications of jobs running in High-Performance Computing (HPC) systems is invaluable for administrators. This research aims to detect and identify applications through job signatures built upon monitoring traces obtained from the LDMS monitoring infrastructure on Cori. By constructing job signatures and applying machine learning models on them, we will be able to detect and identify job applications without user intervention. In addition to application names, job signatures offer the potential to study workloads by computation motif.

1 INTRODUCTION

As HPC systems are entering the exaFLOP era, the scale and complexity of HPC systems have increased significantly over the past few years. Administrators need to understand not only how the hardware system is performing, but also the typical applications that use the system. In addition, resource contention and energy consumption increase with the computation capability. HPC administrators and researchers need to understand the characteristics of running applications and design better resource-aware scheduling policies to improve system efficiency. Moreover, unauthorized applications, such as bit-coin mining programs, could take advantages of the high computing capability, consuming computing hours that supposed be used for scientific discoveries. Therefore, knowing which applications are running will help administrators to ban these malware in a proactive manner.

To address these challenges, it is necessary to detect applications and develop management strategies based on knowledge of the applications. However, this is a no-trivial task if users do not specify the name of the application in their job submission scripts. Earlier works, such as presented in [4], explored identifying applications based on binaries static analysis. Unfortunately, these approaches do not perform well if the same application is compiled by different compiler toolchains or optimization levels. Moreover, obtaining binaries requires additional effort to collect and manage binaries data, which is not always possibles for HPC centers.

In this research, we propose an approach to detect and identify applications through job signatures. Job signatures are patterns within performance metrics of a job collected from existing monitoring frameworks. Specifically, we exploit monitoring metrics collected from LDMS [1] to build job signatures by extracting statistical features from time-series data. Then, we explore a classification algorithm and evaluate its performances in classifying job signatures.

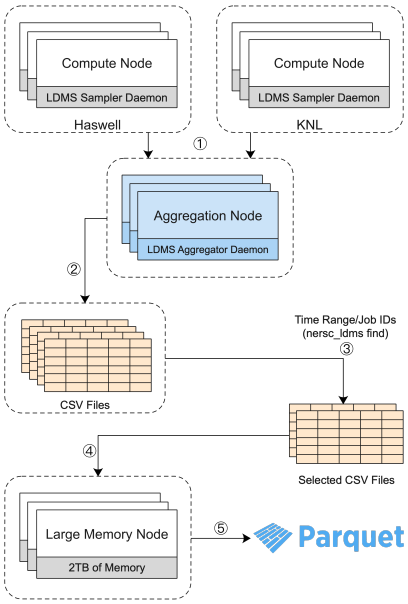

Figure 1: The workflow of LDMS at NERSC

2 BACKGROUND

The monitoring metrics are collected through *LDMS* deployed on the Cori system at NERSC. LDMS stands for lightweight Distributed metric Service, it is a low-overhead, low-latency framework for collecting, transferring, and storing metric data[1]. Note that we do not consider the GPU metrics in this study since LDMS metrics for GPU nodes were not available at the time of submitting this poster. The workflow of LDMS at NERSC is shown in Figure 1. ① LDMS samplers on Haswell and KNL nodes collect in-band metrics and send data to aggregation nodes; ② the raw monitoring data are saved as large CSV files; ③ The NERSC_LDMS tool [3] can find corresponding CSV files by time range or job IDs; ④ the selected CSV files are then submited to large memory nodes for postprocessing; ⑤ The post-processed data (job-level metrics) are saved in parquet format.

3 METHODOLOGY

3.1 Data Acquisition

For this study, we generate the dataset by following the steps below. We utilize the job metadata stored in a MySQL database and select

job metadata collected in April and May 2021. The job metadata has application names labeled. Note that these labeled application names are derived from job submission scripts and not all names are descriptive. We select those names that have explicit meaning and then use job IDs of the selected applications with the NERSC_LDMS tool [3] to collect the corresponding monitoring metrics. These metrics are post-processed and the raw metrics, sampled metrics, and extracted features are saved in an HDF file for each job. The total number of job samples is over 18,000.

3.2 Building Job Signatures

Among the available metrics, currently, we only use power consumption (*power(W)*), Instruction Per Cycle (*IPC*), and memory usage (*MemUsed*), as shown in table 1. These three metrics are applied to nine statistical functions (median, stand deviation, skewness, kurtosis, percentils (5, 25, 50, 75, 95)) to generate features, which form the job signature.

Table 1: Metrics and Samplers

Derived Metric	Raw Metric	Sampler
power(W)	power(W)	cray_aries_sampler
IPC	PAPI_TOT_INS	syspapi_Hsw64
	PAPI_TOT_CYC	syspapi_Knl272
MemUsed	MemTotal	meminfo
	MemFree	

4 MACHINE LEARNING MODEL

To detect and identify applications, we train a machine learning model using the labelled dataset. Specifically, we use *Random Forest* [6] as the classifier; the criterion parameter is set to "entropy"; the rest of the parameters are left unchanged.

Figure 2 shows the precision-recall curve; it plots the precision (y-axis) and the recall (x-axis) of application classification for different thresholds. The random forest classifier performs the best for the WRF application and the worst for the NWCHEM application.

To explain the classification model, we use the SHAP [5] summary plot to analyze the feature importance with feature effects. From Figure 3, we can see that the standard deviation of memory used contributes the most to the prediction of the model, followed by minimum memory used and minimum instruction per cycle. In addition, a high value of standard memory used has a negative impact on the prediction value.

5 DISCUSSION

In this study, we explored the use of monitoring metrics for jobs to build job signatures through which we would be able to detect and identify job applications without user intervention. We explored the random forest algorithm; even though it has been showed in other similar studies that random forest outperforms decision trees and support vector machine classifier for similar problems [2], the classification performance in our case is not always good for certain applications.

In our future work, we will add more statistical features from the available metrics to improve the classification performance and compare the performance of different classifiers. Also, it is worth exploring using the entire monitoring traces as a signature to classify time series data (represented by images as shown in Figure 4) with deep learning. Another interesting topic we would like to explore is whether we can achieve on-time application detection, i.e., detect and identify job applications while the job is still running.

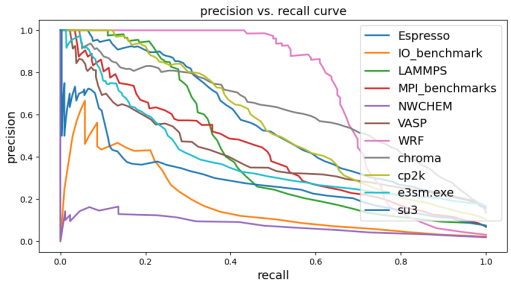

Figure 2: Precision-Recall Plot of Random Forest

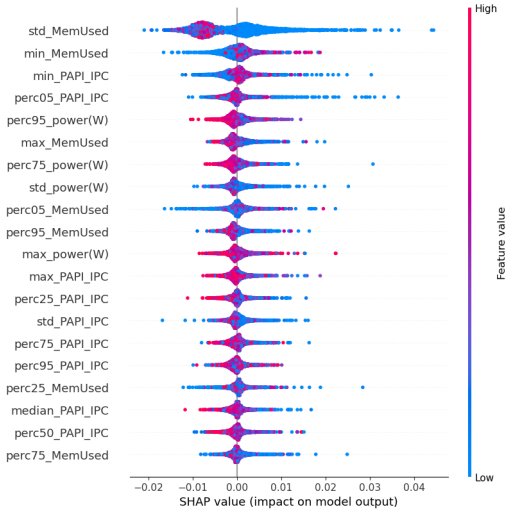

Figure 3: Shap Summary Plot of Random Forest

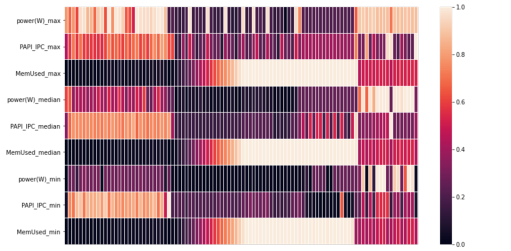

Figure 4: Image Representation of a Job Signature

6 ACKNOWLEDGMENTS

This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231.

REFERENCES

[1] Anthony Agelastos, Benjamin Allan, Jim Brandt, Paul Cassella, Jeremy Enos, Joshi Fullop, Ann Gentile, Steve Monk, Nichamon Naksinehaboon, Jeff Ogden, et al. 2014. The lightweight distributed metric service: a scalable infrastructure for continuous monitoring of large scale computing systems and applications. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 154–165.

[2] Emre Ates, Ozan Tuncer, Ata Turk, Vitus J Leung, Jim Brandt, Manuel Egele, and Ayse K Coskun. 2018. Taxonomist: Application detection through rich monitoring data. In *European Conference on Parallel Processing*. Springer, 92–105.

[3] Brandon Cook. 2021. *NERSC LDMS Pipeline*. Retrieved August, 2021 from https://software.nersc.gov/ldms/nersc-ldms

[4] Manuel Egele, Maverick Woo, Peter Chapman, and David Brumley. 2014. Blanket execution: Dynamic similarity testing for program binaries and components. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 303–317.

[5] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*. 4768–4777.

[6] Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston. 2003. Random forest: a classification and regression tool for compound classification and QSAR modeling. *Journal of chemical information and computer sciences* 43, 6 (2003), 1947–1958.