SCIENCE CHINA
Information Sciences

• **Supplementary File** •

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* 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*

Appendix A Experiments

We set up a cloud environment and use the RUBiS benchmark to evaluate our self-adaptive resource allocation. First, we validate whether our approach is feasible to allocate resources for cloud-based software services dynamically. Second, we compare the accuracy of the progressive QoS prediction model with the original one. Third, we compare the performance of our approach with the traditional and rule-driven ones. Last, we evaluate the overhead of our approach.

Appendix A.1 Experimental Setup

The RUBiS benchmark is an auction site prototype modeled after eBay.com [1]. It provides a client which simulates user behavior for various workload patterns. And the number of clients indicates the amount of workload. We simulate an actual workload which has two types of user behaviors, and the average workload and workload pattern change respectively for each time interval of an hour, as shown in Figure A1.

There are three types of virtual machines in this experiment, as shown in Table A1. The numbers of virtual machines of each type are represented as vm_s, vm_m and vm_l, respectively. Therefore, the allocated resources can be represented as VM_A=(vm_s,vm_m,vm_l).

Table A1 Three types of virtual machines.

Property	Small	Medium	Large
CPU	1core	1 core	1 core
Memory	1G	2G	4G
Cost_L	1.761 RMB	1.885 RMB	2.084 RMB
Cost_D	0.440 RMB	0.471 RMB	0.521 RMB

The quality of service refers to the response time (RT) in this experiment, and its value is calculated by a Sigmoid function, as shown in Figure A2.

The fitness function represents the resource-allocation target given by cloud engineers. Better resource-allocation plan shall get a smaller value of fitness function. The weightages (r_1 and r_2), which are pre-defined by cloud engineers, reflect their different preferences on QoS and resource cost. For instance, a higher r_1 represents a more sensitive preference on QoS, so more virtual machines are needed in order to guarantee the QoS under the same workload; while a higher r_2 represents a more sensitive preference on resource cost, so less virtual machines are needed in order to reduce the resource cost. In practice, the most common fitness function is to balance the QoS and resource cost, which is also challenging to achieve

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

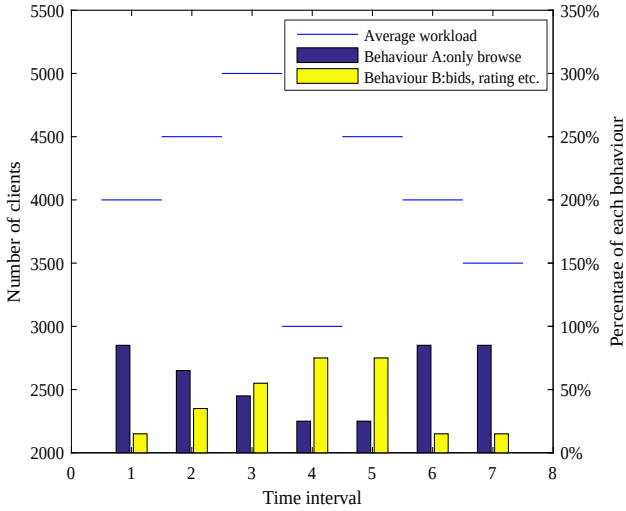


Figure A1 Average workload and workload pattern for each time intervals.

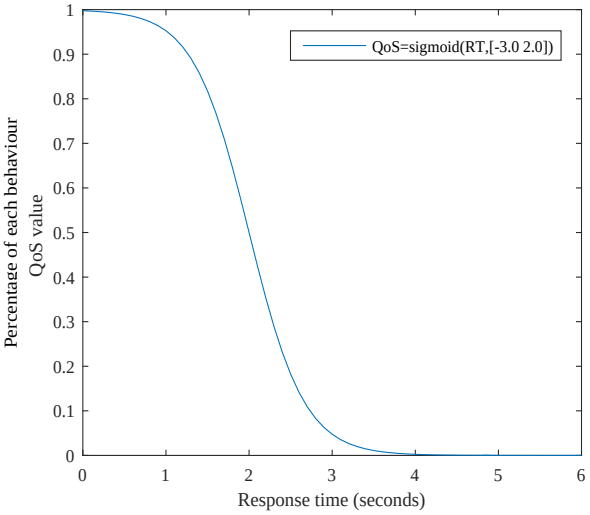


Figure A2 Service level agreement.

due to the complex relationship between the resource and the QoS of cloud services. Therefore, in our experiment, we set $r_1 = 320$ and $r_2 = 10$ based on our experience in order to balance the QoS and resource cost, as shown in Formula A1.

$$Fitness = 320 * \frac{1}{Q} + 10 * Cost\,, \tag{A1}$$

The parameter calculation in self-tuning control, is aimed to set a new coefficient vector $\widetilde{W}$, and the weightages ($0 < \eta < 1$), which is set by cloud engineers, reflects confidence levels on the original coefficient vector W and the local optimal coefficient vector $\widehat{W}$. A higher η represents a higher confidence on the original coefficient vector W, so parameters of the new coefficient vector $\widetilde{W}$ is closer to the original coefficient vector W. In our experiment, we set the weightages ($\eta = 0.9$) based on experience, considering the volume of historical data is much larger than runtime data under the current workload.

The relevant parameters of genetic algorithm are set as follows. The size of initial population is 100, and the maximum number of iterations is 1000. Besides, the algorithm will end if the optimal value is not updated for 20 consecutive iterations.

For each loop, we adjust only a certain proportion of resources based on the difference between the allocated resources and the objective plan. The adjustment proportion (P) is pre-defined by cloud engineers and reflects their preferences on efficiency and overhead of resource allocation. A lower P represents a more sensitive preference on efficiency, which will lead to a higher number of feedback iterations; in this situation, it takes more adjusting time to find out a better resource-allocation plan. In our experiment, we set the adjustment proportion ($P = 25\%$) based on experience, in order to balance the efficiency and overhead of resource allocation.

We use two QoS prediction models in this experiment. In order to describe the accuracy of QoS prediction model, we define the relative error value $R = |QoS_{predicted} \,/\, QoS_{actual} \;\text{-}\; 1|$, and introduce the admissible relative error E, and the confidence level L, as shown in Formula A2. The admissible relative error in this experiment is set to 0.3, and the confidence levels of two models are respectively 75% and 85%.

$$L = Lr(\,|\,QoS_{predicted} \,/\, QoS_{actual} \;-\; 1\,|\leqslant E\,)\,. \tag{A2}$$

For comparison, we apply our approach, the traditional one based on QoS prediction model, and the rule-driven one to support self-adaptive resource allocation for comparison. The traditional one directly uses the genetic algorithm to search for an appropriate resource-allocation plan based on the QoS prediction model. The rule-driven one follows the rules described in Table A2.

Table A2 Rules for resource allocation.

Conditions	Operations
RT >1.4s	vm_l += 1
1.2s <RT $\leqslant$ 1.4s	vm_m += 1
1.0s <RT $\leqslant$ 1.2s	Remain
0.8s <RT $\leqslant$ 1.0s	vm_m -= 1
RT $\leqslant$ 0.8s	vm_l -= 1

Appendix A.2 Validation of Self-Adaptive Resource Allocation

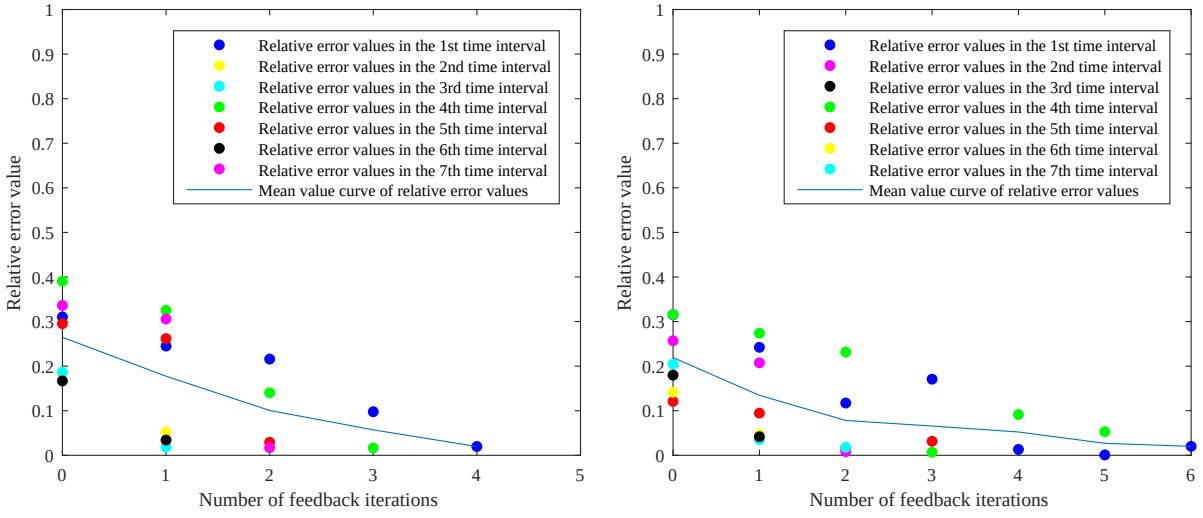
Table A3 shows the resource-allocation plans for each intervals when using three approaches. We can see that, there are many factors which impact the resource-allocation plans of our approach. First, the average workload and workload pattern impact the resource-allocation plans significantly. For instance, there are striking differences between allocated resources of each intervals. Second, allocated resources of the previous interval can impact the resource-allocation plans. For instance, the average workloads and workload patterns of 1^{st} and 6^{th} intervals are the same, while there are differences between allocated resources of the two ones. Third, QoS prediction models also impact the resource-allocation plans. For instance, there are differences between allocated resources of same intervals when respectively using two QoS prediction models.

Table A3 Resource-allocation plans for each intervals (vm_s , vm_m , vm_l).

QoS Prediction Models	Self-Adaptive Approaches	1^{st} Time Interval	2^{nd} Time Interval	3^{rd} Time Interval	4^{th} Time Interval	5^{th} Time Interval	6^{th} Time Interval	7^{th} Time Interval
The One with the Confidence Level of 85%	Our Approach	0,2,5	0,4,5	0,5,5	0,0,5	0,1,7	0,1,7	0,0,6
	Traditional Approach	0,2,6	0,3,6	0,4,6	0,0,6	0,0,8	0,0,8	0,1,6
	Rule-driven Approach	0,1,7	0,2,7	0,3,7	0,1,6	0,1,8	0,2,7	0,2,6
The One with the Confidence Level of 75%	Our Approach	0,3,5	0,3,6	0,4,6	0,2,5	0,2,7	0,1,7	0,1,5
	Traditional Approach	0,2,7	0,3,7	0,4,7	0,1,6	0,2,8	0,1,8	0,1,7
	Rule-driven Approach	0,1,7	0,2,7	0,3,7	0,1,6	0,1,8	0,2,7	0,2,6

Appendix A.3 Accuracy Improvement of QoS Prediction Model

We evaluate the accuracy improvement of QoS prediction model made by our approach based on comparison of prediction error between the original and progressive ones. We calculate each relative error values when using the QoS prediction model, as shown in Figure A3. We can observe that self-tuning control method can significantly improve the accuracy of QoS prediction model.



(A) Using the less accurate one.

(B) Using the more accurate one.

Figure A3 Relative error values when using different QoS prediction models.

First, it is a progressive course to improve the local accuracy of QoS prediction model, and the relative error values of model are reduced when the number of feedback iterations increases. For instance, the average relative error value of

the more accurate one is 0.23, when the number of feedback iterations is 0; in contrast, the average relative error value is reduced to 0.08, when the number of feedback iterations is 3.

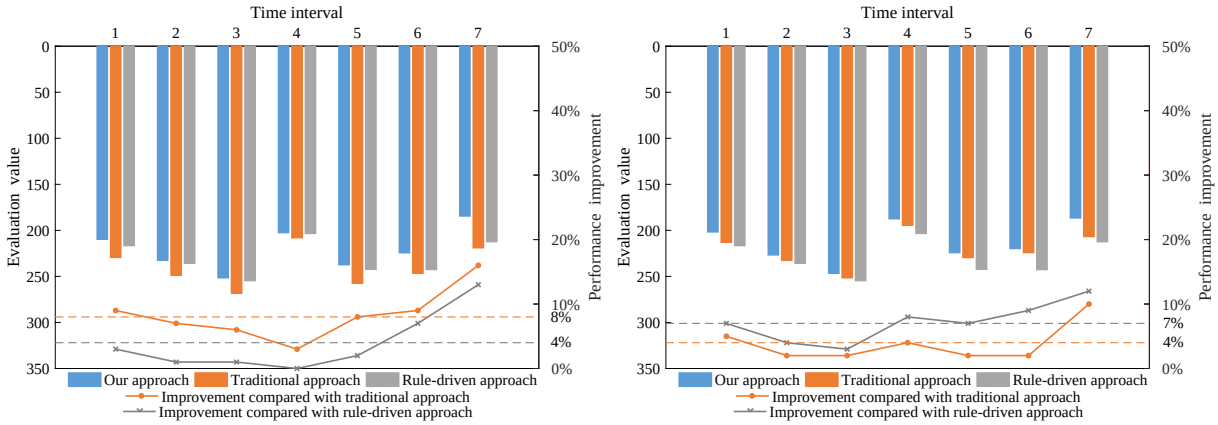
Second, two QoS prediction models can both benefit from our approach, but with the less accurate one, the impact is more pronounced. For instance, the relative error values of the less accurate one is reduced by 24% on average, while the relative error values of the more accurate one is reduced by 20% on average.

Appendix A.4 Performance Improvement of Self-Adaptive Resource Allocation

We evaluate the performance improvement of self-adaptive resource allocation made by our approach, based on comparison of evaluation values of resource-allocation plans between three approaches, as shown in Figure A4. The evaluation values are calculated according to Formula A1. We can see that the evaluation value in our approach is the smallest in each case, which indicates that the plan is usually the best one.

Compared with the traditional approach, our approach can improve performance by 8% on average when using the less accurate QoS prediction model. In contrast, it can only improve performance by 4% on average when using the more accurate one. It is because that the accuracy of QoS prediction model can affect the efficiency of resource allocation. Our approach can improve the accuracy of QoS prediction model, and the impact is more pronounced with the less accurate one, as mentioned in Appendix A.3. Therefore, when compared with the traditional approach in efficiency of resource allocation, the impact of our approach is also more pronounced with the less accurate QoS prediction model.

Compared with the rule-driven approach, our approach can improve performance by 4% on average when using the less accurate QoS prediction model; and it can improve performance by 7% on average when using the more accurate one. It is because that the rule-driven approach follows the rules described in Table A2, and it does not depend on the QoS prediction model. However, the QoS prediction model is used in our approach, and its accuracy can affect the efficiency of resource allocation. Therefore, when compared with the rule-driven approach, the impact of our approach is more pronounced with the more accurate QoS prediction model. In addition, the rules are specifically developed for this system, considering features of the SLA contract, each types of virtual machines and the fitness function, leading to high administrative overhead and implementation difficulty.



(A) Using the less accurate one.

(B) Using the more accurate one.

Figure A4 Performance comparison between three approaches when using different QoS prediction models.

Appendix A.5 Overhead of Proposed Approach

The cloud services are still working during the process of adjusting the resource allocation plan. Therefore, we regard the overhead as the adjusting time that is spent on reaching the optimized resource status. The shorter the adjusting time is, the more resource could be saved and the faster the service could reach the desired QoS. In general, the adjusting time can be divided into two parts: decision making to find out a resource-allocation plan, and VM configuration (such as VM initialization and deletion) to prepare the cloud resources. Since the time of VM configuration can be optimized through various techniques such as maintaining VM pools, the overhead mainly comes from the decision making.

We calculate average time consumption of each approaches in decision making during resource allocation, as shown in Table A4. Our approach is feedback driven so that the total decision-making process consists of several iterations. So we compare our approach with the rule-driven approach that also needs several iterations to make the decision. We can see that the average number of feedback iterations of our approach is 3.35, while the number of the rule-driven one is 3.40. Therefore, our approach can find the optimized allocation plan in acceptable number of iterations. Considering the time spent on each iteration, our approach is longer than the rule-driven one as the rules are explicitly given and smaller amount of computation is needed to get the allocation plan. So we then compare our approach with the traditional approach that is based on QoS prediction model. We can see that the average time spent of each decision of our approach is 7.76 seconds, and very close to the traditional approach (7.18 seconds), but with an overhead of 0.58 seconds. The slight increase of

execution time is due to that in our approach it not only searches for the new objective resource-allocation plan, but also tunes the QoS prediction model. In fact, if the number of evolutions reaches the preset maximum (1000), the time spent of our approach is 9.52 seconds on average. However, with the other stop criterion that the optimal value is not updated for 20 consecutive evolutions, the number of evolutions can be reduced to about 840 on average, and the average time spent of our approach can be reduced to 7.76 seconds too. In addition, for each decision iteration, all the three approaches can compute the allocation plan in several seconds, which is acceptable from the management perspective. On the whole, the overhead of proposed approach is very small.

Table A4 Average time consumption of each approaches in decision making during resource allocation.

Self-Adaptive Approaches	**Average Time Spent of Each Decision(Seconds)**	**Average Number of Feedback Iterations**	**Total Time (Seconds)**
Our Approach	7.76	3.35	26.00
Traditional Approach	7.18	1.00	7.18
Rule-driven Approach	1.08	3.40	3.67

References

1 RUBiS: Rice University Bidding System. http:// http://rubis.ow2.org/.