Towards Practical Machine Learning Frameworks for Performance Diagnostics in Supercomputers

Burak Aksar
baksar@bu.edu
Boston University
Boston, MA, USA

Efe Sencan
esencan@bu.edu
Boston University
Boston, MA, USA

Benjamin Schwaller
bschwal@sandia.gov
Sandia National Laboratories
Albuquerque, NM, USA

Vitus J. Leung
vjleung@sandia.gov
Sandia National Laboratories
Albuquerque, NM, USA

Jim Brandt
brandt@sandia.gov
Sandia National Laboratories
Albuquerque, NM, USA

Brian Kulis
bkulis@bu.edu
Boston University
Boston, MA, USA

Manuel Egele
megele@bu.edu
Boston University
Boston, MA, USA

Ayse K. Coskun
acoskun@bu.edu
Boston University
Boston, MA, USA

ABSTRACT

Supercomputers are highly sophisticated computing systems designed to handle complex and computationally intensive tasks. Despite their tremendous efficiency, performance problems still arise due to various factors, such as load imbalance, network congestion, and software-related issues. Monitoring frameworks are commonly used to collect telemetry data, which helps identify potential issues before they become critical or debug problems. However, telemetry analytics is essentially a big data problem that is becoming increasingly difficult to manage due to terabytes of telemetry data collected daily. Owing to the limitations of manual analysis, recent analytics frameworks leverage automated machine learning (ML)-based frameworks to identify patterns and anomalies in this data, enabling system administrators and users to take appropriate action towards resolving performance problems quickly.

This paper explores the benefits and challenges of ML-based frameworks that automate performance diagnostics, particularly focusing on labeled training data requirements and deployment challenges. We argue that ML-based frameworks can achieve desirable performance diagnosis results while reducing the need for large labeled data sets, and we demonstrate successful prototypes that are suitable for rapid deployment on real-world systems.

CCS CONCEPTS

• **General and reference → Performance**; • **Computing methodologies → Learning settings**.

KEYWORDS

High-performance Computing, Machine Learning, Deployment

ACM Reference Format:
Burak Aksar, Efe Sencan, Benjamin Schwaller, Vitus J. Leung, Jim Brandt, Brian Kulis, Manuel Egele, and Ayse K. Coskun. 2023. Towards Practical Machine Learning Frameworks for Performance Diagnostics in Supercomputers. In *Proceedings of the First Workshop on AI for Systems (AI4Sys '23), June 20, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3588982.3603609

1 INTRODUCTION

Supercomputers are among the most powerful computing systems in the world [20]. The speed and performance of supercomputers are truly impressive, with some of the fastest machines in the world capable of executing quintillions of calculations per second [26]. The architecture of a supercomputer is highly sophisticated, consisting of thousands of interconnected processing nodes that work together to provide massive parallelism and processing power [20, 24, 32]. This architecture enables supercomputers to execute thousands of tasks simultaneously for processing large datasets and performing complex simulations. These capabilities have made them invaluable in scientific research, engineering applications, and many other fields. They have also been applied to solve some of the world's most challenging problems in areas such as climate modeling, drug discovery, and astrophysics, as well as in commercial applications like financial modeling and risk analysis. However, despite the tremendous power of supercomputers, they are not immune to performance problems. Detecting and diagnosing these problems have become increasingly critical as supercomputers have grown more complex, powerful, and expensive (i.e., in terms of operational and energy costs).

Performance problems can arise from various sources, including load imbalances, scheduling problems, network congestion, hardware faults, software inefficiencies, and other factors. Load imbalances and scheduling problems occur when the workload distribution across the system's processors or cores is uneven, leading to some processors being overutilized while others are underutilized [19]. This type of imbalance can result in reduced efficiency and increased resource consumption, ultimately leading to lower overall performance [18]. Network congestion is another common problem in supercomputers [15, 37]. These machines rely on complex interconnect networks to facilitate communication between processors and cores, and if the network becomes congested, it can lead to performance degradation. Software-related problems are also a common source of performance problems in supercomputers [2, 17]. Poorly optimized code, for example, can result in excessive memory usage, increased communication overhead, and inefficient use of system resources. Memory management problems, such as memory leaks or excessive swapping, also often lead to performance degradation.

Detecting and diagnosing performance problems in supercomputers requires advanced analytics frameworks that can ideally provide near-real-time insights into system behavior along with automated alerts. These frameworks can help identify bottlenecks and other performance problems in a fully- or semi-automated way, allowing system administrators to take corrective action quickly. The foundational element behind these frameworks is monitoring frameworks, which have become the de-facto standard [1, 11, 28]. In practice, a supercomputer commonly employs one or more monitoring frameworks to collect information on the resource utilization characteristics of the whole system. Monitoring frameworks often gather telemetry data in the form of multivariate time series through performance counters across different subsystems (e.g., memory, CPU, network), system logs, and traces. This proactive monitoring approach can help minimize system downtime and improve overall system efficiency. For example, network traffic, processor usage, memory usage, and other key metrics can be leveraged to identify potential issues before they become critical.

Although monitoring frameworks have advanced significantly in recent years (e.g., low memory overhead, negligible latency, etc.), the big data problem of monitoring is becoming increasingly difficult to manage. Supercomputers generate vast amounts of telemetry data, accumulating billions of data points over time, leading to terabytes of data generated daily. Analyzing this data is a daunting task, requiring sophisticated data processing and analysis techniques. Fortunately, researchers have developed advanced analytics frameworks to help system administrators extract meaningful insights from available data [5, 7, 13, 23, 29, 33, 36]. These frameworks leverage machine learning (ML) algorithms and other advanced techniques to identify patterns and anomalies, enabling system administrators to take appropriate action quickly. For the rest of the paper, we refer to these frameworks as *ML-based frameworks*.

This position paper offers an overview of various recent ML-based frameworks, specifically focusing on their labeled training data requirements and suitability for deployment. We examine open problems and potential next steps based advantages and disadvantages of the available ML-based frameworks. Our primary objective is to argue that a desirable performance diagnosis score can be achieved with limited reliance on large labeled data sets using ML-based frameworks. Furthermore, we present successful prototypes that are suitable for rapid deployment on real-world systems.

Section 2 provides an overview of telemetry data-based analytics in supercomputers. Section 3 discusses success stories and challenges with ML-based frameworks, including the deployment angle at scale. Finally, Section 4 summarizes the key findings and open problems for leveraging ML-based frameworks in supercomputers.

2 BACKGROUND

Over the last decade, there have been significant advancements in high-performance computing (HPC) systems, particularly in processor performance, networking technologies, storage capabilities, and software tools. These improvements have enabled these systems to process larger workloads and perform computations faster. However, despite these advancements, HPC systems still encounter challenges in maintaining their efficiency and reliability, primarily due to performance anomalies. Performance anomalies can occur due to various factors, such as hardware problems, software bugs, network congestion, and workload imbalances. They can significantly affect the efficiency and reliability of the system, leading to longer processing times, increased energy consumption, and variations in the application's performance. For example, performance anomalies can cause an eight times increase in application running times in production systems [38], impacting efficiency and cost in major ways.

Currently, a popular procedure for detecting and diagnosing issues in large-scale systems involves system administrators manually examining logs and telemetry data and utilizing their domain-specific knowledge to identify problems. However, the complexity and scale of HPC systems make manual analysis extremely challenging to detect or diagnose these anomalies at scale. Traditional rule-based anomaly detection methods are also limited because of several reasons. Firstly, creating these rules is a manual and time-consuming process that requires domain-specific expertise. As systems become larger and more complex, creating rules that can accurately diagnose issues becomes increasingly challenging. Secondly, these rules are not generalizable, as they can only detect issues that have been explicitly defined. As new issues arise, new rules must be created, which can become unmanageable as the system grows. Thirdly, transferring knowledge across different systems is difficult, as the rules that work for one system may not apply to another. This means that the expertise required for diagnostics must be built up separately for each system, which can be inefficient and costly.

Fortunately, ML offers a promising solution to this problem. Unlike traditional rule-based methods that rely on fixed thresholds and predefined rules based on domain expertise, ML-based frameworks can learn and adapt from historical data to identify and detect anomalies automatically. This approach offers several benefits, including runtime anomaly detection, the ability to adjust to new patterns in the data, and improved accuracy over time. Over the past decade, there has been a notable increase in the development of ML-based frameworks designed to detect performance anomalies in HPC systems. These frameworks can be categorized into three main groups: supervised, semi-supervised, and unsupervised, based on the amount of labeled data used during the model training stage.

Researchers have designed supervised ML-based frameworks to address the challenge of detecting and diagnosing performance anomalies in HPC systems [7, 23, 35]. These frameworks utilize a range of algorithms such as support vector machines [30], k-nearest neighbors [21], random forest [31], and Bayesian classifier [25]. These frameworks aim to detect anomalies or identify the type of anomalies (i.e., diagnosis). This is achieved through the use of labeled datasets that contain healthy and anomalous application runs.

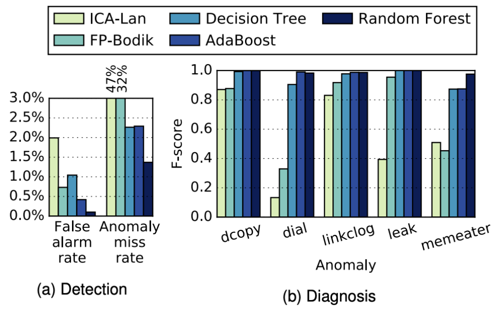

Figure 1: Anomaly detection and diagnosis performance for Random Forest and several baselines using 5-fold stratified cross-validation. Random Forest correctly detects 98% of the anomalies with a 0.08% false alarm rate.

Although supervised ML-based frameworks have shown promising results in detecting performance anomalies, they have a significant limitation of requiring a large amount of labeled data, which can be time-consuming and expensive. In fact, telemetry data collected from HPC systems is often largely unlabeled. Furthermore, the high-dimensional and voluminous nature of time series data in such systems makes it impractical to label each sample manually. As a result, researchers have designed alternative approaches, such as semi-supervised and unsupervised ML-based frameworks.

Semi-supervised ML-based frameworks aim to achieve target anomaly detection or diagnosis performance using much fewer labeled samples than fully supervised methods, thereby reducing the cost and effort required for manual labeling. These methods typically involve training an ML model on a small subset of labeled data and leveraging additional information from the unlabeled data. One such popular approach uses autoencoders [10], which are neural networks trained to reconstruct input data. This approach has been used for detecting anomalies in HPC systems by learning the normal behavior of compute nodes and detecting anomalies based on the reconstruction error [12]. Similarly, Aksar et al. [5] introduce an autoencoder-based semi-supervised framework for diagnosing anomalies in production HPC systems. Their approach involves learning the characteristics of previously encountered performance anomalies in an unsupervised manner, followed by using supervised classifiers to diagnose anomalies on compute nodes. Other semi-supervised methods in the HPC domain for detecting performance anomalies include clustering-based approaches, which group similar time series samples. For each cluster, the labeled samples are used to train a supervised classifier and classify the remaining unlabeled samples in that cluster. Another semi-supervised approach for diagnosing anomalies is based on active learning [4], which involves selecting the most valuable unlabeled samples for labeling by a human expert and using these samples to train a supervised classifier. By leveraging labeled and unlabeled data, semi-supervised methods have shown great potential in improving the accuracy and efficiency of anomaly detection and diagnosis in HPC systems. However, labeling a relatively smaller number of samples (i.e., compared to data requirements of fully supervised methods) can still be time-consuming and challenging in practice, which motivates the design of unsupervised ML-based frameworks.

In the unsupervised setting, an ML model is trained without labeled data. The goal is to discover hidden patterns and relationships in the data without having the label information. Traditional unsupervised learning methods include K-means clustering [22], local outlier factor [6], and kernel density estimation [14]. These methods aim to identify the underlying structure or pattern in the data and detect anomalies based on deviations from that structure or pattern. However, their performance can be limited when data is complex or similar in shape, making it difficult to distinguish normal patterns from anomalous ones. Furthermore, distance-based clustering methods may not be able to handle high-dimensional data or outliers. In contrast, researchers often use deep neural networks in state-of-the-art unsupervised frameworks to detect anomalies in multivariate time series data. These methods include adversarially trained autoencoders [9] for isolating anomalies, graph neural networks for identifying anomalies via attention-based forecasting and deviation scoring [16], deep transformer networks that utilize attention-based sequence encoders [34], and recurrent autoencoder models that capture the healthy time-series characteristics while capturing temporal dependencies in the data [29]. Despite unsupervised frameworks excelling in detecting anomalies, these frameworks lack the ability to diagnose anomaly types as they do not utilize known anomalous labels.

3 SUCCESS STORIES AND CHALLENGES WITH ML-BASED FRAMEWORKS

In this section, we present ML-based framework success stories and challenges. We discuss supervised and semi-supervised frameworks for anomaly detection, as well as unsupervised frameworks. We also discuss the challenges of deploying ML-based frameworks at scale in HPC systems. Despite these challenges, ML-based frameworks improve HPC system performance.

3.1 Evaluation Metrics

To evaluate the performance of ML-based frameworks, we use the F1-score as a primary evaluation metric since it is commonly adopted to assess the classification performance of ML models, especially in scenarios with imbalanced datasets. The F1-score is particularly well-suited for evaluating anomaly diagnosis frameworks due to the rarity of anomalies compared to the abundance of healthy samples. The suitability of the F1-score extends beyond binary classification tasks and also applies to multi-class settings, where anomalies can be represented by one or more classes. By considering both precision and recall, the F1 score provides a balanced assessment of the model's ability to accurately classify anomalies in such imbalanced datasets.

Another popular metric is Area Under the Curve (AUC) [27], which is generally adopted for binary classification tasks such as anomaly detection. It measures the overall performance of a model by assessing the trade-off between a true positive rate and a false positive rate at various classification thresholds. It provides a comprehensive summary of the model's classification power across different thresholds, regardless of the threshold chosen for the final classification. However, the AUC may not be a suitable metric in scenarios where the class distribution is heavily imbalanced. In such cases, where the rare anomalies are of primary interest, a

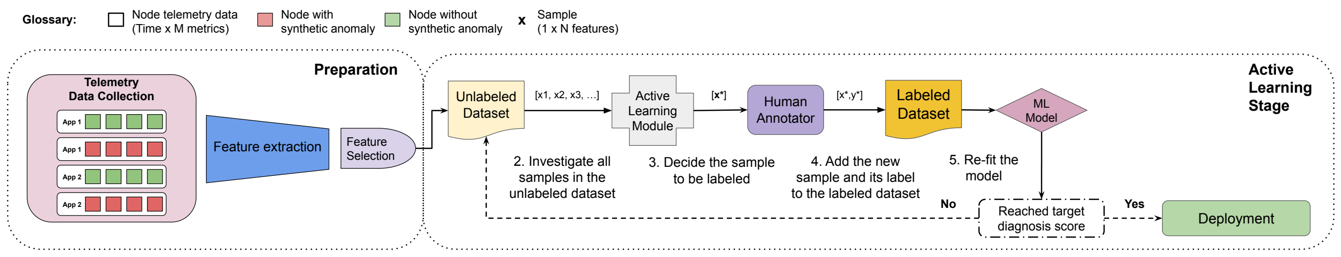
Glossary:
Node telemetry data (Time x M metrics)
Node with synthetic anomaly
Node without synthetic anomaly
x Sample (1 x N features)
Preparation
Telemetry Data Collection
App 1
App 2
Feature extraction
Feature Selection
Active Learning Stage
Unlabeled Dataset
[x1, x2, x3, ...]
Active Learning Module
[x*]
Human Annotator
[x*,y*]
Labeled Dataset
ML Model
2. Investigate all samples in the unlabeled dataset
3. Decide the sample to be labeled
4. Add the new sample and its label to the labeled dataset
5. Re-fit the model
No
Reached target diagnosis score
Yes
Deployment

Figure 2: During the preparation stage, we collect telemetry data from compute nodes while running applications with and without synthetic anomalies. In the active learning stage, ALBADross decides which samples should be labeled from the unlabeled dataset and re-trains the model until a satisfactory diagnosis score is reached.

small number of correct or incorrect predictions can result in a large change in the AUC score.

In the anomaly diagnosis domain, false alarm and miss rates are other important metrics to consider. False positives are generally considered more critical than false negatives. The main reason is that false positives typically require human intervention to investigate and address these alarms, incurring additional costs and efforts. On the other hand, the anomaly miss rate refers to a case where true anomalies are not detected. While minimizing the anomaly miss rate is important, it is generally considered less critical than false positives since missed anomalies may still be identified through rule-based methods or manual intervention.

3.2 Supervised Frameworks

Recently, researchers and practitioners have given growing attention to utilizing supervised anomaly detection and diagnosis frameworks for detecting anomalies in HPC systems. Such frameworks require a large labeled dataset consisting of anomalous and healthy samples. However, labeled data is often scarce in real-world scenarios, particularly anomalies, which are usually rare occurrences. One way to solve this problem is to generate labeled data using synthetic performance anomalies. High-performance anomaly suite (HPAS) is one example that generates realistic synthetic anomalies and can emulate different types of anomalies impacting the main subsystems of HPC systems, including CPU, cache, memory, network, and shared storage [8]. Another popular open-source example is "Global Performance and Congestion Network Tests", which specifically target network-related performance variations [15].

We design a framework that can detect and diagnose anomaly types at runtime [35], which achieves state-of-the-art performance. As a first step, we collect historical telemetry data collected from application runs with and without synthetic anomalies generated using HPAS. Then, we leverage statistical feature extraction and selection steps to reduce the size and dimensionality of the telemetry data. In the last step, we train ML Models (random forest, decision tree, and AdaBoost) to learn the characteristics of previously observed anomalies. During runtime, we process telemetry data collected from individual nodes using a sliding window technique and extract statistical features. The trained model provides a prediction for each window; however, to decide whether a compute node is anomalous, we wait unless anomaly prediction is valid for a certain amount of consecutive sliding windows. Figure 1 highlights one of the key results we achieve. Random forest correctly identifies 98% of the anomalies while leading to only 0.08% false anomaly alarms. While supervised frameworks can achieve almost perfect anomaly diagnosis scores when provided with representative samples of both healthy and anomalous data of sufficient size, acquiring a large labeled dataset may not be practical due to the high cost and time-consuming nature of manual diagnosis required for labeling.

3.3 Relaxing the Label Requirement

To address the challenge of labeling a large number of labeled samples, we approach the problem from two different angles. The first angle focuses on a scenario where there is an abundance of healthy samples, but only a few anomalous samples are available. Our recent work introduces Proctor, a semi-supervised anomaly diagnosis framework that effectively utilizes labeled and unlabeled samples [5]. Proctor performs better than a fully supervised baseline, especially when the number of labeled samples is limited. Proctor has an autoencoder-based unsupervised pretraining stage and a supervised classification layer that diagnoses performance anomalies on compute nodes. The autoencoder model is first trained on a large amount of unlabeled data (anomalous and health samples), followed by a finetuning stage with a few labeled samples. The finetuning stage helps the model use a few labeled samples to adjust its learning process. Proctor outperforms a fully-supervised baseline model [35] by 4.5% on average (and up to 11%) in terms of macro average F1-score when the labeled data percentage is less than 5%.

The second angle frames the problem in a different setting where a few anomalous and a large set of healthy samples are available; however, it also assumes a human annotator is available to provide the label of a selected sample upon request. In this setting, we aim to determine the minimum number of samples, which maximizes the model performance, to be labeled among thousands of unlabeled samples since admins or code developers have limited capacity. Our framework, ALBADRoss [4], trains a supervised model with initially available labeled samples and then utilizes active learning query strategies to determine which sample should be labeled among the thousands of unlabeled samples to achieve a desirable F1-score for anomaly diagnosis. Figure 2 shows the preparation and active learning stages. We evaluate ALBADRoss with a dataset collected from a production HPC system and achieve a 0.95 F1-score (the same score that a fully-supervised framework achieved) and near-zero false alarm rate using 28x fewer labeled samples.

One key takeaway is that semi-supervised frameworks can achieve competitive performance by leveraging both labeled and unlabeled samples, even when the number of labeled samples is limited. These frameworks help mitigate the need for extensive manual labeling efforts while still achieving accurate results.

3.4 Anomaly Detection without Supervision

Ideally, the goal would be to possess a readily available framework that can be immediately implemented in a new system. However, before we achieve this goal, we need to minimize the acquisition cost of an ML-based framework for a new system. One of the biggest obstacles to adopting an ML-based framework at scale is labeled data. Recent research [13, 29] adopts unsupervised ML models with a goal of zero-label or minimal supervision. In this context, we observe two main directions: partially and fully unsupervised. In the partially unsupervised direction, ML models are trained with only healthy samples to detect anomalies. One can argue that this requires knowledge of which samples are healthy, but ML models do not require label information during training, so we categorize them in the unsupervised setting. Borghesi et al. [13] design an autoencoder-based framework that learns the healthy characteristics of compute node telemetry data. The framework achieves 12% higher anomaly detection accuracy than baselines. In the fully unsupervised case, ML models are trained with the overwhelming majority of healthy samples and a limited number of anomalous samples, accurately representing a production system scenario. Molan et al. [29] design a different autoencoder framework, which extracts temporal characteristics of raw telemetry data. Based on the evaluation, the framework achieves a 0.767 AUC score, whereas the previous work [13] achieves a 0.747 AUC score in the same test dataset. Even though the performance gain is minimal, training models without supervision is an important step towards lowering the labeled data barrier and enabling the wide spread of ML-based frameworks in production HPC systems. Another direction to consider while moving to the minimal supervision era is **diagnosing** the anomaly type. In both frameworks, it is impossible to diagnose the anomaly type, which is crucial to understanding root causes and developing effective mitigation policies in the long run.

3.5 ML-based Frameworks at Scale

Even though many novel ML-based frameworks target production systems, the deployment aspect is often overlooked. In one of our earlier works [3], we design an end-to-end architecture to deploy the state-of-the-art fully supervised anomaly diagnosis framework [35] to a 1500-node production HPC system. To train the model, we collect telemetry data from application runs with and without synthetic anomalies. Using the best model, we achieve a 0.92 macro average F1-score in the anomaly diagnosis task. However, the biggest challenges we observe are compiling applications properly with different input decks (since it requires domain expertise) and verifying that synthetic anomalies create the desired impact on application runs. These tasks can be time-consuming, considering they are not the main focus of administrators, which leads to a difficult adoption process. Borghesi et al. [13] design a deployment scenario that includes a generic and a node-specific approach to reduce the adoption barrier from the data collection angle. While using a node-specific approach may be feasible for small computing clusters, it entails high training and maintenance costs, such as tuning hyperparameters or selecting a new detection threshold for each model. Since production systems generally have thousands to tens of thousands of compute nodes, a generic approach may be a more practical solution. However, this brings another challenge regarding the generalizability of the model. For example, node-specific models have an F1-score of 0.898; however, the generic model has a 0.726 F1-score, which reveals a significant performance drop for the only anomaly detection task. Considering the generic model is trained with telemetry data from multiple compute nodes, it is necessary to consider robust architectures or different training approaches. Currently, we are working on highly customizable yet simple software architecture, which supports various feature extraction and selection strategies as well as multiple ML models. Our primary objectives are facilitating easy integration with monitoring frameworks and expediting the deployment process. The architecture requires only a monitoring framework and a backend server. Using the designed architecture, we deploy a variational autoencoder-based framework to a production system with 1488 nodes. This generic model (i.e., not node-specific) achieves a macro average F1-score of 0.9 using only 16 healthy samples, corresponding to 4 jobs where each job runs on four compute nodes. Even though ML-based frameworks are gaining popularity, we need low-overhead and customizable software architectures that can simplify the integration with monitoring frameworks and expedite the deployment process in production systems.

4 OPEN PROBLEMS AND NEXT STEPS

This section discusses open problems and potential next steps to enhance the accuracy, efficiency, and usability of ML-based frameworks. Active learning-based frameworks reduce labeling efforts, but labeling high-dimensional telemetry data remains challenging. Additional efforts, especially automated approaches, are needed for efficient and accurate labeling. Another open problem is developing robust ML-based frameworks to facilitate deployment. Autoencoder-based frameworks work well when trained with only healthy samples but require extensive human monitoring to ensure the long-term stability of the system. Therefore, frameworks that can be trained with healthy and anomalous samples can facilitate easy deployment.

The above directions focus on data and software-related open problems. Designing and developing monitoring frameworks suitable for heterogeneous clusters (CPU and GPU) is necessary, considering the latest supercomputer architectures. Especially collecting telemetry data from GPU clusters with low overhead and fine granularity is critical to support large-scale heterogeneous systems.

ACKNOWLEDGMENTS

This work has been partially funded by Sandia National Laboratories. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and 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] 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 *Proc. of the Int. Conf. for High Performance Computing, Networking, Storage and Analysis (SC)*. 154–165.

[2] Anthony Agelastos, Benjamin Allan, Jim Brandt, Ann Gentile, Sophia Lefantzi, Steve Monk, Jeff Ogden, Mahesh Rajan, and Joel Stevenson. 2015. Toward rapid understanding of production HPC applications and systems. In *2015 IEEE International Conference on Cluster Computing*. IEEE, 464–473.

[3] Burak Aksar, Benjamin Schwaller, Omar Aaziz, Vitus J Leung, Jim Brandt, Manuel Egele, and Ayse K Coskun. 2021. E2EWatch: An End-to-End Anomaly Diagnosis Framework for Production HPC Systems. In *European Conference on Parallel Processing*. Springer, 70–85.

[4] Burak Aksar, Efe Sencan, Benjamin Schwaller, Omar Aaziz, Vitus J Leung, Jim Brandt, Brian Kulis, and Ayse K Coskun. 2022. ALBADross: Active Learning Based Anomaly Diagnosis for Production HPC Systems. In *2022 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 369–380.

[5] Burak Aksar, Yijia Zhang, Emre Ates, Benjamin Schwaller, Omar Aaziz, Vitus J Leung, Jim Brandt, Manuel Egele, and Ayse K Coskun. 2021. Proctor: A semi-supervised performance anomaly diagnosis framework for production hpc systems. In *High Performance Computing: 36th International Conference, ISC High Performance 2021, Virtual Event, June 24–July 2, 2021, Proceedings 36*. Springer, 195–214.

[6] Omar Alghushairy, Raed Alsini, Terence Soule, and Xiaogang Ma. 2020. A review of local outlier factor algorithms for outlier detection in big data streams. *Big Data and Cognitive Computing* 5, 1 (2020), 1.

[7] 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 *Euro-Par 2018: Parallel Processing: 24th International Conference on Parallel and Distributed Computing, Turin, Italy, August 27-31, 2018, Proceedings 24*. Springer, 92–105.

[8] Emre Ates, Yijia Zhang, Burak Aksar, Jim Brandt, Vitus J Leung, Manuel Egele, and Ayse K Coskun. 2019. HPAS: An HPC Performance Anomaly Suite for Reproducing Performance Variations. In *ACM Proceedings of the 48th International Conference on Parallel Processing*. 1–10.

[9] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. 2020. Usad: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3395–3404.

[10] Dor Bank, Noam Koenigstein, and Raja Giryes. 2020. Autoencoders. *arXiv preprint arXiv:2003.05991* (2020).

[11] Andrea Bartolini, Andrea Borghesi, Antonio Libri, Francesco Beneventi, Daniele Gregori, Simone Tinti, Cosimo Gianfreda, and Piero Altoè. 2018. The DAVIDE big-data-powered fine-grain power and performance monitoring support. In *Proceedings of the 15th ACM Int. Conf. on Computing Frontiers*. 303–308.

[12] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. 2019. A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems. *Engineering Applications of Artificial Intelligence* 85 (2019), 634–644.

[13] Andrea Borghesi, Antonio Libri, Luca Benini, and Andrea Bartolini. 2019. Online anomaly detection in hpc systems. In *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 229–233.

[14] Yen-Chi Chen. 2017. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology* 1, 1 (2017), 161–187.

[15] Sudheer Chunduri, Taylor Groves, Peter Mendygral, Brian Austin, Jacob Balma, Krishna Kandalla, Kalyan Kumaran, Glenn Lockwood, Scott Parker, Steven Warren, et al. 2019. Gpcnet: Designing a benchmark suite for inducing and measuring contention in hpc networks. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–33.

[16] Ailin Deng and Bryan Hooi. 2021. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4027–4035.

[17] Matthieu Dorier, Gabriel Antoniu, Rob Ross, Dries Kimpe, and Shadi Ibrahim. 2014. CALCioM: Mitigating I/O interference in HPC systems through cross-application coordination. In *2014 IEEE 28th international parallel and distributed processing symposium*. IEEE, 155–164.

[18] Yuping Fan, Zhiling Lan, Paul Rich, William Allcock, and Michael E Papka. 2022. Hybrid Workload Scheduling on HPC Systems. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 470–480.

[19] Yuping Fan, Zhiling Lan, Paul Rich, William E Allcock, Michael E Papka, Brian Austin, and David Paul. 2019. Scheduling beyond CPUs for HPC. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*. 97–108.

[20] Jiangang Gao, Fang Zheng, Fengbin Qi, Yajun Ding, Hongliang Li, Hongsheng Lu, Wangquan He, Hongmei Wei, Lifeng Jin, Xin Liu, et al. 2021. Sunway supercomputer architecture towards exascale computing: analysis and practice. *Science China Information Sciences* 64, 4 (2021), 141101.

[21] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. 2003. KNN model-based approach in classification. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*. Springer, 986–996.

[22] John A Hartigan and Manchek A Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)* 28, 1 (1979), 100–108.

[23] Jannis Klinkenberg, Christian Terboven, Stefan Lankes, and Matthias S Müller. 2017. Data mining-based analysis of HPC center operations. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 766–773.

[24] Peter M Kogge and William J Dally. 2022. Frontier vs the Exascale Report: Why so long? and Are We Really There Yet?. In *2022 IEEE/ACM International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. IEEE, 26–35.

[25] K Ming Leung et al. 2007. Naive bayesian classifier. *Polytechnic University Department of Computer Science/Finance and Risk Engineering* 2007 (2007), 123–156.

[26] Adam Mann. 2020. Nascent exascale supercomputers offer promise, present challenges. *Proceedings of the National Academy of Sciences* 117, 37 (2020), 22623–22625.

[27] Caren Marzban. 2004. The ROC curve and the area under it as performance measures. *Weather and Forecasting* 19, 6 (2004), 1106–1114.

[28] Matthew L Massie, Brent N Chun, and David E Culler. 2004. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Comput.* 30, 7 (2004), 817–840.

[29] Martin Molan, Andrea Borghesi, Daniele Cesarini, Luca Benini, and Andrea Bartolini. 2023. RUAD: Unsupervised anomaly detection in HPC systems. *Future Generation Computer Systems* 141 (2023), 542–554.

[30] William S Noble. 2006. What is a support vector machine? *Nature biotechnology* 24, 12 (2006), 1565–1567.

[31] Steven J Rigatti. 2017. Random forest. *Journal of Insurance Medicine* 47, 1 (2017), 31–39.

[32] David Schneider. 2022. The Exascale Era is Upon Us: The Frontier supercomputer may be the first to reach 1,000,000,000,000,000,000 operations per second. *IEEE Spectrum* 59, 1 (2022), 34–35.

[33] Denis Shaykhislamov and Vadim Voevodin. 2018. An approach for dynamic detection of inefficient supercomputer applications. *Procedia Computer Science* 136 (2018), 35–43.

[34] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. 2022. Tranad: Deep transformer networks for anomaly detection in multivariate time series data. *arXiv preprint arXiv:2201.07284* (2022).

[35] Ozan Tuncer, Emre Ates, Yijia Zhang, Ata Turk, Jim Brandt, Vitus J Leung, Manuel Egele, and Ayse K Coskun. 2018. Online diagnosis of performance variation in HPC systems using machine learning. *IEEE Transactions on Parallel and Distributed Systems* 30, 4 (2018), 883–896.

[36] Cong Xie, Wei Xu, and Klaus Mueller. 2018. A visual analytics framework for the detection of anomalous call stack trees in high performance computing applications. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 215–224.

[37] Yijia Zhang, Taylor Groves, Brandon Cook, Nicholas J. Wright, and Ayse K. Coskun. 2020. Quantifying the impact of network congestion on application performance and network metrics. In *2020 IEEE International Conference on Cluster Computing (CLUSTER)*. 162–168. https://doi.org/10.1109/CLUSTER49012.2020.00026

[38] Yijia Zhang, Taylor Groves, Brandon Cook, Nicholas J Wright, and Ayse K Coskun. 2020. Quantifying the impact of network congestion on application performance and network metrics. In *2020 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 162–168.