A Cost-Aware Auto-Scaling Approach Based on Workload Predictions in Service Clouds

Jingqi Yang, Chuanchang Liu, Yanlei Shang, Zexiang Mao, Bo Cheng, Junliang Chen
State Key Lab of Networking and Switching Technology
Beijing University of Posts and Telecommunications, Beijing 100876, China
Email:{yangjingqi,lcc3265,shangyl,zxiangmao,chengbo,chjl}@bupt.edu.cn

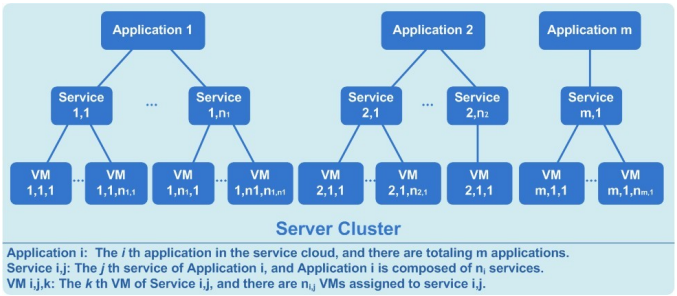

Fig. 1. Logical architecture of server cluster

I. Introduction

Service platforms have become a very popular way to provide services. But service platforms have disadvantages such as they have long construction periods, low resource utilizations and isolated constructions. Migrating service platforms into clouds can solve these problems. Some researches point out that the scalability become important for taking advantages of service cloud computing technologies. But there are not so many researchers working on the scalability of service clouds.

The main contributions of this work are as follows: (1) We propose a linear regression model to predict the workload of service cloud platforms; (2) We present a novel service cloud architecture which offers different services and an auto-scaling approach in service clouds. It combines the real-time scaling and the pre-scaling, and we consider three scaling techniques at different resource levels: self-healing scaling, resource-level scaling and VM-level scaling [1]; (3) Our experiment results show that our auto-scaling approach can meet the user SLA with less costs.

II. Auto-Scaling Approach in Service Clouds

First we present a scenario of a service cloud. Figure 1 shows the logical architecture of the server cluster in our service cloud. There are several different applications in the server cluster. Each application is composed of one or more services. Services are loosely coupled, and each of them runs on one or more VMs which have a variety of processing capabilities.

In order to achieve scalability the server cluster should add or release virtual resources dynamically according to platform running states. The general approach of scaling platform is to add virtual resources when the system detects that the system utilization exceeds the threshold. We can execute the horizontal scaling or the vertical scaling. The horizontal scaling (VM-level scaling) is not subject to the resource constraints, but it will take a while to complete it. This may lead to user SLA violations. The scaling time taken by the vertical scaling (self-healing scaling, resource-level scaling) can be ignored, but the vertical scaling is limited by resources. As a result, it is desirable if the platform can be pre-scaled earlier based on predictions, so we propose an approach to predict workloads.

Workloads of service clouds are irregular, we need a method which can adjust its model quickly according to the variation trend of workloads. We also observe that the workload trend is linear in a relatively short period of time. As a result of these, we use a linear regression model (LRM) to solve this problem. The generic form of the linear regression model is

$$Y_i = \beta_1 + \beta_2 X_i \tag{1}$$

In our scenario, Y is the workload, X is the time, and i indexes the interval. The coefficients β_1, β_2 are determined by solving a linear regression equation based on previous workloads. We use the Ordinary Least Squares to solve this equation based on previous workloads, and results are shown in (2)(3).

$$\beta_1 = \frac{\sum X_i^2 \sum Y_i - \sum X_i \sum X_i Y_i}{n \sum X_i^2 - (\sum X_i)^2} \tag{2}$$

$$\beta_2 = \frac{n \sum X_i Y_i - \sum X_i \sum Y_i}{n \sum X_i^2 - (\sum X_i)^2} \tag{3}$$

Then we can calculate the workload at next interval using (1). We compare predicted workloads with actual workloads, and a set of alternative workload prediction methods are also implemented for comparing. All of these methods use a sliding window of previous workloads. The first one is **A second order autoregressive moving average method filter (ARMA)** [2]. The equation for the filter used is given by $Y_{t+1} = \beta * Y_t + \gamma * Y_{t-1} + (1 - (\beta + \gamma)) * Y_{t-2}$. The value for the variables β and γ are given by the values 0.8 and 0.15, respectively. The other two are **Mean/Max**. The predicted workload is the mean/maximum workload in the window.

We collect workloads of a video service for 6 hours in a service cloud. The time interval is set to the time spending on booting a VM. This actual workload is compared with the other four predicted workloads including the LRM. Experimental results are shown in Table I. We can see that the LRM outperforms other methods.

Because of the abnormal and burst of workloads, there may be predicted deviations. We execute the real-time scaling in order to minimize the impact of deviations. When the

TABLE I. RESULTS OF WORKLOAD PREDICTION

Metric (%)	LRM	ARMA	Mean	Max
Min predicted deviation	0	1	1	0
Max predicted deviation	72	99	166	288
Avg predicted deviation	9	12	17	20

TABLE II. COST-AWARE AUTO-SCALING

algorithm Cost-Aware Auto-Scaling

```
1 Begin
2 Predict an initial workload, and boot VMs;
3   While (the system is running and in the beginning of a interval)
4     For (every service in the cloud)
5       Monitor platform running states;
6       Conduct the real-time vertical scaling;
7       Predict the workload at the next interval;
8       Conduct the pre-scaling at three resource levels.
9 End
```

utilization goes beyond the threshold, the vertical scaling will be applied to add virtual resources. Profiting from the short time the virtual scaling takes, this approach is effective.

In our approach a cost model is proposed to measure the scaling cost (SC). SC is composed of a virtual resource cost (VRC) and a license cost (LC). VRC is the cost of CPU, memory and other resources used by services. LC is the cost we must pay for license fees if there are business softwares running on VMs. Three scaling technologies we used (self-healing scaling, resource-level scaling and VM-level scaling) have different unit SCs. We formulate the scaling problem as an integer programming problem. In order to reducing computational overheads, we calculate the near-optimum cost solution with a greedy approach. When the platform need to be scaled, we choose a strategy with the smallest SC.

Above all, pseudo code for the auto-scaling approach is given in Table II. This algorithm is started after the service cloud platform is deployed, and it keeps running until there are no services running in the service cloud. This approach manages virtual resources of each service every time interval. When the platform is deployed, the algorithm predicts an initial workload of every service based on the experience, then boots an appropriate number of VMs for them (line 2). At every interval the algorithm first collects information of platform running states (line 5), then it performs the real-time vertical scaling (line 6), and finally it predicts the workload and pre-manages virtual resources at the next interval (line 7-8).

III. EVALUATION

The experiment is carried out on the CloudSim cloud simulator [3]. Using Amazon EC2 [4] as a reference, three types of VMs are provided in our experiment: small, medium and big, and they have different capacities and different costs. There are two applications in the experimental service cloud, and each of them is composed more than one services. The first application is a social network sites (four services), and the other one is a video site (two services).

In the experiment, we compare our approach with two baseline methods to verify the effectiveness. The first strategy adds fixed size VMs whenever the existing VMs are fully utilized, which is labeled "**Horizontal scaling**". The second strategy uses the method in [1] which is labeled "**Lightweight Scaling**". Our approach is labeled "**Auto-scaling**".

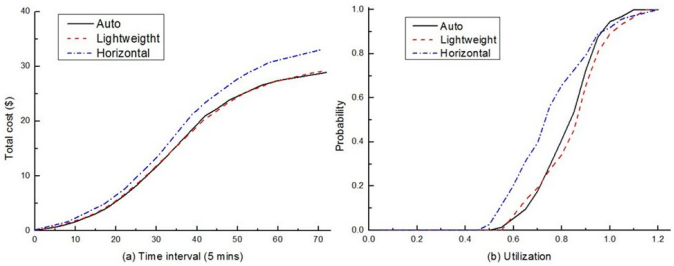

Fig. 2. (a) Cumulative SC at different intervals and (b) CDF of utilizations

From the results shown in Figure 2, the horizontal scaling performs the worst. The lightweight scaling and our auto-scaling both scale virtual resources at different levels and take costs into consideration, as a result of these in most intervals these two approaches have similar costs and average utilizations. But comparing with the auto-scaling, the lightweight scaling incurs a higher SLA violation. Situations of request dropped are observed at 9% of intervals in the lightweight scaling, while this rate of the auto-scaling is only 5% benefiting from the workload prediction and the real-time scaling.

IV. CONCLUSION

In this paper, we investigate the cost-aware auto-scaling problem in service clouds. We use a linear regression model to predict the workload, and we also propose an approach to scale the service cloud platform in both the real-time scaling and the pre-scaling. We formulate the pre-scaling problem as an integer programming problem and present a greedy algorithm to solve it. In order to implement these mechanisms, a cloud scaling architecture is presented to support our approach. According to the experiment results, our approach predicts more accurately, costs less and has a lower user SLA violation than other methods. Our approach is also generic, and it can be used in most service cloud scenarios. It also can be expanded to construct a more complex environment easily.

ACKNOWLEDGMENT

The work is supported by the 973 program of National Basic Research Program of China (Grant No.2011CB302500, 2011CB302506 and 2011CB302704), the Novel Mobile Service Control Network Architecture and Key Technologies (Grant No.2010ZX03004-001), the National Natural Science Foundation of China (Grant No.61132001, 61001118, 61171102) and the Program for New Century Excellent Talents in University (Grant No.NCET-11-0592).

REFERENCES

[1] R. Han, L. Guo, M.M. Ghanem, Y. Guo, Lightweight Resource Scaling for Cloud Applications, Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 644-651 (2012).

[2] N. Roy, A. Dubey, A. Gokhale, Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting, Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing, 500-507 (2011).

[3] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, R. Buyya, Cloudsim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisoning Algorithms, Software: Pratice and Experience, vol. 41, no. 1, 23-50 (2011).

[4] Amazon elastic compute cloud, http://aws.amazon.com/ec2/.