SCIENCE CHINA
Information Sciences

• LETTER •

November 2019, Vol. 62 219101:1–219101:3
https://doi.org/10.1007/s11432-018-9750-2

Self-adaptive resource allocation for cloud-based software services based on progressive QoS prediction model

Xing CHEN[1,2], Junxin LIN[1,2], Yun MA[3], Bing LIN[2,4], Haijiang WANG[1,2] & Gang HUANG[5*]

[1]*College of Mathematics and Computer Science, Fuzhou University, Fuzhou* 350116, *China;*
[2]*Fujian Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, Fuzhou* 350116, *China;*
[3]*School of Software, Tsinghua University, Beijing* 100084, *China;*
[4]*College of Physics and Energy, Fujian Normal University, Fuzhou* 350117, *China;*
[5]*Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing* 100871, *China*

Received 15 July 2018/Revised 30 September 2018/Accepted 12 January 2019/Published online 18 September 2019

Citation Chen X, Lin J X, Ma Y, et al. Self-adaptive resource allocation for cloud-based software services based on progressive QoS prediction model. Sci China Inf Sci, 2019, 62(11): 219101, https://doi.org/10.1007/s11432-018-9750-2

Dear editor,

Self-adaptation is a promising approach to allocate resources for cloud-based software services [1, 2]. Traditional self-adaptive resource-allocation methods are rule-driven, which leads to high administrative cost and implementation complexity. Machine learning techniques and control theory are two kinds of solutions to decrease manual efforts. However, machine learning techniques require a huge amount of historical data to build an accurate quality of services (QoS) prediction model, leading to errors in resource allocation, while approaches based on control theory need a large number of iterations of feedback to find an appropriate resource-allocation plan, leading to a high overhead of discontinuing virtual machines.

We propose a self-adaptive approach to allocate resources for cloud-based software services based on progressive QoS prediction model. First, the QoS prediction model is locally improved by self-tuning control based on runtime data under the current workload. Second, a new feedback loop is designed to allocate resources for cloud-based software services, considering the additional overhead of dynamically adjusting the allocated resources. We evaluate our approach on RUBiS benchmark, and the results show that our approach can considerably help in improving the accuracy of QoS prediction model and performance of self-adaptive resource allocation.

Problem statement. The quality of cloud-based software service changes over time, as its runtime environment changes over time. Environment changes fall into two types: external and internal, according to initiating factors. The external factors mainly refer to the workloads (R), whose changes are given in this study; the internal ones refer to the allocated resources (VM). When allocating resources for cloud-based software services, cloud engineers or self-adaptive systems should balance between the QoS (Q) and cost of resources (Cost) according to preset targets. The preset targets refer to evaluation values (Fitness) calculated by the fitness functions, as shown in (1), and better resource-allocation plans will have smaller values of fitness function.

$$\text{Fitness} = r_1 \times \frac{1}{Q} + r_2 \times \text{Cost}. \tag{1}$$

One component of evaluation value is resource cost, which mainly comes from leased cost (Cost_L)

* Corresponding author (email: hg@pku.edu.cn)

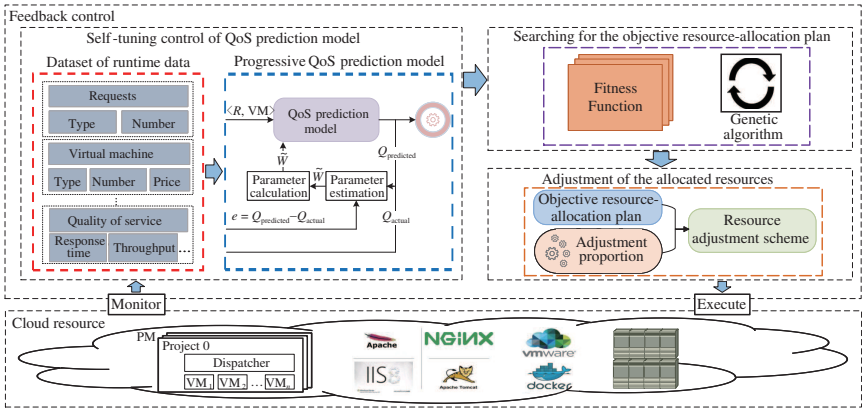

Figure 1 (Color online) Overview of the proposed approach.

and discontinued cost (Cost_D) of virtual machines, as shown in (2). Cost_L is the total price of all allocated virtual machines, and Cost_D is the total penalty value as the allocated virtual machines are shut down.

$$\text{Cost} = \text{Cost}_L + \text{Cost}_D. \tag{2}$$

The other component of evaluation value is QoS value, which can include metrics that service level agreements (SLA) usually specify, such as response time, data throughput. However, these metrics cannot be used to predict the QoS value, because they can only be monitored after the resources have been allocated. The needed expert knowledge is regarded as the QoS prediction model, as shown in (3). Its inputs contain different types of requests (R), and numbers of different types of virtual machines (VM). Its output is the predicted value of QoS (Q).

$$Q_{\text{predicted}} = \text{QoS}(R, \text{VM}). \tag{3}$$

However, it requires huge quantities of historical data to build an accurate QoS prediction model. In practice, historical data are usually inadequate and limited in variations, not covering different cases of workload and allocated resources. As a result, the QoS prediction model is not sufficiently accurate, leading to inefficiency of resource allocation.

In this study, we suggest using feedback control to improve the accuracy of the given QoS prediction model, which would improve the accuracy of the resource-allocation plan.

Our approach. We introduce feedback loop into self-adaptive resource allocation for cloud-based software services, as shown in Figure 1. For each loop, there are three steps. First, we use self-tuning control to improve the local accuracy of QoS prediction model based on runtime data under the current workload. Second, we use genetic algorithm (GA) to search for the new objective resource allocation plan based on the progressive QoS prediction model. Third, we compare the current allocated resources with the new objective plan and adjust only a certain proportion of resources based on their difference, during which virtual machines are added or deleted one by one and the runtime data are collected. The above steps are repeated until the resource assigned is the same as the new objective plan and no longer needs to be adjusted.

(a) Self-tuning control of QoS prediction model. The QoS prediction model is a calculation model, as shown in (3). The workload $R = (x_{i,1}, x_{i,2}, \ldots, x_{i,m})$ has m types of requests, and $x_{i,k}$ represents the amount of k-th type of request. The allocated resources $\text{VM} = (x_{i,m+1}, x_{i,m+2}, \ldots, x_{i,m+n})$ have n types of virtual machines, and $x_{i,m+s}$ represents the number of s-th type of virtual machine. The given QoS prediction model can be fitted by a corresponding polynomial $\text{QoS}(X)$, which is described as [3]

$$\text{QoS}(X) = \sum_{q=0}^{m+n} w_q x_q^{a_q}. \tag{4}$$

To improve the local accuracy of QoS prediction model, we use self-tuning control to adjust $\text{QoS}(X)$'s coefficient vector W based on runtime data, including information about the workload, allocated resources, predicted and actual QoS values (Q_{actual}).

There are two steps in the self-tuning control of QoS prediction model, including parameter estimation and parameter calculation. The parameter estimation aims to calculate the local optimal coefficient vector $\widehat{W}$ of $\text{QoS}(X)$ based on the original coefficient vector W and runtime data under the current workload. According to the QoS prediction model, it is possible to get a set of predicted QoS values based on the runtime data, calculated by (5). Then we use the least square method to do the estimation where the optimal coefficient vector

$\widehat{W}$ is to minimize the sum of square of the loss between the predicted and actual QoS values, which is defined as $e = Q_{\text{predicted}} - Q_{\text{actual}}$.

$$Q_{\text{predicted } j} = w_0 + w_1 x_{j,1}^{a_1} + w_2 x_{j,2}^{a_2} + w_3 x_{j,3}^{a_3} + \cdots + w_{m+n} x_{j,m+n}^{a_{m+n}},\ 1 \leqslant j \leqslant u. \quad (5)$$

The parameter calculation is aimed to set a new coefficient vector $\widetilde{W}$, which makes a trade-off between the original coefficient vector W and local optimal coefficient vector $\widehat{W}$, as shown in (6). The relative weightage η is defined by experts according to requirements for different systems.

$$\widetilde{W} = \eta W + (1 - \eta)\widehat{W}. \quad (6)$$

(b) Searching for the objective resource-allocation plan. Using the QoS prediction model, the evaluation value can be predicted according to the given workload and resources to be allocated. Thus, the self-adaptive resource allocation can be transformed to the search for most appropriate resource-allocation plan. Thus, we use GA to search for an appropriate resource-allocation plan. Assume that there are p types of virtual machines, and the number of virtual machines of each type is represented as $\text{vm}_v (1 \leqslant v \leqslant p, \text{vm}_v \geqslant 0)$. Therefore, the chromosome coded for resource-allocation plan can be defined as $\text{VM} = (\text{vm}_1, \text{vm}_2, \ldots, \text{vm}_p)$. The preset targets refer to evaluation values calculated by the fitness functions, as shown in (1), this means that a better resource allocation plan is allotted a smaller value of fitness function.

(c) Adjustment of the allocated resources. For each loop, the QoS prediction model is progressive and the objective resource-allocation plan also usually changes. If the adjustment of the allocated resources goes completely according to the objective plan (VM_O), we will always need to discontinue some leased virtual machines for each loop, which leads to high discontinued cost (Cost_D) of virtual machines. Thus, we adjust only a certain proportion (P) of resources based on the difference (VM_D) between the allocated resources (VM_A) and the plan for each loop. In addition, we define the sum (VM_S) of VM_D and the rounding function Intpart() as follows:

$$\text{VM}_D = (\text{vm}_1^O - \text{vm}_1^A, \text{vm}_2^O - \text{vm}_2^A, \ldots, \text{vm}_p^O - \text{vm}_p^A), \quad (7)$$

$$\text{VM}_S = \sum_{v=1}^{p} \text{VM}_D(v), \quad (8)$$

$$\text{Intpart}(a) = \begin{cases} \lceil a \rceil, & a \geqslant 0, \\ -\text{Intpart}(-a), & a < 0. \end{cases} \quad (9)$$

We propose the decision algorithm to calculate the adjustment scheme of the allocated resources. If each item of the array VM_D is 0, it indicates that the resource assigned is the same as the new objective plan and no longer needs to be adjusted. Otherwise, we guide the overall trend of resource allocation (increase or decrease) based on the value of VM_S. If VM_S is greater than or equal to 0, the allocated resources need to be increased, else the allocated resources need to be decreased. Then we check each item in the array VM_D and calculate the adjustment scheme. For instance, when VM_S is greater than or equal to 0 and the checked item is positive, we multiply it by the adjustment proportion P and round it by the rounding function Intpart(); if the checked item is negative, we take its value as 0.

Experimental results. We set up a cloud environment and use the RUBiS benchmark to evaluate our self-adaptive resource allocation. Results show that the proposed approach can help improve the accuracy of QoS prediction model by 20%–24% and the performance of self-adaptive resource allocation by 4%–8%. For detailed results of the experiment, please refer to Appendix A.

Conclusion. We have proposed a self-adaptive approach to resource allocation for cloud-based software services based on progressive QoS prediction model. This approach can significantly help in improving the accuracy of QoS prediction model and the performance of self-adaptive resource allocation.

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2018YFB-1004800), National Natural Science Foundation of China (Grant No. 61725201), and Talent Program of Fujian Province for Distinguished Young Scholars in Higher Education.

Supporting information Appendix A. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Hummaida A R, Paton N W, Sakellariou R. Adaptation in cloud resource configuration: a survey. J Cloud Comput, 2016, 5: 7
2 Chen X, Lin J X, Lin B, et al. Self-learning and self-adaptive resource allocation for cloud-based software services. Concurr Comput Pract Exper, 2018, 19: 4463
3 Drwal M, Borzemski L. Prediction of web goodput using nonlinear autoregressive models. In: Proceedings of the 23rd International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, Cordoba, 2010. 347–356