VNU Journal of Science: Comp. Science & Com. Eng., Vol. 32, No. 2 (2016) 43-48

An Efficient Log Management System☆

Tran Van Cuong, Nguyen Van Nam*

Vietnam National University, Hanoi Capital City, Vietnam

Abstract

Server monitoring is really necessary because this can help administrators to track users' activities in order to improve user management ability, load balancing as well as to detect DDoS attacks. Usually, server monitoring is based on logging. However, logging system is always considered to be expensive in term of storage, data collection, data searching and analysis. In this paper, we introduce eLMS, an efficient and scalable log management system. In our system, the log files can be collected from multiple servers, stored in a scalable manner, appropriately indexed and fast analysed. eLMS acts both in online and off-line modes and provides a practical web-based monitoring interface. eLMS is based on a famous open-source core called ELK including three main modules: ElasticSearch, LogStash and Kibana. However, by using queue and streaming technique instead of uploading technique used in LogStash, eLMS is tested to be at least ten times faster than existing ELK.

Received 15 July 2015, revised 25 August 2015, accepted 28 September 2015

Keywords: Log Files, Big Data, Streaming, Index and Search.

1. Introduction

Today, server monitoring becomes important and essential for most of the IT companies and organizations. This can help administrators not only to protect the server from the outside attacks such as DDoS but also to tract customer's activities and hobbies in order to improve individual service providing ability. In fact, server monitoring focuses mainly on log file processing. These are considered to be big data in terms of volume, variety and velocity. In fact, logs exist under different formats such as log file, xml, meta-data, etc. Moreover, log also grows very fast during the time.

The log management is also a big data application which is often based on Hadoop platform. However, Hadoop primarily supports large-scale systems such as Google, Yahoo ... For small and medium-sized systems, Hadoop becomes expensive and impractical to implement. Moreover, Hadoop is not fast enough for on line data processing.

In this paper, we present a novel and efficient log management system (eLMS). This is a light system that can be used for online processing and is really beneficial for small and medium companies and organizations. The system is also extensible for future use in large scale platforms.

eLMS log management system is based on three open source projects: Logstash, Elasticsearch and Kibana of Elastic technology platform. Logstash is used to transport content and structure of data logs. Elasticsearch is responsible for data indexing and providing the ability to full-text searchi

To build eLMS, we gradually optimize different modules from Elastic platform. Firstly, we simplify the data queue of the system to reduce the consumption of computer resources

☆ This work is dedicated to the 20th Anniversary of the IT Faculty of VNU-UET

* Corresponding author. E-mail: van.nam@vnu.edu.vn

such as CPU and RAM. Secondly, we propose several techniques to read log from individual server. Thirdly, we apply some strategies for data indexing in database. Fourthly, we improve mechanisms of filtering, formatting and data tagging.

The rest of our paper is structured as follows. Section II presents background of logs and some difficulties in log processing. Section III provides an overall view of log management systems. In section III, we introduce related works of log processing systems. In section IV, we propose our efficient log management system named eLMS. Our experiments is presented and discussed in section V. Finally, section VI concludes our work and states several future works.

2. Background

2.1. Log overview

Log [3, 6] is a record of events that are occurred within the network system of an organization. This is considered as a black box to reflect the status of the system. It records everything that happens on the system in a detail manner. Log allows us to know what the system has done well as which problems occur.

Log is generated from most of the activities and behaviors of the systems. For example, the events of the router, firewall, database transactions, the attacks, the connections, the activities of the users, etc. There are many sources that can generate logs. Moreover, there are also many log types. For example, the log files of the host such as Unix, Linux, Windows, VMS, etc. is completely different from the log of network equipment such as switch, router. Similarly, the log of information security applications such as firewall, IDS, anti-DDoS equipment, defense systems, etc. is also very different from both host log and network log.

A log line that is generated by computer is often comprehensible by human. The content of most of the logs is structured as follows:

LOG = TIMESTAMP + DATA

where *Timestamp* represents the time when the event happens. *Data* contains the content that includes all the information about the valid and invalid events which was happened in the system [3].

How can we find those which are not allowed and how can we learn the mistakes of the past from log to predict the future? Can we expect to search in every gigabyte of log files to identify those activities that are not allowed to happen? The answer lies in the necessary issues of centralized log processing systems.

2.2. Log life cycle

We can see that there are many centralized log integration products with variety of processing architecture that meet different management needs. Basically, they have the same life cycle [10] as in Fig. 1 that including six main steps:

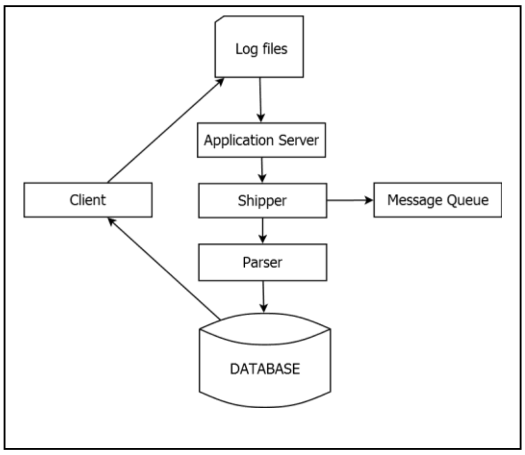

Figure 1: Log Life Cycle.

● Step 1: *The application generates log files*: An event is generated by an application. The life cycle begins when a data log is recorded and stored on a server. In most cases, it will be a single line written to a text file.

● *Step 2: Data collection by shipper:* The logs are collected by the shipper and sent to the message broker. The shipper will look up for log files and retrieve new data to send them to the message broker or push them directly to the parser.

● *Step 3: Logs are temporarily queued:* Since logs are collected from multiple sources and theses can generate a huge amount of data. Then, the processes such as data analysis, extraction and formatting in the parser can not keep up data collection process which causes data loss. In that case, we will have to place an intermediate module which acts as a queue, temporary data storage and allows the sequential data processing of the parser. However, the continuous reading and deleting operations in the parser is time-consuming, leading to systems with certain lag

2.3. Centralized Log Management Difficulties

In an organization which uses multiple operating systems (Linux, Windows, Mac OSX, etc), the software's services and other applications generate log and store them in multiple formats. This causes difficulties in log management focusing in the following fundamental issues [4,6]:

Multiple log sources: There are many sources of logs to be processed. They are located on different servers within the institution. Moreover, in a source log, there may be hundreds of log files with different contents and sizes.

Heterogeneous log content: Each type of log sources has different values. Assuming the first log contains an IP address, but no user name, while the second log has a user name but does not contain the IP address. This situation leads to the difficulties in linking the records together because we do not have the records containing common values.

Heterogeneous log format: There are many types of log sources using different formats such as syslog, databases, SNMP, XML, binary files, meta-data, resulting in difficulty of standardization for all structures of logs.

Log protection and transmission: Logs need to be protected to ensure the confidentiality and integrity and need to be always ready for the investigation. Therefore, log needs to be transmitted over a secure channel to the storage, and must have a separate channel to ensure operational readiness.

2.4. Real-time Log Management Difficulties

As we already know, logs can be seen as big data which is normally held by volume, velocity and variety. The log processing in real time is considered very complicated and difficult. Firstly, the system must be able to collect real-time data from the input data stream at a rate of millions of messages per second. Secondly, it is necessary to perform parallel data processing tasks as soon as it is collected. Thirdly, we must have methods to filter the data to extract and refine the meaningful information. Besides, the data have to be restructured to a common format [2,6].

3. Related works

Over the last years, to overcome the above difficulties, many log management system have been proposed including Hadoop, Splunk and ELK (ElasticSearch-Logstash-Kibana). Hadoop [9] is usually used for high scale system such as Google, Yahoo, etc. The framework is opened for many third-party plugins. However, the framework requires a master node and lots of distributed slave nodes with high performance. Hadoop is therefore not really efficient for light and centralized log management system.

Splunk [7] is a closed log management system. It offers paid services such as data storage, analyzing, searching and visualization. With Splunk, end-users do not aware of the system's underlying infrastructure, operations and managements. However, in this case, the confidentiality and integrity of logs are questionable that worrying users.

ELK [8] (Fig. 2) is a simple open source log management system. ELK contains several fundamental modules of a log management system. Firstly, LogStash [1] is responsible for log collection from multiple sources. Due to the heterogeneous content, logs are then tagged and/or filtered. Log tagging is a process in which log file format can be identified thanks to several key words detected in its content. Log filtering is a process in which each log line is extracted in to data field. Thus, LogStash inputs heterogeneous log files from multiple log sources and outputs structured log data so that it can be stored in a database.

Secondly, ElasticSearch is used for log data indexing and storing. As in many other large scale systems, ElasticSearch uses inverted index technique. This means that ElasticSearch stores not only the log content but also the reference to the location of the content. This technique aims to accelerate log content searching.

Finally, Kibana is a web-based interface which analyzes and visualizes searching log. Kibana offers also different options to data graphical representation.

In general, ELK is centralized and much more compact than Hadoop. ELK ensures the confidentiality and integrity of log data since this must not be sent to third-party servers to process. ELK provides also the basical functions of a log management system. However, in ELK, when loading log file from multiple sources to a central server, LogStash usually causes a dead-lock. Thus, these three systems can not be used to fully address all the problems of a centralized log management system.

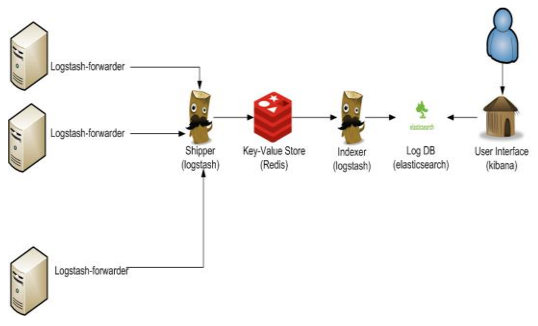

Figure 2: The ELK architecture [8].

4. Efficient Log Management System (eLMS)

In this section, because of its certain advantages, we aims to build a novel centralized log based on ELK which is called eLMS (i.e efficient Log Management System). In this system, we propose two models to avoid the dead-lock problem of ELK. The first one, we use streaming technique [4] instead of uploading technique in LogStash so that the log data can be loaded to ElasticSearch in nearly real time. This is called the single source model. In the second model, we replace LogStash with a better queue, so that log files from multiple sources can be cyclic queued without causing loading dead-lock.

4.1. Single source model

In this model (Fig. 3), eLMS shipper pushes log file directly to the eLMS-parser by the use of in-streaming technique. In this approach, instead of sending a large log file which may cause the dead-lock, we partition the file in to small pieces, store them in a buffer and frequently send them to the eLMS-Parser. Then, eLMS-parser is responsible for log filtering according to business rules. eLMS-parser is also used for log data tagging based on a data mining model. This model helps us to predict log source such as Linux, MacOS, Windows, etc.

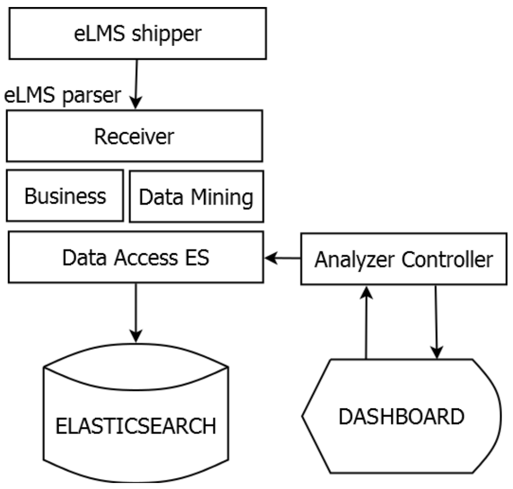

Figure 3: Single source model.

We also develop a Data Access library that provides API to access to ElasticSearch (ES). Our system inherits the two modules ES (ElasticSearch) and Dashboard (Kibana) from ELK.

4.2. Multiple source model

In the multiple source model (Fig.4), we replace LogStash in ELK with a novel queue (Message Queue) so that eLMS can receive log from multiple sources without causing dead-lock. The queue can be Kafka [5]. This is a message cycle. Kafka uses Zeokeeper [6] as a message buffer. Kafka acts as a central data backbone so that it can handle hundred megabytes of reads and writes per second from thousands of clients.

Kafka message queue partitions messages in categories called topics. Each partition is an ordered sequence of messages. Kafka uses a parallelism instead of asynchronous ordering technique. Therefore, it has stronger ordering guarantees than traditional messaging systems.

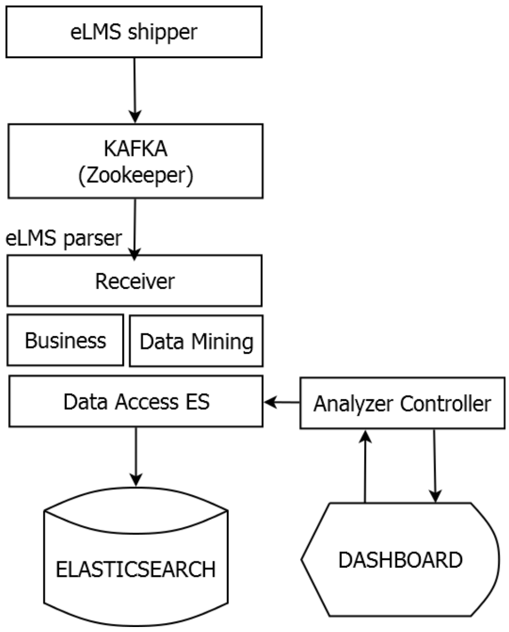

Figure 4: Multiple source model.

5. Model evaluation

In our works, we have built the single source model for testing. We use a laptop Lenovo X230, Core i7, CPU: 3.4 GHz, RAM: 8G to run the program. We first generate random data with 15 fields with different volumes (1MB, 2MB, etc.). Then we calculate the running-time of eLMS and ELK from inputting log files to the system until visualizing the results in Kibana. The evaluation results can be found in Fig. 5. In this figure, the vertical axis represents the running-time and the horizontal axis represents the size of log files. The blue columns display the running-time in ELK while the red ones represents that of eLMS. The figure shows that the eLMS runs much faster than the ELK. Up to 100MB of log file size, we can see that eLMS is at least 10 times faster than ELK. Althouth, the evaluation data is of not very large size, we can see that our improvements in eLMS are significant.

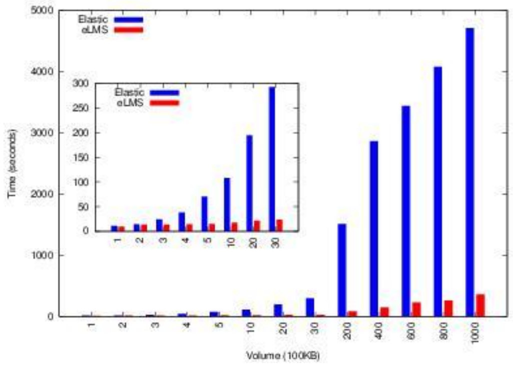

Figure 5: Running-time comparison between eLMS and Elastic (ELK).

6. Conclusion

In this paper, we present the difficulties of a centralized log management system. We also explain certain systems that are recently proposed in literature. We find that, ELK is the most basic system and can be extended to address the above issues. However, ELK causes dead-lock when using LogStash with muliple sources. ELK is really slow since LogStash is based on uploading technique for log file collection. We have built therefore a new system which is based on ELK called eLMS to avoid this dead-lock in order to accelerate the system. Our system uses streaming technique and an additional queue.

We have implemented eLMS. From the evaluation results, eLMS is much faster than ELK. In the future, we aims to continue testing eLMS with larger log file sizes and with both proposed models.

References

[1] James Turnbull. The Logstash Book. Version v1.4.3. Published By You Lulu Inc. 2014.

[2] Patricio Córdova. Analysis of Real Time Stream Processing Systems Considering Latency. University of Toronto patricio@cs.toronto.edu. 2015.

[3] Peter Matulis, Centralised logging with rsyslog. Canonical, 2009.

[4] Radomır Sohlich, Jakub Janostık, Frantisek Spacek. Centralized logging system based on WebSockets protocol. 13th International Conference on telecommunications and informatics, Istanbul,Turkey, 2014.

[5] Jay Kreps , Neha Narkhede , Jun Rao. Kafka: a Distributed Messaging System for Log Processing. LinkedIn Corp, 2015.

[6] MIT College of Engineering University of Pune. Real Time Generalized Log File Management and Analysis using Pattern Matching and Dynamic Clustering. International Journal of Computer Applications (0975 8887) Volume 91 - No. 16, April 2014.

[7] Johnvey Hwang. 2009. Splunk, innovation behind. In Proceedings of the Symposium on Computer Human Interaction for the Management of Information Technology (CHiMiT '09). ACM, New York, NY, USA, , pages. DOI=http://dx.doi.org/10.1145/1641587.1814304

[8] Alberto Paro. 2013. Elasticsearch Cookbook. Packt Publishing

[9] Tom White. 2009. Hadoop: The Definitive Guide (1st ed.). O'Reilly Media, Inc..

[10] Alien Vault, Life Cycle of a Log, 2014, https://www.alienvault.com/doc-repo/usm/security-intelligence/AlienVault_Life_cycle_of_a_log.pdf