ModuleLog: Module Based Approach to Anomaly Detection In Parallel File System Logs

Chris Egersdoerfer

Computer Science
University of North Carolina at Charlotte
Charlotte, NC, United States
cegersdo@uncc.edu

Dong Dai

Computer Science
University of North Carolina at Charlotte
Charlotte, NC, United States
dong.dai@uncc.edu

Di Zhang

Computer Science
University of North Carolina at Charlotte
Charlotte, NC, United States
dzhang16@uncc.edu

***Abstract*—With the increasing prevalence of Parallel File Systems (PFSes) in the context of vast and complex server networks, the importance of accurate anomaly detection on runtime logs of parallel file systems is increasing. But as it currently stands, many state-of-the-art methods for log-based anomaly detection, such as DeepLog, have encountered numerous challenges when applied to parallel file system logs due to their irregularity and a lack of identifying characteristics. Although a previous work, SentiLog has shown promising results, the sentiment based model lacks analysis of temporal dependencies within a log sequence, and hence misses important sequence-based anomalies. To circumvent these problems, this study proposes ModuleLog, a log anomaly detection solution which analyzes the temporal sequence of logging modules to detect irregularity. The key distinction from existing sequence-based anomaly detection solutions is the attempt to reduce the granularity of using individual log keys by grouping these keys by the module they reside in, based on the PFS source code. We apply an RNN architecture with regular LSTM cells to the sequence of modules. This method allows ModuleLog to be able to detect transition points between normal and abnormal logs in a given sequence, as well as detect sequences of abnormal logs.**

I. Introduction

In light of growing datasets, increasing problem complexity, and more compute intensive algorithms, it is no question that High Performance Computing is a growing area of interest. Additionally, with high prevalence of decentralized computing, leveraging many server nodes at once to provide globally connected storage, file systems must also adapt to meet these complex needs. This need for adaptation has brought Parallel File Systems (PFS) into focus, mainly due to their inherent scalability and high bandwidth support. With increased use of these systems, and on such large scale, anomaly detection is obviously a prevalent issue, but current State of the Art file system anomaly detection techniques do not seem to generalize well to this problem area. This is due to the fact that PFS logs are not only very high volume, but they also often do not include session or block identifiers making them highly irregular. To further explain these key issues, PFSes such as Lustre can generate logs across all servers in milliseconds so algorithms meant to individually identify logs as normal or abnormal must be light enough to keep up with incoming logs. Additionally, the aforementioned lack of session or block identifiers which are commonly seen in Distributed File Systems, presents a high degree of irregularity among incoming logs and rules out a number of algorithms designed to classify abnormality on a per

session basis.

To face these challenges, this study proposes ModuleLog, a PFS log analysis technique for anomaly detection based on time series log data. ModuleLog is based on the simple idea of reducing the granularity of log keys by encoding log keys by the module they reside in. Reducing the granularity of log keys this way eliminates much of the noise present in PFS log key sequences allowing for a predictive model based on time series data to much more easily separate normal from abnormal behavior.

II. Related Works

There is a large catalog of anomaly detection approaches for file system logs in existence today. These approaches generally can be broken into non-machine learning approaches and machine learning approaches, the latter of which can be further broken into supervised and unsupervised learning. Of the more successful non-machine learning approaches, the general approach can be summarized as rule based, where domain experts formulate a set of rules which can be used to classify logs as anomalies [1], [9], [5], [8], [2]. These approaches can be quickly ruled out as a result of their need for expert level knowledge and their understandable inability to generalize across various systems. In contrast, both unsupervised and supervised machine learning approaches have shown to provide more robust solutions.

Among unsupervised techniques, pattern seeking [10], [11] and invariant mining [6] based algorithms have proven effective on a variety of file system logs, but their results do not hold up on PFS logs due to the aforementioned irregularity of these logs. Additionally, sentiment analysis [12] has proven to be highly effective for PFS logs but this approach is unable to pick up on sequence based anomalies as logs are only analyzed one at a time. Among supervised learning algorithms, additional methodologies have been presented, some of which are similar to the work presented in this work. The first, and least similar, approach, LogAnomaly[7], borrows from NLP by proposing a log key embedding technique which vectorizes log keys and calculates similarity between vector embeddings to predict anomalies. An additional approach in this domain, LogBERT[4], utilizes the Bidirectional Encoders Represented by Transformers model which has provided State of the Art Results in multiple domains. In this approach, it is not the next log key that is predicted, but rather a given sequence that is masked and then classified as being normal or abnormal. While this approach does seems to work well for some file system logs, specifically those based on sessions, the lack of these grouping characteristics in PFS logs and resultant irregularity makes it difficult for this solution to be effectively applied. A final approach, and the one most similar to this work, DeepLog[3], considers a window of ten sequential log keys which are fed into a two layer LSTM model which predicts the probability distribution over the log keys representing the most likely key to come next. From this distribution, only the top 9 log keys are selected and accuracy is calculated on this basis. Though even this method does not hold up against the irregularity in PFS logs.

III. Design and Implementation

ModuleLog design is largely based on the approach utilized by DeepLog which analyzes sequential log key input and attempts to predict the next key. The key distinction in this work is the introduction of module encoding which reduces the granularity of log key encoding

A. Module Encoding

The first step to implement ModuleLog is the module encoding process. The idea behind this process is to group the log key encodings by the modules they reside in. This process greatly reduces the granularity of the log encodings, and in turn reduces the noise present in the sequential data. In the case of Lustre, a well known and widely used PFS, the dataset contained a total of 73 unique log key encodings. Following the module encoding process, this number was reduced to just 13 module encodings. This reduction also resulted in a visibly obvious decrease in noise in the sequential data as the module encoding sequence made far fewer jumps during the enitre sequence. In general, this intentional reduction of granularity is meant to improve model performance on the highly irregular PFS logs as a coarse granularity ideally filters out

irregularity resulting from noise that may otherwise be confused with anomalous behaviour.

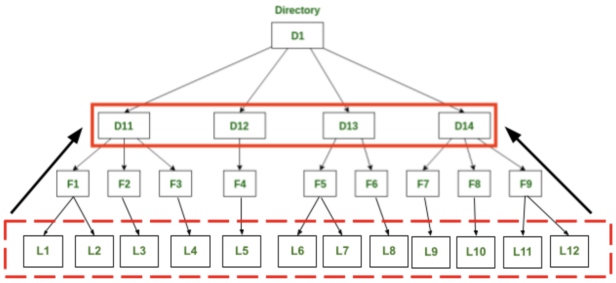

Fig. 1: Module encoding process

As shown in Figure 1, the module encoding process simply consists of generalizing log key encodings which originate from file system source code files to higher level folders or modules. This idea can be generalized by exploring further encoding options such as encoding log keys by the files they reside in or in the case of deeper repositories, by even higher level modules. The choice to encode our dataset by the lowest level of modules was due to the fact that it provided a great balance of reducing noise and retaining information

B. Sequential model

The second part of ModuleLog is the sequential model. This model is largely based on the one proposed by Deeplog. The architecture of this model (as outlined in Figure 2) is broken down as 2 layers of LSTM blocks connected to a fully connected layer. This results in a probability distribution among all of the encoded modules, of which the most likely candidates are selected. If the next log in the sequence is not part of the predicted candidates, the log is treated as an anomaly.

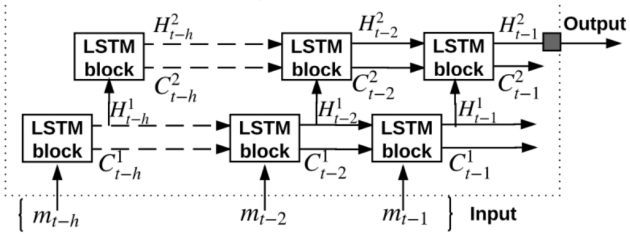

Fig. 2: Sequential model architecture

In the sequential model, there are a few parameters which may be tuned in correspondence to the file system being used. As some file systems may have deeper source code than others, the level of generalization may need to be tuned to provide optimal results. Additionally, the window size, which is used to specify how long the input sequence to the sequential model is may need to be tuned based on the frequency and irregularity of the output logs. Finally, the number of candidates used to represent a normal output based on the prediction output distribution needs to be tuned in order to scale appropriately to the amount of available log keys. For reference, the model seems to perform well when the number of candidates is equal to approximately one third the amount of total encodings.

IV. Evaluation

ModuleLog was initially evaluated against Lustre, a well known Parallel file system which is in use by more than half of the top 100 supercomputers in the world. The results shown in figure 3 are thereby significant as the proposed approach beats State of The Art methods on the given dataset.

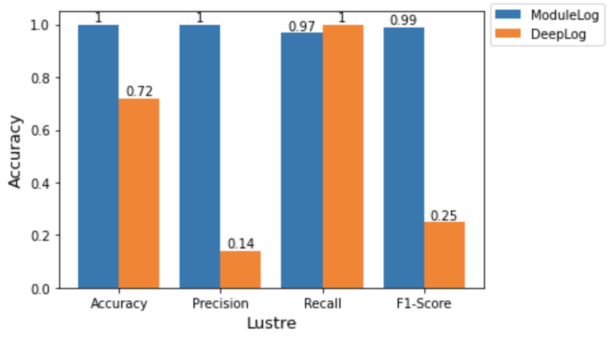

Fig. 3: Lustre evaluation results

In part, the success of ModuleLog on the Lustre dataset can be explained by the distribution of anomalous logs in the source code. Essentially, in the Lustre source code, it is the case that when tasks are running normally, they tend to reside in similar areas of the source code with high variability among log keys, but low variability among modules. In the case of an anomaly however, the execution path changes drastically, even introducing variability among modules which the sequential model is easily able to pick up on. This more drastic change between normal and abnormal logs compared with

log key based encodings is due to just the right amount of generalization or reduction of granularity as this process is effectively able to reduce the inherent noise of log key encodings without losing information regarding anomalies. This is further supported by DeepLog's low Precision as DeepLog is grossly over-predicting anomalies due to the high degree of noise and irregularity of log key sequences in the Lustre dataset.

However, Lustre's source code characteristic of drastically changing the execution path when an anomaly is present is not representative of all file systems. As shown in Figure 4, the results of ModuleLog are not as good when evaluated against logs originating from HDFS. This is due to the structure of the source code, as some anomalies do not take vastly different execution paths. Instead, they often originate from within the same module or even file as they occur. This is obviously a challenge for ModuleLog's approach as reducing granularity may in fact lose some of the information regarding anomalies. It must be noted, however, that HDFS is not a PFS, but rather a Distributed File System (DFS), and the reason for evaluating on this dataset was to test ModuleLog's generalizability to different types of file systems.

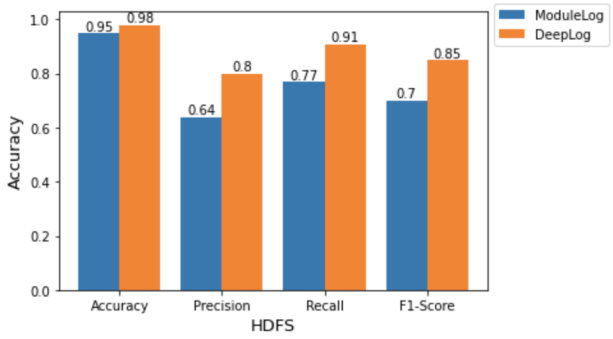

Fig. 4: HDFS evaluation results

V. Conclusion and Future Work

This paper presents ModuleLog, an anomaly detection tool for Parallel File System Logs based on time series analysis. This work is unique from previous solutions to File System anomaly detection as it aims to coarsen the granularity of individual log keys by generalizing their encodings to the modules they reside in based on the source code. This effectively is meant to reduce the noise of PFS logs without losing any information regarding the location of anomalies. When evaluated against a dataset originating from Lustre, the most common file system among modern supercomputers, ModuleLog did just that, as the underlying LSTM model was easily able to predict anomalies based on a sequence of logs. This proved ModuleLog's ability to provide State of the Art on representative Parallel File Systems, showing massive improvement over previous methods, such as DeepLog. However, when applied to further file systems, specifically distributed file systems, like HDFS, the results of ModuleLog do not quite match up. This is primarily due to the nature of the HDFS source code as anomalous logs often reside in the same module or even file, which is in contrast with Lustre, where anomalies tend to lead to vastly different execution paths and log origins.

A. *Future Work*

With the goal in mind to be able to generalize ModuleLog more effectively across more file systems, there is some future work which must be completed. First, it is important to gather more labelled data. While there are plenty of unlabelled sets of file system logs available and easily created, it is challenging to accurately label logs or log sessions as anomalies while they are being created. One method to do this is to manually inject faults in the system, but this runs the risk of the created logs not being representative of real world logs, with few ways to prove that they are across the board. Another way is for experts to manually inspect log keys and sessions to label them as anomalies. This method also leaves room for error as some types of anomalies have a chance to be overlooked given the general volume of File system logs. Additionally, this method is time intensive and arduous. The second area of future work is further analysis of HDFS logs and source code to hopefully find more ways to reduce granularity but without losing important information. This will require intensive analysis of HDFS log session sequences, log producing files, and individual log keys. In extension of this, it could also be beneficial to evaluate ModuleLog against additional file systems as some ideas may become more clear when applied back to HDFS logs. Also,

more file systems would allow for more detailed testing of ModuleLog's robustness when applied further file system source code architectures.

REFERENCES

[1] Marcello Cinque, Domenico Cotroneo, and Antonio Pecchia. Event logs for the analysis of software failures: A rule-based approach. *IEEE Transactions on Software Engineering*, 39(6):806–821, 2012.

[2] Biplob Debnath, Mohiuddin Solaimani, Muhammad Ali Gulzar Gulzar, Nipun Arora, Cristian Lumezanu, Jianwu Xu, Bo Zong, Hui Zhang, Guofei Jiang, and Latifur Khan. Loglens: A real-time log analysis system. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1052–1062. IEEE, 2018.

[3] Min Du, Feifei Li, Guineng Zheng, and Vivek 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*, pages 1285–1298, 2017.

[4] Haixuan Guo, Shuhan Yuan, and Xintao Wu. Logbert: Log anomaly detection via bert. 2021.

[5] Stephen E Hansen and E Todd Atkins. Automated system monitoring and notification with swatch. In *LISA*, volume 93, pages 145–152, 1993.

[6] Jian-Guang Lou, Qiang Fu, Shengqi Yang, Ye Xu, and Jiang Li. Mining invariants from console logs for system problem detection. In *USENIX Annual Technical Conference*, pages 1–14, 2010.

[7] Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, et al. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *IJCAI*, volume 7, pages 4739–4745, 2019.

[8] Alina Oprea, Zhou Li, Ting-Fang Yen, Sang H Chin, and Sumayah Alrwais. Detection of early-stage enterprise infection by mining large-scale log data. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 45–56. IEEE, 2015.

[9] Sudip Roy, Arnd Christian König, Igor Dvorkin, and Manish Kumar. Perfaugur: Robust diagnostics for performance anomalies in cloud services. In *2015 IEEE 31st International Conference on Data Engineering*, pages 1167–1178. IEEE, 2015.

[10] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael Jordan. Online system problem detection by mining patterns of console logs. In *2009 Ninth IEEE International Conference on Data Mining*, pages 588–597. IEEE, 2009.

[11] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 117–132, 2009.

[12] Di Zhang, Dong Dai, Runzhou Han, and Mai Zheng. Sentilog: Anomaly detecting on parallel file systems via log-based sentiment analysis. In *13th ACM Workshop on Hot Topics in Storage and File Systems*, 2021.