Research on a Method of Log Aggregation

Lai Xinming¹, Wang Haitao², Zhao Jing², Zhang Fan², Zhao Chao², Wu Gang²*,

¹Aisino Corporation, Beijing, China
²China National Institute of Standardization, Beijing, China
*Corresponding author
Email: wugang@cnis.ac.cn

Abstract. This paper proposed a log aggregation method in scale container platform. This method collects container log data from cluster nodes through system management tool and container governance tool, provides the indiscriminate collection service for different platforms, aggregates log data in time based on Storm stream processing to implement real-time performance, implement high performance and high availability of container log data aggregation under scale clusters based on Kafka as the aggregation queue of log data.

1. Introduction
With the continuous development of container technology, application containerization brings more and more practical benefits. Compared to traditional applications, containerized applications are easy to automate deployment, run multiple instances simultaneously in the same environment without affecting each other, new versions of automated upgrades, fewer external environment dependencies, and ease of horizontal scaling to handle traffic spikes. As a result, more and more companies are beginning to develop container technology and containerize more IT services.

At the same time, in response to the rapidly growing demand for container operations, container management technology has emerged. After years of development and competition, Kubernetes has become the industry's de facto standard for container governance platforms. Kubernetes guarantees the smooth running of the container through the container running state probe, and at the same time ensures the load balancing of the cluster through the scheduling algorithm. Kubernetes not only enables the container application to dynamically extend the service according to the real-time load condition, but also provides the gray-scale publishing capability of the container application, which greatly improves the efficiency of the cluster and reduces the operation and maintenance cost of the enterprise.

The container orchestration management capabilities provided by Kubernetes can effectively ensure the load performance of the application layer services running on them is stable, but it also has some impact. The first is the impact of deploying applications. Initially deploying an application service on Kubernetes is more difficult than traditional stand-alone deployment. In addition to deploying applications on Kubernetes, you need to be able to containerize the application, correctly write the deployment configuration file, and also master the Kubernetes management API view. Start the log to track the application deployment process. If there is a deployment problem, the node that needs to be returned to the application scheduling should continue to track the Kubernetes component logs. The second is the impact of application debugging. When an error occurs during the application test run, the program logic needs to be performed. Debugging, because the Kubernetes cluster is not in the same network domain as the local environment, the micro services application is usually associated with
several other services in the background. Therefore, it is not possible to run the debugging in real time after modifying the code logic locally. Usually, you need to manually use the Kubernetes management API to view the container. The running log, even need to enter the container to view the relevant log output of the container; the third is the Kubernetes cluster fault location is difficult, most of the Kubernetes cluster problem will make the application state fail, then use the management API to directly view the application Start the log, the log may have internal difficulties cannot be located, usually you need to run the application to view the log docker scheduling nodes based on experience, and even need to use journalctl view the system log to locate the particular topic. For example, when the most common IP resources are insufficient and the DNS domain name resolution fails, you cannot directly locate the problem from the application log. You need to check the running log of the Kubernetes component to locate the problem. Based on the above three problems, it takes a lot of effort for the operation and maintenance personnel to complete an application deployment, application debugging, and fault location. The larger the cluster size, the more energy is needed, which causes the enterprise to invest more in the Kubernetes cluster.

In summary, the three main effects of Kubernetes are that the log data are scattered but related to each other. The operation and maintenance personnel spend more energy in the operation and maintenance process to check the log data on different machines of different levels and different components. This patent proposes a Kubernetes log data distributed aggregation method. Based on storm streaming calculation, the distributed log data is classified, grouped and structured, and then the structured log data is divided, aggregated and stored based on the Kafka message queue. The database is accessible to the upper level restful API.

2. Technology Proposal
To solve the deployment, debugging and operation and maintenance difficulties caused by the scattered application logs in Kubernetes cluster. The distributed structure based on Kubernetes cluster includes three parts: log data collection, aggregate processing and storage. The architecture of this proposal as the figure 1 shows.

2.1. Log Data Collection
The log collection service is run on each node of the Kubernetes cluster. The log collection service collects application-level logs and system-level logs on the node. The log collection service on the master node of Kubernetes is also responsible for collecting event-level logs of the application. The event-level log uses Kubernetes management api collection, which is mainly used to track the process of Kubernetes governing the Pod; the application-level log includes the Pod internalizer status data and the in-container program output log, the container status data is collected by Kubernetes management api, and the container internal program The output log is collected by Docker management api; the system-level log is mainly the running log data of Docker and Kubelet service itself, including service status data and systemd output log. The status data is collected by systemd management work, and systemd log is collected by journalctl tool.

After the log collection service collects the log data of the local device, the log data is compressed and packaged, and the identification information such as the timestamp, the node name, and the level is attached to the log data aggregation service.

2.2. Log data aggregation processing
The log aggregation service is run on the Kubernetes cluster. The log aggregation service is deployed on multiple nodes in the cluster. The log aggregation service is based on the Storm flow calculation engine. It has four log processing units, which correspond to event level and application level. System level log processing unit and log aggregation unit.

All the log data collected by the log collection service on each node of the Kubernetes cluster will be sent to the first three log processing units without any difference. The log processing unit will filter the logs that are not in the processing level according to the processing logic, and only process the
corresponding levels. Log information.

After receiving the log packet of the corresponding level sent by the log collection service, the log processing unit decompresses the log packet and parses the namespace, resource name, and Pod name information in the log data as the location identifier of the log packet, and then packages the log data and the location identifier. Sent to the log aggregation unit.

The log aggregation unit receives the log packet data of the three levels of the log processing unit, and aggregates logs belonging to the same application from different processing units according to the location identifier and the level information, and finally merges the location identifiers such as <namespaceid_resourceid_level>, and according to the positioning identifies the topic corresponding to the kafka service sent to the cluster.

2.3. Log data storage
The log data storage service and the kafka service are run on the Kubernetes cluster. The kafka service is responsible for storing all log data from the log aggregation unit. Kafka guarantees the high performance and high reliability of the log data under large-scale clusters.

The log data storage service classifies the data of different topics in kafka, and classifies the information according to the topic information, the location identifier of the log data, and finally stores the log data in the database system for access by an external system or other API.

The architecture proposed by the present invention is as follows:

Figure 1 shows the architecture of the container platform log aggregation method. At the far right is the various nodes of Kubernetes, and the Log collector log collector is run on each node. The log collector collects all the log information of the node and sends it to the three processing units in the log processor. Each processing unit retains only one level of logs. After the log data is generated, the corresponding positioning identifier is generated, and the compressed data is sent to the log aggregation unit. The log aggregation unit aggregates the distributed log data according to the location identifier and sends it to Kafka. Finally, the data in kafka is stored and stored in the database. External applications, such as the log visualization system and the monitoring and early warning system, read the data in the database to implement their own business.

3. Conclusion
The large-scale container platform log aggregation method proposed in this paper uses distributed computing and message queuing technology to realize distributed log data aggregation of large-scale container clusters. This paper mainly has the following key points:

1. Automatic collection of container related logs;
2. Log aggregation method for large-scale distributed clusters;
The solution is applied to the physical machine group of the container management platform, can efficiently collect the logs related to the cluster container and aggregate them into the database, and effectively reduce the difficulty of deployment, debugging and operation and maintenance of the application on the container management platform by integrating the log visualization system, and reduce the enterprise in Operation and maintenance investment on the container management platform.

The log aggregation method proposed in this paper can be applied to a large-scale container platform with high performance, high availability, and high scalability architecture.

Acknowledgments
This paper is supported by grants from National Key R&D Program of China (522016Z-4973) and China National Institute of Standardization (522018Y-5941, 522018Y-5948).

References
[1] Design and implementation of real-time log analysis system of map world platform. Zhang, Hongping (National Geomatics Center of China, Beijing, China); Huang, Wei; Yang, Jing. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives, v 42, n 4, p 121-127, September 19, 2018.
[2] http://kubernetes.io
[3] Bell C. (2018) Introduction to High Availability. In: Introducing InnoDB Cluster. Apress, Berkeley, CA.
[4] Zitan Chen, Alexander Barg, "The repair problem under connectivity constraints: Explicit MSR codes for the rack-aware model of distributed storage", Communication Control and Computing (Allerton) 2018 56th Annual Allerton Conference on, pp. 710-714, 2018.
[5] Analysis farm: A cloud-based scalable aggregation and query platform for network log analysis Wei J., Zhao Y., Jiang K., Xie R., Jin Y. (2011) Proceedings - 2011 International Conference on Cloud and Service Computing, CSC 2011, , art. no. 6138547 , pp. 354-359.
[6] Semantics-based event log aggregation for process mining and analytics Deokar, Amit V. (Sam and Irene Black School of Business, Pennsylvania State University, 5101 Jordan Road, Burke Center 268, Erie; PA; 16563, United States); Tao, Jie Source: Information Systems Frontiers, v 17, n 6, p 1209-1226, December 1, 2015.