Workload Failure Prediction for Data Centers

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

Rui Wang
Texas Tech University, USA
rui.wang@ttuhsc.edu

Ghazanfar Ali
Texas Tech University, USA
Ghazanfar.Ali@ttu.edu

Tommy Dang
Texas Tech University, USA
tommy.dang@ttu.edu

Alan Sill
Texas Tech University, USA
alan.sill@ttu.edu

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

2023 IEEE 16th International Conference on Cloud Computing (CLOUD) | 979-8-3503-0481-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/CLOUD60044.2023.00064

***Abstract*—Failed workloads that consumed significant computational resources in time and space affect the efficiency of HPC data centers significantly and thus limit the amount of scientific work that can be achieved. While the computational power has increased significantly over the years, detection and prediction of workload failures have lagged far behind and will become increasingly critical as the system scale and complexity further increase. In this study, we analyze workload traces collected from a production cluster and train machine learning models on a large amount of data sets to predict workload failures. Our prediction models consist of a queue-time model that estimates the probability of workload failures before execution and a runtime model that predicts failures at runtime. Evaluation results show that the queue-time model and runtime model can predict workload failures with a maximum precision score of 90.61% and 97.75%, respectively. By integrating the runtime model with the job scheduler, it helps reduce CPU time, and memory usage by up to 16.7% and 14.53%, respectively.**

***Index Terms*—High-Performance Computing, Data Center, Failure Prediction, Predictive Analytic, Big Data, Machine Learning**

I. Introduction

The scale and complexity of many High-Performance Computing (HPC) data centers have significantly increased over the years. Meantime, the demand from the user community for computational and storage capability has considerably increased too. This combination of the increased scale of data centers and the size of workloads with different requirements and characteristics has resulted in growing node and workload failures, posing a threat to the reliability, availability, and scalability (RAS) of data centers.

Reactive strategies, such as Checkpoint/Restart (C/R) [1], [2], are conventional approaches for fault tolerance. As an example, a reactive fault tolerance strategy for a node failure is to reschedule a workload to a new node and restart from a specific checkpoint. However, checkpointing a job in a large-scale system could incur large I/O overhead when writing and reading workloads state [3], and takes an overhead of more than 15% of the total execution time [4], [5], which significantly impedes science productivity. As a result, researchers on failure management have found that prevention is better than cure and shifted to proactive management strategies [2], [6]–[10]. In contrast to reactive strategies, proactive strategies develop models based on the failure data in data centers to predict node or workload failures in the near future and take preventive measures to improve the RAS of data centers.

Numerous research efforts have developed node failure detection and prediction methods by utilizing temporal and/or spatial correlations of failures [11]–[14]. They usually investigate system behavior via Syslog analysis and have developed supervised and unsupervised approaches for predicting failures in data centers. A number of studies have attempted on workload-centric failure detection and prediction based on the resource usage or requested resources [15]–[18]. However, only a limited amount of workload data is publicly available due to confidentiality or other reasons. In addition, analyzing and extracting insightful knowledge from massive amounts of data is daunting, given the increasing scale and complexity of data sets.

This research aims at using a machine learning-based approach to predict workload failures in HPC data centers. In particular, we investigate two months of workload traces collected from a production cluster in order to find a correlation between workload attributes with the *exit* status. We seek to train supervised learning models to predict: (1) the failure probability of a workload at queue time, and (2) the likelihood of failure over the life-span of a workload. Having the knowledge of whether jobs will likely fail or not can be valuable for both users to be alerted of the potential failures and for the resource manager to be proactive in preventing wasting computational resources. Consequently, the RAS and productivity of data centers can be improved in return by better managing the workloads that are likely to fail.

We make the following contributions in this study:

- We analyze workload traces collected from a production data center and perform an extensive characterization study of workload failure rates across nodes and users.
- We apply machine learning algorithms on our dataset and train two prediction models: a *Queue-time model* and a *Runtime model*. Experimental results show that these two models predict workload failures with a maximum Precision of 90.61% and 97.75%, respectively.
- We quantify the resource savings achieved by applying the runtime prediction model on workloads at different times. Based on the prediction results, proactive failure

2159-6190/23/$31.00 ©2023 IEEE
DOI 10.1109/CLOUD60044.2023.00064

management may achieve CPU and memory savings by up to 16.7% and 14.53%, respectively.

The rest of this paper is organized as follows. Section II describes the background of this research, including the monitoring infrastructure, and source of anomalies. In Section III, we analyze the workload data. Section IV describes the machine learning algorithms we have investigated in this research and explains our methodology. The experimental results are presented in Section V. Section VI provides an overview of related work, and we conclude this research in Section VII.

II. Background

This research study is conducted on a production academic HPC data center called *Quanah*, where scientists from all major scientific fields, such as astrophysics, computational chemistry, bioinformatics, etc., perform simulations and scientific computations. The Quanah cluster and monitoring framework are described in the next section, followed by the analysis of sources of failures in common data centers.

A. Quanah Cluster

The Quanah cluster at the High Performance Computing Center (HPCC) of Texas Tech University is comprised of 467 nodes with Intel XEON processors providing 36 cores per node [19]. The system has a total of 16,812 cores with a benchmarked total computing power of 485 Teraflops/s and provides 2.5 petabytes of storage capacity. The cluster is based on Dell EMC PowerEdge™ C6320 servers, which are equipped with the integrated Dell Remote Access Controller (iDRAC) [20] providing Redfish API [21] for accessing Baseboard Management Controller (BMC).

B. Monitoring Infrastructure

The monitoring data in the Quanah cluster is obtained through a monitoring tool that utilizes the Redfish API to retrieve sensor metrics from the BMC on each compute node and the resource manager for job accounting data [22]. Sensor metrics and resource usage data are collected at the node level in 60-second intervals, which include power usage, fan speed, CPU usage, memory usage, node-job correlations, etc. Job accounting data is collected per job and includes job submission time, start time, end time, total CPU time, integral memory usage, IO operations, etc. The monitoring data is managed in a Postgres-based time-series database.

C. Sources of Failures

In an HPC data center cluster where computing resources are shared by different workloads submitted by users from various domains, the number of failed workloads can be large. There are three main reasons. First, domain scientists, while skilled in their scientific fields, do not always have sufficient experience and background in computing, especially in large-scale, parallel computing. Second, diverse workloads depend on different libraries, and bugs and missing updates in dependent libraries can lead to unexpected failures. Third, the data center is complex, and any misconfiguration or hardware errors can cause workload termination.

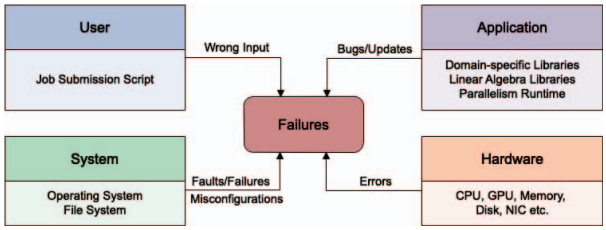

Figure 1: Sources of Failures in Data Centers

TABLE I: Exit Status Summary for Failed Workloads

Exit Code	Meaning	Number	Percentage
1	Miscellaneous errors	21367	58.16%
2	Missing keyword or command	3032	8.25%
7	Argument list too long	6549	17.83%
127	Command not found	528	1.44%
137	(System signal 9) Kill	190	0.52%
255	Exit status out of range	4598	12.52%
Others		475	1.28%

Figure 1 summarizes the high-level root cause categories in data centers, where failures are attributed to user, application, system, or hardware problems.

- *Users*: insufficient resource requests or wrong input in the job submission script can cause workloads to fail [23]. Note that user interruptions, such as issuing a command like "scancel", can interrupt a running workload and cause a failure too. However, since the effect of canceling a running workload is obvious to the user, we do not categorize it as a source of failures.
- *Applications*: misconfigured applications increase the risk of poor performance. In addition, buggy codes, missing dependent library updates, and/or bugs can cause applications to terminate unexpectedly too.
- *Systems*: misconfigurations or failures of system resources and components may considerably affect the performance of workloads or cause failures.
- *Hardware*: hardware errors are one of the most devastating issues for data centers. Severe hardware issues can lead to the malfunction of the entire system. Events such as memory hardware errors, CPU overheating, etc., will result in workload errors and crashes.

III. Workload Analysis

To predict workload failures in data centers, it is crucial to understand the characteristics of failed workloads. In this section, we first present an overview of the workload trace, then quantitatively analyze the percentage of failures in the workloads and the computational resource consumption characteristics. After that, we further study the failure rate across the nodes and users.

A. Workload Overview

The workload trace is derived from the monitoring data collected from the Quanah cluster for the period of August 1, 2020, to October 1, 2020, which contains 324,358 instances

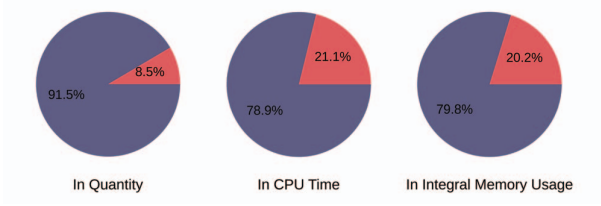

Figure 2: Proportion and Resource Consumption Characteristics of workloads. Red indicates failed workloads and dark blue indicates successful and canceled workloads (non-failures).

submitted from 204 unique users. Each workload trace has categorical fields, such as owner, group, and job name, as well as numeric fields such as CPU time usage, integral memory usage, etc. When a batch job exits, the scheduler generates an `exit_status` field of the job accounting data. We summarize the exit status that indicates failure in Table I. We find that the most common exit status was `1` (58.16%), indicating that there are miscellaneous errors causing the failures. The next most significant exit status is `7` (17.83%), which occurs anytime a user feeds too many arguments in the job submission script. We also notice that there are 190 (0.52%) workloads that are killed by users through system signal `9`. Since we do not consider canceled workloads as failures, we drop these 190 workloads with exit_status `137`. In addition, we do not intend to predict exact errors, so we convert all non-zero `exit_status` to `1`, representing workloads that face problems during run time.

B. Proportion of Workload Failures

Figure 2 presents the proportion and resource consumption characteristics of workload failures. As shown in the figure, the workload failure rate is 8.5% for all submitted jobs in quantity. We further analyze the CPU time consumed by failed workloads and find that failed workloads cost 21.1% of the total CPU time. The proportion of CPU time for failed workloads is larger than the proportion of the number of failed workloads, indicating that the more processors a workload uses and the longer it runs, the higher the probability that this workload will fail. Additionally, we quantify the integral memory usage consumed by failed workloads. As shown in Figure 2, the wasted memory resource rate is 20.2%. All these statistics imply that failed workloads waste a significant amount of computational resources and therefore degrade the system's efficiency.

C. Failures Distribution by Nodes

We depict the distribution of failures across the nodes of the system by physical locations in Figure 3. These 467 nodes in the Quanah cluster are hosted in 10 racks and each node can be uniquely addressed by the rack and chassis number. Each column shown in Figure 3 represents one rack of nodes and each row represents one chassis. From the figure we observe that nodes in racks 1, 3, and 7 have a relatively high workload failure rate. Since the power, temperature, and connectivity of

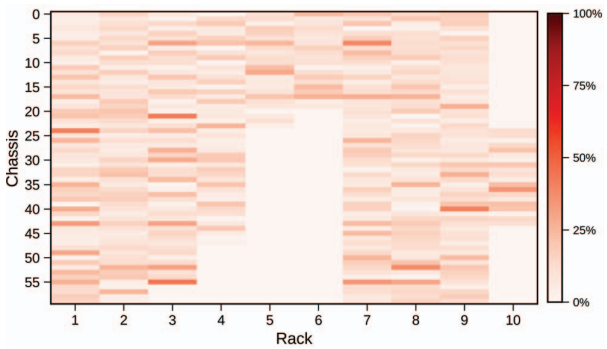

Figure 3: Percentage of workload failures distributed by physical location. The darker the color means the higher the workload failure rate.

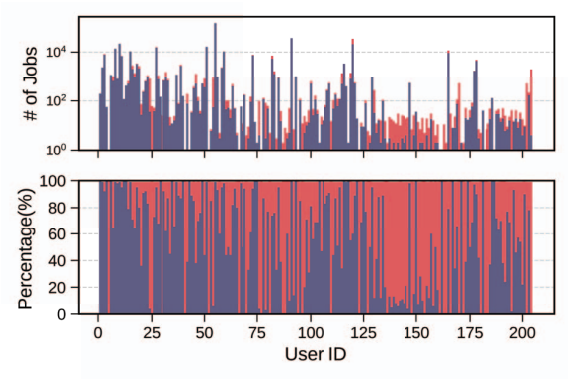

Figure 4: Number and percentage of workload failures distributed by user ID.

all nodes located in a rack are controlled together, problems in these areas can cause failures to occur in physical location vicinity [14].

D. Distribution by Users

To find out the correlation between users and failed workloads, we plot the workload distribution by user ID, as shown in Figure 4. The total number of jobs submitted by users ranges from a few to over 10,000. We observe that the failure rate of workloads per user varies significantly and those users who submit a small number of workloads have a large portion of failed workloads. These statistics suggest that users' experience in properly configuring their applications and/or requesting computational resources varies widely and that these inexperienced users contribute a large fraction of failed workloads.

IV. Methodology

In this section, we describe the workflow of predicting workload failures in HPC data centers. As shown in Figure 5, the workflow consists of four phases: (1) data collection: collecting metrics from data centers; (2) data preparation: preprocessing the data into a structured format and extracting features for machine learning models; (3) model training:

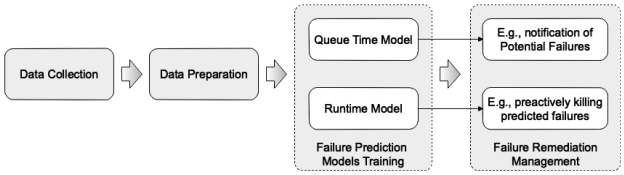

Figure 5: Workflow of Predicting Workload Failures in Data Centers

training *Queue-time* and *Runtime* models using machine learning algorithms; (4) remediation management: applying remediation management techniques to leverage the prediction results to optimize the management of data centers. Data collection has already been discussed in Section II-B. Therefore, we focus on data preparation and model training in this section. We leave the failure remediation management and data center optimizations as near-future research work.

A. Data Preparation

1) Preprocessing: In the data preprocessing phase, we convert raw features into a format more suitable for machine learning training. Specifically, we create dummy variables for the categorical variables by using one-hot encoding [24] and scale the numerical features to avoid features with high variability from having more influence in the prediction. We also deduct irrelevant attributes and derive appropriate features from the original attributes. In our case, we drop the feature *job_id* because it is assigned by the job scheduler and does not reveal the characteristics of the workload. We derive hours of the day and days of the week from time-related features to augment the data. We also derive several numeric features from the resource usages data, such as CPU intensity and average memory usage.

In addition, the collected data does not contain information about applications and libraries. To overcome this limitation, we apply Natural Language Processing (NLP) techniques to job names and identify similar job names submitted by the same user. We then assign a uniform name to these workloads as their job names. This process aims to categorize workloads that use the same libraries. An underline hypothesis of this process is that workloads with similar job names submitted by the same user tend to be the same applications and use the same libraries, differing only in parameters or parts of the code.

As discussed in Section III-A, we do not intend to predict the exact workload error, therefore the prediction is a binary classification problem (i.e., success or failure). We convert exit_status to 1 if the workload fails and 0 otherwise, and use this as the class label.

2) Features Selection: In this study, we trained two models, one for predicting pre-run failures (i.e., queue-time model) and the other for predicting runtime failures (i.e., runtime model). The queue-time model is trained with categorical features such as owner, group, job name, department, etc. The runtime model uses not only categorical features but also resource usage features, such as CPU time, integral memory usage, data transferred in IO, etc.

B. Model Training

We use two classification algorithms to implement machine learning models and train them with our 324,358 instances to predict workload failures. These algorithms and the corresponding hyper-parameters are described below. (1) **Gaussian Naive Bayes**. We use the default value of "None" for `priors` parameter in our model. (2) **Random Forest**. We set the `criterion` parameter to "gini" and the `max_features` to "sqrt". The rest of the parameters are left unchanged.

Since our data set is very large, we use the holdout method instead of the cross-validation method to save computational costs. The data set is partitioned into 65% training data, 15% validation data, and 20% testing data. The training set learns the relationship between the features and the target variables (i.e. 0 for success and 1 for failure). The validation set is used to check how accurately the model defines the relationship between features and known outcomes. The testing data provide a final estimate of the model performance after the model has been trained and validated.

V. Experimental Results

In this section, we describe the evaluation metrics we used for our experiments and present the experimental results including the performance of the machine learning algorithms and the potential resource savings that benefit from the prediction and proactive interference. The models in this study are implemented in the *scikit-learn* Python library [25].

A. Evaluation Metrics

1) Prediction Metrics: In order to measure the performance of ML algorithms, it is important to specify evaluation metrics. We use *recall* (i.e., true positive rate), *precision*, and *F1 Score* as our measurements. The recall represents the ratio between the number of correctly predicted failed workloads to the total number of actual failures. Precision is calculated by dividing the total number of predicted failures by the number of correctly predicted failed workloads. The F1 score is the weighted average of recall and precision. A higher score for these three metrics means that the model's classification results are more accurate.

2) Resource Savings Metrics: The basic proactive failure remediation management is to simply kill workloads that are predicted to fail. This strategy is sensitive to false positives, where workloads are incorrectly predicted to fail. Killing workloads inappropriately will result in wasted resources, as the killed workloads will be restarted and run at a later time. Therefore, We define the resource saving (R_{saving}) as: $R_{saving} = (R_s - R_w)/R_{total}$, where R_{total} is the total resources consumed by failed and successful workloads, R_s is the resource saved by proactively killing failed workloads, and R_w is the resource wasted by killing successful workloads.

TABLE II: Performance of Queue-Time Model

Model	recall	precision	F1 Score	Training Time(s)
GNB	**99.44**	15.65	27.04	3.5
RF	85.00	**90.61**	**87.71**	149.81

TABLE III: Performance of Runtime Model

Model	recall	precision	F1 Score	Training Time(s)
GNB	**99.44**	15.65	27.05	5.13
RF	94.14	**97.75**	**95.91**	145.71

B. Failure Prediction

Table II presents the performance of the queue-time model. Specifically, we observe that Gaussian Naive Bayes (GNB) achieves a better recall score of 99.44%; Random Forest (RF) performs better with a precision score of 90.61% and an F1 score of 87.71%. The performance of the runtime model is shown in Table III. Again, RF achieves a better performance with a precision of 97.75% and an F1 score of 95.91%. Although GNB achieves a better recall score, its precision score is the lowest, indicating a low number of successful failure predictions in its total failure predictions and it predicts most successful workloads as failures. When evaluating the overall performance, we choose RF as the classification algorithm for both models.

C. Resource Savings

The numeric features such as CPU time and integral memory usage are only known after the workload has completed execution. This fact may raise the question of how we can estimate resource savings at different run times of a workload and use these features to predict workload failures early. To apply the run time model on a running workload, we make the following assumption: resource usage is linearly proportional to run time, so its resource usage at different times can be calculated as: $CRU = (FRU * Time)/Wallclock$, where *CRU* stands for Current Resource Usage and *FRU* stands for Final Resource Usage. Based on this formula, we generate a series of test data sets from the original test data (20% of 2-month workload traces in Quanah cluster, i.e., 12-day workload traces) containing synthetic resource usage at different times, and then, apply the runtime model on these data sets. Figure 6 shows the resource savings. From this figure, we observe the same pattern of savings in CPU time and integral memory usage; they both achieve the highest savings at the beginning of the time, 16.7% and 14.53%, respectively. Overall, the resource savings decrease over time, except for a few ups and downs at around 4200s and 14400s.

To better understand the resource savings, we plot the number of workloads and prediction performance at different times, as shown in Figure 7 and Figure 8. Figure 7 shows that the total number of workloads participating in the failure remediation management decreases exponentially and some workloads are completed before the runtime model is applied. Therefore, the resource savings that can be achieved decrease with time. Figure 8 presents the recall, precision, and F1

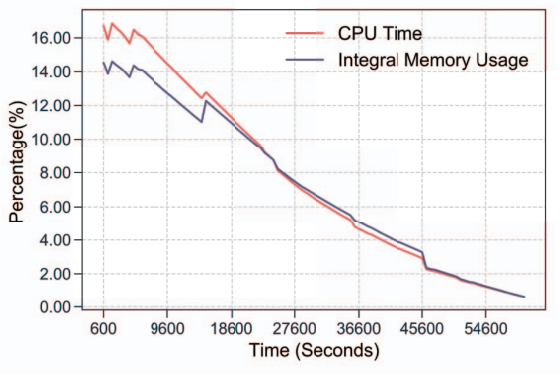

Figure 6: Resource savings in percentage at different times.

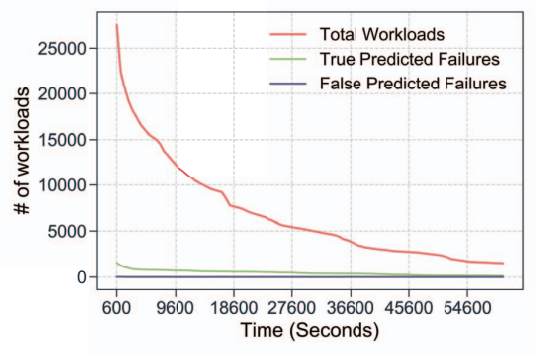

Figure 7: Number of workloads participated in the prediction model at different time.

scores. Throughout time, the precision scores are at high values. The recall and F1 scores are decent (both scores are above 72%) although some fluctuations exist.

VI. Related Work

Characterizing and quantifying failures in data centers are invaluable for system administrators to understand the behavior of the systems and thus develop strategies to improve the RAS of the systems. Many prior works have investigated failures on large-scale systems [11], [12], [14], [15], [26]–[28]. For example, Zheng et al. [27] presented a co-analysis of RAS and job logs that help in understanding failure patterns and system/user behavior. There are also studies that looked specifically into the reliability of particular components such as DARMs, disks, and GPUs [29]–[31].

Considering the failure characteristics and the correlations between failures and performance metrics, and components, several studies investigated machine learning models to predict failures on large-scale systems [16], [17], [32], [33]. For example, Fu et al. [32] proposed a hybrid failure detection framework using one-class and two-class support vector machines (SVM). Chen et al. [16] proposed a prediction method based on Recurrent Neural Network (RNN) that predicts application failure in the cloud using the Google cluster

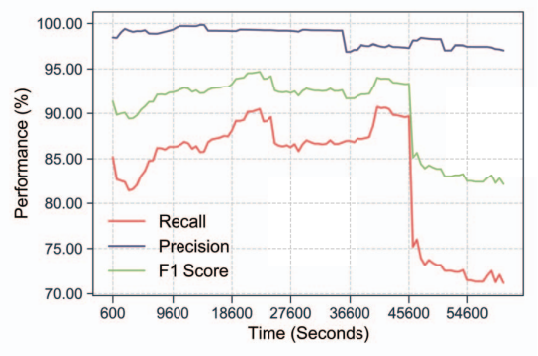

Figure 8: Prediction performance of Random Forest at different time.

workload traces. Many of the proposed approaches are limited to certain performance metrics [16], [17], or are limited to certain components of the system, such as studies focused on GPUs [33].

VII. Conclusions and Future Work

In this study, we analyzed two months of job monitoring data collected from a production HPC data center and found that failed workloads waste a significant amount of compute resources. In addition, we quantified the workload failure rates across nodes and users. Based on the comprehensive understanding of workload traces, we develop two prediction models (queue-time model and runtime model) with two machine learning algorithms and have found that Random Forest performed better with 90.61% and 97.75% precision scores, respectively. Our experimental results show that the workload failure prediction model can help save CPU time and integrated memory usage by up to 16.7% and 14.53%, respectively.

However, there are several areas where our study can be further improved. Firstly, exploring additional machine learning models could enhance the prediction accuracy even further. Secondly, the resource savings predictions are currently based on synthetic data, which may not accurately reflect real-world scenarios. Incorporating time-series monitoring data, rather than a subset of integral data, into the prediction model would provide more accurate results. Lastly, extending the prediction model to infer the causality of workload failure would offer deeper insights into the factors contributing to failures.

In large-scale HPC data centers, where workload failures become the norm, proactive failure management is critical to improving system reliability, availability, and scalability. Understanding the causality of workload failures is important for both system administrators and users. We hope to conduct causal inference studies when detailed provenance is available. Moreover, failure-aware resource scheduling is also a promising research direction and deserves further studies.

VIII. Acknowledgement

We would like to express our gratitude to the anonymous reviewers for their insightful comments and suggestions. This research is supported in part by the National Science Foundation under grants OAC-1835892, CNS-1817094, and CNS-1939140. We are also very grateful to the High-Performance Computing Center of Texas Tech University for providing HPC resources for this project.

References

[1] P. H. Hargrove and J. C. Duell, "Berkeley lab checkpoint/restart (blcr) for linux clusters," in *Journal of Physics: Conference Series*, vol. 46, no. 1. IOP Publishing, 2006, p. 067.

[2] M. Rodríguez-Pascual, J. Cao, J. A. Moríñigo, G. Cooperman, and R. Mayo-García, "Job migration in hpc clusters by means of checkpoint/restart," *The Journal of Supercomputing*, vol. 75, no. 10, pp. 6517–6541, 2019.

[3] R. Garg, T. Patel, G. Cooperman, and D. Tiwari, "Shiraz: Exploiting system reliability and application resilience characteristics to improve large scale system throughput," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 83–94.

[4] E. N. Elnozahy and J. S. Plank, "Checkpointing for peta-scale systems: A look into the future of practical rollback-recovery," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 2, pp. 97–108, 2004.

[5] F. Cappello, "Fault tolerance in petascale/exascale systems: Current knowledge, challenges and research opportunities," *The International Journal of High Performance Computing Applications*, vol. 23, no. 3, pp. 212–226, 2009.

[6] R. K. Sahoo, A. J. Oliner, I. Rish, M. Gupta, J. E. Moreira, S. Ma, R. Vilalta, and A. Sivasubramaniam, "Critical event prediction for proactive management in large-scale computer clusters," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 426–435.

[7] P. Yalagandula, S. Nath, H. Yu, P. B. Gibbons, and S. Seshan, "Beyond availability: Towards a deeper understanding of machine failure characteristics in large distributed systems." in *WORLDS*, 2004.

[8] J. W. Mickens and B. D. Noble, "Exploiting availability prediction in distributed systems." in *NSDI*, vol. 6, 2006, pp. 73–86.

[9] A. Nukada, H. Takizawa, and S. Matsuoka, "Nvcr: A transparent checkpoint-restart library for nvidia cuda," in *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*. IEEE, 2011, pp. 104–113.

[10] A. Rezaei, G. Coviello, C.-H. Li, S. Chakradhar, and F. Mueller, "Snapify: Capturing snapshots of offload applications on xeon phi manycore processors," in *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*, 2014, pp. 1–12.

[11] N. El-Sayed and B. Schroeder, "Reading between the lines of failure logs: Understanding how hpc systems fail," in *2013 43rd annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2013, pp. 1–12.

[12] S. Ghiasvand, F. M. Ciorba, R. Tschüter, and W. E. Nagel, "Lessons learned from spatial and temporal correlation of node failures in high performance computers," in *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*. IEEE, 2016, pp. 377–381.

[13] T. Kimura, A. Watanabe, T. Toyono, and K. Ishibashi, "Proactive failure detection learning generation patterns of large-scale network logs," *IEICE Transactions on Communications*, 2018.

[14] S. Ghiasvand and F. M. Ciorba, "Anomaly detection in high performance computers: A vicinity perspective," in *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE, 2019, pp. 112–120.

[15] H. Fadishei, H. Saadatfar, and H. Deldari, "Job failure prediction in grid environment based on workload characteristics," in *2009 14th International CSI Computer Conference*. IEEE, 2009, pp. 329–334.

[16] X. Chen, C.-D. Lu, and K. Pattabiraman, "Failure prediction of jobs in compute clouds: A google cluster case study," in *2014 IEEE International Symposium on Software Reliability Engineering Workshops*. IEEE, 2014, pp. 341–346.

[17] T. Islam and D. Manivannan, "Predicting application failure in cloud: A machine learning approach," in *2017 IEEE International Conference on Cognitive Computing (ICCC)*. IEEE, 2017, pp. 24–31.
[18] D. Andresen, W. Hsu, H. Yang, and A. Okanlawon, "Machine learning for predictive analytics of compute cluster jobs," *arXiv preprint arXiv:1806.01116*, 2018.
[19] HPCC. (2021) High Performance Computing Center. [Online]. Available: http:www.depts.ttu.edu/hpcc/
[20] D. Technologies. (2021) Integrated Dell Remote Access Controller (iDRAC). [Online]. Available: https://www.delltechnologies.com/en-us/solutions/openmanage/idrac.htm
[21] DMTF. (2021) DMTF's Redfish®. [Online]. Available: https://www.dmtf.org/standards/redfish
[22] J. Li, G. Ali, N. Nguyen, J. Hass, A. Sill, T. Dang, and Y. Chen, "Monster: An out-of-the-box monitoring tool for high performance computing systems," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2020, pp. 119–129.
[23] H. Li, D. Groep, L. Wolters, and J. Templon, "Job failure analysis and its implications in a large-scale production grid," in *2006 Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*. IEEE, 2006, pp. 27–27.
[24] Wikipedia. (2021) Dummy variable (statistics). [Online]. Available: https://en.wikipedia.org/wiki/Dummy_variable_(statistics)
[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.
[26] B. Schroeder and G. A. Gibson, "A large-scale study of failures in high-performance computing systems," *IEEE transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 337–350, 2009.
[27] Z. Zheng, L. Yu, W. Tang, Z. Lan, R. Gupta, N. Desai, S. Coghlan, and D. Buettner, "Co-analysis of ras log and job log on blue gene/p," in *2011 IEEE International Parallel & Distributed Processing Symposium*. IEEE, 2011, pp. 840–851.
[28] C. Di Martino, Z. Kalbarczyk, R. K. Iyer, F. Baccanico, J. Fullop, and W. Kramer, "Lessons learned from the analysis of system failures at petascale: The case of blue waters," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 610–621.
[29] A. A. Hwang, I. A. Stefanovici, and B. Schroeder, "Cosmic rays don't strike twice: understanding the nature of dram errors and the implications for system design," *ACM SIGPLAN Notices*, vol. 47, no. 4, pp. 111–122, 2012.
[30] V. Sridharan, J. Stearley, N. DeBardeleben, S. Blanchard, and S. Gurumurthi, "Feng shui of supercomputer memory positional effects in dram and sram faults," in *SC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 2013, pp. 1–11.
[31] B. Nie, J. Xue, S. Gupta, C. Engelmann, E. Smirni, and D. Tiwari, "Characterizing temperature, power, and soft-error behaviors in data center systems: Insights, challenges, and opportunities," in *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2017, pp. 22–31.
[32] S. Fu, J. Liu, and H. Pannu, "A hybrid anomaly detection framework in cloud computing using one-class and two-class support vector machines," in *International conference on advanced data mining and applications*. Springer, 2012, pp. 726–738.
[33] B. Nie, J. Xue, S. Gupta, T. Patel, C. Engelmann, E. Smirni, and D. Tiwari, "Machine learning models for gpu error prediction in a large scale hpc system," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 95–106.