Advances in Intelligent Systems Research, volume 147
International Conference on Network, Communication, Computer Engineering (NCCE 2018)

Auto-Scaling Web Application in Docker Based on Gray Prediction

Shuangyan Wu[1, a)], Dong Zhang[2, b)], Bingheng Yan[2, c)], Feng Guo[2, d)], and Dongpu Sheng[1, e)]

[1]*School of Industrial Technology Research Institute, Zhengzhou University, Zhengzhou, China.*
[2]*System Software R&D Department, Inspur Electronic Information Industry Co., Ltd, China.*

[a)] wusyany@163.com, [b)] zhangdong@inspur.com, [c)] yanbh@inspur.com, [d)] guofengbj@inspur.com, [e)] shengdongpu@163.com

Abstract. The on-demand use and the elastic scaling feature of cloud computing make more and more companies deploy web applications on cloud platforms. True elasticity and cost-effectiveness in the pay-per-use cloud business model, however, have not yet been achieved. To address this challenge, we propose an algorithm which can auto-scaling web application in docker. We use docker container technology to deploy web applications and add or delete docker containers to adjust web applications resources. The use of dynamic weighted average method to calculate the weight of web application indicators combined with the gray prediction algorithm to predict container scale up and down. Experimental results show that the algorithm in this paper can achieve better results.

Key words: Docker, web application, dynamic weighted average method, gray prediction.

INTRODUCTION

Cloud computing is the integration of limited infrastructure, abstraction and fine-grained resources and provided to users on demand [1]. Because of on-demand and elastic scaling of cloud computing is ideal for deploying web applications. More and more companies are gradually deploying web applications on the cloud platform, which not only saves the expenditures and processes of purchasing physical devices, but also does not worry about possible waste of resources on the server [2].

However, in order to respond to dynamically changing workloads, application scenarios, and service quality goals, application providers hope to dynamically adjust cloud computing resources in a scalable manner. Traditional virtual machine-based resource management is overloaded, making it difficult to implement fine-grained resource dynamic adjustments and rapid migration of application services in the cloud platform.

Google has developed a new virtualization technology based on Linux containers, which abstracts resources more advanced, enables services to control resources more finely, and gradually becomes an alternative to virtual machines. Due to the container occupies less resources, starts quickly, and has high concurrency and can run in a variety of environments, the containers have been widely used at home and abroad, such as Docker Cloud, WeChat grabs red envelopes, JingDong 618 commodity spikes, and so on. Another Gartner research report, about 70% of the world's applications will be moved to container infrastructure in 2019 [3]. However, there is a big difference between the underlying implementation of the container and virtual machine make the scheduling method is very different. The traditional method of resource management of the virtual machine can not applicable to the container technology.

Container technology is a new supplement, consumption and delivery model for cloud computing. Each service is deployed in a container, which avoids the shortcomings of traditional VM-based startup and resource granularity. We use docker container technology to deploy web applications and add or delete docker containers to adjust web

applications resources. The use of dynamic weighted average method to calculate the weight of web application indicators combined with the gray prediction algorithm to predict container expansion and shrinkage.

Web applications usually provide 7*24 hours of service [4]. Once the server is unable to respond due to high application load, it will cause long service interruptions. According to a statistic by Amazon, if the increase in service response time caused by cluster performance is only 10%, it will bring an additional 1% overhead. Google, an internet technology company, also found that if the response time of a service on a cluster increases by 0.5 second, the quantity of visiting may drop by 1/5.

Different Web applications cause different loads on the server [5]. There are many metrics available for Web application load. From the QoS point of view, there are many indicators such as the number of concurrent users, the number of active connections, the number of requests per second, and the average request response time and so on. Only from the load of the server, the web application load can divide into CPU, memory, I/O, bandwidth, and so on [6]. If you randomly select an indicator to consider, it may not be able to accurately reflect the application of the load. A variety of load forms have brought challenges to the scale up or down container.

We discard the single indicator measurement method and use the dynamic weighting method to determine the contribution of each indicator to the load based on the measurement of multiple indicators, then calculate a comprehensive indicator that can reflect the load, and finally compare the comprehensive indicator and the size of threshold to decide whether to increase or decrease the container.

Due to the differences in the underlying implementation of container technology and virtual machines, traditional VM-based scheduling does not apply to container technology. [7] Use stable matching theory to generate an optimal mapping from containers to physical servers. This resource scheduling approach based on the container in cloud environments to reduce response time of customers' jobs and improve providers' resource utilization rate. [8] Use nonlinear functions to more accurately model the cloud computing resources. At the same time, genetic algorithm is used for parameter tuning, and a container-friendly resource architecture scheme and resources are designed. The scheduling method has a delay of scaling. [9] Ant colony algorithm is used to optimize container scheduling. While the ant colony algorithm easily falls into local optimum. [10] This paper use cpu utilization as historical data and use time series to predict resource utilization. This method has a single performance index. For applications that use more memory, this method can't get better prediction and scheduling results.

MATHEMATICAL MODEL

Dynamic Weighted Average

Different types of web applications cause different loads on the server, including CPU-intensive, memory-intensive, I/O-intensive, etc. If one single indicator is considered, it will be inaccurate. Instead, we use dynamic weight average method to convert multiple indicators into single indicator Xt.

Suppose an application contains m docker containers at a time $(C_1, C_2, C_3, \dots C_m)$,The resource dimension is $\mathrm{D}[1,2,3, \dots \mathrm{d}]$, used resources are $\mathrm{Q}[\mathrm{q}_1, \mathrm{q}_2, \mathrm{q}_3, \dots \mathrm{q}_d]$, the total amount of resources allocated to a container is $\mathrm{R}[\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3, \dots \mathrm{r}_d]$.

The resource utilization of the application $\mathrm{U}[\mathrm{u}_1, \mathrm{u}_2, \mathrm{u}_3, \dots \mathrm{u}_d]$:

$$\mathrm{U} = \frac{\sum_{i=1}^{m} Q}{\sum_{i=1}^{m} R} \tag{1}$$

Threshold for a dimension resource $\mathrm{M}[\mathrm{m}_1, \mathrm{m}_2, \mathrm{m}_3, \dots \mathrm{m}_d]$:

$$\mathrm{m}_j = \frac{u_j}{\sum_{i=1}^{d} u_i} \tag{2}$$

The load applied at one moment is:

$$X_t = \sum_{i=1}^{d} m_i * u_i \tag{3}$$

Gray Prediction

A gray system is a system in which part of the information is known and part of the information is unknown. The theoretical essence of the gray system is to accumulate irregular raw data to generate a series and then re-model. The data obtained by the generated model is obtained by accumulating the generated inverse operations—the cumulative reduction generates the restored model, and then the restored model serves as the predictive model.

The grey forecasting model is established as follows:

1): Set the sequence of time series $x^{(0)}$ to be a series of n elements:

$x^{(0)} = (x^{(0)}(1), x^{(0)}(2), x^{(0)}(3), \dots x^{(0)}(n), the$ AGO generation sequence of $x^{(0)}$ is as follows:

$$x^{(1)} = (x^{(1)}(1), x^{(1)}(2), x^{(1)}(3), \dots x^{(1)}(n) \tag{4}$$

$$x^{(1)}(\mathrm{k}) = \sum_{i=1}^{k} x^{(0)}(\mathrm{i}) \ \ (k = 1,2,3, \dots n) \tag{5}$$

Mean sequence:

$$z^{(1)}(k) = 0.5z^{(1)}(k) + 0.5z^{(1)}(k-1), k = 2,3, \dots \tag{6}$$

2): According to the grey prediction method, the GM (1,1) model is derived and the differential equation established is:

$$x^{(0)}(k) + az^{(1)}(k) = b, k = 2,3, \dots n \tag{7}$$

The corresponding bleaching differential equation is:

$$\frac{dx^{(1)}}{dt} + ax^{(1)}(t) = b \tag{8}$$

3): Rememberu $= (a, b)^T$ .By least squares solution:

$$\hat{u} = (B^T B)^{-1} B^T Y \tag{9}$$

$$\mathrm{Y} = (\mathrm{x}^{(0)}(2), \mathrm{x}^{(0)}(3), \dots \mathrm{x}^{(0)}(\mathrm{n}))^{\mathrm{T}} \tag{10}$$

$$\mathrm{B} = \begin{bmatrix} -\mathrm{z}^{(1)}(2) & 1 \\ -\mathrm{z}^{(1)}(3) & 1 \\ \vdots & \vdots \\ -\mathrm{z}^{(1)}(\mathrm{n}) & 1 \end{bmatrix} \tag{11}$$

4): Solving the Algebraic Differential Equation (8):

$$\hat{x}(k+1) = \left(x^{(0)}(1) - \frac{b}{a}\right) e^{-ak} + \frac{b}{a}, k = 0,1, \dots, n-1, \dots \tag{12}$$

among them:

$$\hat{x}^{(0)}(k+1) = \hat{x}^{(1)}(k+1) - \hat{x}^{(1)}(k), k = 1,2, \dots, n-1, \dots \tag{13}$$

5): Accuracy test. Calculate the relative residual of original data and prediction data according to formula (14)

$$\varepsilon(k) = \frac{x^{(0)}(k) - \hat{x}^{(0)}(k)}{x^{(0)}(k)}, k = 1,2, \ldots . n \quad (14)$$

If $|\varepsilon(k)| < 0.2$, it is considered to meet the general requirement, if $|\varepsilon(k)| < 0.1$, it is considered to meet higher requirements.

Resources Scheduling

We define H_{max} as the maximum threshold value of the service load and H_{min} as the minmum threshold value of the service load. When the average load of the service is higher than H_{min} and lower than H_{max}, it indicates that the container cluster in the service is running normally. When the average load of the service is higher than H_{max}, it indicates that the container cluster in the service is in a state of high load, which can add some new containers to ensure that the application provides a reliable service. When the average load of the service is less than H_{min}, it indicates that the container cluster in the service is in a low load state, which can reduce the number of containers and save energy consumption.

EXPERIMENTS

We have 3 nodes running Ubuntu 14.04 with 4G memory and 4-core 2.33GHz CPU. Each node has been deployed with Docker 1.10. We also have deployed a sample web application in the docker container, using the tomcat container, and the database uses a specialized MySQL server. We define the highest threshold of service load H_{max} is 0.65, and the minimum threshold of service H_{min} is 0.20.

The experiment uses the algorithm proposed in this paper to predict the load of the application and get the error between the predicted value and the actual value, as shown in FIG. 1. It can get more accurate predicated value with 8.1% average forecast error in most time.

$$err = \frac{|v_{prediction} - v_{actual}|}{v_{actual}} * 100\% \quad (15)$$

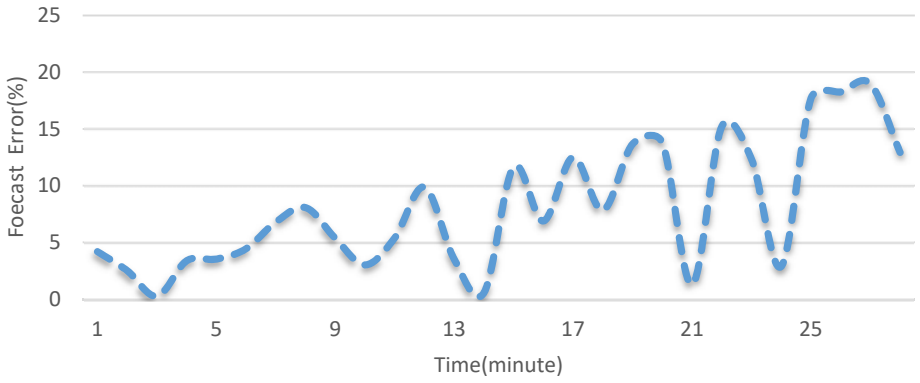

FIG 1. Forecast error

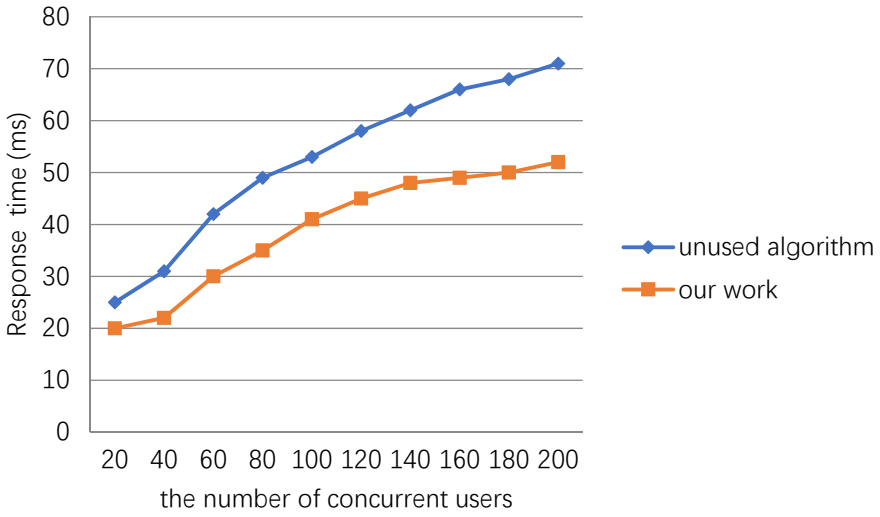

FIG 2. Response time

CONCLUSION

In this paper, we use docker to deploy web application and propose an auto-scaling algorithm. The experiments show our algorithm can scale-up or scale-down the number of container to meet the requirement of various types of Web applications.

ACKNOWLEDGMENTS

This work is supported by China National Key Research and Development Plan “Cloud Computing and Big Data Key Projects” under the grants NO. 2017YFB1001700.

REFERENCES

1. Haowei Yu. Design and implementation of Web container and scheduling in PaaS cloud [D]. Beijing University of Posts and Telecommunications,2015. (in Chinese).
2. T. Truong Huu, G. Koslovski, F. Anhalt, J. Montagnat, P. Vicat-Blanc Primet, Joint elastic cloud and virtual network framework for application performance-cost optimization, Journal of Grid Computing, 2011, pp. 1-21.
3. Petrucci V, Laurenzano M, Doherty J, et al. Octopus-man: QoS-driven task management for heterogeneous multicores in warehouse-scale computers. 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA). IEEE Computer Society. 2015. 246–258.
4. Junfeng Bian. Design and implementation of resource scheduling and application container cluster management system based on Docker [D]. Shandong University,2017. (in Chinese).
5. Yunchun Li. Yumeng Xia. Auto-Scaling web application in hybrid cloud based on Docker 2016 5th International Conference on Computer Science and Network Technology (ICCSNT).2016.75-79.
6. Jiang J, Lu J, Zhang G, et al. Optimal cloud resource auto-scaling for web applications[C]//Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on IEEE, 2013:58-65.
7. Xin Xu, Huiqun Yu, Xin Pei. A novel resource scheduling approach in container based on clouds.2014 IEEE 17th International Conference on Computational Science and Engineering. 2014.257-264. (in Chinese).

8. A Shu, Xin Peng, Wenbiao Zhao. Adaptive cloud computing resource management method based on container technology [J]. Computational Science. 2017, 44(7):120-127.
9. Chanwit Kaewkasi. Kornrathak Chuenmuneewong. Improvement of Container Scheduling for Docker using Ant Colony Optimization.2017, IEEE.
10. Li Xiao Miao. Research on Hybrid ouster scaling method based on Docker container[D].
11. Xing Lu. Large scale cloud data center load optimization scheduling method [D]. Zhejiang University, 2014. (in Chinese).
12. Baohua Yang, WangJian Dai. Introduction and practice of Docker technology, [M]. Mechanical Industry Press, 2015. (in Chinese).