Log Analytics in HPC: A Data-driven Reinforcement Learning Framework

Zhengping Luo[†], Tao Hou[†], Tung Thanh Nguyen[‡], Hui Zeng[‡] and Zhuo Lu[†]
[†]University of South Florida, Tampa, FL, USA, {zhengpingluo@mail., taohou@mail., zhuolu@}usf.edu
[‡]Intelligent Automation Inc., Rockville, MD, USA, {tnguyen@, hzeng@}i-a-i.com

***Abstract*—High Performance Computing (HPC) has been employed in many fields such as aerospace, weather forecast, numerical simulation, scientific research etc. Security of HPC, especially anomaly/intrusion detection, has attracted many attentions in recent years. Given the heavily instrumented property of HPC systems, logs become an effective and direct data source that can be utilized to evaluate the system status, further, to detect anomalies or malicious users. In this paper, we offer a novel perspective, treating the anomaly detection in HPC as a sequential decision process, and further applying reinforcement learning techniques to learn the state transition process, based on which we build a framework named as ReLog to detect anomalies or malicious users. Besides, a common challenge of employing machine learning techniques is lacking sufficient data, we provide a generative adversarial network (GAN)-based solution to generate sufficient training data in HPC. The experimental validations are conducted based on real-world collected MPI logs, and our results demonstrate a 93% of detection accuracy on the collected dataset.**

***Index Terms*—High performance computing, security, reinforcement learning, defenses and attacks, log analytics**

I. Introduction

High Performance Computing (HPC) has been widely used in domains that require intensive computing such as scientific numerical simulation, financial prediction, weather forecast etc. The research on HPC in the past mainly focused on how to improve the computing performance of the system. Security of HPC has been an active research topic in recent years, especially of how to detect unwanted anomalous jobs or malicious abusive users [1]–[5].

The security problem in HPC is somehow very much similar to those in typical IT systems. As they are both connected to the Internet, and they usually run on Linux-based systems [2]. Besides inherited vulnerabilities in typical IT systems, some vulnerabilities in HPC systems can have disastrous consequences. For example, malicious users in HPC systems can lead to data leakage, and misuse of computing cycles may cause delay or negative affect on other jobs [1].

Typically, HPC systems can be decoupled into 4 layers [6], [7]: application layer, middleware layer, operating system layer and network layer. In nowadays systems, each of the 4 layers has been well-instrumented for security evaluation. Information like percentage of CPU time, memory utilization, I/O time, MPI operations, access information etc. are all well-logged and can be used to analyze and evaluate the performance and security of HPC. Log analytics in HPC has been a more straightforward and effective strategy to secure HPC systems against malicious users [2], [5], [8].

There are many anomaly/intrusion detection methods based on log analytics in HPC have been proposed [2]–[4], especially when machine learning techniques are involved [5], [8]. Machine learning techniques such as deep learning, support vector machine (SVM) have their unique advantages in dealing with large volumes of logs generated by HPC on each computing node [9].

Nevertheless, anomaly/intrusion detection based on log analytics in HPC is not quite the same as traditional classification problems. As the logs in HPC are generated continuously when the system is running, thus the input of anomaly/intrusion detection mechanism is also streaming data. We need to parse the streaming data into tokens, which is defined as a small group of log lines, based on which we can extract feature vectors. Further the feature vectors can be used to perform anomaly/intrusion detection.

The decision of whether the evaluated user is malicious or not depends not only the pattern of all feature vectors, but also the order. Therefore, we can treat detection process of anomaly/intrusion detection in HPC as a sequential decision process, in which final decision of the normality of a user depends on a set of small decisions based on all previous feature vectors. When we think the problem from a sequential decision perspective, we can employ reinforcement learning techniques to model the anomaly/intrusion detection problem in HPC.

In this paper, we propose a framework, named as ReLog, based on reinforcement learning techniques to perform log analytics in HPC for anomalous user detection. We observed that MPI operations used by computing nodes are usually a subset of pre-defined operations [10]. Thus, we construct feature vectors based on MPI logs and treat them as different states in reinforcement learning. Feature vectors can be obtained from parsing of logs using sliding windows. The ultimate reward comes from final classification decision. When malicious users are detected, the reward will reach a maximum value.

Lacking sufficient training data is a common challenge in many reinforcement learning problems, and various synthetic data generation methods have been studied in machine learning [11]. We provide a synthetic data generation method based on Generative Adversarial Networks (GAN) [11] in HPC to tackle this challenge. When no anomalous data are available,

we can get initial data through Gaussian-based sampling from normal data through changing sampling parameters. In scenarios where we have only a small number of available data, we can generate enough data through GAN.

The major contributions of this paper can be listed as follows:

- We build a reinforcement learning-based framework named as ReLog to perform log analytics in HPC. We provide a new perspective, treating the anomaly/intrusion detection as a sequential decision problem, to detect anomalous/malicious users.
- Based on GAN, we give a solution to generating enough training data based on available logs, such that deep neural network models can be fully trained.
- We collect real-world MPI logs and perform comprehensive experiments to validate ReLog and compared it with existing other anomaly/intrusion detection mechanisms. The results show that ReLog can achieve 93% of the detection accuracy on our collected dataset.

The background and related work is shown in Section II; Section III elaborate the design of ReLog; Experimental results are given in Section IV and Section V concludes the paper.

II. Background and related work

In this section, we give the detailed background of log-based anomaly/intrusion detection in HPC systems and the mathematical model of reinforcement learning, which is the foundation of our ReLog framework.

A. Log-based anomaly/intrusion detection in HPC systems

Anomaly/intrusion detection based on logs in HPC systems using machine learning techniques usually have four basic steps: log collection, log parsing, filtering/feature extraction and anomaly/intrusion detection. The basic workflow is shown in Fig. 1.

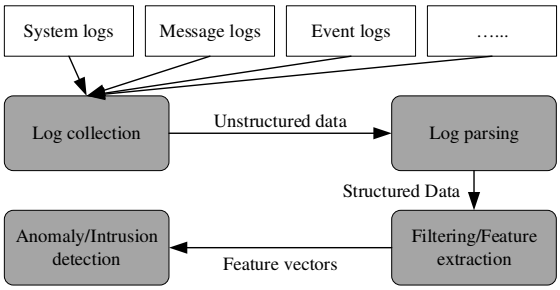

Fig. 1. General workflow of using logs to detect intruders [5], [9], [10], [12], [13].

Log collection is the foundation of log analytics. Reliability, availability and serviceability (RAS) logs of HPC systems can collect information from different hardware and software sensors [8]. Benefited from the heavily instrumentation property in HPC, various logs are generated both from hardware and software at different layers [8]. Most of HPC systems are based on Linux operating systems, log files in Linux systems are stored in text form under the directory of /var/log and its subdirectories, which record the information about system, kernel, access control, package managers, etc.

Log parsing aims to get a structured representation of system information contained in raw logs, which requires a lot of domain knowledge to design rule-based detectors [14]–[16]. As the collected raw data usually is unstructured text files with different formats and semantics. An online streaming method is proposed in [5] using the longest common subsequence to parse logs. Beschastnikh et al. [17] uses regular expressions to parse log lines of interest. Source code are also leveraged in [18] to get structured files. Data mining approaches are employed in [19], [20] to parse log files based on log characteristics.

Filtering/feature extraction is a process to group the separated events from the log parsing step and encode them as feature vectors for machine learning techniques. Grouping techniques such as fixed windows, sliding windows etc. can be used to form feature vectors [9].

Anomaly/intrusion detection involves classifying extracted feature vectors into different patterns. One strategy is to take logs as a language model, thus building relationships between logs and normal/abnormal behaviors. Long Short-Term Memory (LSTM) models are employed in [5] to detect malicious users through treating logs as a structured language model. Besides, a rule ensemble method is adopted in [10] to connect logs with source codes. The relationship of logs and source codes can be further used to predict malicious users.

Based on the above workflow, we designed a novel anomaly/intrusion detection mechanism in this work, in which we treat the detection process as a sequential decision problem, thus reinforcement learning techniques can be employed to improve the detection performance.

B. Reinforcement learning

Reinforcement learning process is usually modeled as a Markov decision process (MDP) with a state space $\mathcal{S}$, an action space $\mathcal{A}$ and the Markov property in terms of stationary transition dynamics as $p(s_{t+1}|s_1, a_1, ..., s_t, a_t) = p(s_{t+1}|s_t, a_t)$ of any sample trajectory $s_1, a_1, s_2, a_2, ..., s_T, a_T$, $s \in \mathcal{S}, a \in \mathcal{A}$ with initial state distribution $p(s_1)$. The reward/cost function is termed as $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The reward $r_t^\gamma = \sum_{k=t}^{\infty} \gamma^{k-t} r(s_k, a_k)$ is the total discounted reward starting from time t, $0 < \gamma < 1$ is the discount factor.

The rule of selecting an action given a state in MDP is named as policy functions, which have the form of $\pi_\theta : \mathcal{S} \to P(\mathcal{A})$, $P(\mathcal{A})$ is the probability distribution over the actions. $\theta \in \mathbb{R}^n$ is a vector of parameters of the policy function. Value functions are the expected total discounted reward. For a given state s, the value $V^\pi(s) = \mathbb{E}[r_1^\gamma | s; \pi_\theta]$ and for a given action a over state s, the value $Q^\pi(s, a) = \mathbb{E}[r_1^\gamma | s; a; \pi_\theta]$. Therefore, we can write the general objective of the reinforcement learning as an expectation $J(\pi_\theta) = \mathbb{E}_{a_k \sim \pi_\theta}[r(s, a)]$ [21].

There are three main different perspectives to solve $J(\pi_\theta)$:

- *Value function algorithms* [22] update value functions after each sample trajectory. The optimal action at each state is the action that can achieve optimal value functions V_* or $Q_*^{\pi_\theta}$ according to Bellman optimality equations, in which $V_*(s) = \max_a \mathbb{E}[R_{t+1}+\gamma V_*(S_{t+1})|S_t = s, A_t = a]$ and $Q_*^{\pi_\theta}(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} Q_*^{\pi_\theta}(S_{t+1}, a')|S_t = s, A_t = a]$. R_{t+1} is the transition reward from state $S_t = s$ to S_{t+1}. $S_t, S_{t+1} \in \mathcal{S}, A_t \in \mathcal{A}$. a' is the action at state S_{t+1}.
- *Policy gradients* [23] update policy parameter vector θ directly through $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$, in which $\nabla_\theta J(\theta) \approx \sum_i (\sum_t \nabla_\theta \log \pi_\theta(a_t^i|s_t^i))(\sum_t r(s_t^i, a_t^i))$ if the commonly known REINFORCE algorithm is used. α is an ajustable parameter to control the step size. a_t^i, s_t^i denote the action and state at time t from sample trajectory $\{\tau^i\}$.
- *Actor-critic algorithms* [24] combine both the value functions and policy gradient update methods. Using value functions, e.g., advantage functions, to estimate $\sum_t r(s_t^i, a_t^i)$ of $\nabla_\theta J(\theta)$ and following the same update strategy in (ii).

We model our anomaly/intrusion detection in HPC as a sequential decision problem. Each feature vector is modeled as a state, and we employ value function algorithms to update the transition reward. Based on the cumulative reward metric, we make the decision of whether the evaluated user is anomalous or not.

III. ReLog

We name the proposed anomaly/intrusion detection framework in HPC systems through LOG analytics based on REinforcement learning techniques as ReLog. Motivation and design details are articulated in this section. We first give the overall architecture of ReLog as shown in Fig.2. Details about the ReLog will be elaborated in following subsections.

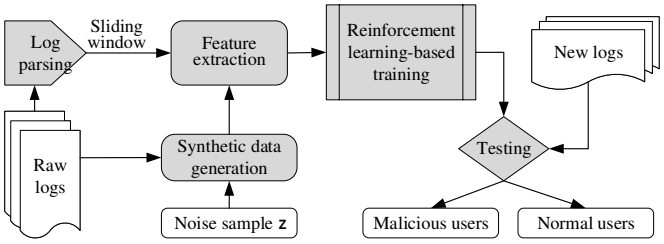

Fig. 2. Overall architecture of ReLog.

A. Motivation

The Message Passing Interface (MPI) is a communication protocol that specifies how HPC passes information among computing nodes and clusters. There are various implementations of MPI, including MPICH, Open MPI etc., and SKaMPI is a benchmark that measures the performance of an MPI implementation on a specific hardware. It is an advanced method for measurements, especially for collective operations of MPI [25].

Unlike traditional classification problem in which they have fixed input format, log analytics in HPC systems is a streaming process as the generation of logs from a running system is continuous. The streaming logs can be parsed into tokens, which is a small group of lines from logs, further using the tokens to extract feature vectors. As we are detecting anomalous users from their behavior history, i.e., whether an evaluated user is anomalous or not depends not only the pattern of all feature vectors, but also the order. We can treat the detection process as a sequential decision process, the final decision depends on the cumulative decisions from each feature vector and from each transition of two feature vectors.

Reinforcement learning is a sequential decision process. The goal of reinforcement learning is to achieve the maximum reward at the end of all decisions. In log analytics of HPC systems, what we are interested in is also the final decision after analyzing all the feature vectors. The similarity of these two problems inspires us to find a solution to detecting anomalous users through log analytics in HPC systems through reinforcement learning techniques.

B. Design of ReLog

The basic idea of ReLog is to treat the anomaly/intrusion detection based on logs of HPC systems as a reinforcement learning problem. As we observed that in MPI logs, the MPI operations used by computing nodes are usually a subset of pre-defined operations [10], thus we can build feature matrix and treat the feature vectors as states in reinforcement learning. Using the idea of sliding window, we can parse log files of MPI logs into a series of states. Therefore, feature vectors extracted from logs can be employed as a state transition process in reinforcement learning. Thus, value function algorithms [22] can be employed to obtain the ultimate reward, and further to classify the evaluated users.

ReLog framework has two steps: training and test. We first train the reinforcement learning model based on feature vectors and then test the new logs on the trained model to classify normal and anomalous users.

In the training process, we first build the state space S from logs through counting frequencies of each operation. Each dimension of feature vector denotes one type of MPI operation. The type information of MPI operation is pre-defined [10]. To reduce complexity or dimensions of feature vectors, we can only include information of the top n most occurred operations rather than all. Because jobs running on HPC systems usually are data-intensive jobs, which requires frequent information exchange between computing nodes, and no exception for malicious attackers. Therefore, it opens the feasibility of using on the top n most occurred MPI commands to diagnose the running status.

Besides the feature vector dimension, we also need to assign appropriate size of sliding windows, such that the frequency information can be counted within that window. The size of sliding window should reflect dynamics of the job running on HPC systems; thus, a too large or too small size of the

window will make the training and test steps too sensitive or too insensitive.

When the examined feature vectors are detected as behaviors from an anomalous user, we set this scenario has the maximum reward. Contrarily, when the feature vectors are classified as behaviors from a normal user, we set the reward as 0, i.e., the minimum reward. Thus, the training process can be treated as a procedure learning the reward of each state transition. In the testing process, we'll get a cumulative final reward from the state transitions of the testing feature vectors. Based on the reward we can classify the feature vectors of the evaluated logs into normal or abnormal user.

C. Synthetic data generation

In real world log analytics of HPC, usually it is not easy to obtain enough training data, especially malicious training data for ReLog. Other cases that require synthetic data generation include (i) when the training data is poorly sampled, i.e., it focuses only on some specific regions, while samples from other regions are necessary to train an efficient and general model; and (ii) when required training data cannot represent what we are looking for, or worse, it is twisted or noised by some unknown reasons.

Synthetic data generation is a critical step in the training of ReLog, there are many ways have been proposed to address data generation problem [11], [26]. In ReLog, we need to generate two kinds of data, both normal data and abnormal data when the training data is limited. Generating synthetic data based on existing available data is comparatively easier than when there is no available data at all.

In case where no malicious data is available, we adopt a Gaussian-based sampling method to generate malicious data based on available normal data. The two parameters needed are the mean value and the standard deviation. The probability density distribution of the Gaussian distribution is denoted as:

$$p(\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(\mathbf{x}-\mu)^2}{2\sigma^2}}, \tag{1}$$

in which $\mathbf{x}$ is the feature vector data. Through changing the mean value and the variance of normal data, we can generate anomalous feature vectors that are required in the ReLog through sampling from the modified Gaussian distribution.

Given the available feature vectors, both normal and abnormal, we plan to construct a Generative Adversarial Network (GAN) [11] based synthetic data generator to generate more normal and abnormal feature vectors to feed the training step in reinforcement learning. In GAN, there are two players: a generator G and a discriminator D. Let p_{data} denotes the distribution that feature vectors are drawn from. The generative model G aims to generate a probability distribution p_g over feature vector data $\mathbf{x}$, which is an estimate of p_{data}. Typically, the generator and discriminator are represented by two deep neural networks. The objective of the GAN framework can be shown as

$$\begin{aligned}\min_G \max_D V(D,G) =& \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log\ D(\mathbf{x})] \\ &+ \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}(\mathbf{z})}}[\log\ (1 - D(G(\mathbf{z})))],\end{aligned} \tag{2}$$

in which $p_{\mathbf{z}}(\mathbf{z})$ is a prior on input noise variables.

The GAN let two players (G and D) play against each other in a game. In ReLog, the feature vector data can also be generated through the generative model. We train two deep neural network models, generative model G and discriminator model D. The two models will play the game to optimize their own objective function. However, to avoid the problem of finding an exact Nash equilibrium, which is challenging in real world, here we use the accuracy of the generated data in discriminator D as a stop requirement, which means after the training process, when the generated data from G can have a higher probability than the pre-set threshold to be misclassified by D, the game stops. The probability is estimated by the percentage of the generated data that are misclassified by D into the real dataset samples in all the generated data.

IV. Experimental validations

We collect real-world MPI dataset from HPC systems and validate ReLog framework in this section. Detailed experimental results are given, and analysis are specified.

A. Dataset

In our work, we use SKaMPI to collect the traces of basic collective operations, such as MPI_Bcast, MPI_Reduce, and so on. According to the communication process of whether to consider the number of processors, we divide the collective operations into two parts, processors-associating (MPI_Scatter, MPI_Gather, etc.) and non-processors-associating (MPI_Bcast, MPI_Redu, etc.). Thus, we collect different processors and counts about different collective operations, such as MPI_Allreduce, MPI_Allgather, MPI_Alltoall, MPI_Bcast, MPI_Gather, MPI_Reduce, MPI_Scam, MPI_Scatter etc.

In our experimental validation, we mainly focused on four types of logs from directories of Isend_Rev, Send_Irecv, Send_Recv and Ssend_Recv. Each of the directory has 24 log files, 96 log files in total are used in our experiments.

B. Experimental results and analysis

We first analyze statistical properties of the collected dataset, which is the foundation to construct feature vectors. We chose 6 logs from the dataset and show frequency information of MPI operations in Fig. 3. The overall frequency information of all logs are shown in Fig. 4. From the frequency information depicted in the two figures, we can observe that only a small number of MPI operations (around 30%) are frequently executed, which paves the path for the application of reinforcement learning frameworks.

1) Synthetic data generation: We divide the collected dataset, which has 4×24 log files, into two categories. Each category has 4×12 log files. One category of logs is used for training, while the other for testing. Initial anomalous data are generated using Gaussian-based sampling method, we generate sufficient anomalous training and testing feature vectors based on the normal training and testing data through changing the mean value of each dimension of the feature

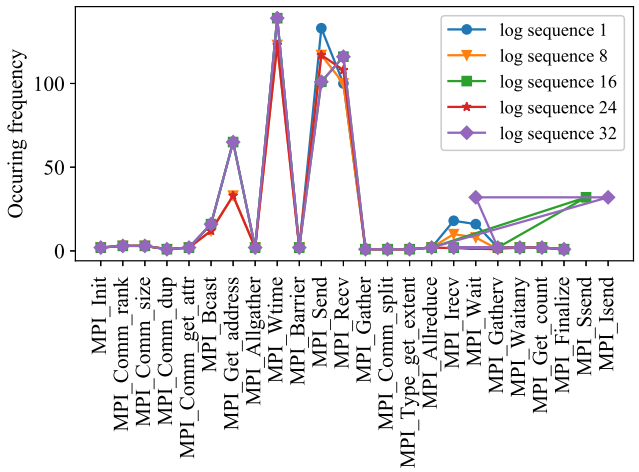

Fig. 3. The frequency information of the training logs

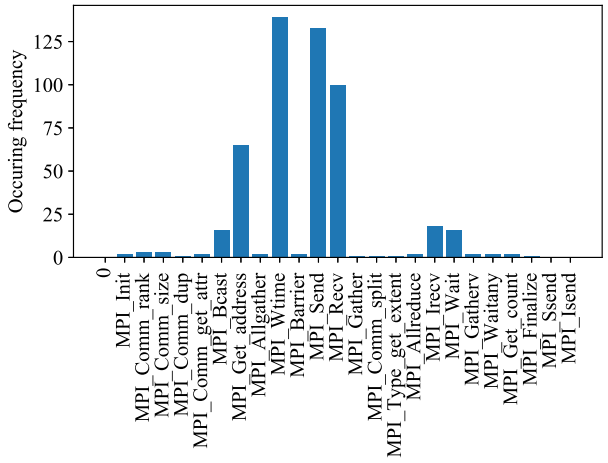

Fig. 4. The overall frequency information of all the training logs

vectors while maintain the same variance of training and testing data. We set the sliding window size as 150 to form feature vectors in this group of experiments. In Fig. 5, we give the relationship between the mean square error (MSE) and the training iterations of GAN-based generation framework. The experimental results show that at least 800 iterations are needed in order to stabilize the MSE.

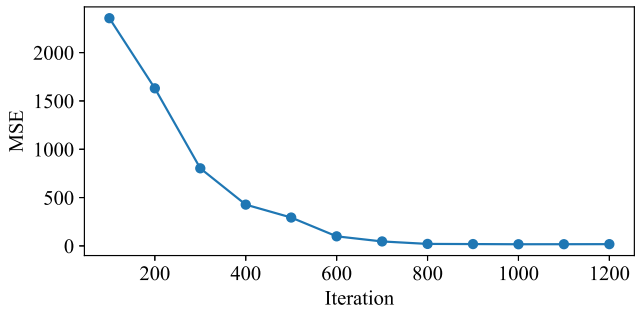

Fig. 5. Relationship of MSE and training iterations.

TABLE I
RELATIONSHIP BETWEEN SLIDING WINDOW SIZE AND DETECTION ACCURACY.

Window size	**100**	**120**	**140**	**160**	**180**	**200**	**220**
Detection accuracy	0.36	0.42	0.54	0.78	0.93	0.93	0.93

2) ReLog Training: To train an effective ReLog model, we need to choose an appropriate sliding window size to form feature vectors. In this group of experiments, we compare the performance of ReLog under different size of the sliding window. The relationship between the sliding window size and detection accuracy is illustrated in Table I. It demonstrates that when the window size is too small, the performance of ReLog in detecting malicious users will suffer. However, as the sliding window size increases to a point (size of 180 in our experiments), the performance of overall detection accuracy tends to be stable, that's because when the sliding window reaches that point, the information contained in each feature vector already can encode most of the classification information.

3) Comparison of ReLog with other methods: To demonstrate the performance of ReLog and show the advantage of using reinforcement learning techniques to detect anomaly/intrusion attackers. We compared ReLog with existing other anomaly/intrusion detection methods in terms of detection accuracy and the time complexity. The methods we compared include: DeepLog [5], which is a deep neural network model employing Long Short-Term Memory (LSTM) to treat logs as natural language sequences; Support Vector Machine (SVN) method, an well-known supervised classification method, which is used in [9] to detect anomaly patterns of logs.

Detailed experimental results on our collected dataset is shown in Table II. We can observe from the results that SVM based detection method has the smallest time cost, that's because the training process involves only finding support vectors, which has been well-solved and the complexity is already optimized. DeepLog and ReLog has similar time cost as a result of training deep learning models in both frameworks. ReLog is especially time-consuming due to the training of multiple deep neural networks in reinforcement learning frameworks. However, ReLog has the best detection performance than both DeepLog and SVM, which we believe also contributed to the reinforcement learning models, which refines the detection process into a sequential decision process. The experimental results demonstrate the promising future of reinforcement learning techniques in detecting anomalous users in HPC systems.

V. CONCLUSION

With the rapid growing of HPC in terms of scale, complexity and widely application in various fields, machine learning-based security inspection will have a more fundamental impact on HPC security. Anomaly/intrusion detection based on log

TABLE II
COMPARISON OF RELOG WITH OTHER EXISTING METHODS

Detection methods	Time cost (seconds)	Detection accuracy
DeepLog [5]	56	0.91
SVM [9]	13	0.86
ReLog	107	0.93

analytics provides a straightforward and effective strategy to evaluate system status of HPC systems. we believe employing machine learning techniques, especially deep learning model-based frameworks such as reinforcement learning, to process the huge volumes of generated logs will result in more robust defense mechanisms for HPC systems than traditional methods.

We offered a new perspective in this paper through employing reinforcement learning techniques to detect anomalies based on MPI logs in HPC. The ReLog works through treating the detection process as a sequential decision process. We also provide a strategy to generate sufficient training data for the training process in case of lacking enough data. The experiments on real-world collected data demonstrate the performance of our proposed methods. How to reduce the complexity, adapt and refine reinforcement learning techniques to log analytics in HPC systems will be our future work.

ACKNOWLEDGEMENT

This material is based upon work supported by the U.S. Department of Energy, Office of Science under Award Number DE-SC0017180.

REFERENCES

[1] S. Peisert, "Security in high-performance computing environments," *Communications of the ACM*, vol. 60, no. 9, pp. 72–80, 2017.

[2] S. Campbell and J. Mellander, "Experiences with intrusion detection in high performance computing," *Proceedings of the Cray User Group (CUG)*, 2011.

[3] R. Bulusu, P. Jain, P. Pawar, M. Afzal, and S. Wandhekar, "Addressing security aspects for hpc infrastructure," in *2018 International Conference on Information and Computer Technologies (ICICT)*. IEEE, 2018, pp. 27–30.

[4] M. Kumar and M. Hanumanthappa, "Scalable intrusion detection systems log analysis using cloud computing infrastructure," in *2013 IEEE International Conference on Computational Intelligence and Computing Research*. IEEE, 2013, pp. 1–4.

[5] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1285–1298.

[6] R. L. Braby, J. E. Garlick, and R. J. Goldstone, "Achieving order through chaos: the llnl hpc linux cluster experience," Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), Tech. Rep., 2003.

[7] F. Gadaud, M. Blanc, and F. Combeau, "An adaptive instrumented node for efficient anomalies and misuse detections in hpc environment," in *CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005.*, vol. 1. IEEE, 2005, pp. 140–145.

[8] B. H. Park, S. Hukerikar, R. Adamson, and C. Engelmann, "Big data meets hpc log analytics: Scalable approach to understanding systems at extreme scale," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2017, pp. 758–765.

[9] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: system log analysis for anomaly detection," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2016, pp. 207–218.

[10] O. DeMasi, T. Samak, and D. H. Bailey, "Identifying hpc codes via performance logs and machine learning," in *Proceedings of the first workshop on Changing landscapes in HPC security*. ACM, 2013, pp. 23–30.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[12] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "Towards automated log parsing for large-scale log data analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 931–944, 2017.

[13] P. Ma, "Log analysis-based intrusion detection via unsupervised learning," *Master of Science, School of Informatics, University of Edinburgh*, 2003.

[14] N. Nakka, A. Agrawal, and A. Choudhary, "Predicting node failure in high performance computing systems from failure and usage logs," in *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*. IEEE, 2011, pp. 1557–1566.

[15] X. Lin, P. Wang, and B. Wu, "Log analysis in cloud computing environment with hadoop and spark," in *2013 5th IEEE International Conference on Broadband Network & Multimedia Technology*. IEEE, 2013, pp. 273–276.

[16] N. Taerat, N. Naksinehaboon, C. Chandler, J. Elliott, C. Leangsuksun, G. Ostrouchov, S. L. Scott, and C. Engelmann, "Blue gene/l log analysis and time to interrupt estimation," in *2009 International Conference on Availability, Reliability and Security*. IEEE, 2009, pp. 173–180.

[17] I. Beschastnikh, Y. Brun, M. D. Ernst, and A. Krishnamurthy, "Inferring models of concurrent systems from logs of their behavior with csight," in *Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 468–479.

[18] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 2009, pp. 117–132.

[19] Q. Fu, J.-G. Lou, Y. Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis," in *2009 ninth IEEE international conference on data mining*. IEEE, 2009, pp. 149–158.

[20] A. A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "Clustering event logs using iterative partitioning," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 1255–1264.

[21] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 2, no. 4.

[22] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[23] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," 2014.

[24] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[25] R. Reussner, P. Sanders, and J. L. Träff, "Skampi: A comprehensive benchmark for public benchmarking of mpi," *Scientific Programming*, vol. 10, no. 1, pp. 55–65, 2002.

[26] S. Tripathi, S. Chandra, A. Agrawal, A. Tyagi, J. M. Rehg, and V. Chari, "Learning to generate synthetic data via compositing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 461–470.