www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 6 Issue 11 November 2017, Page No. 23072-23076
Index Copernicus value (2015): 58.10 DOI: 10.18535/ijecs/v6i11.12

Research and implementation of Docker performance service in distributed platform

Liu Lijuan
College of Information Engineering Nanjing Normal University Taizhou College
Dongfeng South Road No. 518, Hailing District, Taizhou, Jiangsu, China

Abstract:
Aiming at Docker currently only support queries for performance monitoring and logging services for a single container, the performance monitoring and log service scheme in distributed platform are designed and we also conduct test and analysis on it, so that the Docker container cluster can be monitored as a whole. The application of the Docker cloud platform is more stable and optimized, and the integrated management of complex logs is realized. The final result has certain guiding significance to the integration and perfection of Docker platform service.

Key words: Docker, cloud computing, performance monitoring, log service

1. Introduction

With the development of network technology[1], cloud computing uses a large number of computers to form a pool of resources to facilitate user access to on-demand computing services. As a new cloud platform, Docker has broken the cloud computing model based on the virtual machine since 2013, and has been supported by many large enterprises, becoming the most popular project[2] in the field of cloud computing. But at present, Docker is mainly for single container, and there is no reliable and stable performance monitoring scheme. It is impossible to monitor the cluster as a whole, so it is necessary to find a solution to cluster container monitoring through research. In the aspect of log service, Docker does not provide special log service scheme in the face of huge log generated by cloud platform. So it is necessary to use advanced package design components to realize complex log extraction、storage、retrieval, provide the data base for application debugging 、 performance optimization.

2. Docker related technologies

With the development of cloud computing, container technology is favored by enterprises for better performance. Docker is a container engine based on LXC[3], which uses container technology for software development and deployment, and gives it to users in mirror mode. Users can get the required software when it is running. It has the advantages of portability 、cross platform、easy to use etc..

Docker includes several core elements of mirror[4]、container and warehouse. Mirroring is the foundation of building a container, providing the basic environment for application running. container supports application instances running in it, warehouses are collections of mirrors. Docker follows the service architecture[5] of C/S, as shown in figure 1:

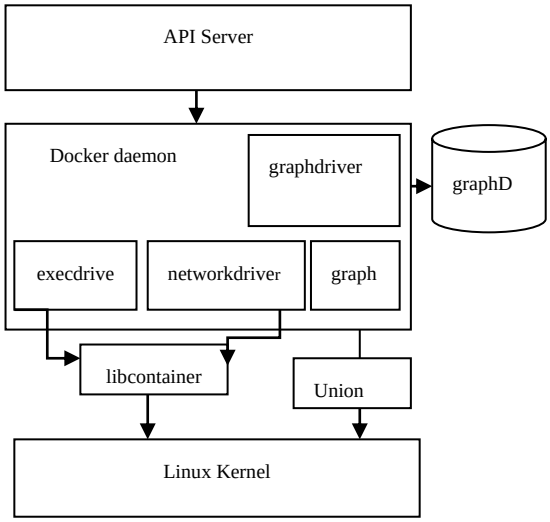

Figure 1: Docker architecture diagram

The Docker client sends a request to the Docker server, and the Docker Daemon responds to the request[6]. To create

a container, you need to download the mirror and save it in graph, use the networkdriver to configure the network environment of the container. In order to ensure the security, namespace and cGroups are used to realize the isolation and restriction of resources.

Etcd is a distributed、strongly consistent key value storage repository. It is simple、safe、fast and reliable. It is the foundation of distributed container construction. In the construction of distributed container applications, it is necessary to use Etcd to connect distributed nodes and applications within the container, and to monitor the cluster as a whole.

3. Module design

3.1 Design of Docker performance monitoring service

In the performance monitoring services, mainly for the container level, while taking into account the cluster and the host level. For users to use real-time control platform resources, data analysis and summary, provide intuitive charts to display CPU、memory、network、hard disk and other[7] indicators. In addition, alerts are provided on the basis of performance monitoring to improve the stability of applications.

Docker container performance monitoring is divided into three layers: collection、processing/storage and application, sets the proxy、server、database and interface several modules, the overall structure as shown in figure 2.

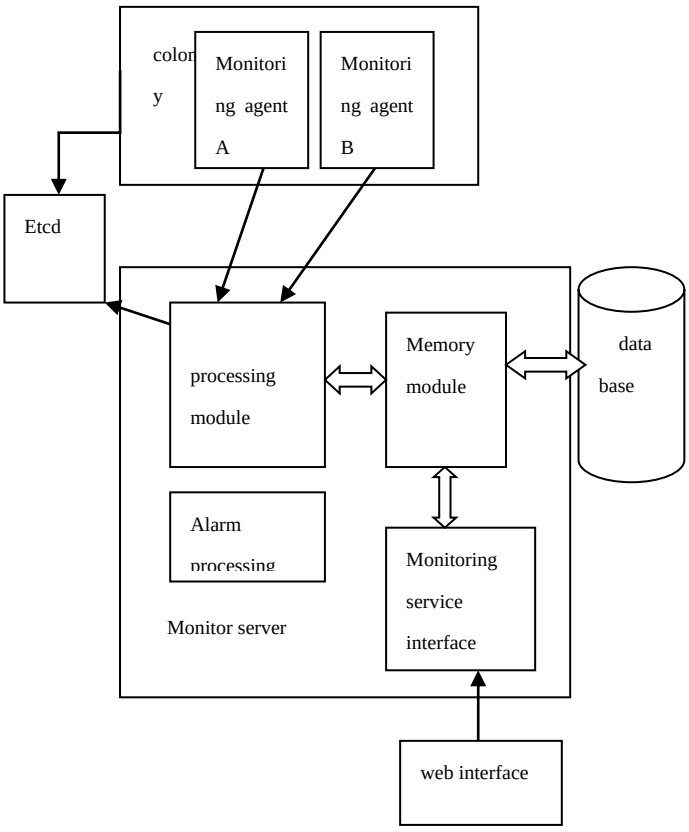

Figure 2： overall architecture of performance monitoring services

In the entire architecture, hosts in each node of the cluster run multiple containers. The monitor agent is equivalent to the collector, collects the container performance data and transmits the relevant information. The server is responsible for receiving data、 processing and saving data, listening to the interface information, responding to the request and sending data to the Web interface, alarming when monitoring node and container exception. The database is mainly responsible for storing and processing some information and providing query services. Etcd is responsible for storing the information of nodes and containers in the cluster for the server to monitor. The monitoring service interface is responsible for transmitting requests and data.

3.2 Docker application log service design

The log service provides collection、transmission、storage function. The log is classified and stored, and the efficient and reasonable log storage module is used to make the application storage more orderly and better retrieval. The whole service is divided into three layers: collection、 transmission and storage, its architecture is shown in figure 3.

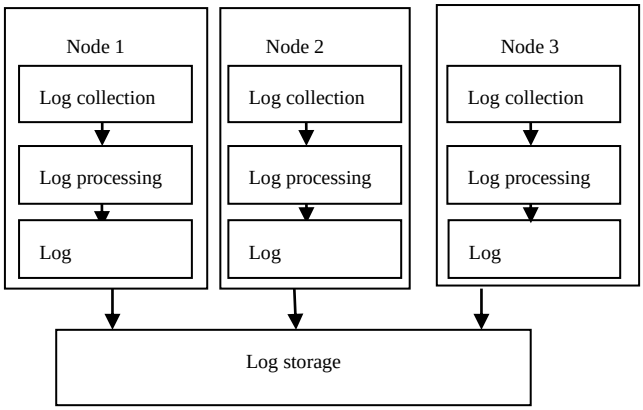

Figure 3： overall architecture of application log service

The collector Fluentd collects and sorts the source data, indicates the relevant information, and prepares for the subsequent collection and indexing. The transmission module collects the source data transmitted by each node from the collection module to the storage module. With the increase of application instances, a message queue is designed to alleviate the transmission pressure, so as to ensure the reliable and efficient transmission of log data. The storage module is saved by index, which makes the log service easy to use and provides information retrieval function.

4. Experimental test

4.1 Cluster building

Based on the cloud platform, this experiment uses Kubernetes[8] to build a container cluster, provides 15 nodes of virtual resources, deploys Docker platform for each node host and starts the corresponding container. The main configuration of performance monitoring and log service

testing is shown in table 1.

Table 1： main configuration for performance monitoring and logging service tests

Performance monitoring	*Log service*
Several containers	Some Fluented containers
One monitoring server	Log transfer component
Two databases	Some MongoDB containers
One Etcd	
One Web server	

4.2 Docker container performance monitoring service testing

4.2.1 Cluster performance monitoring

For performance monitoring services, the cluster environment test is first built, and the specific conditions are shown in table 2.

Table 2： overall state of the cluster

Project	*State*
Cloud State	available
Host number	15
Memory	12854. 36MB
Storage	94.58GB
CPU kernel number	6
Network uplink	5kbps
Network downlink	7kbps

As shown in Table 2, the state of the cluster、the number of nodes、the use of resources、the amount of data on the network and so on can be monitored. In addition, the details of each node can be further examined, as shown in table 3.

Table 3： node status of clusters

Node	*CPU utilization rate*	*Memory usage*	*Network uplink*	*Network downlink*	…
1	52%	41%	68kbps	123 kbps	…
2	34%	30%	56 kbps	109 kbps	…
3	67%	89%	50 kbps	78 kbps	…
…	…	…	…	…	…
15	57%	42%	123 kbps	189 kbps	…

By table 3, it is concluded that the state of any node in the cluster can be selectively tested, including its CPU、memory、network and other content.

4.2.2 Container performance monitoring

In a distributed platform, you can also test the status of each container, including its CPU、memory、network and other information. The specific content is shown in table 4.

Table 4： CPU performance monitoring of containers

Node	*Container*	*Before experiment*	*After experiment*
A	1	70%	50%
A	2	Not started	50%

Table 4 shows that the original A only runs the container 1, its share of CPU for an average of 70%, then the node manually run a new container 2, the original 1 CPU utilization rate[9] is less than 50%, consistent with the expected results, the performance of container monitoring can be a verification good.

4.2.3 Alarm function test

When the application appears abnormal, it will send alerts to the user, the system automatically gets the abnormal state, through the page and mail to the user to send information, specific as shown in figure 4.

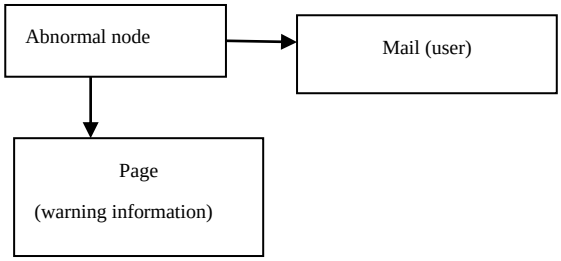

Figure 4 ： alarm function

After connecting the abnormal nodes and closing the background program on the node, the system gets the exception immediately, sends the alert mail to the user in the cluster, and sends out the page to describe the warning information.

4.3 Docker container application log service testing

4.3.1 Log collection performance test

On a certain node, different parameters are set to test the performance of log collection, and the results are shown in table 5.

Table 5： logs collection of concurrent test results

Total data	*Concurrent requests*	*Packet size*	*Success number*	*Fail number*
5120	512	1KB	5120	0
5120	512	1KB	5120	0
10240	1024	2KB	10240	0
10240	1024	2KB	10240	0
20480	2048	4KB	20480	0
20480	2048	4KB	20480	0

Table 5 shows that the number of data requests increases from 512 to 2048, the packet size increases from 1KB to 4KB, and the successful service rate is 100%, which shows that the

collection module has good concurrency.

4.3.2 Transmission module test

The transport module mainly uses message queues to improve throughput. 3 nodes are set to deploy producers、queues and consumers. The throughput[10] test of message producers and consumers is shown in Table 6 and table 7 respectively.

Table 6： throughput test of producer node

Message data	*Experiment 1*	*Experiment 2*	*Experiment 3*
512	5.12	6.13	5.24
1024	10.15	11.62	9.89
2048	19.56	20.72	21.44

Table 7： throughput test of consumer node

Message data	*Experiment 1*	*Experiment 2*	*Experiment 3*
512	5.42	6.03	5.94
1024	9.15	10.62	11.80
2048	19.43	20.35	21.67

From table 6、table 7 shows that, under the same conditions, repeat the test 3 times, with the increase of message data, a corresponding increase in throughput, message transmission speed is stable, good performance, in line with the log transmission service function.

4.3.3 Performance test of data storage cluster

MongoDB cluster is divided into 3 parts, each part has the main piece、copy、arbitration. When the log service is running, the state of the log database shows that the total data of the log is divided into 3 segments, and the data obtained by each fragment is similar in size to meet the load balancing requirements of the database. Specifically shown in table 8.

Table 8： log total data allocation table

Project	*Data quantity*
Log total data	5489892
Patch 1	1613214
Patch 2	1845210
Patch 3	2031468

Finally, the performance of database index query is tested. According to the same field to retrieve, using two groups of experiments without index and index to test different data, get the results shown in table 9.

Table 9： performance test of database index

Data quantity	*Retrieval log number*	*No index time*	*Index time*
100	654	1351	65
100	1342	1569	136
200	1178	2436	132
200	2463	2813	258
400	3462	5127	420
400	6714	6238	789

The results show that if the database is retrieved by the same field, when no index, it takes more time to traverse the entire record because each search needs to traverse the whole database. Therefore, the results return time does not change with the number of logs, and the performance is poor. But the time when the index is returned is basically proportional to the result log number, and the performance is better.

5. Conclusion

This paper takes the popular Docker technology as the starting point, in order to meet the needs of the development of container cloud, the performance and service of Docker in the distributed platform are studied. It solves the shortcomings of the current Docker container only for single node application, and expects further research on the Docker platform in the future, to improve the development、deployment、operation and maintenance efficiency of the Docker platform.

Reference

1. Miao Liyao, Chen Lijun, “A cluster expansion method based on Docker container,” Computer Applications and Software, pp. 34-39, 2017.
2. Zhang Hanbo, Ni Ming, Lu Shenglin, “Research on performance optimization of Docker virtualization technology based on RBD,” Information Technology, pp. 125-129, 2016.
3. Dong Bo, Wang Xue, Sophie, et al, “Research on virtualization technology based on Docker,” Journal of Liaoning University, Natural Science Edition, pp. 327-330,2016.
4. Zhao Lele, Huang Gang, Ma Yue, “Research on Hadoop platform architecture based on Docker,” Computer Technology and Development, pp. 99-103, 2016.
5. Lu Shenglin, Ni Ming, Zhang Hanbo, “Optimization of scheduling policies based on Docker Swarm cluster,” Information Technology, pp.147-152,2016.
6. Liu Minxian, Design and implementation of service

invocation topology analysis and performance monitoring system based on Docker, Zhejiang: Zhejiang University, 2016.
7. Mdhaffar A, Halima R B, Jmaiel M, et al, "CEP4CMA: Multi—layer Cloud Performance Monitoring and Analysis via Complex Event Processing//Networked Systems," Springer International Publishing, pp.138-152,2014.
8. Zhang Yi, Design and implementation of virtualization application platform based on Docker, Guangdong: South China University of Technology, 2016.
9. Liang Junjie, Research and implementation of cloud resource scheduling based on application container, Sichuan: University of Electronic Science and Technology of China, 2015.
10. Miao Liyao, Research on hybrid cluster expansion method of container based on Docker, Shanxi: Xi'an University of Posts and Telecommunications, 2016.

Author Profile

Liu Lijuan received a master's degree from Nanjing University of Aeronautics and Astronautics, where she was an assistant in Nanjing Normal University Taizhou College (2008) and lecturer (2010). Her research interests include cloud computing、 big data theory and practical application.