uPAD: Unsupervised Privacy-Aware Anomaly Detection in High Performance Computing Systems

Siavash Ghiasvand
Technische Universität Dresden, Germany

Keywords: Pattern Detection, Time Series Analysis, Anomaly Detection, Neural Networks, Noise Mitigation.

Abstract: Rapid growing complexity of HPC systems in response to demand for higher computing performance, results in higher probability of failures. Early detection of failures significantly reduces the damages caused by failure via impeding their propagation through system. Various anomaly detection mechanism are proposed to detect failures in their early stages. Insufficient amount of failure samples in addition to privacy concerns extremely limits the functionality of available anomaly detection approaches. Advances in machine learning techniques, significantly increased the accuracy of unsupervised anomaly detection methods, addressing the challenge of insufficient failure samples. However, available approaches are either domain specific, inaccurate, or require comprehensive knowledge about the underlying system. Furthermore, processing certain monitoring data such as system logs raises high privacy concerns. In addition, noises in monitoring data severely impact the correctness of data analysis. This work proposes an unsupervised and privacy-aware approach for detecting abnormal behaviors in general HPC systems. Preliminary results indicate high potentials of autoencoders for automatic detection of abnormal behaviors in HPC systems via analyzing anonymized system logs using fast-trainable noise-resistant models.

1 INTRODUCTION

Complexity of computing systems rapidly increases via employing additional components, in response to demand for higher computing performance. Higher complexity results in higher probability of failure occurrence. The first exascale computers are expected to arrive by 2020 (Service, 2016). However, failures already became an integral part of high performance computing (HPC) systems (Dongarra et al., 2011). Addressing failures is a vital requirement towards exascale era.

Advances in machine learning techniques, particularly neural networks, enabled further possibilities to address failures beside traditional methods such as failure prevention and failure recovery (El-Sayed and Schroeder, 2013). Various data generated by computing systems, enables comprehensive monitoring of the systems behavior. *Monitoring data* in form of discrete time-series are particularly useful for behavioral analysis. System logs are discrete time-series which provide invaluable information about the behavior of individual components in different layers of computing systems. The syslog (RCF5424, 2009) daemon is available on all Linux-based operating systems and is active by default. All current TOP500 (TOP500, 2018) supercomputers are powered by Linux-based operating systems. Therefore, for the purpose of this work, system logs are chosen as the monitoring data. Although, system logs are substitutable with any other monitoring data which can be represented as discrete time-series, informational richness of system logs provides accurate insights about the systems behavior. However, processing and analyzing system logs raise privacy concerns due to the existence of sensitive information such as usernames, IP addresses, and detailed information about user activities in log entries. Furthermore, processing the high volume of system logs generated by thousands of computing nodes in an HPC system requires additional computing and storage resources.

This work proposes an efficient unsupervised and privacy-aware approach for detecting abnormal behaviors in general HPC systems. To achieve this goal, all syslog entries are fully anonymized and encoded via *PaRS* anonymization approach (Ghiasvand and Ciorba, 2018) to guarantee the data privacy and reduce the data size. Neural networks are used to process the anonymized syslog entries and generate models representing normal systems behavior. The generated models are used to monitor and evaluate

Ghiasvand, S.
uPAD: Unsupervised Privacy-Aware Anomaly Detection in High Performance Computing Systems.
DOI: 10.5220/0007582208520859
In *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2019)*, pages 852-859
ISBN: 978-989-758-351-3

the behavior of Taurus[1], a production HPC cluster. Preliminary results indicate high potential of machine learning techniques for automatic detection of abnormal behaviors in HPC systems via anonymized system logs. Figure 1 illustrates the workflow of the proposed approach for detecting anomalies and predicting systems behavior.

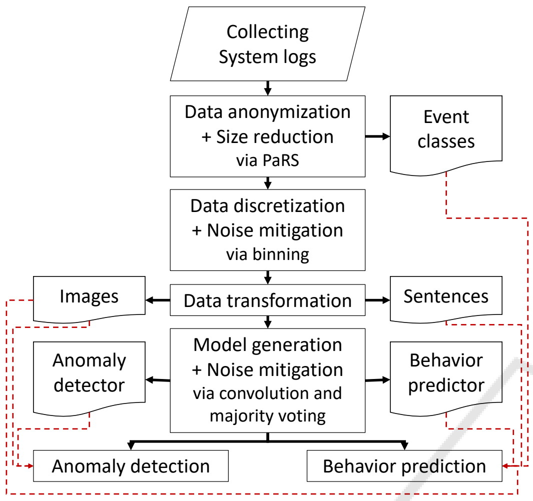

Collecting System logs
Data anonymization + Size reduction via PaRS
Event classes
Data discretization + Noise mitigation via binning
Images
Data transformation
Sentences
Anomaly detector
Model generation + Noise mitigation via convolution and majority voting
Behavior predictor
Anomaly detection
Behavior prediction

Figure 1: The workflow of anomaly detection via syslog analysis.

The remainder of this work is structured as follows: Related works are reviewed in section 2. The data preparation and anomaly detection approach is described in Section 3. Preliminary results are shown in Section 4, and the paper is concluded and important future work directions are introduced in Section 5.

2 RELATED WORKS

Existence of system logs on virtually all computing systems, motivated numerous syslog analysis researches (Girardin and Brodbeck, 1998; Liu et al., 2018). System logs are invaluable resources of information for analyzing systems behavior due to their availability and rich content. In the past decades various syslog-based anomaly detection mechanisms were proposed. However, system logs are not the only monitoring data which are used for detecting anomalies in computing systems. CASPER (Baldoni et al., 2015) monitors the network activities, TIRESIAS (Williams et al., 2007) observes CPU, memory, and context switches, SEAD (Pannu et al., 2012) monitors the hypervisor, and ALERT (Tan et al., 2010) collects various metrics including CPU load, memory usage, input/output data rate and buffer queue length.

Anomaly detection methods are divided into three main categories: rule-based, supervised, and unsupervised (Patcha and Park, 2007). Although many rule-based methods have been proposed, the unstructured form of syslog messages extremely limits the functionalities and the detection domain of rule-based approaches. Supervised methods on the other hand require both normal and abnormal patterns to train a functional behavioral classifier. Therefore, rule-based approaches as well as supervised approaches are not able to detect anomalies which are not seen before. Furthermore, automatic extraction of rules and patterns for rule-based and supervised approaches respectively, are time consuming and inaccurate (Chandola et al., 2009).

In contrast, unsupervised approaches are able to automatically extract an accurate behavioral pattern from the monitoring data. However, most of the unsupervised approaches and available tools are domain specific. They are built specifically for a certain domain of problems e.g., to detect security threads (Yen et al., 2013), identify DNS poisoning attacks (Oprea et al., 2015), or detecting performance bottlenecks (Roy et al., 2015). More general approaches such as invariant log mining (Beschastnikh et al., 2012) and principal component analysis (Jolliffe, 2011) only consider the chronological order of the events, discarding the temporal correlation among log entries.

Recent advances in machine learning techniques further improved unsupervised methods of syslog analysis (Du et al., 2017; Vaarandi et al., 2018; Li et al., 2018; Aussel et al., 2018). Despite the rapid improvements in performance and accuracy of unsupervised syslog analysis approaches via machine learning, certain challenges remained unsolved. As the volume of generated system logs on HPC systems is rapidly increasing, the storage and processing of syslog entries became challenging. Processing system logs which are packed with various personal data, raises high privacy concerns. Due to the heterogeneity of various components in modern HPC systems, each component projects a different behavior which cannot be accurately modeled via a single general model. Software and hardware updates, various applications and the multi-user environment of HPC systems, continuously change the systems behavior. Therefore, a static behavioral model of the HPC system is not sufficient to accurately model the dynamic behavior of modern HPC systems. In addition, system logs are generated by individual computing nodes, thus, any failure directly affects syslog entries via introducing noises, interrupting log generation, or impeding log

[1]https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/SystemTaurus

Table 1: Sample of syslog entries with their respective severity level and event class.

Timestamp	Source	Message	Severity	Event class
1517266801	T-1020	(siavash) CMD (/usr/bin/check)	Information	62440f7d
1517266925	T-1020	(root) CMD (/fast/sbin/start)	Information	62440f7d
1517266929	T-1020	Accepted publickey for siavash from 192.68.31.32	Notice	ea6f83c9
1517267050	T-1020	pam_unix(sshd:session): session opened for user siavash	Notice	feaec917

collection. Furthermore, even harmless errors may introduce noises in syslog entries.

The proposed approach in this work, analyzes anonymized and compressed syslog entries to precisely model the behavior of any general HPC system, via a set of dynamic behavioral models extracted using fast-trainable noise-resistant neural networks. To the best of our knowledge, this is the first work to detect anomalies in computing systems using fully anonymized monitoring data. Transparent noise mitigation via utilizing the *neighborhood homogeneity* within convolutional neural networks, as well as porting monitoring data into image and text processing domains are other main contributions of this work.

3 ANOMALY DETECTION

The proposed approach is intended to be applied on production HPC systems without modifying the HPC systems configuration. Therefore, it is assumed that all behavioral analysis are performed out of the HPC system, and the type and time of the previous failure occurrences are not recorded. The first assumption requires guaranteed data privacy, and the later rules out all supervised and rule-based learning approaches.

The proposed approach collects the monitoring data (syslog entires), anonymizes and transforms the data into predefined formats. Three different networks process the transformed data to extract models which represent the common behavior of the computing nodes. The trained models are used to compare the divergence of new monitoring data against the extracted common behavior to detect abnormal behaviors. Concurrently the trained models are improved via further training using new monitoring data. Following subsections describe the proposed approach in more details.

3.1 Monitoring Data

System logs are used as monitoring data for analyzing computing systems behavior. Each syslog entry denotes the occurrence of an event. Events vary from users logging in and out, to temperature changes, buffer overflows, or kernel panics. Syslog entries does not represent and store all events happening in an HPC system. In certain situations such as boot time, computing systems generate massive amount of syslog entries. Each syslog entry beside its optional metadata (e.g., severity, facility) consists of three main parts: *timestamp*, *source*, and *message*. The *timestamp* and *source* contain structured data while the *message* part is unstructured. *Timestamp* denotes the time of events occurrence, *source* identifies the origin of events, and *message* describes events properties.

Taurus HPC cluster consists of 2046 heterogeneous computing nodes, divided into 6 groups (islands) based on their hardware architecture. Each island itself is divided into several racks, and each rack consists of 18 homogeneous computing nodes. Table 1 contains four sample syslog entries from Taurus, separated into their *timestamp*, *source*, and *message* fields. The *severity* denotes syslog entries importance (specified by log generator) based on one of the 8 standard severity levels of *debug*, *information*, *notice*, *warning*, *error*, *critical*, *alert*, and *emergency* defined in (RCF5424, 2009). The last column in Table 1, contains the *event class* of each entry. The first and second syslog entires in Table 1 share a similar *event class*, thus, represent a similar event which reports: a user executed a command. *Event class*es are extracted from the *message* part of syslog entries via generalization and anonymization. The anonymization process is described in subsection 3.2. Precision of syslog entries timestamp on Taurus is 1 second.

Figure 2 illustrates the occurrence pattern of 6 common events on a single node of Taurus for a period of 32 minutes. Each row represents a certain event class, and each column represents a period of 60 seconds. Each colored cell indicates at least one occurrence of the respective event. Darker colors represent additional occurrences of the respective event during each time interval.

3.2 Data Anonymization

To guarantee the data privacy, all syslog entries are fully anonymized via the *PaRS* anonymization approach before any behavioral analysis. *PaRS* removes sensitive terms[2] such as IP addresses, and usernames

[2]Any string consisting of one or more characters is a term.

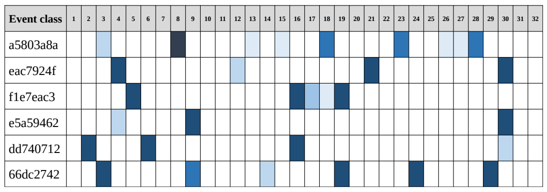


Figure 2: Occurrence pattern of 6 event classes on one node, in 32 minutes. Darker shades of blue indicate more re-occurrences.

from each syslog entry, substitutes all variable terms with relevant constants, and encodes the entire entry using a collision-resistant hash function such as SHAKE-128 (Bertoni et al., 2018). The output of *PaRS* is time-series consisting of timestamp, origin (source), and the hash-key of syslog entries. The metadata remains unchanged.

The final encoding step in *PaRS* beside guaranteeing the full anonymization of syslog entries, significantly reduces the data volume. Figure 3 illustrates the workflow of *PaRS* anonymization approach (Ghiasvand and Ciorba, 2018).

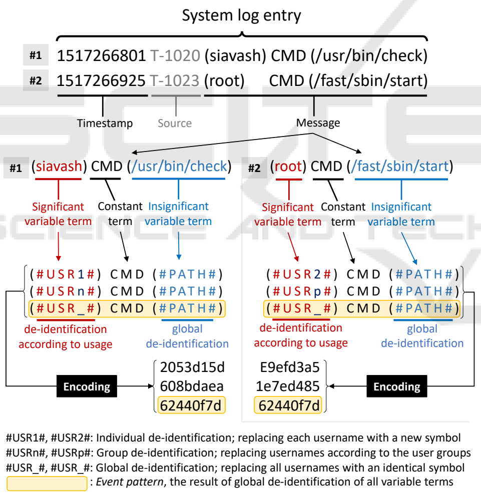

System log entry
#1 1517266801 T-1020 (siavash) CMD (/usr/bin/check)
#2 1517266925 T-1023 (root) CMD (/fast/sbin/start)
Timestamp
Source
Message
#1 (siavash) CMD (/usr/bin/check)
#2 (root) CMD (/fast/sbin/start)
Significant variable term
Constant term
Insignificant variable term
(#USR1#) CMD (#PATH#)
(#USRn#) CMD (#PATH#)
(#USR_#) CMD (#PATH#)
(#USR2#) CMD (#PATH#)
(#USRp#) CMD (#PATH#)
(#USR_#) CMD (#PATH#)
de-identification according to usage
global de-identification
Encoding
2053d15d
608bdaea
62440f7d
E9efd3a5
1e7ed485
62440f7d
#USR1#, #USR2#: Individual de-identification; replacing each username with a new symbol
#USRn#, #USRp#: Group de-identification; replacing usernames according to the user groups
#USR_#, #USR_#: Global de-identification; replacing all usernames with an identical symbol
: Event pattern, the result of global de-identification of all variable terms

Figure 3: Workflow of *PaRS* anonymization approach (Ghiasvand and Ciorba, 2018).

3.3 Data Preparation

System logs are collected and stored by local syslog daemons on each computing node. Afterwards, syslog entries can be either directly forwarded to a central syslog storage or actively being collected from each individual node. The entire procedure of syslog generation, transmission, and collection is prone to errors caused by software and hardware failures. Same principle applies to all other monitoring data collected from computing nodes such as power consumption, CPU load, or memory utilization. Therefore, data preparation is necessary.

3.3.1 Data Discretization

The proposed approach in this work is based on time-series analysis. Therefore, prior to analysis the continuous monitoring data collected from computing systems should be discretized. System logs are discrete time-series by nature, therefore, further discretization is not required. However, discrete binning of syslog entries can be used to reduce the sampling rate of collected system logs and mitigate certain noises. In this work a one minute time binning is applied on syslog entires. Each bin contains the accumulated number of events occurred in one minute per node and per event class.

3.3.2 Noise Mitigation

Noise is an erroneous presence or absence of log entries within the monitoring data. To identify the normal behavior of computing systems, it is necessary to remove the random noises. Beside software and hardware failures which may inject random noises into the monitoring data, other actions such as software updates, administration activities, and system maintenances can also introduce noises. In addition, most production HPC systems are used by various groups of users and for different applications. Therefore, existence of random noises in monitoring data is highly plausible due to human errors and application misbehaviors. Part of these noises can be removed via discrete binning of the monitoring data introduced in subsubsection 3.3.1. However, an extreme discrete binning can decrease the accuracy of anomaly detection by decreasing the monitoring data precision.

This work utilizes the *neighborhood homogeneity* of HPC systems to mitigate random noises. Computing nodes in HPC systems are divided into smaller subsets such as chassis or racks. Majority of these small subsets consist of homogeneous computing nodes which share various physical resources such as power supply, cooling system, and network infrastructure. Homogeneous computing nodes which are physically collocated (adjacent) and share similar physical resources tend to project similar behaviors. Therefore, in a homogeneous subset of computing nodes, common behavior of the majority can be considered as the normal behavior in that particular subset. The extraction of common node behavior from noisy syslog entries on Taurus in a subset consisting of 8 homogeneous computing nodes is shown in Figure 4. Colored cells mark the occurrences of event (a5803a8a) on 8 adjacent nodes during 32 minutes.

Bottom row indicates the normal pattern of event occurrences, extracted via majority voting among the 8 computing nodes. Events are placed in each bin according to their relative time passed since midnight. Further time synchronization is not required.

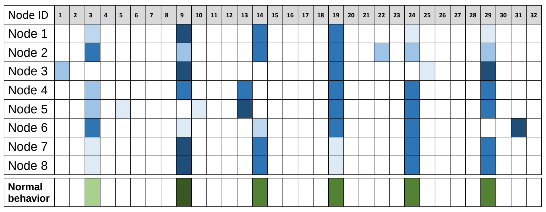


Figure 4: Occurrence pattern of one event class (a5803a8a) on 8 nodes, in 32 minutes. The bottom row shown in green, holds the result of majority voting on all 8 nodes.

Outliers are valid events which distant from the norm. Outliers can impede correct analysis of systems behavior. However, in contrast to noises, outliers are part of the systems behavior, thus, they should not be removed. Outliers are not always indicators of an abnormal behavior. It is worth to emphasize that the goal of this stage is extracting the pattern of normal (healthy) system behavior. Therefore, standardizing the data range (scaling) is sufficient to omit the negative effect of outliers.

Considering the noise mitigation approach shown in Figure 4, when the majority of nodes project abnormal behavior, the extracted behavior pattern will be incorrect. However, analyzing Taurus behavior revealed that except global system failures which affect the majority of computing nodes, utilizing the *neighborhood homogeneity* and majority voting correctly extracts the common event patterns. Nevertheless, the system log entries collected during global system failures must be excluded from the training data to prevent unexpected results.

3.4 Model Generation

Having insufficient amount of failure samples was one of the main motivations to use an unsupervised approach for anomaly detection in this work. On Taurus, except certain failures which are caused by distributed file system, rest of the failures are not frequent (less than 10 occurrences per year). Due to insufficient amount of failure samples, automatic extraction of abnormal system behavior patterns leading to non-frequent failures is not feasible. Therefore, rather than extracting the pattern of abnormal behaviors and detecting similarities between the current behavior and abnormal behavior, this work considers the common system behavior as the normal behavior and evaluates the divergence of current behavior from the common behavior.

In this work, three neural networks are used to extract the common system behavior patterns from system logs. For the first two networks explained in subsubsection 3.4.1, syslog entries are transformed into images and processed via image processing techniques. While the third network explained in subsubsection 3.4.2, uses a text auto-completion technique to predict the upcoming events. The usage of each network is described in section 4.

3.4.1 Image Processing

Many periodic events in HPC systems have static time intervals. Longest interval between two consecutive occurrences of a periodic event on Taurus is 60 minutes, thus, every periodic syslog entry appears at least once during an hour. Therefore, the observation window of one hour was chosen to monitor Taurus behavior. To simplify future calculations, the width of observation window is extended to 64 minutes (2^6). However, in each observation the window is shifted forward by 60 minutes such that the observation window always starts exactly on the hour. Hereafter, the data which is captured in an observation window is referred to as a *frame*. Shifting the observation window and capturing data frames for a duration of four hours are shown in Figure 5.

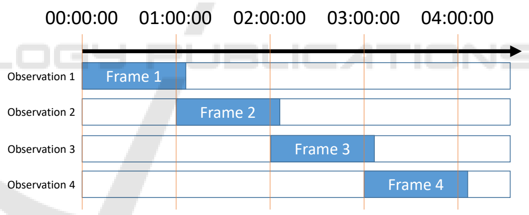


Figure 5: Capturing data frames from monitoring data.

Each frame is represented as a two dimensional matrix of F_e, with t columns and n rows. t represents the time bins of one minute and is equal to the width of the observation window (64 in this work) and n is the number of nodes which have been observed. The value of each cell (v_{nt}) denotes the number of event e re-occurrences for all events of the same severity level within the time bin of t on node n. A sample frame is shown in Figure 6(a). This frame represents all events marked as *emergency* and occurred on 18 adjacent computing nodes (a rack) during a 64-minute time window. Two nodes (rows) are randomly chosen to be removed from the frame to simplify the future calculations ($18 - 2 = 2^4$). To eliminate potential accuracy penalties caused by random node removals, two different copies of each frame are generated. For the second copy, two nodes other than those which were removed from the first copy are randomly chosen to be removed. During the learning phase, networks are trained on both copies.

For each of the eight syslog severity levels a separate frame is captured as shown in Figure 6(b). Frames containing *emergency*, *alert*, and *critical* events are merged into a single frame. Similarly the *error* frame is merged with *warning* frame, and the *notice* frame is merged with *information* frame. The *debug* frame remains unchanged. Furthermore, the values of each cell is normalized to the range of 0 and 255. The four resulting frames are stored as a three dimensional matrix shown in Figure 6(c), representing a 16 by 64 pixels RGBA PNG image (W3C, 2018).

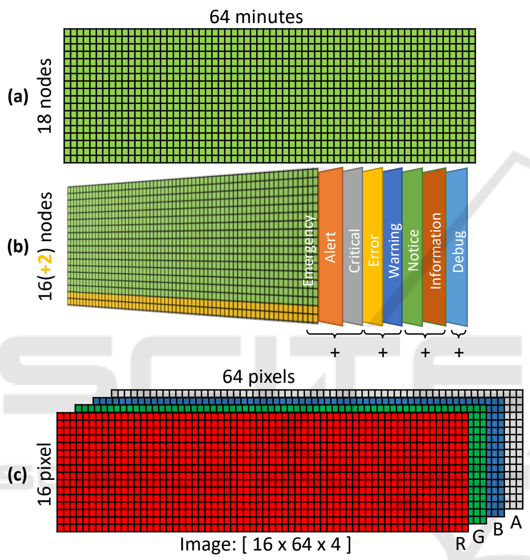


Figure 6: Transforming syslog entries to an RGBA image. Value of each cell indicates the number of event reoccurrences per node and per minute. (a) 2D representation of syslog entries. (b) Removing extra rows and merging events with similar severity levels. (c) RGBA representation of syslog entries.

Two different autoencoders are used to model the normal behavior of Taurus via image data. Both networks are trained via a sequence of 24 RGBA images in size of 16 by 64 pixels. The first approach is training a state-less convolutional autoencoder shown in Figure 7. The expected output of the network is the same image (frame) as the input.

The second approach is training a long short term memory (LSTM) autoencoder. The expected output of the network in this approach is the next image (frame) in the input sequence. In another word, the network should predict the next image of the sequence. A similar network as shown in Figure 7 is used, with the convolutional layers substituted by convolutional LSTMs.

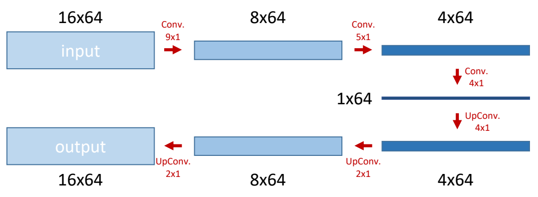
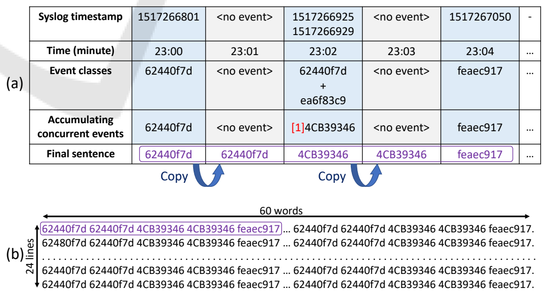


Figure 7: The lightweight convolutional autoencoder for extracting behavioral patterns. Encoding layers perform the majority voting and noise mitigation. While decoding layers are fine tuning the results.

3.4.2 Text Auto-completion

The third approach is using a text auto-completer. The input of network is a sequence of anonymized syslog entries (event classes) and the expected output is the upcoming entries. In another word, this network predicts the future events and completes the sequence. In this approach, one day of syslog entries is considered as a text with 24 sentences, and 60 words per sentence. Each word is the *event class* of a syslog entry. A sequence of 60 words (one word per minute) forms a sentence, and 24 sentences form a full text (one day). Multiple occurrences of an identical event within a minute are ignored, while the occurrence of concurrent distinct events are accumulated. Empty minutes are filled with the *event class* of the previous event. Figure 8(a) illustrates the transformation of sample syslog entries shown in Table 1 into an incomplete sentence of event classes. The network used in this approach consists of two layers, a dense layer attached to an LSTM.


(a)

Syslog timestamp	1517266801	<no event>	1517266925 1517266929	<no event>	1517267050	-
Time (minute)	23:00	23:01	23:02	23:03	23:04	...
Event classes	62440f7d	<no event>	62440f7d + ea6f83c9	<no event>	feaec917	...
Accumulating concurrent events	62440f7d	<no event>	[1]4CB39346	<no event>	feaec917	...
Final sentence	62440f7d	62440f7d	4CB39346	4CB39346	feaec917	...

(b)

```
62440f7d 62440f7d 4CB39346 4CB39346 feaec917... 62440f7d 62440f7d 4CB39346 4CB39346 feaec917.
62480f7d 62440f7d 4CB39346 4CB39346 feaec917 ... 62440f7d 62440f7d 4CB39346 4CB39346 feaec917.
.................................................................................
62440f7d 62440f7d 4CB39346 4CB39346 feaec917 ... 62440f7d 62440f7d 4CB39346 4CB39346 feaec917.
62440f7d 62440f7d 4CB39346 4CB39346 feaec917 ... 62440f7d 62440f7d 4CB39346 4CB39346 feaec917.
```

Figure 8: (a) Transforming sample syslog entries from Table 1 into an incomplete sentence. (b) 24 hours of syslog entires represented as 24 sentences with the constant length of 60 words.

4 PRELIMINARY RESULTS

The three proposed approaches were applied on one month of system logs from 270 Taurus nodes (15 racks) collected in April 2017. An independent instance of each network was trained with each rack data, resulting in 45 trained networks (3 per rack). The state-less autoencoder generated outputs with 65% accuracy (correct detection of normal behavior using noisy system logs) after only 10 epochs. This number raised to about 30 epochs for the LSTM autoencoder and 400 epochs for the text auto-completer. The accuracy improves via additional epochs. However, to achieve higher accuracy with less epochs, hyper-parameters must be further tuned.

Based on the preliminary results, the state-less autoencoder could be efficiently used for de-noising the monitoring data. The LSTM autoencoder could be used to quickly model the behavioral pattern of the HPC system. The text auto-completer needs further improvements such as additional noise removal, since this approach projected high sensitivity to random noises.

The required time for the first training of the state-less autoencoder, LSTM autoencoder, and text auto-completer networks with 720 samples on an Intel Xeon E5 took 30, 65, 40 seconds respectively. Small size of the networks, low volume of the required input data and fast training curve makes them good candidates for modeling the HPC systems behavior.

To evaluate the current behavior of each computing node, the nodes syslog entries are monitored using a 64-minute observation window. The resulting 1x64x4 frame is transformed into a 16x64x4 frame using up-sampling (row copy). Evaluating the observed frame via any of the three proposed networks provides a prediction of upcoming events and a value indicating the divergence of current behavior from the common behavior. The state-less network is intended to be used for subsets of the HPC systems with static behaviors such as nodes which are not accessible by users and are not under constant maintenances, while the LSTM is intended to be used for dynamic parts. The text auto-completer, due to its short response time is more suitable for larger HPC systems.

5 CONCLUSION AND FUTURE WORKS

High performance computing systems are rapidly growing in size and complexity. Failures already became an integral part of HPC systems and the exascale systems are expected to arrive soon. Recent advances in machine learning techniques dramatically improved the methods in addressing failures on computing systems. Anomaly detection approaches can efficiently reduce the negative side effects of failures via early detection and prediction of upcoming failures. Although the accuracy of pattern detection mechanisms is highly improved using neural networks; Because of the heterogeneous nature of the modern computing systems, the behavioral patterns of these systems cannot be precisely described via a single global model. Different subsets of HPC systems, project different behaviors and independently change during the time. A global model cannot reflect the continuous changes in various components of HPC systems.

This work proposed an approach using autoencoders to generate behavioral models for smaller subsets of high performance computing systems. These small models are more precise in compare to single global models, faster to train and easier to update. In addition, the autoencoders are designed to transparently mitigate random noises in monitoring data via utilizing the *neighborhood homogeneity* feature of HPC systems. Transforming the monitoring data into the RGBA format, enables direct application of various image processing methods on monitoring data. Furthermore, the anonymized monitoring data can securely be processed via various text processing methods without raising privacy concerns.

In this work monitoring data from Taurus HPC cluster is used solely to exemplify the proposed approach. The RGBA format is also chosen due its visualization capabilities. The proposed models can easily adapt to any other monitoring data via extending the depth of transformed image from four channels (RGBA) to an arbitrary number, e.g., for syslog entries *event classes* can be used instead of severity levels.

Fine-tuning the networks hyperparameters, further training of the networks via additional monitoring data, and improving the performance of text auto-completer approach are planned future works. In addition, the networks will be trained via other monitoring data (beside syslog entries) and their accuracy and performance will be compared.

ACKNOWLEDGEMENTS

This work is in part supported by the German Research Foundation (DFG) within the Cluster of Excellence ‘Center for Advancing Electronics Dresden (cfaed)’.

REFERENCES

Aussel, N., Petetin, Y., and Chabridon, S. (2018). Improving performances of log mining for anomaly prediction through nlp-based log parsing. In *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 237–243.

Baldoni, R., Montanari, L., and Rizzuto, M. (2015). Online failure prediction in safety-critical systems. *Future Gener. Comput. Syst.*, 45(C):123–132.

Bertoni, G., Daemen, J., Peeters, M., and Van Assche, G. (2018). The KECCAK SHA-3 submission. `keccak.noekeon.org`. Accessed: 2018-11-15.

Beschastnikh, I., Brun, Y., Ernst, M. D., Krishnamurthy, A., and Anderson, T. E. (2012). Mining temporal invariants from partially ordered logs. *SIGOPS Oper. Syst. Rev.*, 45(3):39–46.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):1–58.

Dongarra, J., Beckman, P., Moore, T., Aerts, P., Aloisio, G., Andre, J.-C., Barkai, D., Berthou, J.-Y., Boku, T., Braunschweig, B., Cappello, F., Chapman, B., Chi, X., Choudhary, A., Dosanjh, S., Dunning, T., Fiore, S., Geist, A., Gropp, B., Harrison, R., Hereld, M., Heroux, M., Hoisie, A., Hotta, K., Jin, Z., Ishikawa, Y., Johnson, F., Kale, S., Kenway, R., Keyes, D., Kramer, B., Labarta, J., Lichnewsky, A., Lippert, T., Lucas, B., Maccabe, B., Matsuoka, S., Messina, P., Michielse, P., Mohr, B., Mueller, M. S., Nagel, W. E., Nakashima, H., Papka, M. E., Reed, D., Sato, M., Seidel, E., Shalf, J., Skinner, D., Snir, M., Sterling, T., Stevens, R., Streitz, F., Sugar, B., Sumimoto, S., Tang, W., Taylor, J., Thakur, R., Trefethen, A., Valero, M., van der Steen, A., Vetter, J., Williams, P., Wisniewski, R., and Yelick, K. (2011). The international exascale software project roadmap. *The International Journal of High Performance Computing Applications*, 25(1):3–60.

Du, M., Li, F., Zheng, G., and Srikumar, V. (2017). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1285–1298, New York, NY, USA. ACM.

El-Sayed, N. and Schroeder, B. (2013). 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)*, pages 1–12.

Ghiasvand, S. and Ciorba, F. M. Assessing Data Usefulness for Failure Analysis in Anonymized System Logs. In *Proceedings of the 17th International Symposium on Parallel and Distributed Computing (ISPDC)*.

Ghiasvand, S. and Ciorba, F. M. (2018). Anonymization of System Logs for Preserving Privacy and Reducing Storage. In *Proceedings of the 2018 Future of Information and Communications Conference (FICC)*, pages 440–447, Singapore.

Girardin, L. and Brodbeck, D. (1998). A visual approach for monitoring logs. In *12th Systems Administration Conference (LISA)*, volume 98, pages 299–308.

Jolliffe, I. (2011). *Principal Component Analysis*, pages 1094–1096. Springer, Berlin, Heidelberg.

Li, Z., Davidson, M., Fu, S., Blanchard, S., and Lang, M. (2018). Converting unstructured system logs into structured event list for anomaly detection. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, ARES 2018, pages 15:1–15:10, New York, NY, USA. ACM.

Liu, Z., Qin, T., Guan, X., Jiang, H., and Wang, C. (2018). An integrated method for anomaly detection from massive system logs. *IEEE Access*, 6:30602–30611.

Oprea, A., Li, Z., Yen, T., Chin, S. H., and Alrwais, S. (2015). Detection of early-stage enterprise infection by mining large-scale log data. In *45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 45–56.

Pannu, H. S., Liu, J., and Fu, S. (2012). A self-evolving anomaly detection framework for developing highly dependable utility clouds. In *2012 IEEE Global Communications Conference (GLOBECOM)*.

Patcha, A. and Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*.

RCF5424 (2009). The syslog protocol. `tools.ietf.org`. Accessed: 2018-11-15.

Roy, S., König, A. C., Dvorkin, I., and Kumar, M. (2015). Perfaugur: Robust diagnostics for performance anomalies in cloud services. In *2015 IEEE 31st International Conference on Data Engineering*.

Service, R. F. (2016). Racing to match China's growing computer power, U.S. outlines design for exascale computer. `sciencemag.org`. Accessed: 2018-11-15.

Tan, Y., Gu, X., and Wang, H. (2010). Adaptive system anomaly prediction for large-scale hosting infrastructures. In *Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '10, pages 173–182, New York, NY, USA. ACM.

TOP500 (2018). TOP 500 Supercomputer Sites. `top500.org`. Accessed: 2018-11-15.

Vaarandi, R., Blumbergs, B., and Kont, M. (2018). An unsupervised framework for detecting anomalous messages from syslog log files. In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6.

W3C (2018). PNG (Portable Network Graphics) Specification. `w3.org`. Accessed: 2018-11-15.

Williams, A. W., Pertet, S. M., and Narasimhan, P. (2007). Tiresias: Black-box failure prediction in distributed systems. In *2007 IEEE International Parallel and Distributed Processing Symposium*, pages 1–8.

Yen, T.-F., Oprea, A., Onarlioglu, K., Leetham, T., Robertson, W., Juels, A., and Kirda, E. (2013). Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *Proceedings of the 29th Annual Computer Security Applications Conference*, ACSAC '13, pages 199–208, New York, NY, USA. ACM.