PoS PROCEEDINGS OF SCIENCE

A Blueprint of Log Based Monitoring and Diagnosing Framework in Large Distributed Environments

Yining Zhao[1]
Computer Network Information Center, Chinese Academy of Sciences
No.4 Zhongguancun Nansijie, Beijing, China
E-mail: `zhaoyn@sccas.cn`

Xiaodong Wang
Computer Network Information Center, Chinese Academy of Sciences; University of Chinese Academy of Sciences
No.4 Zhongguancun Nansijie, Beijing, China
E-mail: `wangxd@sccas.cn`

Haili Xiao
Computer Network Information Center, Chinese Academy of Sciences
No.4 Zhongguancun Nansijie, Beijing, China
E-mail: `haili@sccas.cn`

Xuebin Chi
Computer Network Information Center, Chinese Academy of Sciences; University of Chinese Academy of Sciences
No.4 Zhongguancun Nansijie, Beijing, China
E-mail: `chi@sccas.cn`

Distributed systems have kept scaling upward since this concept appears, and they soon evolve to environments that contain heterogeneous components playing different roles, making it difficult to understand how the large environment works or if any undesired matters happened from security point of view. Logs, produced by devices, sub-systems and running processes, are a very important source to help system maintainers to get relative security knowledge. But there are too many logs and too many kinds of logs to deal with, which makes manual checking impossible. In this work we will share some of our experiences in log processing and analyzing. We have summarized some common major steps that appear in most of the existing log analysis approaches, including log selection, log classification, information analyses and result feedback. We also represent a general framework that monitors events, analyzes hidden information and diagnoses the healthy state for large distributed computing environments bases on logs. Although the framework we initially designed was for the maintenance for CNGrid, its process is adaptable to other distributed computing environments.

International Symposium on Grids & Clouds 2019, ISGC2019
31st March - 5th April, 2019
Academia Sinica, Taipei, Taiwan

[1] Speaker

© *Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0).*

1. Introduction

Distributed systems have grown larger and larger since this concept appears, and as the scale of network kept expanding, distributed systems soon evolve to large environments that may composite of machines from all over a country, or even throughout the world. These large distributed environments may contain heterogeneous components playing different roles, e.g. data centers and computing units, and human users access these computing resources from separated client devices. From security point of view, such large distributed environments bring new challenges, as it is an extremely difficult task to get an idea of how a large environment is working or if there are any undesired matters happened in the environment.

Logs, produced by devices, sub-systems and running processes in computer systems, are a very important source to help system maintainers to get relative security knowledge. Nowadays nearly all released programs and services use logs to record events that happen at runtime. But there are too many logs and too many kinds of logs to deal with, which makes manual checking impossible. Therefore many researchers have been working on log processing and analyzing methods to help humans simplify this job and automatically find information hidden in logs.

After reviewing a number of works related to log processing and analyzing methods, we have discovered some common steps that can be found in almost all these methods. There are 4 major steps in log processing works: 1) identifying necessary logs that are produced by key modules in the system or environment; 2) classifying these logs either using variables or non-variable contents to obtain a set of log types; 3) performing analyses from various angles such as user behaviors, type associations, etc., using data mining and machine learning techniques with the help of the obtained set of log types; 4) gathering results from previous analyses and find some metrics that can help system maintainers locate and solve problems, improve the provided services, and produce a better represented display with techniques such as visualization.

We will also introduce our practices on log processing and analyzing works that have been carried in CNGrid [10] to demonstrate these 4 steps. CNGrid is a good example of a large distributed environment, as it is the national High-Performance Computing environment in China, currently consists of 19 HPC clusters contributed by many research institutes and universities. First, we pick up system logs and middleware logs from CNGrid as the target to be analyzed, according to our demands of responding to system failures, defending on malicious attacks and providing user services. Second, log pattern extraction algorithms [22] are used to classify types for system logs and middleware logs. We organize these log types as the log library and implement interfaces to access it. Third, by applying the log library, a number of log analyzing applications are developed to perform analyses, including log flow detection, fault prediction bases on log type associations and user behavior analysis. Finally, results of these analyses would be turned into some measurable factors, so that they can be combined to reach an overall conclusion. We plan to weight these factors in order to make the produced diagnosis report more adequate to describe the vulnerability level of the environment.

In Section 2 of this paper, we will talk about some related works on log processing and analyzing. In Section 3 the common steps in log processing is explained with more details. We will introduce our practices in CNGrid in Section 4.

PoS(ISGC2019)033

2. Related Works on Log Processing

Log processing and analyses are a commonly used method in fields of abnormal detecting, fault locating and performance analyzing in computer science. Yuan in his work [1] has shown the importance of logs and how to adjust log configurations to better utilize logs. Oliner and co-mates [8] generally explains the reason, applying fields and difficulties of log analysis.

There are plenty of works about processing logs. Zhao proposed Stitch [2] that is a tool to build the System Stack Structure Graph using logs to understand the execution flows of distributed software and provide helps for system performance analyses and fault locating. Another contribution made by Zhao is called Log20 [4], which can analyze program traces and best optimizing log printing statements in codes. This is done by constructing control flow graphs and finding crucial blocks in different parts of codes. Zhang introduced Pensieve [3] which also analyze program codes, trace and reproduce the error to locate failure points in codes when there is a log representing the failure. Pensieve uses the concept of event chain to back-trace the sources and conditions that cause failures.

Vaarandi worked on mining frequently appeared patterns from logs [5][6]. This was done by finding all frequent combinations of words using Apriori algorithm and depth-first algorithm. There are also tools implemented this approach to print the found frequent log patterns. Xu proposed a way of automatically detecting anomaly logs [7]. He converted logs into feature vectors and used the algorithm of PCA to distinguish normal and abnormal logs. His work also included a decision tree generator to visualize and explain the auto-detected result. Peng shows an analyzing method of log association [9]. This method classifies logs in Naïve Bayes algorithm and hidden Markov model. Then the visualization is applied to show the associations among log types.

Logs that record user operations are commonly used in analyzing user behaviors, so that service providers can improve their product designs and layouts. There are many works that retrieve data from web server logs that record user actions to provide accurate information to users. Wu [13] has proposed a method to enhance recommendation systems by turning user sessions into graphs and applying graph neural network. Zhou and mates [14] proposed the Deep Interest Network to learn user interest from history behaviors to show more related advertisements. Hassan [15] models user behavior records to predict whether the search query is successful or not. In other works, Müller [16] studied user interaction log files to learn feature weights to improve the results of content-based image retrieval systems. Anderson [17] studied how badges of online communities can influence user behaviors and keep users paricipate site activities.

There are also tools and software that are designed to help log processing, and one of the most famous products is the combination of ELK, which consists of Logstash, ElasticSearch and Kibana (and Beats) that are all developed by Elastic [18]. These tools can construct a complete chain that providing functionalities from log collecting, data storing and indexing to data visualizing. There are many works that use the ELK combination to achieve the goal of data processing [19][20][21]. However, ELK only provides a basic data processing workflow, and it still need system maintainers to make effort on further analyses.

PoS(ISGC2019)033

3. Common Steps in Log Processing

By learning approaches introduced in the previous section, we have found that there are some common clues that are shown in many works. After summarizing and abstraction, 3 major steps emerged and we believe these are essential parts in almost all log processing workflows. These steps are: log classification, information analysis, and result feedback.

However, the log analyzing approaches mentioned in the previous section are mostly aiming to deal with a single matter, usually with an explicit purpose or requirement. In this article we are discussing logs in large distributed environments, which differ from the above situations. In a large distributed environment there might be much more issues to be deal with, and the required information could be scattered in many different log files. Therefore we need to add one more step in front of the other three steps, and this step is log selection. See Fig.1 for a sketched view of the common steps.

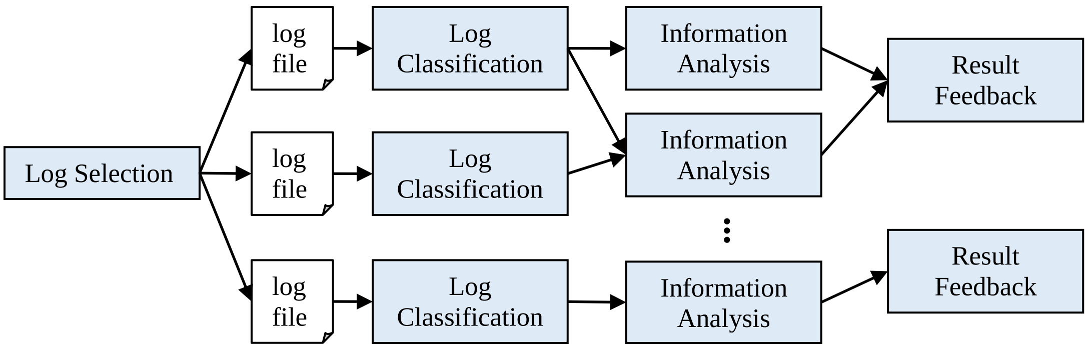

Fig.1 A Sketch of Common Steps in Log Processing

3.1 Log Selection

Before any further operations, log selection is the first thing you need to do when analyzing logs in a large distributed environment. This step includes some necessary considerations, and how to select analyzing targets heavily relies on the purpose of the analysis to be launched. There usually are plenty of log files produced by different services and applications. System maintainers should have the knowledge of what these services and applications work for and what are expected to be output in their log files. Hence when performing an analysis, only relative log files are needed to be selected and processed in following steps.

There are two issues that should be considered when selecting log files: 1) the aim of log analyses; 2) the necessity of logs and the difficulty of analyses. Assuming we would like to analyze user access behaviors to the environment, apparently logs generated by devices and programs that provide user services are the prime target to be selected, and we possibly will also looking at the network security logs. In another case, if we want to inspect the usage history of computing units, the system logs on those computing units and logs of any monitoring tools would be good choices. In the meanwhile the user access logs are not likely to be chosen in this case, as they are not directly relate to our aim and hard to interpret, even if they somehow reflect the usage.

PoS(ISGC2019)033

This step is largely depend on human experiences. Only human can identify and decide which log files are to be selected so that the essential information can be obtained. We may possibly be helped by machines to find those frequently accessed large log files, as they are very likely to contain more and useful information. But be aware that it still need humans to determine whether these log files are applicable when performing log analyses.

3.2 Log Classification

Log classification is an indispensible part in the process of log analysis. Almost all works on logs contain such a step. The reason for log classification is that the amount of logs is extremely huge, and it is impossible to manually check each log line by line. Hence it is necessary to find a solution to turn the large number of logs into a relatively small number of log types.

Depending on the feature selected for classifying logs, we can divide log classification methods in two major approaches: variable and non-variable. Assuming there is a line of log "Session opened for user Alice from 192.168.1.1". Two parts of 'Alice' and '192.168.1.1' represent variables, while the other part 'Session opened for user * from *' can be recognized as non-variable information, which statically generated by program codes.

The non-variable approach classifies logs by the static information part. This approach can explain 'what events can happen in the case' to a large extent. System maintainers use this approach to obtain log types, and can made analyses and feedbacks on those log types. For example, people may want to know generally that kinds of events happened in the system and the percentage information of each kind, or maintainers want to receive an alert when a certain type of log appears. There are two options to get log types through non-variable approach: summarizing from existing log records, or learning from source codes. Vaarandi's work [5][6] adapted the formal option by finding all frequent items in log records, and produces a set of frequent log patterns bases on non-variable content. Peng [9] also choose this option by using the Naïve Bayes classifier to group log records into a few major categories. Xu [7] classifies logs with the latter option, as he obtained all log print statement from source code or decompiled programs. Note that the option to get log types from codes is only feasible for open-source software or programs that have decompilers available.

The variable approach classifies logs using variables in logs. Examples of types of variables include username, time period, IP address or hostname, thread ID, storage block ID, etc. This approach is particularly suitable for building an event chain for a certain object such as users, explaining 'what has happened to that object'. Stitch [2] extract variables from logs and interpret them into IDs (IP, process ID, thread ID, filename, etc.), and use the System Stack Structure Graph to build a hierarchy of these ID types. Logstash [18] can use the *grok* block to matching non-variable content by regular expressions, and obtain variable values to be further analyzed.

Note that these two approaches are not mutual exclusive. For those log classifiers that base on variables, the static information of logs is still essential, as one would need to know what kinds of events these variables are actually happening. The non-variable classifiers help to understand what events happened in the system, but it also need variable information to know who/what causes the happened events.

PoS(ISGC2019)033

3.3 Information Analysis

Information analysis is the core step of the general log processing work. It affects the strategy combinations of all the other steps. There are many methods and algorithms from data-mining and AI fields that can be applied in the information analysis step, and the selection largely depends on purposes or needs of the overall log processing. Common purposes of log processing include but not restrict to:

- **Program code analysis and improvement**. Logs produced by print statements are essential for tracking traces and conditions of programs. A better designed print statement principle provides better information for analyzing the running program, thus helpful for avoiding failures and deadlocks caused by coding. Pensieve [3] and Log20 [4] are good examples of using logs to help modify codes to achieve better running performance and faster failure locating.
- **System performance analysis**. This usually relates to large scaled distributed computing tasks. In such tasks a number of devices and processes are involved, and low performance in one point may cause the degradation of overall performance. By analyzing logs from related devices and processes, we could find the point that causes the degradation, and even the solution for upgrading. Stitch [2] can provide such analysis. This topic also includes the observation of hardware and software performance history that derived from logs, and conjecture about reasons for performance changes.
- **Failure locating and prediction**. In a large distributed environment it is usually difficult to find the root cause when an anomaly or fault happens. Logs can help locating failures by gathering relevant information, and infer sources of failures from variables of the gathered failure logs. It is also possible to make predictions of potential failures through log analysis, and this could be done by analyzing sequences of log types. If certain log types are confirmed as indications of failures, then when their predecessors appear, the prediction that potential failures exist. Another way of failure prediction could be made by analyzing the system monitoring logs, such as high CPU temperature, low disk and low memory prompts.
- **User behavior analysis**. The concept of user behaviors in this context means compositions of user operations. It is believed that users using computing resources in large distributed environments have general patterns, which are related to workflows of performing computing tasks or formed by user habits. The aim is to summarize behavior patterns for users so that we can find if there are any improving spaces for user services [13][14][15]. Or we may try detecting any cases that some users behave extraordinarily distinct from the normal behavior pattern, as this implies user anomalies, which means there are possibly security problems appeared.
- **Hidden information and relation discovering**. There might be some information that is not directly shown in log contents. A good example is the hidden association among log types, e.g. the log types 'failed to send an email' and 'low disk space' may not look strongly related at the first glance, but in some systems it is the latter event causing the former one, thus when system maintainers find email failures they would remember to check disk spaces. Log flow is another kind of hidden information. Assuming there is

PoS(ISGC2019)033

a log type that does not indicate failures by itself, but when it keeps happening in a short period of time there is clearly an anomaly.

- **Event alert and response**. This majorly related to log monitoring. If some log types are marked as indications of failures, we can configure to generate alerts when logs of those types appear. We may also set some scripts that will be actively executed when the designated logs appear, e.g. banning user IP for a short time if there is a log indicating the user has failed in password attempt for several times.

3.4 Result Feedback

The step of information analysis will produce some results, but in most cases they are not applicable directly. Output results may be sets of numbers, classified groups of data, models of data processors, etc. These are difficult to understand or explain by human beings, and this is especially true when black box methods are used. Principle Component Analysis and Deep Learning are good examples. In order to make the analyzed results usable and provide valuable feedback to improve performance and service levels of the system, it is necessary to put some effort on reasoning and applying of results.

A very common method in this field is data visualization, which could turn numbers into shapes and colors that show more impacts to human's vision. Stitch [2] creates a hierarchical timeline and labeled each log event as a colored circle so that it is easy to find performance bottlenecks and failure points. Xu [7] draws decision trees to explain conditions that determine the classified result of normal logs and anomalies. Peng's 3D cube view shows three factors Time, State and Component in one plot [9]. Kibana itself is a visualization tool that provides different charts and diagrams to make data easily understood, and it also provides a dashboard view for users to customize their visualization layouts [18].

It is also important to improve systems according to the result of log analyses. For those code checking analyses, we should modify any code blocks that could possibly cause performance bottleneck or system failures. For those system failure analyses, we need to fix faults and patch any vulnerable points to upgrade security level. For any user behavior related analyses, we can improve user services while defending systems from malicious attackers.

4. CNGrid Practices

In this section we will introduce how we applied the log processing techniques mentioned in previous sections to CNGrid. CNGrid is the national high-performance computing environment in China. It consists of a number of computing clusters and storages contributed by research institutes and universities all over the country, and have provided quality scientific computing resources to users from many research fields include high-energy physics, computing chemistry, biomedical research, astronomy, atmospheric physics, mechanic engineering, etc. The SCE middleware [11], which is developed by the Computer Network Information Center (CNIC) of Chinese Academy of Sciences, is the core task scheduler and service provider in CNGrid. SCE integrates computing resources from all nodes in CNGrid, and provide unified accessing clients and web portals to users. In addition, there is also an interface called SCEAPI that helps developers to build communities for special scientific fields.

CNGrid is a typical large distributed environment. There are currently 19 nodes in the environment and they use heterogeneous batch processing and storage systems. Data exchanges

PoS(ISGC2019)033

through high-speed networks, while SCE provides login server as the main entrance to computing resources for users. Our team is responsible in taking the maintenance work for CNGrid, and log processing and analyzing is an important part. In the following subsections we will show how we use our understandings of log processing that described in the previous section to build a diagnosis framework for a large distributed environment.

4.1 Design of the Framework

The framework of log processing in CNGrid is illustrated in Fig.2. To start with all the processes, we first decide to use two major types of logs as our information sources: syslog and SCE traces. Syslog is generated by the rsyslog service of Linux operating systems. It covers most OS-level events, plus records of some programs that inject their logs into syslog files. Syslog is split into several files, each stores logs for a special category. In our context, we mainly interest in syslog files of cron, maillog, secure and messages. Cron records events for crond service that periodically executes designated commands. Maillog records events that relate to email services. Secure contains events of local and remote user accesses to the system. Messages is the file for general information, where all events that are not recorded in other syslog files would be written in the file of messages. SCE traces are logs generated by the SCE middleware. This records all events at the environmental level, including message exchanging among local servers and central servers, user operations, status of submitted jobs and computing resources, and so on.

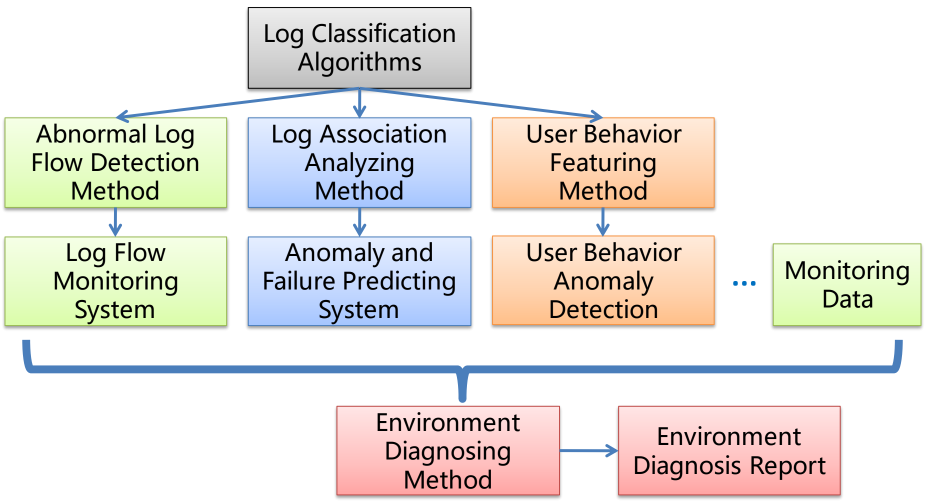
Log Classification Algorithms
Abnormal Log Flow Detection Method
Log Association Analyzing Method
User Behavior Featuring Method
Log Flow Monitoring System
Anomaly and Failure Predicting System
User Behavior Anomaly Detection
...
Monitoring Data
Environment Diagnosing Method
Environment Diagnosis Report

Fig.2 CNGrid Log Processing Framework

Logs from distributed servers are first collected into the log storing and processing device. This can be done by any proper data transfer tools, and we use a combination of Logstash and Apache Kafka. We have constructed a system called ESPADS [23] for events collecting, filtering and distributing. ESPADS is expected to serve multiple applications for different purpose including event grouping, event prompting, behavior tracking, etc. In this case functions of ESPADS are involved in our general framework for log processing.

After we have obtained events from devices in CNGrid, the log classification is carried out to obtain the set of log types that identifying all kinds of events that happened in the environment. For system logs we use the non-variable classifying methods called log pattern

PoS(ISGC2019)033

extraction algorithm [22] to generate log types. For SCE traces which only contains variables, we construct a data cube that stores information of all traces and can produce log types for any variable types.

Then we can perform the analytic work according to our requirements. The individual analyses will be demonstrated in the next subsection, but in general each of them is set to return a result representing the evaluation from a certain dimension. Finally, these evaluation results will be gathered and integrated into an overall evaluation or diagnosis of the environment.

4.2 Log Analysis Methods and Applications

CNGrid is an open HPC environment that provides computing resources to users. To maintain such a large distributed environment, it is necessary to look at different aspects from different angles. At the current stage, we have designed and implemented several log analyzing methods to monitor and represent the status of the environment, and try retrieving some hidden information or relations that cannot be easily discovered. They may need distinct preprocesses from raw logs, but still follows the common steps described in the previous section. In this paper we select three of the analyzing methods and briefly explain how they work in CNGrid.

4.2.1 Log Flow Anomaly Detection

This is an analysis and monitoring method mainly about syslog. The idea is that we assume that in the majority of time when the environment is running properly and smoothly (i.e. in normal cases), the number of logs for each classified log type should be kept in a nearly stable level in a unit time. But in some cases where anomalies happen, the number of appearances for certain log types may change dramatically. Note that log types that are in abnormal conditions may not represent anomalies by themselves. For example, the log type 'User xxx established a connection to the server' do not stand for abnormal, but when logs of this type repeatedly appears for a great number of times in a few minutes, it is highly possible that something wrong happened. Since this is not easy to be captured by direct log monitoring bases on log types, the method that bases on log flow is therefore introduced.

PoS(ISGC2019)033

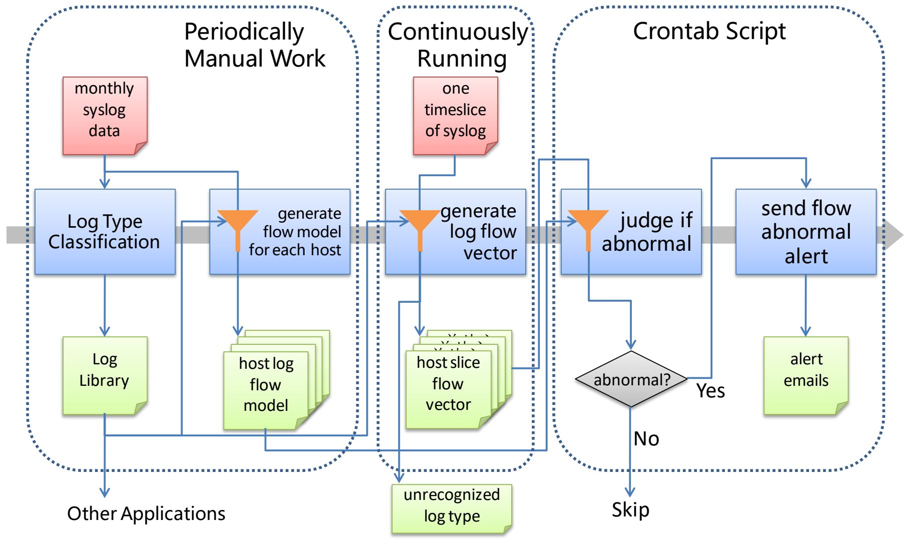

Fig.3 Workflow of Abnormal Log Flow Detecting System

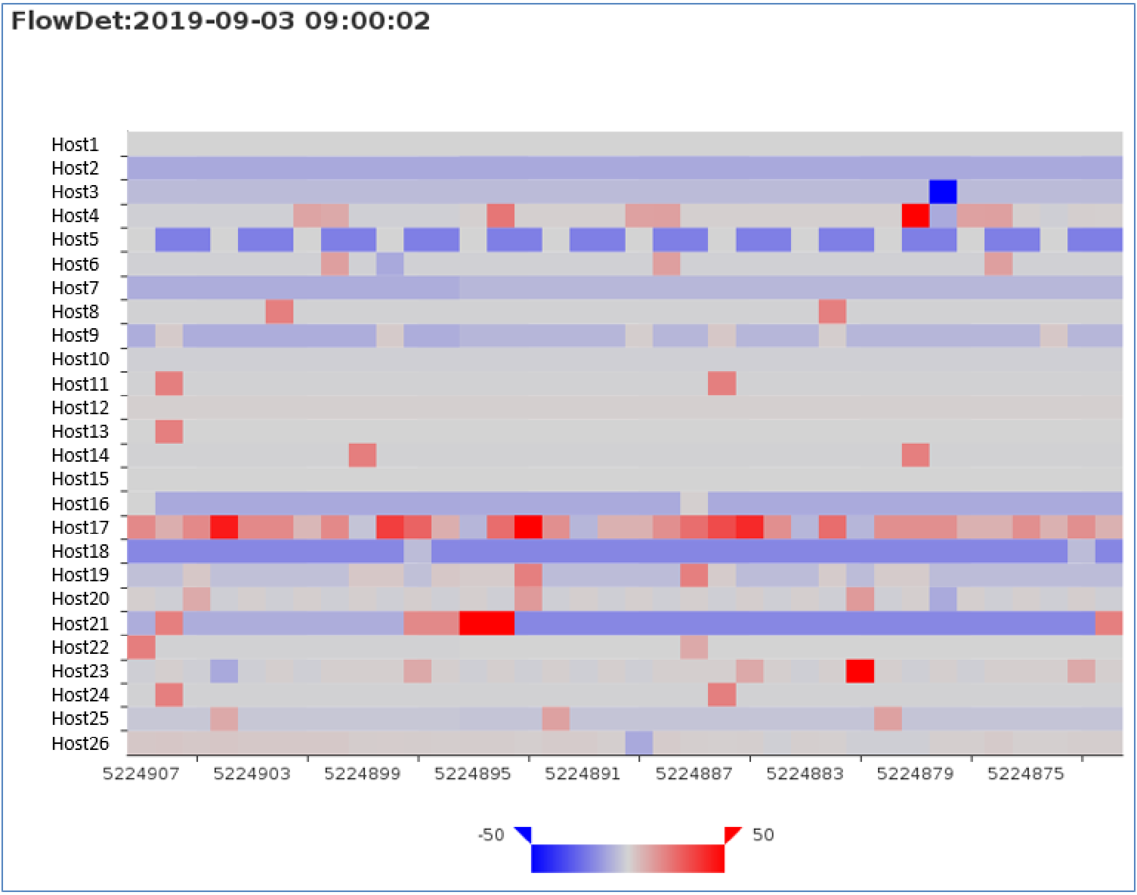

Fig.4 Examplified Visualization of Abnormal Log Flow Detecting System

We first cut history log records in to a number of time-slices, each contains logs in a unit time that is set to be 5 minutes. Then for each time-slice we make a statistics about the number logs in each log type, so time-slices are turned into slice-vectors having the form '[n_1, n_2, n_3, …, n_m]', where n_i represent the number of appearance for log type i in this time-slice. After that, the algorithm of Principle Component Analysis (PCA) is applied to these normal slice-vectors to build a general model representing the normal running status of CNGrid. There is a program recording the latest slice-vector for every unit time i.e. 5 minutes, and this new slice-vector is examined by the general model. A conclusion of whether this time-slice is normal or abnormal is generated. And if abnormal, an alert email including detailed information and the level of this anomaly is sent to our maintainers. The general workflow of this system is shown in Fig.3.

We have applied the abnormal log flow detecting system with our log records, and used some visualization techniques to illustrate the results. Fig.4 shows an example view of the visualization work for the system. Each row represents a host in the environment of CNGrid, and each column represents a single time-slice. Red color denotes an increase of log flow on that host, while blue color denotes a decrease. The content in the visualization is automatically reloaded every 5 minutes (i.e. after each new time-slice). By this effort we can see the log flow changes in the past 3 hours for all reported hosts in one sight, so that the system maintainers can rapidly find when and where those anomalies happened.

Details of our log flow abnormal detection method is illustrated in another work [12].

PoS(ISGC2019)033

4.2.2 Anomaly Prediction bases on Log Type Association

In many cases, events happened as chains in a system, where an event triggers another, or a change in configurations causes a series of faults. From log view, these relationships among logs are not explicitly shown, but they can be revealed through data mining techniques like association analyses. In our work, we perform statistics on the preceding and following events for each log type and identifying the time distribution for all other log types near it. The aim is to find potential candidates of causes of anomalies, which happens before abnormal log types with a small time interval. If these causes also have significant preceding events, it could build the event chain regarding to anomalies. This approach can find cases where there are multiple events mutually interfering and causing the anomaly or failure, and hence suitable for large environments that have a number of interacting components.

Once event chains on log types are built, it is clearly an advantage on anomaly prediction. A library of anomaly event chains can be found bases on the association analysis, and then we can monitor logs to be aware of any events that can be the beginning of chains forwarding to anomalies, and send alerts to system maintainers. There is a problem that some preceding events of anomalies are frequently appeared normal events, thus there might be a large number of false prediction alerts. To solve such a problem, we need to make better configurations and argument adjustments, which requires further research.

4.2.3 User Behavior Featuring and Prediction

As an open HPC environment, CNGrid keeps receiving operation commands through its client or portal from users. To understand users and their behaviors can help maintainers to provide pertinent user services or find anomalies caused by users. In our context, we care more about correct accessing and using of computing resources. This includes two points: 1) helping normal users to use resources in a better way; 2) discovering undesired user behaviors and preventing malicious actions.

To achieve these goals, we have analyzed user behavior features from multiple angles. By examining the frequency of user accesses and numbers of operations in logs, users are clustered into several groups with distinct customs, and this can help us to provide advices to users accordingly. In another approach, we classified user operations as a number of operation types, and use the deep learning method to build behavior models for each user. These behavior models will be used to make predictions of user operations. If a user behaves significantly different from his/her past behaviors or different from most normal users, there is a high probability that: a) the user changes the way he/she usually behaves, and there should be a reason for that, which we may be care about; b) somebody cracks into this user’s account, in which case we must take actions; c) some exceptions happened in the environment and forced the user to act differently. In any possible case, system maintainers need to review the cause and take responses.

Early results over the user operation logs shows that the learning model has a high accuracy of over 90% when the number of operations in a single session is less. But for those user sessions with long sequences of operations, the accuracy may drop to 60%-70%. We are examining different machine learning algorithms and compositions to find a model that best fits the situation that included all categories of user behaviors.

PoS(ISGC2019)033

4.3 Diagnosis Generation

There are several log analyzing methods designed and implemented, and each has a separate result. These results reflect issues from different angles, and it is not easy for system maintainers to look through all of them to get a general conclusion of how the current status of our environment is. Therefore we would like to find an approach to integrate these analyzed results to produce a general diagnosis.

We aim to build such diagnosis reports with visualization and quantization techniques. For each analyzed result, we use colored diagrams to display the status and numbers contained in the result. For example, in the log flow analysis, the result would be the difference of the expected number of log flow versus the actual number in the given time unit. We can define the level of anomaly to be the quotient of the two numbers. This value will be integrated into the overall level of anomaly for the environment. We can also turn this level of anomaly into a colored square, and if we place a series of colored squares as an array by the order of time units, it is much easier to observe the historical trend of log flows. This could be done to other analyses as well, e.g. we could define the deviation of a user action from normal behaviors to be the level of anomaly in user behavior dimension, and contribute to the overall value with an adjusted weight.

For a general diagnosis of the environment, there would be an overall value sketchily reflect the level of anomaly for the whole environment. Below that there are a number of panels with detailed and visualized analysis results. For those severe abnormal parts in the result, there would be more information to help system maintainers to fast locate the source of the anomaly. We expect such display of diagnosis can greatly help us to understand the running status of our environment and decrease the reaction time to deal with anomalies.

5. Conclusion

In this paper we have reviewed a number of log processing and analyzing works contributed by other researchers and summarized some common techniques during the process of log analyses. Four common steps including log selection, log classification, information analysis and result feedback are explained with possible options that can be made in these steps, and pointed out how the related works adopt the common steps and what decisions they have made when facing those options.

We also introduced some practice efforts that have been applied in the Chinese national HPC environment of CNGrid. We aims to build a general framework for analyzing logs that generated from CNGrid and provide various analyses from different angles, so that the produced results can help us to improve the maintenance work and service of the environment. Finally results from these analyses are supposed to be combined to generate diagnosing report that can generally represent the running status of CNGrid and shows any anomalies or potential threats. This is a goal that has yet to be achieved, and we are building the general framework step by step.

The work described in this article is practiced in CNGrid but it can be adapted to other large distributed environments.

PoS(ISGC2019)033

Acknowledgment

This work is supported by National Key R&D Program (2018YFB0204002) and National Natural Science Foundation of China (61702477).

References

[1] Ding Yuan, Soyeon Park, Yuanyuan Zhou. *Characterizing Logging Practices in Open-Source Software*. Proceedings of the 34th International Conference on Software Engineering. IEEE Press, 2012.

[2] Xu Zhao, Kirk Rodrigues, Yu Luo, Ding Yuan, Michael Stumm. *Non-intrusive Performance Profiling for Entire Software Stacks based on the Flow Reconstruction Principle*. Usenix Conference on Operating Systems Design & Implementation. USENIX Association, 2016.

[3] Yongle Zhang, Serguei Makarov, Xiang Ren, David Lion, Ding Yuan. *Pensieve: Non-Intrusive Failure Reproduction for Distributed Systems using the Event Chaining Approach*. The 26th ACM Symposium on Operating Systems Principles, 2017.

[4] Xu Zhao, Kirk Rodrigues, Yu Luo, Michael Stumm, Ding Yuan, Yuanyuan Zhou. *Log20: Fully Automated Optimal Placement of Log Printing Statements under Specified Overhead Threshold*. The 26th ACM Symposium on Operating Systems Principles, 2017.

[5] R. Vaarandi, *A data clustering algorithm for mining patterns from event logs*. Ip Operations & Management, IEEE, 2003:119-126.

[6] R. Vaarandi, *A Breadth-First Algorithm for Mining Frequent Patterns from Event Logs*. Intelligence in Communication Systems, Springer Berlin Heidelberg, 2004:293-308.

[7] Wei Xu. *System Problem Detection by Mining Console Logs*. PhD Dissertation. University of California at Berkeley, 2010.

[8] Adam Oliner, Archana Ganapathi, Wei Xu. *Advances and Challenges in Log Analysis*. ACM, 2011.

[9] Wei Peng, Tao Li, Sheng Ma. *Mining Logs Files for Data-Driven System Management*. Acm Sigkdd Explorations Newsletter, 2005, 7(1):44-51.

[10] Z. Xu, X. Chi, N. Xiao. *High-performance computing environment: a review of twenty years of experiments in China*. National Science Review Advance Access, 2016, 3(1):36-48.

[11] H. Xiao, H. Wu, X. Chi, *SCE: Grid Environment for Scientific Computing*. Lecture Notes of the Institute for Computer Sciences Social Informatics & Telecommunications Engineering, pp. 35-42, 2009.

[12] Xiaodong Wang, Yining Zhao, Haili Xiao, Xuebin Chi, Xiaoning Wang. *Multi-node System Abnormal Log Flow Mode Detection Method*. Journal of Software, to be published.

[13] Wu Shu, Tang Yuyuan, Zhu Yanqiao, et al. *Session-based Recommendation with Graph Neural Networks*. 2018.

[14] Zhou Guorui, Song Chengru, Zhu Xxiaoqiang, et al. *Deep Interest Network for Click-Through Rate Prediction*. 2017.

[15] A. Hassan, R. Jones, K.L. Klinkner. *Beyond DCG: User behavior as a predictor of a successful search*. Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010. ACM, 2010.

[16] H. Müller, T. Pun, D. Squire. *Learning from User Behavior in Image Retrieval: Application of Market Basket Analysis*. Kluwer Academic Publishers, 2004.

PoS(ISGC2019)033

[17] A. Anderson, D. Huttenlocher, J. Kleinberg, et al. *Steering user behavior with badges*. Proceedings of the 22nd international conference on World Wide Web - WWW \"13 - 2013:95-106.

[18] Products of Elastic, https://www.elastic.co/products/.

[19] S. Bagnasco, D. Berzano, et al., *Monitoring of IaaS and Scientific Applications on the Cloud using the Elasticsearch Ecosystem*. Journal of Physics: Conference Series, vol. 608, 2014.

[20] Alan Hardy, Pere Urbon, *Revealing Insights from Data*, CERN Computing Seminar, May 2015, https://cds.cern.ch/record/2019744.

[21] Qing Zou, *A Novel Open Source Approach to Monitor EZproxy Users' Activities*. Code4lib Journal, vol. 29, 2015.

[22] Yining Zhao, Xiaodong Wang, Haili Xiao, Xuebin Chi. *Improvement of the log pattern extracting algorithm using text similarity*. IEEE 32nd International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2018, p 507-514, August 3, 2018.

[23] Yining Zhao, Haili Xiao. The Design of Event Stream Processing and Distributing System in CNGrid. Proceedings of HPC CHINA 2017, October 19, 2017

PoS(ISGC2019)033