IJCSI International Journal of Computer Science Issues, Volume 13, Issue 6, November 2016
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org https://doi.org/10.20943/01201606.814

A dynamic deployment method of micro service oriented to SLA

Zhen-Ling Ji[1] and Yong Liu[2]

[1] Information Engineering College, Henan University of Science and Technology, Luoyang, 471023, China

[2] Information Engineering College, Henan University of Science and Technology, Luoyang, 471023, China

Abstract

In order to solve the problem of frequent dynamic deployment of micro service in cloud computing environment under different load conditions, a dynamic deployment method of micro service oriented to SLA is proposed. A dynamic description model of micro-service is built, endow the micro service and its dependencies with dependency weight, and offer a dynamic update mechanism of the dependency weight ,then deploy a corresponding number of multilayer dependencies according to the dependent weights when need to deployment the micro service. Finally, it compared and analyzed the actual effect, the results show that this method can effectively reduce the times of micro-services dynamic deployment and guarantee the minimum violation rate of service-level objectives.

Keywords: *cloud computing, microservice, dynamic deployment, SLA, dependency, service level agreement.*

1. Introduction

In recent years, with the development of cloud computing and the rapid expansion of web applications, it is need to change the web application development and deployment model urgent to accelerate the application of continuous integration and delivery, and reduce the costs of web application on the development and maintenance continuously[1]. The concept of micro-service architecture was proposed in this background[2-6]. At present, most of the web application is still the monolithic architecture, that is, regards all the web application as a entirety to develop and deploy.In this architecture, the application does not have the error isolation and on-demand scalability, companies have to run the application and maintenance of increased investment, according to statistics more than 90% of the enterprise budget for system maintenance [7].With the large-scale deployment and the increasing complexity of web applications ,its architecture model transformed into micro-service architecture gradually to enhance the fault tolerance, scalability and on-demand deployment capabilities.However, we will face the following problems at the same time: the dependence of micro-services on different runtime environments, the dependency between micro-services and how to guarantee the timely response of user requests.

Different micro-services can be developed by different technology stack, and the run-time environment is generally not the same. Micro services are usually packaged in the docker image, and distributed through the docker mirror to Adapt to different run-time environments[11]. The cloud infrastructure provider provides and assigns computing resources to the web application owner and signs a service level agreement (SLA) that includes one or more service level objectives. This level of service goal is the application performance that the application owner must assure to the application user, such as "web application response time can not be higher than 100ms", "web application throughput not less than 200 requests / s" and so on. IaaS providers should improve resource utilization and reduce unnecessary waste of resources while allocating sufficient resources to meet the service level objectives of the web application.In order to meet the service level agreement while enabling resource utilization to maximize,how to dynamically deploy micro services must be considered in the cloud computing environment.

Front-end micro-services and back-end micro-services are two types of micro-services. The back-end is a multi-level structure, different levels of micro-services have dependency relationships, and an upper micro-services may dependent on multiple lower-level micro-services.Micro-services are generally difficult to meet the needs of users under different load requirements with the number of micro-services after the initial deployment. In general, there is a problem that the request is difficult to get a timely response with a high load state and the utilization rate of resources can not be effectively utilized with a low load state. In order to solve this problem, we must deploy the micro-services dynamically with the load at runtime.At present, there are few researches on the dynamic deployment of micro-services. The literature[13] implemented an automated deployment of micro services by defining a model of the monitoring infrastructure that provides an interface between the user and the cloud management system.The literature[14] presents an approach for the incremental integration of microservices that allows the application developers to specify and

design microservice integration, and provide mechanisms with which to automatically obtain the implementation code for business logic and interoperation among microservices along with deployment and architectural reconfiguration scripts specific to the cloud environment in which the microservice will be deployed. Although the above two methods can deploy micro-services dynamically, but they just deploy a single micro-service without considering its dependencies, then the dependencies will become the bottleneck of the response, and it will lead to frequent deployment problem.However, dynamically increasing or decreasing the number of micro-services will take some time, and affect other activities on the nodes, the frequent deployment of micro-services will lead to increased cloud load and application performance degradation, so that the service level agreement will not be completed.

To solve the problem, this paper proposes a method for dynamic deployment of micro service oriented to SLA by analyzing the distribution of resources in micro-service clusters. This method gives the dependence weight between the micro service and the dependent micro service by constructing the micro service dynamic deployment description model, and updates the weight dynamically based on the response time of the dependent micro-service.When the dependency weight reaches a certain threshold, the bottleneck point of the response is found by analyzing the dependency tree of the micro service, then calculate the number of deployments required of the bottleneck point and the multilayered micro services.The dynamic deployment method proposed in this paper can guarantee the lowest violation rate of the service level agreement, and can effectively reduce the times of dynamic deployment of micro-services.

2. Dynamic deployment method

In order to achieve the purpose of deploying multi-tier dependencies automatically when deploying micro-services,In this paper, the dynamic deployment model oriented to SLA of micro-services is designed on the basis of the above papers, which consists of deployment server and deployment client composition, as shown in Figure 1.

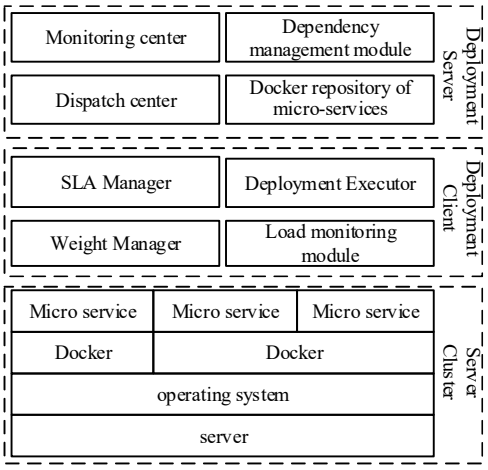

Fig. 1 Dynamic deployment architecture

Deployment server:
1) Monitoring center: For detecting load module acquisition node load information collection, summary, analysis of the current cluster load state, and provide the basis for node selection before deployment.
2) Dependency management module: Responsible for analyzing and finding out the bottleneck points of the request of the micro service dependency tree, as well as all the dependent quantities needed to calculate the dynamic deployment according to the weight of each micro service during the dynamic deployment.
3) Dispatch center: Work with dependent management module and monitoring center before deploying and responsible for the deployment of nodes.
4) Docker repository of micro-services It is used to store all the micro services for web applications. The micro service and its runtime environment are packaged together into a docker image, and the docker mirror will be uploaded to the docker repository of micro-services after the test. The advantage of packaging a mirror in a docker container is: a) It simplifies the installation process of the micro service, simply copy the image to a docker running environment on the machine and use a unified command to complete the installation and operation of a single micro service. b) Docker container as the intermediate layer of the micro service operation, ensure the consistency of the micro service operation environment, reduce the risk of the application can not run due to the operation environment of the development test and deployment phase.

Deploy client:
1) Deployment executor: According to the dispatch center issued the deployment instructions and server node communication, complete the micro service installation, destruction, operation and migration and other tasks.

2) SLA Manager: In charge of managing the response time information of the micro service, including the average response time of the request and the service response time of the SLA.
3) Load monitoring module: Responsible for monitoring the load status of all server nodes, including CPU usage, memory usage, hard disk usage and network bandwidth usage, and report to the deployment of server monitoring center.
4) Weight Manager: Responsible for forecasting and record the weights between the micro service, when the instantaneous weight exceeds the threshold value will depend on the weight information submitted to the server, to distribute the pressure on the management module of the server.

The dynamic deployment of the micro service is triggered by the change of the micro service response time. In this paper, we design the automatic deployment process according to the dynamic deployment architecture, as shown in figure 2.

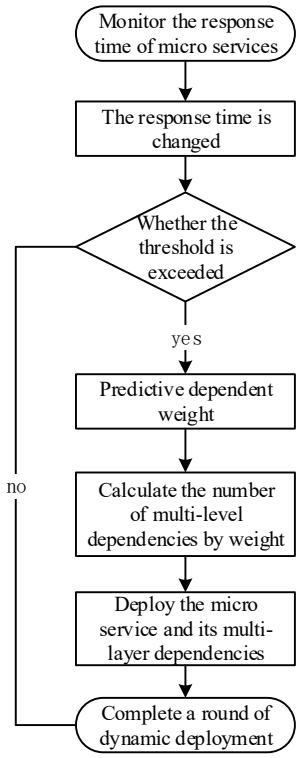

Fig. 2 flow chart of micro service dynamic deployment

The specific steps are as follows:
1) SLA manager collects the response time of the micro service and judges whether it exceeds the threshold value. The response time information will be passed to the weight manager in order to calculate the current dependency if the response time exceeds the threshold value.
2) According to the historical data and the current response time to predict the weight, the weight manager reports the response time and the predict results to the server's dependency management module.
3) The dependency management module analyzes the entire dependence tree, analyzes the bottleneck points of the response in the dependent tree according to the dependent weight information, and then calculates the number of the micro services which directly and indirectly depends on the bottleneck point and informs the dispatch center of the dynamic deployment response.
4) According to the deployment requirements, the dispatch center requests the monitoring center to acquire the node information in the server cluster and select the appropriate node from the server cluster to issue the deployment instructions to the deployment executor.
5) The deployment executor downloads the specified micro-service image from the micro-service image repository according to the deployment instructions, copies the micro service image to the designated server node and starts the micro service.

From the micro-service architecture and dynamic deployment process, it can be seen that the dynamic deployment needs to select the appropriate server nodes according to the basic properties of the micro-services and the hardware parameters of the server before implementation, and it is necessary to maintain dependency weights between micro services to trigger dynamic deployment at runtime. This paper abstracts the deployment description model of micro-service, encapsulates the basic attributes, physical constraints, dependencies and deployment configurations of the micro-services, and selects and maintains the dependent weights and dependencies among the micro-services . The model structure is shown in figure 3.

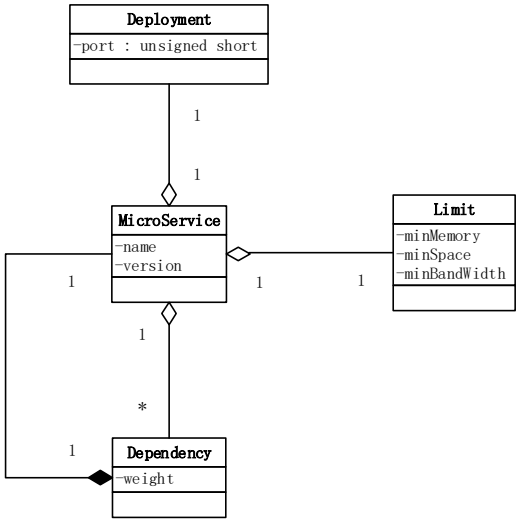

Fig. 3 micro-service deployment description model

The micro-service including name, version, constraint information (Limit), the deployment of information (Deployment) and the dependent set; constraints information (Limit) refers to the deployment of micro-services hardware constraints, such as minimum memory, minimum disk space , Minimum bandwidth, etc .; Dependency (Dependency) refers to the micro-service dependent on other micro-services, including micro-service information and rely on the weight (weight); Deployment Information (Deployment) refers to the deployment of micro-service-related parameters, such as outbound services When using the port number.Taking a certain crawler system as an example, this micro-service includes MS-Control, MS-Grab, MS-Extract, MS-ResultCollect, where MS-ResultCollect depends on MS-Extract, MS-Extract depends on MS-Grab, and MS-Grab relies on MS-Control. MS-Extract corresponding deployment description model example shown in Figure 4.

```
{
"name":"MS-Extract",
"version":"1.0",
"limit":{
"minMemory":"40MB",
"minSpace":"27MB",
"minBandWidth":"500KB/s"
},
"dependencies":[
{
"microService":{
"name":"MS-Grab",
"version":"1.0"
},
"weight":1
}
],
"deployment":{
"port":8081
}
}
```

Fig. 4 deployment description model example

In the micro-service dynamic deployment descriptive model, the dependency weight is used to represent the upper-level dependence on the underlying micro-services, and the dependent weights of the micro-services are changed as the basis for dynamic deployment. However, in order to prevent the micro-service in a short period of time response time is too long and lead to micro-services frequently initiated dynamic deployment of the phenomenon occurs, this paper introduces the concept of instantaneous weight, used to express the current short-term reliance on weight. When the response time of the micro-service is increased, the instantaneous dependency weight is appropriately increased, whereas the instantaneous dependent weight is appropriately reduced. When the instantaneous dependent weight reaches the threshold preset by the administrator, the dynamic deployment of the micro-service is triggered and the micro-service deployment right is updated value. The flow shown in Figure 5.

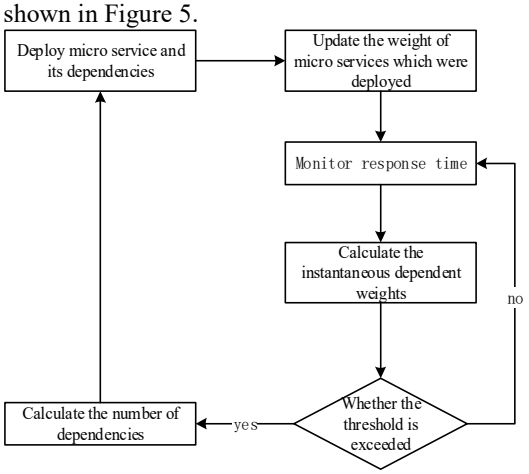

Fig. 5 update process of the dependent weight

The size of the threshold will affect the dynamic deployment of the system agility, it can be adjusted by the type and nature of the application. If the threshold is too large, the system will not be able to self-adjust in a timely manner so that web applications can not respond to user requests in a long time; if the threshold is too small, it may lead to frequent self-adjustment of the system and increase the burden on the cloud environment And a waste of resources.

The dependence weight is related to the response time of the lower layer micro-service when the upper layer accesses the lower layer micro-service. When the response time increases, the dependency weights between the micro services are increased to trigger the dynamic deployment, and the load is distributed to reduce the response time. Because the artificial neural network has a strong ability of fitting [15], this paper uses the three-layer BP artificial neural network to forecast the dependence weight. The response time is taken as input, and the dependent weight is output.Since the input and output have only one parameter, the input-output relational function is simplified to

$$Y=f(WX-\theta) \quad (1)$$

θ is a threshold value and is initialized to a random number close to zero. $f(\bullet)$ is the S function. The principle of the neural network algorithm is to find the appropriate weight coefficient W to produce the expected Y_d for the

sample X and to adjust the weight W according to the deviation between the expected output and the actual output until the deviation e is within the acceptable range. In order to facilitate data processing and speed up the convergence, the tesponse time and dependency weights should be normalized at first, the normalized formula is

$$X' = \frac{X - min}{max - min} \quad (2)$$

min is the minimum value in the sample and max is the maximum value in the sample.

Then the network training through the sample, assume that the actual output at time t is

Then the expected output and the actual output deviation is

$$e = Y_d - Y(t) \quad (4)$$

After correction is

$$W(t+1) = W(t) + e \bullet \eta \bullet X \quad (5)$$

η is the learning rate, $0 < \eta \leq 1$.

ε is the minimum allowable deviation . The actual output is considered to be approximate to the desired output when $e < \varepsilon$. At this time, the weight coefficient W can be obtained after the network training is completed. Otherwise, the next round of training will be continued.

When the response time of the micro-service is changed, it can be predicted by trained neural network, and then the corresponding weight of the response time can be obtained by processing the prediction result by inverse normalization. The dependence of micro-services from top to bottom can be seen as a dependency tree, the path between parent and child nodes is dependent weight, the number of deployments per microservice in the dependency tree is the product of the path from that node to the root node, it is denoted by

$$WML = W_1 \bullet W_2 \bullet \ldots \bullet W_n \quad (6)$$

3. Results and Analysis

In this paper, the deployment client is implemented on the OpenStack platform. The order management module of a mall application is deployed to the cloud platform, and the response time of front-end micro-services and the number of deployed micro-services are collected under different loads. Thus proving the effectiveness of the deployment method. In the experiment, 10 virtual machines are used to simulate the cluster of servers, which are configured as single core CPU, 1G memory, 20G hard disk and CentOS6.5 operating system. The dependencies of web application order management module are shown in figure 6.

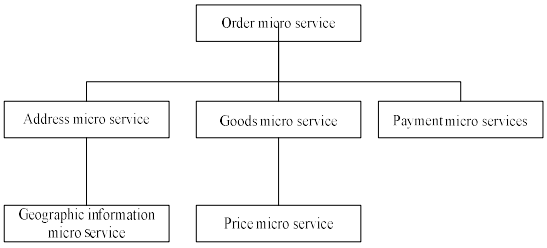

Figure 6 order management micro service dependency graph

Using the curl-loader simulation to generate the load after the first deployment of the micro-service, and record the response time of web applications under different loads and the number of micro services that make up the web application, as shown in Table 1.

Table :1 micro-service automated elastic telescoping deployment of experimental data

Load (times / minute)	1000	5000	10000	5000	1000
Average response time(ms)	42	47	45	44	43
Totals of microservices	14	21	28	21	14
Consumption time (s)	-	10	12	9	7

The dynamic deployment of micro-services was triggered when the load from 1000 / min gradually increased to 10,000 times / min, the number of micro-services deployed gradually increased, while the average front-end micro-service response time remained unchanged. As the same, the dynamic deployment of micro-services was triggered when the load from 10000 times / min and gradually reduced to 1000 times / min, the number of micro-services deployed to reduce the corresponding front-end micro-service average response time remained basically stable.

The experimental data shows that the deployment system can automatically adjust the number of micro-services in a short period of time to meet the current load demand and the front-end micro-service response time will not change significantly when the load changes to a certain extent.

Figure 7-8 depicts the results of an evaluation of the dynamic deployment method of the same experimental configuration in literature13, literature14, and this paper. The dynamic deployment method of the literatures 13 and 14 does not consider the dependence of the micro services. When the load increases, the response time of the uppermost micro-service increases, and the number of the top-level micro-services was dynamically increased in order to ensure the timely response of user requests. However, the number of micro services underneath did not increase. In the micro-service dependency tree, the response bottleneck begins to move to the underlying micro-service, the deployment system again increases the number of lower-level micro services, and so on, frequent deployments will be triggered. Conversely, the

deployment system is still reduce the number of the micro services frequently. The performance of the entire web application is reduced due to frequent deployment, and the violation rate of the service level agreement is increasing. In this paper, when the load changes, the instantaneous dependence weight is first changed, and the bottleneck point of the response is found by analyzing the dependence tree of the micro service when the instantaneous dependence weight reaches the threshold. Then calculate the bottleneck and the number of the following dependencies based on the dependent weight. Finally, the deployment of the bottleneck point micro-services at the same time the deployment of the corresponding number of dependent micro-services, which reduces the times of dynamic deployment of micro-services, and to ensure that the SLA violation rate is lowest

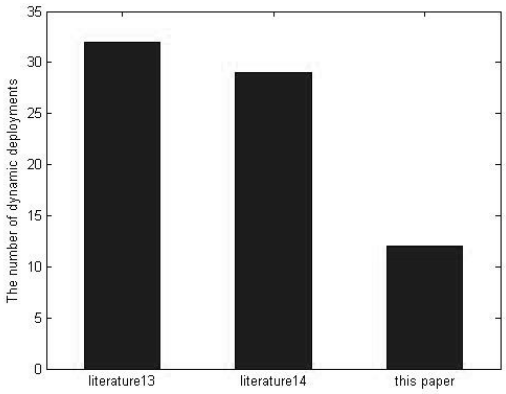

Fig. 7 The times of dynamic deployment triggers under different deployment methods

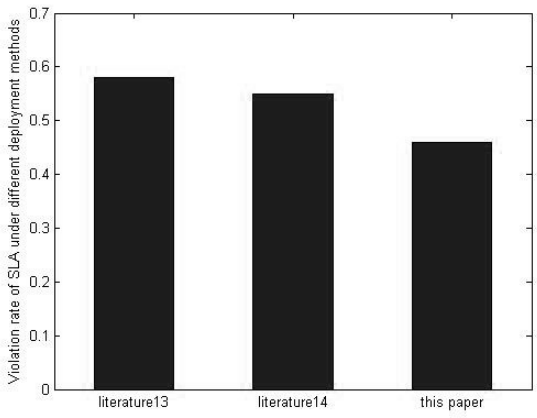

Fig. 8 Violation rate of SLA under different deployment methods

4. Conclusions

This paper studies the dynamic deployment of micro services under cloud environment, and proposes a dynamic deployment method of micro services oriented to SLA, which can effectively reduce the times of dynamic deployment of micro-services, not only guarantee the SLA, but also reduce the resource consumption and reduce the operating costs . However, the automated deployment of micro-services still has a lot of work to do, and it needs to be improved in future work, such as micro-service migration, fault detection and automatic repair.

References

[1] Guha R, Al-Dabass D. Impact of web 2.0 and cloud computing platform on software engineering[C]//Electronic System Design (ISED), 2010 International Symposium on. IEEE, 2010: 213-218.

[2] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing[J]. Communications of the ACM, 2010, 53(4): 50-58.

[3] Namiot D, Sneps-Sneppe M. On micro-services architecture[J]. International Journal of Open Information Technologies, 2014, 2(9).

[4] Kang H, Le M, Tao S. Container and Microservice Driven Design for Cloud Infrastructure DevOps[C]//Cloud Engineering (IC2E), 2016 IEEE International Conference on. IEEE, 2016: 202-211.

[5] Villamizar M, Ochoa L, Castro H, et al. Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures[C]//Cluster, Cloud and Grid Computing (CCGrid), 2016 16th IEEE/ACM International Symposium on. IEEE, 2016: 179-182.

[6] Lewis J, Fowler M. Microservices: a definition of this new architectural term[J]. 2014.

[7] Koskinen J. Software maintenance costs[J]. Jyväskylä: University of Jyväskylä, 2010.

[8] Newman S. Building Microservices[M]. " O'Reilly Media, Inc.", 2015.

[9] Fowler M, Lewis J. Microservices a definition of this new architectural term[J]. URL: http://martinfowler. com/articles/microservices. html [accessed: 2016-02-12], 2014.

[10] Guo D, Wang W, Zeng G, et al. Microservices Architecture Based Cloudware Deployment Platform for Service Computing[C]//2016 IEEE Symposium on Service-Oriented System Engineering (SOSE). IEEE, 2016: 358-363.

[11] Merkel, D.: Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux Journal 2 (2014)

[12] Wu L, Buyya R. Service Level Agreement (SLA) in utility computing systems[J]. IGI Global, 2012.
[13] Ciuffoletti A. Automated deployment of a microservice-based monitoring infrastructure[J]. Procedia Computer Science, 2015, 68: 163-172.
[14] Zúñiga-Prieto M, Insfran E, Abrahao S, et al. Incremental Integration of Microservices in Cloud Applications[J]. 2016.
[15] Farrell T J, Wilson B C, Patterson M S. The use of a neural network to determine tissue optical properties from spatially resolved diffuse reflectance measurements[J]. Physics in medicine and biology, 1992, 37(12): 2281.

Zhen-Ling Ji male, born in 1990, is a master candidate in computer software and theory at the Information Engineering College, Henan University of Science and Technology, Luoyang, China. His research direction is cloud computing and computer architecture.

Yong Liu .male, born in 1966, professor. He is a graduate tutor, mainly engaged in cloud computing and software architecture direction of the study.