Distributed Cloud Monitoring Platform Based on Log In-Sight

E. Haihong, Yuanxing Chen(✉), Meina Song, and Meijie Sun

School of Computer Science,
Beijing University of Posts and Telecommunications, Beijing 100876, China
{ehaihong,mnsong}@bupt.edu.cn, chenyuanxing_bupt@foxmail.com,
sun_magine@163.com

Abstract. Log management plays an essential role in identifying problems and troubleshoot problems in a distributed system. However, when we conducted log analysis on big data cluster, Kubernetes cluster and Ai capability cluster, we found it was difficult to find a Distributed cloud monitoring platform that met our requirements. So, we propose a Distributed cloud monitoring platform based on log insight, which can be used to achieve unified log insight of big data clusters, K8s clusters, and Ai capability clusters. At the same time, through this system, Developers can intuitively monitor and analyze the business system data and cluster operation monitoring data. Once there is a problem in the log, it will immediately alert, locate, display, and track the message. This system is helpful to improve the readability of log information to administrators, In the process of data collection, Filebeat and Metricbeat will be combined to collect data, therefore, the system can not only collect ordinary log data but also support to collect the indicator data of each famous mature system (Such as operating system, Memcached, Mysql, Docker, Kafka, etc.). Besides, the system will monitor and manage the status of cluster nodes through BeatWatcher. Finally, we develop the system and verify its feasibility and performance by simulation.

Keywords: Log insight · Distributed cloud monitoring · Log analysis

1 Introduction

As an essential part of the entire operation and maintenance and even the whole product life cycle, the monitoring system not only needs to be able to feed back the current state of the system in real time. But also promptly detects faults before-hand, and needs to provide sufficient data for fast positioning and tracking after-wards Question, find out the root cause of the problem.

In this paper, we will present a multi-dimensional log insight and analysis system based on Elasticsearch [1]. Most of the monitoring systems using Elasticsearch as data storage and search engine [2] directly use the ELK [3] open source component. In the recent monitoring of a gateway system [4], we have implemented our monitoring function in this way. This way is easy to implement, and its monitoring function is not

X. Zhang et al. (Eds.): CloudComp 2019/SmartGift 2019, LNICST 322, pp. 76–88, 2020.
https://doi.org/10.1007/978-3-030-48513-9_6

weak, but we found that the defects achieved in this way are also obvious. First of all, for many of the same data collection of the cluster, it is necessary to download and start the collector of one server one by one, which brings a vast workload; At the same time, the components of ELK are almost independent of each other, and the entire system cannot be managed uniformly.

Here, we propose a distributed cloud monitoring platform [5] based on log insight that will be able to collect data conveniently from different system clusters. Through the configuration center provided by the insight and analysis system, it can realize the configuration multiplexing of collectors of various categories. And no longer need to deploy each server separately through the command line, We implemented a one-click start for the entire cluster monitoring. The distributed cloud monitoring platform will realize the integration of each component, and the state of the data acquisition node can be synchronized and effectively controlled through the distributed cloud monitoring platform. The insight and analysis platform combines Flink's [6] powerful data processing capabilities to parse and transform various types of log data [7]. The log data will be transformed from unstructured data into structured data [8] for subsequent processing. Finally, the monitoring platform cannot only visualize [9] the log data according to time, label, type, hostname, etc. But also provide multi-dimensional aggregate statistics function and early warning function for analyzing the data. For example, according to the index log data of the system, Analyze whether there is a certain indicator in the current time of each cluster that exceeds the warning line. According to the running log data of the monitored system, you can calculate out whether the log data contains Error, Fail, Warn, etc. And whether the administrator needs to be notified. According to the log data accessed by the user, We can analyze the user's unit time visit amount and regional access heat map [10]. After the design is implemented, the system will perform lots of tests (such as unit test, functional test, and performance test) to verify its feasibility.

The rest of this paper is organized as follows. The second section describes the current work related to monitoring. The third section presents a monitoring system based on Elasticsearch. The fourth part has carried on detailed development to the system. Section five gives the simulation results of operation and performance. Section six is a summary of the entire article.

2 Related Work

We introduce the state of the art for monitoring systems: Splunk [11], Fluentd [12], Loggly [13], Logstash [14], and Graylog [15].

Splunk is a software that provides services such as log-based data search [16], monitoring, and analysis. It can be used for security, management, and applications. Splunk can capture, correlate, collect, monitor, and analyze real-time data from any source. It can also create alerts, dashboards, and identify data patterns. It can be software or cloud services, but the cost of using Splunk in terms of capital and complexity is too high.

Fluentd is an open source data collector designed to handle data flow. It is a bit like syslog, but using JSON as the data format. It features a plug-in architecture with high scalability and high availability, meanwhile also enabling highly reliable message forwarding. As a free data collector, Fluentd is already quite good, But the research data on Fluentd is less than other collectors, and it does not provide any built-in visualization tools.

Loggly is a robust log analysis tool with a very user-friendly interface, with centralized log management and filtering and alerting capabilities, and it is also very easy for developers to customize performance dashboards. But the downside is that after a period of free use, you have to pay to continue using it.

Logstash is part of the ELK stack. With 200 plug-ins, Logstash can access multiple data sources and import data streams to a central analytics system on a large scale. Logstash is designed with scalability in mind, providing APIs for the community to quickly develop plug-ins. The advantage is its freedom and open source and easy integration with other Elastic products. But the disadvantages are more obvious. The filter is difficult to write. And then it is implemented in Java and contains a lot of filtering functions, So if used as a log collector, Logstash takes up a lot of space.

Graylog is an open source solution that claims to be able to do the same thing as Splunk. It is written in Java, and the web interface is written by Ruby on Rails(an open source web application framework written in the Ruby language). The advantage is that it can be easily set up, supports REST APIs [17], and it can be extended with plugins. But the disadvantages are also visible. Graylog only supports system logs and GELF(Graylog Extended Log Format). And Graylog can't read system log files directly. It requires the server to send log messages directly to Graylog, which obviously makes development users more troublesome.

3 System Architecture

This section will show the architectural design of the distributed cloud monitoring platform, which consists of four modules: Monitoring Node Management, Data Collection and Parsing, Data Cleaning and Processing, Data Analysis, and Visualization. The whole system is based on the monitoring of the log data on the monitoring node. The user can monitor and manage the monitoring node according to the visual interface provided by the system (Fig. 1).

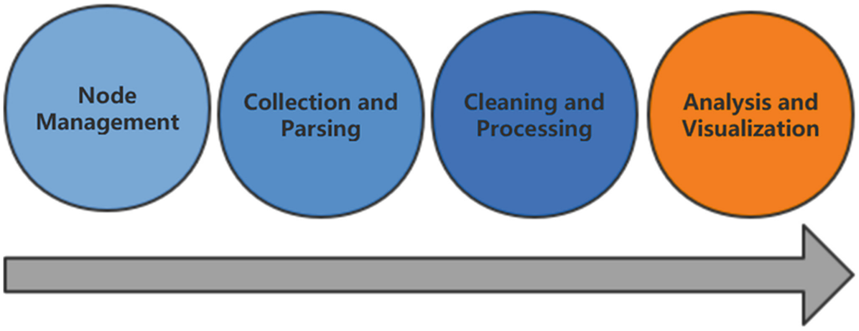


Fig. 1. System architecture diagram

3.1 Monitoring Node Management

The design goal of this system is to make a unified log insight and management platform, which can support access to various servers or systems for management. Therefore, the first need is the management of monitoring nodes. The system uses the self-developed Beatwatcher to manage the monitoring nodes.

The functions of node management mainly include the following parts: one-click download and start, extract monitoring test data, transfer analysis configuration file, Heartbeat Mechanism, Data collector start and stop.

3.2 Data Collection and Parsing

Considering the diversity of monitoring data sources, data collection and simple parsing are done using Filebeat and Metricbeat. Collect and easily parse log type data through Filebeat, Collect metric types and data with standard template types via Metricbeat. The data is then sent to the message queue for subsequent processing.

3.3 Data Cleaning and Processing

The format of the log data is various, so it is necessary to clean and process it before sending it to the log repository. For example, the conversion of the log format, the conversion of the date form, the conversion of the geographical location, the clearing of the invalid data, the processing of various tags, the detection of logs with abnormalities, and subsequent warnings and the like.

3.4 Data Analysis and Visualization

The analysis and visualization of data include two aspects. The first is the analysis of log data. Through various filtering conditions, the log data of each data source that needs to be viewed is displayed through the interface, help users quickly locate the data they need. Secondly, the dashboard visualization part realizes the visualization of the data by implementing the custom creation of the dashboard. It can quickly and conveniently view the required information in the form of a chart and adds data source fusion, multi-field fusion, and other multidimensional display in the visualization process. Data analysis and visualization make the function of the entire system more complete.

4 Implementation

4.1 System Design

The overall module architecture design of the unified log insight analysis platform is as follows (Fig. 2):

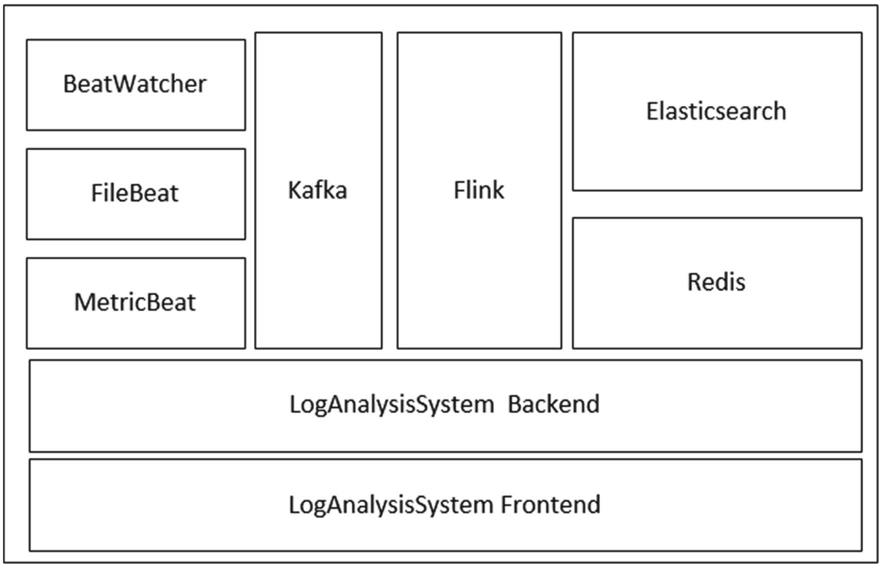


Fig. 2. Distributed cloud monitoring platform overall module architecture diagram

The collectors used by the system to collect data are Filebeat and Metricbeat, and the self-developed BeatWatcher is used to manage and control the collector.

In order to cope with traffic spikes and achieve module decoupling, the system uses Kafka as an external message queue for log data transmission.

Taking into account the real-time requirements of the system, the system uses Flink streaming to process the log data to ensure real-time processing in the case of large log data.

The system uses the Elasticsearch search engine to store and search log data.

At the same time, the system uses Redis as the database of non-log data during system operation, which ensures the high concurrent access requirements of the system.

Finally, the entire cloud monitoring infrastructure platform is built by Spring Boot, and the front and rear ends are separated to achieve real front-end decoupling.

4.2 System Implementation

System implementation includes three dimensions, including monitoring system interface implementation, log monitoring platform layer implementation, data layer implementation. The hardware and software specification of the servers used are shown in Table 1.

Table 1. Development specification

Software/Hardware	Specification
Operating system	Linux/Ubuntu
Number of servers	3
Single server memory	64 GB
Single server storage	1T
Java version	1.8.0_111
Go version	go 1.12
Filebeat version	6.5.4
Metricbeat version	6.5.4
Elasticsearch version	6.5.4
Redis version	4.0.8
Flink version	v1.7.2
Kafka version	Kafka 2.0.0
Browser	Chrome/Firefox/IE/Edge

Monitoring System Visualization

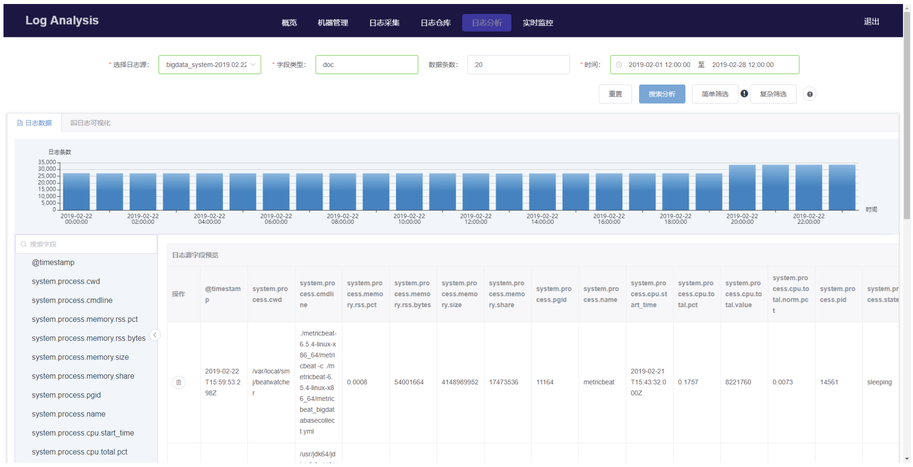
The system adopts the separation of front and rear ends to realize the system separation and data decoupling. On the client side, the interface is realized through the React framework. The user completes a series of functions from machine management, log collection, log processing, log warehouse, log analysis and real-time monitoring through the client interface.

The front end and the back end of the distributed cloud monitoring system interact with each other through the Rest API. The front end and the back end are respectively deployed. The user accesses the client interface by accessing the Http request, and then the client will access the back end of the system according to the user's click request. After receiving the request, the backend will be distributed to different Controllers according to the request parameters. The Controller will call different Services to provide services, and finally, return the request to the client and the user.

In this system, the user can complete the visual configuration of data collection just by using the client, including the configuration of system basic metric data and configuration of common log data. In terms of basic metric data configuration, the system provides a wealth of basic data collection sources to choose from, including Docker's running state, Mysql's running state, Prometheus node's running state, Apache server's running basic indicators, server System's own basic operating indicators, etc. They can be monitored by just clicking and selecting in the visual interface.

The system uses the configuration distribution startup method to start the data collector. Therefore, the collection mode of the startup collector and its configuration can be reused. Data collection of the same type on a cluster can be configured only once and distributed to different nodes via a visual interface to get it up and running.

During the process of log visualization, the user uses the log source, log type, number of displays, and view time to capture data. At the same time, the system designs and provides two filtering options: simple filtering and complex filtering. Simple filtering mode Easy to use. The complex filtering pattern can be continuously combined and nested by the JSON data structure designed in this system. Our system provides users with easy-to-use data filtering options while retaining powerful filtering capabilities (Fig. 3).



* 选择日志源: bigdata_system-2019.02.22
* 字段类型: doc
数据条数: 20
* 时间: 2019-02-01 12:00:00 至 2019-02-28 12:00:00

@timestamp
system.process.cwd
system.process.cmdline
system.process.memory.rss.pct
system.process.memory.rss.bytes
system.process.memory.size
system.process.memory.share
system.process.pgid
system.process.name
system.process.cpu.start_time
system.process.cpu.total.pct

日志源字段预览

操作	@timestamp	system.process.cwd	system.process.cmdline	system.process.memory.rss.pct	system.process.memory.rss.bytes	system.process.memory.size	system.process.memory.share	system.process.pgid	system.process.name	system.process.cpu.start_time	system.process.cpu.total.pct	system.process.cpu.total.value	system.process.cpu.total.norm.pct	system.process.pid	system.process.state
	2019-02-22T15:59:53.298Z	/var/local/smj/beatwatcher	./metricbeat-6.5.4-linux-x86_64/metricbeat -c ./metricbeat-6.5.4-linux-x86_64/metricbeat_bigdatabasecollect.yml	0.0008	54001664	4148989952	17473536	11164	metricbeat	2019-02-21T15:43:32.000Z	0.1757	8221760	0.0073	14561	sleeping
			/usr/jdk64/jd												

Fig. 3. Visualization of log data

In the log data analysis module, the user performs the creation of the Panel through various fields of the log data according to his specific needs and then puts it into a specific Dashboard of a specific folder. In the real-time monitoring module, Users can choose whether to turn on real-time monitoring to update the visual data. As shown in the figure, the system provides a graph fusion technology for different data sources and different filtering conditions. The user can view a comparison graph of the running status of each index of each node in a Panel in the Dashboard, which is convenient for the user to perform data analysis and horizontal and vertical comparisons (Fig. 4).

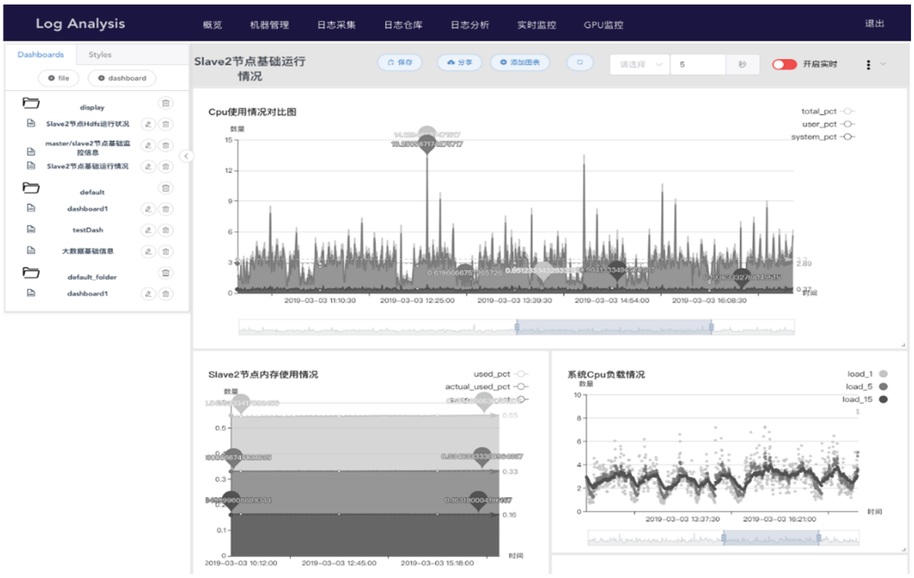


Fig. 4. An example of log analysis dashboard

Log Monitoring Platform Layer Implementation

Node Manager Beatwatcher

Beatwatcher is implemented by the Go language. Its functions mainly include the following parts: one-click script to download and launch, extract monitoring test data, transfer analysis configuration file, the heartbeat mechanism, data collector start and stop.

One-click script download launch: Make it easy to add nodes. The entire process from downloading the complete Shell script to downloading the executable file to start can be completed through a Shell script command containing Key and tags generated by the platform. Greatly simplify the user's operation steps.

Extract monitoring test data: Beatwatcher enables the extraction of test data, ensuring correct configuration when modifying the collector configuration.

Transfer and parse configuration files: Through the interaction with the log management platform, the configuration file is accepted and parsed.

Heartbeat mechanism: The collector sends survival information to the log monitoring platform every 5 s to ensure that the node's survival status can be grasped in the log monitoring management platform.

The data collector starts and stops: the Beatwatcher can be used to start and stop the collector.

Log Monitoring System Center Management Platform

The management platform of the log monitoring system center will be implemented on the basis of the Spring boot framework, receiving the request data of the front end, and calling the corresponding Service according to different Controllers with different parameters. In various such service processing, it will manage the scheduling of Redis, BeatWatcher, Elasticsearch, Kafka, Flink and other components to complete all the functions required by the user.

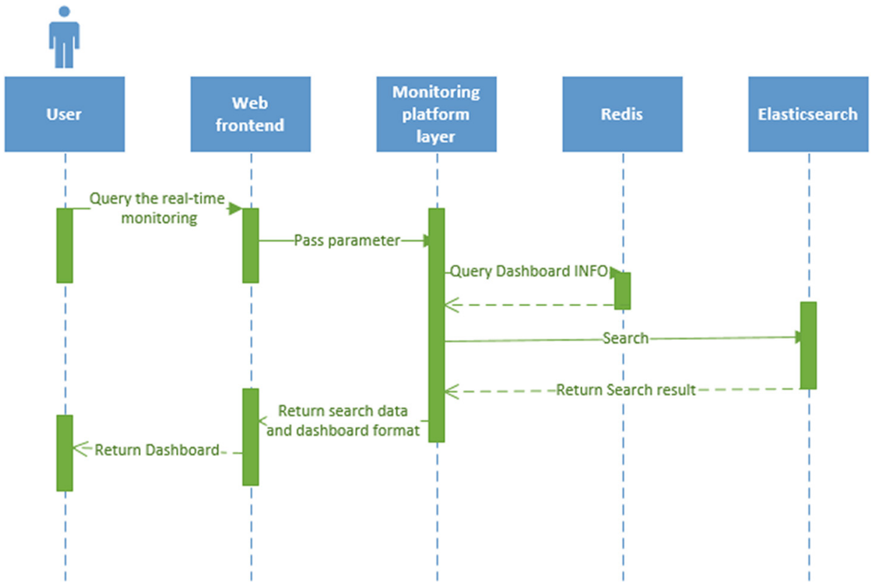


Fig. 5. Query real-time monitoring dashboard

When the user query real-time monitoring dashboard (such as the resource utilization of the big data cluster, the running status of the K8s cluster, the running status of the distributed business system, etc.), the sequence diagram is as shown in Fig. 5. The user will first request the web frontend, the front end will request a specific API, and pass the corresponding parameters. Then, the monitoring center management layer will obtain the attribute information of the Dashboard through the Redis cache, and then initiate an aggregate search request to Elasticsearch according to the Current Para, and obtain the data of each indicator. Finally, the chart style information and indicators data of Dashboard are returned to web frontend uniformly.

The log data in Elasticsearch comes from the big data cluster, K8s cluster, API gateway and other platforms, data collection by Filebeat and Metricbeat managed by BeatWatcher and then processed by Flink, and then imported into Elasticsearch. In Elasticsearch, Indexes are created based on the date and cluster type.

Data Layer Implementation

Data Collector

BeatWatcher is used to manage Filebeat and Metricbeat and to interact with data and command interactions with the monitoring platform (Fig. 6).

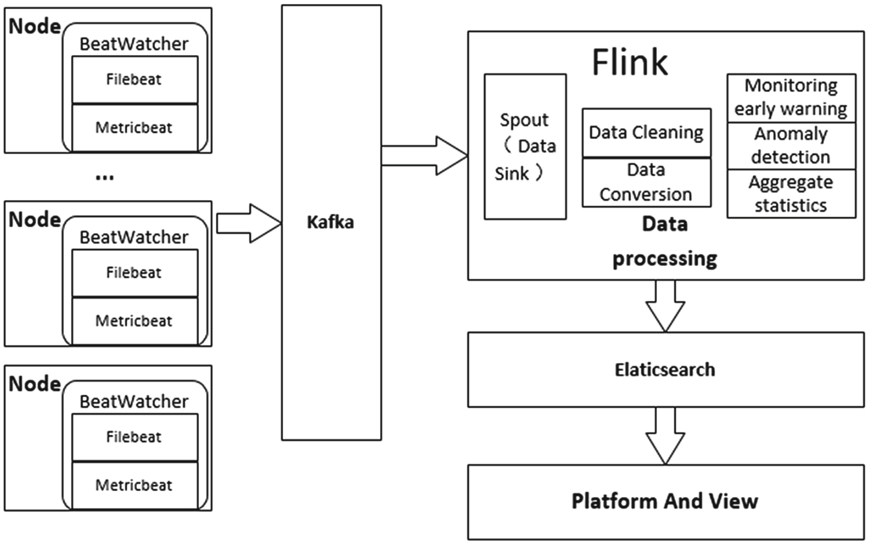


Fig. 6. Data flow graph of data layer

Filebeat is an open source log data collector. When logging data is collected, the log directory or specific log files are monitored with Filebeat. Metricbeat can periodically collect operating system and server operating indicators (CPU, memory, hard disk, IO, read and write speed, processes, etc.), and can also collect running information of many general-purpose systems (such as Kafka, Kubernetes, Memcached, Mysql, etc.).

In our system, they are used to collect multi-source data, and then the collected indicators and data are sent to the specified output (such as Kafka, Elasticsearch, etc.).

Log Message Queue

Apache Kafka is an open source, distributed, partitioned, and replicable publishing-subscription messaging system based on log submissions. It has message persistence, high throughput, distributed, low latency and other features.

In our system, data is sent from the data collector to the corresponding Topic of Kafka, and Flink subscribes to the data from Kafka. It avoids the situation that the server is overwhelmed by the sudden peak value, and can achieve the isolation of data collection and processing well.

Data Processing

Apache Flink is a distributed open source computing framework for data stream processing and bulk data processing. Its core is the distributed stream data flow engine written in Java and Scala, which can support both stream processing and batch processing.

In our system, Flink is used for data processing, and transform the unstructured log data generated by each system (such as the basic data generated by the big data cluster, the operational data generated by the service platform, the basic indicator data of the service platform, etc.) into structured data. In the data processing module, it also includes the operation of deleting useless data, adding tags to the data. The data will be imported into Elasticsearch for data storage and search after Flink processing.

Data Storage and Search
Elasticsearch is an open-source, distributed, RESTful full-text search engine built on Lucene. It can store, search and analyze large amounts of data in a very short time.

Our system uses Elasticsearch to store and search log data. The log data will be divided into different indexes according to the source and date of the log data. Then our monitoring platform will query and aggregate the log data from Elasticsearch according to certain rules. User access The platform can view the filtered log data and the platform provides the Dashboard function for users to analyze the log data.

5 Testing and Results

This section will test this log monitoring system. It includes unit testing, functional testing, integration testing, and performance testing. Only the performance test results are given here. By using JMeter to constantly increase the number of requests and concurrency, We verified the compression resistance and stability of the system by comparing the average response time, accuracy and other indicators of the system.

The system consumes the most performance when performing an aggregate search on all log data. Therefore, what is shown here is the performance of the system in extreme cases where all requests are aggregated search requests (Fig. 7).

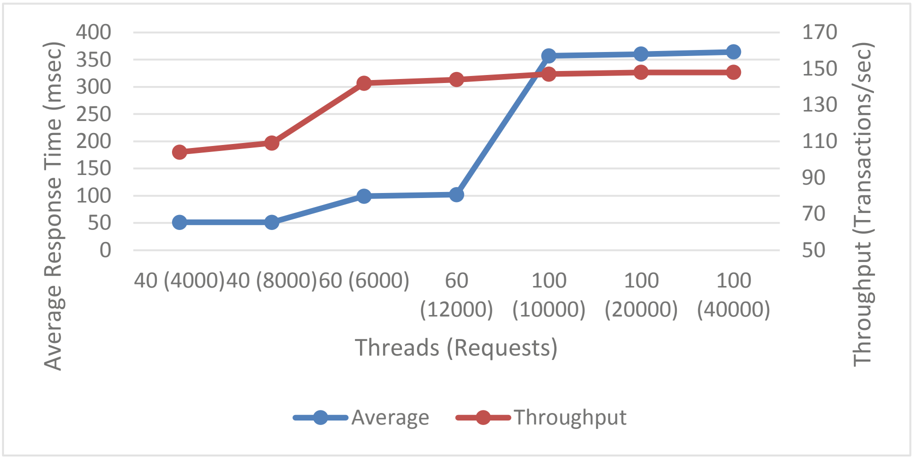


Fig. 7. Load test results (average response time and throughput)

When the system load is 40 threads to access 4000 and 8000 times, the average response time is 51 ms. When the system load is increased to 60 threads to access 6000 and 12000 times, the average response time is 99 ms and 102 ms. When the load is upgraded to 100 threads to access 10000, 20,000, and 40,000 times, the average response time are 345 ms, 360 ms, and 364 ms, respectively.

Figure 8 depicts the amount of data received and transmitted by the system in the above test environment. The resulting curves are almost coincident and similar to throughput. In all of the above tests, the system error rate was 0%.

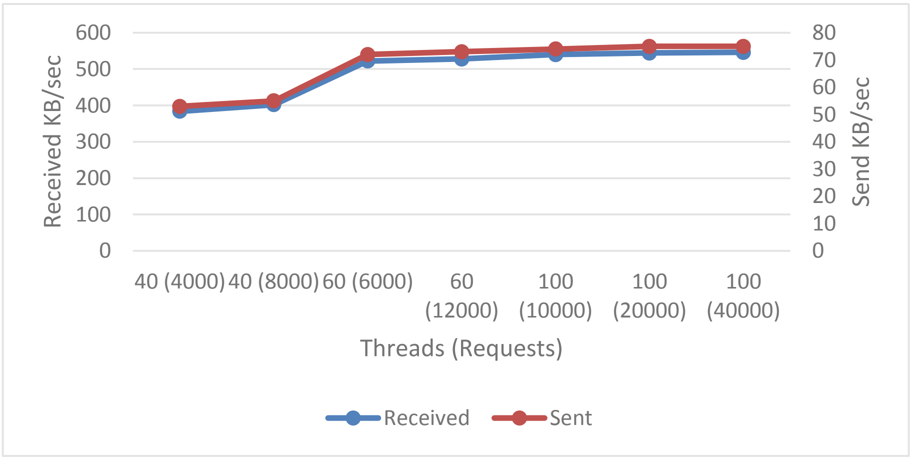


Fig. 8. Load test results (received and sent amount)

6 Conclusions

In this Paper, we provide a solution for log monitoring of distributed systems. Through the distributed cloud monitoring platform based on log insight built on the basis of Spring Boot, We realize functions such as machine management, data collection, data cleaning, log warehouse, data analysis, and real-time monitoring. Users can complete the monitoring and deployment of any server through a unique shell command generated by the distributed cloud monitoring platform.

At the same time, based on the Go language, the Log monitoring analysis platform is implemented to monitor each user cluster node. After the data is collected by Filebeat and Metricbeat managed by BeatWatcher, it will be sent to Kafka for subsequent processing.

The distributed streaming data engine Flink will perform data cleaning and abnormal detection by subscribing to Kafka's corresponding Topic. The functions of Flink module in this system mainly include data deduplication, data enhancement, data conversion, abnormal detection according to rules, unsupervised abnormal detection, etc.

Finally, the data is sent to Elasticsearch for persistent storage. The operator will perform centralized filtering and display and analysis of log data in an effortless way through our system.

In future work, we will intend to add the predictive analysis function of the log data and predict future trends through certain prediction algorithms to assist the operators in making decisions.

Acknowledgment. This work was supported in part by the National Science and Technology Major Project under Grant 2018ZX03001016; Engineering Research Center of Information Networks, Ministry of Education.

References

1. Kononenko, O., et al.: Mining modern repositories with elasticsearch. In: MSR (2014)
2. Zobel, J., Moffat, A.: Inverted files for text search engines. ACM Comput. Surv. **38**, 6 (2006)
3. Prakash, T.R., et al.: Geo-identification of web users through logs using ELK stack. In: 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence), pp. 606–610 (2016)
4. Montesi, F., Weber, J.: Circuit breakers, discovery, and API gateways in microservices. CoRR abs/1609.05830 (2016)
5. Arpitha, P., Kumar, P.V.: Big data computing and clouds: trends and future directions (2018)
6. Carbone, P., et al.: Apache Flink™: stream and batch processing in a single engine. IEEE Data Eng. Bull. **38**, 28–38 (2015)
7. Carbone, P., et al.: State management in apache Flink®: consistent stateful distributed stream processing. PVLDB **10**, 1718–1729 (2017)
8. Tovarnák, D., Pitner, T.: Normalization of unstructured log data into streams of structured event objects. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 671–676 (2019)
9. Tang, J., et al.: Visualizing large-scale and high-dimensional data. In: WWW (2016)
10. Dumais, S., Jeffries, R., Russell, D.M., Tang, D., Teevan, J.: Understanding user behavior through log data and analysis. In: Olson, J.S., Kellogg, W.A. (eds.) Ways of Knowing in HCI, pp. 349–372. Springer, New York (2014). https://doi.org/10.1007/978-1-4939-0378-8_14
11. Splunk. https://www.splunk.com/
12. Fluentd. https://www.fluentd.org/
13. Loggly. https://www.loggly.com/
14. Logstach. https://www.elastic.co/
15. Graylog. https://www.graylog.org/
16. He, P., et al.: Towards automated log parsing for large-scale log data analysis. IEEE Trans. Dependable Secure Comput. **15**, 931–944 (2018)
17. Surwase, V.: REST API modeling languages - a developer's perspective (2016)