Hindawi
Scientific Programming
Volume 2017, Article ID 4358536, 9 pages
https://doi.org/10.1155/2017/4358536

Research Article

An Invocation Cost Optimization Method for Web Services in Cloud Environment

Lianyong Qi,[1] Jiguo Yu,[1] and Zhili Zhou[2]

[1]*School of Information Science and Engineering, Qufu Normal University, Rizhao 276826, China*
[2]*School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China*

Correspondence should be addressed to Lianyong Qi; lianyongqi@gmail.com

Received 2 January 2017; Revised 27 February 2017; Accepted 19 April 2017; Published 9 May 2017

Academic Editor: Basilio B. Fraguela

The advent of cloud computing technology has enabled users to invoke various web services in a "pay-as-you-go" manner. However, due to the flexible pricing model of web services in cloud environment, a cloud user' service invocation cost may be influenced by many factors (e.g., service invocation time), which brings a great challenge for cloud users' cost-effective web service invocation. In view of this challenge, in this paper, we first investigate the multiple factors that influence the invocation cost of a cloud service, for example, user's job size, service invocation time, and service quality level; and afterwards, a novel Cloud Service Cost Optimization Method named *CS-COM* is put forward, by considering the above multiple impact factors. Finally, a set of experiments are designed, deployed, and tested to validate the feasibility of our proposal in terms of cost optimization. The experiment results show that our proposed *CS-COM* method outperforms other related methods.

1. Introduction

The advent of cloud computing technology has provided us with a light-weight resolution for building various complex business applications [1–3]. With the flexible provision of cloud computing infrastructure, a service user can invoke his/her interested web services in an "easy-to-access" and "pay-as-you-go" manner, which significantly benefits the users who request dynamic and variable computing resources [4–6].

However, the flexible pricing model in cloud environment brings users a challenging task to find the optimal service invocation cost [7], as the cost is often varied with many impact factors, such as user job size, service invocation time, and user's requested service quality [8]. For example, let us consider a video-on-demand scenario [9, 10] where a service $service_{VOD}$ can help users to enjoy movies located on remote servers. Then when $service_{VOD}$ is invoked by a user, the invocation cost often depends on many context factors, such as movie size, service invocation time, and movie display quality. Therefore, from the perspective of a cloud user, it becomes a necessity and a challenge to find the optimal service invocation time as well as the minimal service invocation cost, when he/she utilizes $service_{VOD}$ to enjoy his/her preferred movies.

In view of this challenge, in this paper, we first investigate and analyze the impact factors that influence the invocation cost of a cloud service; afterwards, a novel Cloud Service Cost Optimization Method, that is, *CS-COM*, is brought forth in this paper, by combining the above multiple impact factors.

The rest of paper is structured as follows. In Section 2, we first investigate the impact factors that influence the invocation cost of a cloud service. Afterwards, an optimization method for invocation cost of cloud services, that is, *CS-COM*, is put forward in Section 3, by considering the investigated multiple impact factors. A set of experiments are designed and deployed in Section 4, to validate the feasibility of our proposal in terms of cost optimization. Related works and further discussions are presented in Section 5. And finally, in Section 6, we conclude the paper and point out our future research directions.

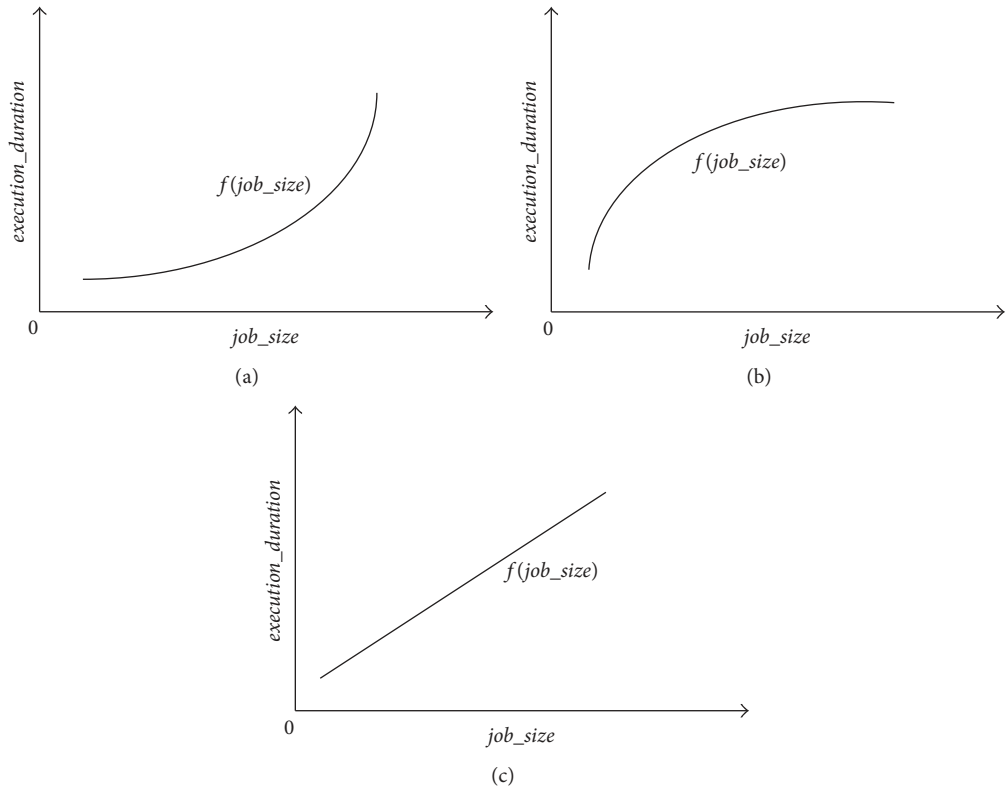

Figure 1: Positive correlation between *job size* of a cloud user and *execution duration* of a cloud service.

2. Impact Factor Analyses for Web Service Invocation Cost in Cloud

Due to the flexible pricing models in cloud environment, the invocation cost for a cloud service is often not static but varied with many impact factors. In this section, we will investigate these context factors. Concretely, the following three context factors play important roles in cloud service charging.

2.1. Job Size. Size of a cloud user' job (or task), whose units are KB, MB, GB, TB, PB, and so forth. Generally, for a job of a cloud user, a larger *job size* often leads to longer service execution duration. Let us consider the example of $service_{VOD}$ introduced in Section 1. A 2 GB movie often takes more time cost (e.g., decoding time or transmission time) than a 1 GB movie does.

With the above observation, a conclusion could often be drawn that there is a positive correlation between a user's *job size* and a cloud service's *execution duration*. Here, we utilize formula (1) to depict the relationship between them. Then according to the above analyses, in (1), the first-order derivative $f'(job\_size) > 0$ often holds. Concretely, for simplicity, we utilize the three submodels in Figure 1 to depict the positive correlation between a cloud user's *job_size* and a cloud service's estimated *execution_duration*. As Figure 1 shows, a cloud service's estimated *execution_duration*s all increase with the growth of *job_size*. The major difference among these three submodels is as follows: *execution_duration* increases faster with the growth of *job_size* in Figure 1(a) (e.g., when the cloud load is becoming heavier and heavier), while in contrast, *execution_duration* increases more slowly when *job_size* grows in Figure 1(b) (e.g., when the cloud load is becoming smaller and smaller); and in Figure 1(c), *execution_duration* increases linearly with the growth of *job_size* (e.g., when the cloud load stays approximately stable).

$$execution\_duration = f\left(job\_size\right). \tag{1}$$

For a cloud service, larger *execution duration* often means higher service invocation cost. Therefore, there is an indirect positive correlation between *job size* of a cloud user and invocation cost of a cloud service. For example, a 2 GB movie may be charged more than a 1 GB movie when a user invokes $service_{VOD}$.

2.2. Service Invocation Time. It means the time point that a cloud service begins to execute. Generally, a cloud user would be charged more when he/she invokes a cloud service in busy hours (e.g., 08:00 am~18:00 pm of a day) or on busy days. In contrast, when the cloud service is not busy, a cloud user would be charged less. Let us take the $service_{VOD}$ introduced in Section 1, for example. If a user invokes $service_{VOD}$ service on free days (e.g., Monday~Thursday), a small fee would be charged (i.e., cost per hour *cph* is low in Figure 2), while on other busy days (e.g., Friday~Sunday), the invocation cost of $service_{VOD}$ would rise significantly due to the heavy network load on weekends.

Here, we utilize the following pricing model in (2) to depict the relationship between a cloud service's *cph* and

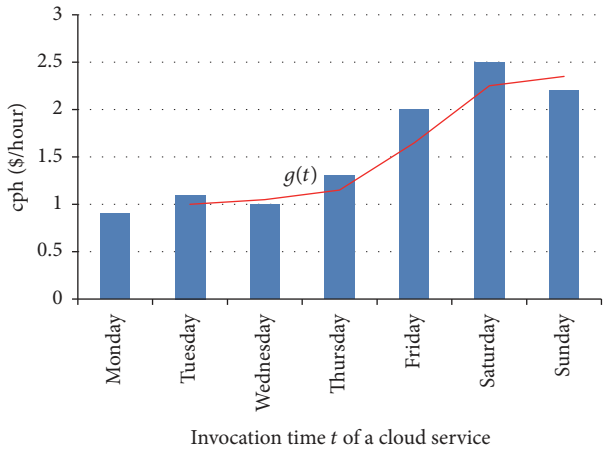

FIGURE 2: Cloud services' pricing model regarding invocation time t: an example.

invocation time point t. Generally, the time-aware pricing model of a cloud service is often provided by its service provider.

$$cph = g(t). \tag{2}$$

2.3. Service Quality Level. It means a cloud service's quality level that is requested by a cloud user. Generally, a cloud provider often publishes its cloud service with multiple quality levels so as to accommodate the various preferences of different cloud users [11]. Here, we utilize set $\{ql_1, \ldots, ql_n\}$ to denote a cloud service's n service quality levels that could be delivered to its cloud users (ql_1 denotes the lowest quality level, while ql_n denotes the highest quality level). Generally, the service invocation cost would be high if a cloud user requests a high service quality level. For example, let us consider the example of $service_{VOD}$ (introduced in Section 1) as well as its two service quality levels shown as follows. As 1080 P > 720 P holds, a user would be charged more if he/she selects *service-quality-level*-1 instead of *service-quality-level*-2, as *service-quality-level*-1 requires more transmission cost than *service-quality-level*-2 does:

service-quality-level-1: 1080 P (high video_quality)

service-quality-level-2: 720 P (middle video_quality)

3. Cost Optimization Method for Cloud Service Invocation

In Section 2, we have analyzed the three important impact factors that are related to the invocation cost of a cloud service, that is, *job size*, *service invocation time*, and *service quality level*. Next, we combine the above three impact factors together to develop a novel cost optimization method for cloud services, that is, *CS-COM*, so as to help cloud users to find the optimal service invocation time and the minimal service invocation cost. Concretely, our proposed *CS-COM* method consists of the following four steps.

TABLE 1: An example of parameters a and b in (4).

Parameter	ql_x			
	ql_1	ql_2	⋯	ql_n
a	1	2		n
b	$1/n$	$2/n$		1

Step 1. Estimate a cloud service's *execution duration* based on a cloud user's *job size*.

In Section 2 (see Figure 1 and formula (1)), we have introduced three pricing submodels between a cloud user's *job_size* and a cloud service's estimated *execution_duration*. Therefore, given a cloud user's job size, we can estimate a cloud service's execution duration based on (1). Here, please note that formula (1) can be in the form of Figure 1(a) or Figure 1(b) or Figure 1(c), depending on the service provider's pricing strategy. Next, we utilize range $[t_0, t_0 + execution\_duration]$ to denote the running period of a cloud service, where t_0 is the time point that the cloud service starts to execute.

Step 2. According to the estimated service *execution_duration* (in Step 1) and the time-aware pricing model (in (2)), calculate the original service invocation cost P (without considering *service quality level*).

As introduced in Step 1, a cloud service starts to execute at $t = t_0$ and ends at $t = t_0 + execution\_duration$. Therefore, through the integral operation over cph in (2), we can obtain the original service invocation cost P (without considering *service quality level*). Concretely, P could be calculated by (3), where $cph = g(t)$ holds (see formula (2)).

$$\begin{aligned} P &= \int_{t_0}^{t_0 + execution\_duration} cph \, dt \\ &= \int_{t_0}^{t_0 + execution\_duration} g(t) \, dt. \end{aligned} \tag{3}$$

Step 3. Calculate the comprehensive service invocation cost $P^{(x)}$ based on the requested *service quality level* ql_x and the original invocation cost P derived in Step 2.

In Step 2, we have derived the original invocation cost P of a cloud service without considering the service quality level. As analyzed previously in Section 2, service quality level often plays an important role in service charging. Therefore, in this step, original service invocation cost P is modified to be $P^{(x)}$ by considering the cloud user's requested service quality level ql_x where $ql_x \in \{ql_1, \ldots, ql_n\}$ holds (here, we assume that there are n quality levels for a cloud service; ql_1 and ql_n denote the lowest and highest service quality levels, resp.).

As analyzed in Section 2, a higher service quality level often leads to larger service invocation cost. In view of this intuitive observation, we utilize the simple linear formula in (4) to depict the correlation between $P^{(x)}$ and P. In (4), a and b are two parameters that are determined by the service quality level ql_x. A concrete example is presented in Table 1

Input: *user*: a cloud user
cs: a cloud service ready to be invoked by *user*
job_size: user's job (or task) size
ql_x: service quality level of *cs* requested by *user*
$f(job\_size)$: $execution\_duration = f(job\_size)$ in (1)
$g(t)$: $cph = g(t)$ in (2)
Output: $t_{0(optimal)}$: optimal service invocation start time of *cs*
$P^{(x)}_{(optimal)}$: optimal invocation cost of *cs* by *user*

(1) Set variable t_0 // service invocation start time
(2) Get models $f(job\_size)$ and $g(t)$ from service provider
(3) Get parameters a and b in (4) from service provider based on ql_x
(4) Estimate *cs*'s *execution_duration* based on *job_size* and (1)
(5) Calculate P based on t_0, *execution_duration* and (3)
// P: original service invocation cost of *cs* by *user*
(6) Calculate $P^{(x)}$ based on a, b, P and (4)
// $P^{(x)}$: comprehensive invocation cost of *cs* by *user*
(7) Set objective function: Minimize $P^{(x)}$
(8) Determine $t_{0(optimal)}$ by combining (1)–(5)
(9) Determine $P^{(x)}_{(optimal)}$ based on (6)
(10) Return $t_{0(optimal)}$ to *user*
(11) Return $P^{(x)}_{(optimal)}$ to *user*

Algorithm 1: *CS-COM* $(user, cs, job\_size, ql_x, f(job\_size), g(t))$.

to demonstrate the relationship between parameter values of (a, b) and service quality level ql_x. Generally, parameters a and b could be obtained from the cloud service provider.

$$P^{(x)} = h\left(ql_x, P\right) = a * P + b. \tag{4}$$

Step 4. Optimize cost $P^{(x)}$ derived in Step 3.

Our final goal is to minimize the comprehensive service invocation cost $P^{(x)}$ derived in (4). Next, through combining (1)–(4) and objective function (i.e., Minimize $P^{(x)}$), we can obtain an optimal value for service invocation start time t_0, denoted by $t_{0(optimal)}$ in (5). And correspondingly, when $t_0 = t_{0(optimal)}$ holds, the optimal service invocation cost $P^{(x)}_{(optimal)}$ is achieved, which could be calculated by (6).

$$t_{0(optimal)} = \left\{t_0 \mid P^{(x)} = \min\left\{P\left(x\right)\right\}\right\} \tag{5}$$

$$P^{(x)}_{(optimal)} = a * \int_{t_{0(optimal)}}^{t_{0(optimal)}+execution\_duration} g\left(t\right) \mathrm{d}t + b. \tag{6}$$

With Steps 1–4 of our proposed *CS-COM* method, we can determine the optimal service execution start time t_0 as well as the optimal (i.e., the lowest) service invocation cost $P^{(x)}_{(optimal)}$, by considering a cloud user's job size and requested service quality level ql_x. Next, more formally, the pseudocode of our proposed *CS-COM* method is specified as Algorithm 1. Here, the functions $execution\_duration = f(job\_size)$ in (1), $cph = g(t)$ in (2), and parameters a and b are all regarded as known already, as they all depend on the pricing models of cloud service providers.

4. Experiments

In this section, a set of simulated experiments are designed and tested, to validate the feasibility of our proposed *CS-COM* method in terms of cost optimization.

4.1. Experiment Settings. Next, we introduce the concrete parameters or environment settings adopted in the experiments.

4.1.1. Relationship between execution_duration and job_size (See Formula (1)). As work [12] indicates, a cloud service's *execution_duration* (without considering the data transfer between user client and cloud server) mainly depends on the CPU processing speed and user *job_size*. Due to the flexible resource provision in cloud environment, we can assume that the CPU processing speed stays approximately stable. In this situation, there is an approximately linear relationship between estimated *execution_duration* (unit: hour) of a cloud service and *job_size* (unit: GB) of a cloud user. So in the experiments, we utilize the linear function in (7) to model their relationship where k ($k > 0$) is a parameter. Here, we utilize the well-known cloud simulation tool *CloudSim* [13] developed by Melbourne University for generating the user job 1000 times randomly, through which the user *job_size* could be obtained.

$$execution\_duration = f\left(job\_size\right) = k * job\_size. \tag{7}$$

4.1.2. Relationship between cph and t (See Formula (2)). Generally, a cloud service's pricing model, that is, *cph* (cost per

hour), heavily depends on the service invocation time t [14]. In the experiments, we generate the random pricing models with the help of *CloudSim* (concretely, cloud services including pricing models in (1)–(4) are encapsulated in a service entity in *CloudSim* and registered in CloudInformationService component; our proposed cost optimization method *CS-COM* and other related methods are located in the component of Data Center Proxy so as to estimate the service invocation cost).

4.1.3. Relationship between Invocation Cost $P^{(x)}$ and Service Quality ql_x (See Formula (4) and Table 1). According to the experiment results observed by [14], there is an approximate linear relationship between service invocation cost $P^{(x)}$ and service quality ql_x (see formula (8)). In the experiments, we adopt this experienced data in (8) to approach formula (4) in our paper.

$$P^{(x)} = 2 * ql_x + 0.4. \quad (8)$$

Besides, we test and compare our *CS-COM* method with four related methods: *FCFS* (First Come First Serve) [15], *FL-FL* (cost evaluation based on historical records) [16], *Cost-plus* (considering service invocation cost and user benefit simultaneously) [17], and *CB* (considering time-depended pricing model only) [14].

The experiments were conducted on a HP laptop with 2.40 GHz processors and 4.0 GB RAM. The machine is running under Windows 7 and JAVA 1.5. Each experiment was carried out 10 times and the average results were adopted. Concretely, three experiment profiles are tested and compared.

4.2. Experiment Results

4.2.1. Profile 1: Invocation Cost Comparison with respect to Job Size. In this profile, we test and compare the service invocation costs of our proposed *CS-COM* method and other four methods. Here, in *CS-COM*, parameters $k = 2$, $a = 2$, and $b = 0.4$ hold (see (8)); parameter L in *FL-FL* is equal to 3; parameter *conversion* = 0.4 holds in *Cost-plus*. Cloud user's *job_size* is varied from 1 GB to 10 GB.

The experiment results are presented in Figure 3. As Figure 3 shows, the service invocation costs of five methods all increase with the growth of *job_size* approximately; this is because processing a larger job often takes more time cost and hence leads to a higher invocation cost. Moreover, the service invocation cost of *Cost-plus* method is high as it considers the user benefit as an optimization object, while more user benefit often means higher charging fees. Besides, the service invocation cost of *FCFS* method often fluctuates frequently as its service invocation time is randomly selected, while different service invocation time means varied service charging. The rest three experiment curves increase approximately in polynomial manners, where *FL-FL* method utilizes the past service invocation costs to estimate the future invocation cost and *CB* method considers the time-dependent pricing models of cloud services, while our proposed *CS-COM* method consider a cloud user's job size, service invocation time, and requested service quality level simultaneously. Therefore, *CS-COM* method outperforms the other four methods in terms of service invocation cost, which could also be observed from Figure 3.

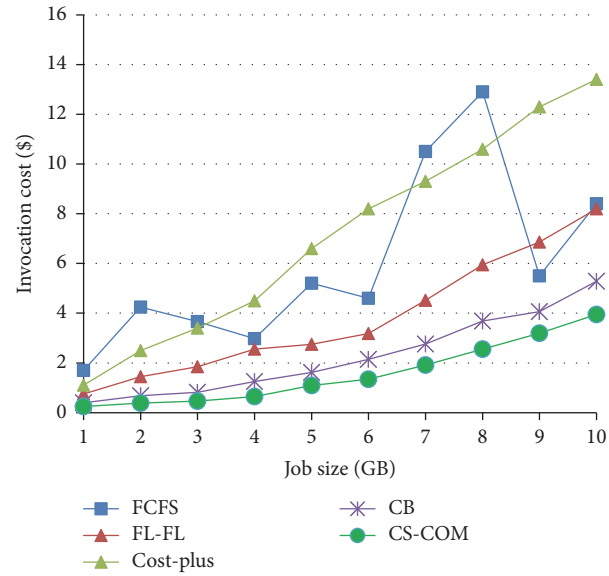

Figure 3: Invocation cost comparison of five methods with respect to job size.

4.2.2. Profile 2: Invocation Cost Comparison with respect to t. In this profile, we compare the service invocation costs of five methods with respect to the service invocation time. As the "*cph-t*" charging models (see formula (2)) randomly generated by *CloudSim* make it hard to observe the stable variation trend of invocation cost, in this profile, we choose a randomly generated but fixed "*cph-t*" charging model ($t \in$ [1:00 pm, 12:00 pm]) where [6:00 pm, 9:00 pm] is the busy hour. To observe the cost variation trend with invocation time t, we tune parameter *job_size* so that the user job could be finished within one hour. Other parameter settings are the same as in Profile 1.

The experiment results are shown in Figure 4. As Figure 4 indicates, the service invocation cost of *Cost-plus* is high as it considers both user benefit and service cost, while larger user benefit often means higher charging fees. The *FCFS* method achieves the approximate cost variation trend with the preset "*cph-t*" charging model as no cost optimization strategy is adopted in *FCFS*. In *FL-FL* method, sampling technique is recruited to approximately approach original "*cph-t*" charging model, which achieves the similar cost variation trend as in *FCFS*. The rest two methods, that is, *CB* and *CS-COM*, perform better than the previous three methods in terms of cost optimization, as the dynamic time-aware cost optimization strategy is considered in these two methods. Besides, as supposed in this profile, the execution duration of user job is one hour; therefore, at the last o'clock (i.e., 12:00 pm), time-aware cost optimization strategy does not work anymore and hence, the five cost variation curves converge.

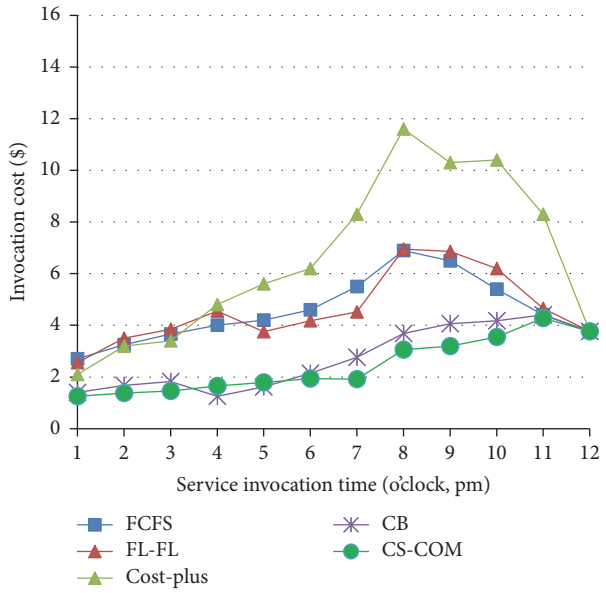

Figure 4: Invocation cost comparison of five methods with respect to service invocation time (1:00 pm~12:00 pm).

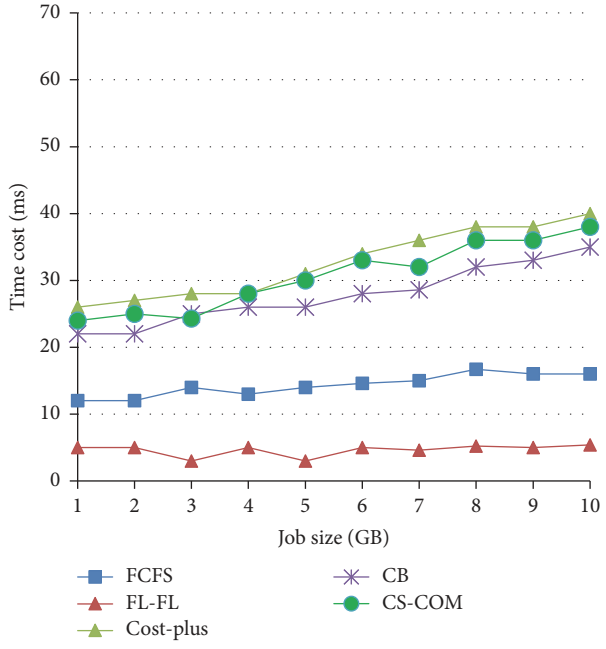

Figure 5: Time cost comparison of five methods.

4.2.3. Profile 3: Time Cost Comparison. In this profile, we test the time costs of five methods. Here, sampling technique is recruited to convert the continuous function $cph = g(t)$ in (2) into d discrete values with same intervals, so as to facilitate the further computation of service invocation cost. Concretely, $d = 100$ holds in this profile. Other parameter settings are the same as those in Profile 1.

The concrete experiment results are presented in Figure 5. As can be seen from Figure 5, *FL-FL* method achieves the least time cost as it only considers the past few historical service invocation costs of a cloud service, without further complicated computation. The rest four time cost curves all increase approximately linearly with the growth of job size. The time cost of *FCFS* method is not very large as it only refers to an integral operation associated with $cph = g(t)$ and t. The execution efficiencies of rest three methods, that is, *Cost-plus*, *CB*, and *CS-COM*, are very close, as they all contain some extra computation processes associated with cloud users' job size; concretely, *Cost-plus* needs to calculate the benefit of a cloud user based on the service invocation cost (derived based on job size) and *CB* needs to optimize the service invocation cost based on the time-dependent pricing model (depends on job size), while our proposed *CS-COM* method employs job size to estimate the service execution duration. As Figure 5 shows, the time costs of all the five methods are not high (at "millisecond" level).

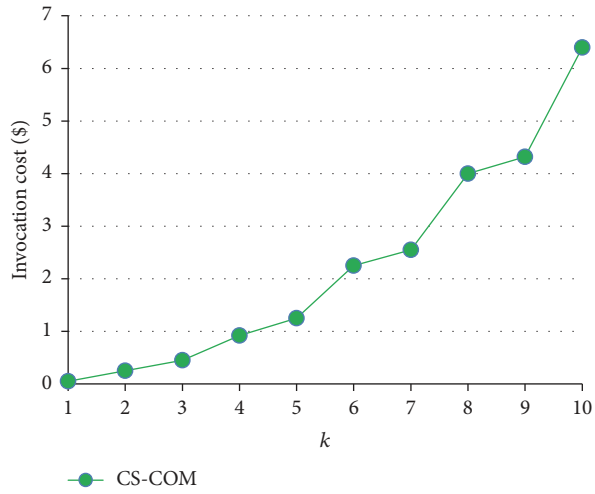

Figure 6: Service invocation cost of *CS-COM* with respect to k.

4.2.4. Profile 4: Service Invocation Cost of CS-COM with respect to k. As formula (7) indicates, k is an important parameter that bridges the estimated service *execution_duration* and a cloud user's *job_size* and consequently influences the finally derived optimal service invocation cost. In this profile, we test the relationship between k and the optimal service invocation cost in our proposed *CS-COM* method. Concretely, k is varied from 1 to 10, *job_size* is equal to 1 GB, and parameters $a = 2$ and $b = 0.4$ hold in (4).

The concrete experiment results are presented in Figure 6. As can be seen from Figure 6, the invocation cost of *CS-COM* method increases approximately in a polynomial manner with the growth of k, this is because k appears in the upper bound of t (i.e., $t_{0(optimal)} + execution\_duration = t_{0(optimal)} + f(job\_size)$) of invocation cost integration $\int_{t_{0(optimal)}}^{t_{0(optimal)}+execution\_duration} g(t)\,dt$ in (6), while $g(t)$ is often a polynomial function associated with t. So after the integral operation, the service invocation cost becomes a polynomial function associated with parameter k.

From the above three sets of experiment results, we can conclude that our proposed *CS-COM* method outperforms

the rest four methods in terms of service invocation cost (concretely, compared to *FCFS*, *FL-FL*, *Cost-plus*, and *CB*, the cost reduction ratios of our proposal are 74.9%, 62.8%, 80.1%, and 35.1%, resp.). Besides, the time cost of our proposal is at the "ms" level, which is acceptable for most business applications. Finally, in our proposed *CS-COM* method, the derived optimal service invocation cost has a positive correlation with parameter k in formula (7), which approximately coincides with most existing cloud pricing models. Actually, for a cloud service, the value of parameter k could be published by its service provider in a flexible manner so as to maximize the economic gains.

5. Related Work and Further Discussions

5.1. Related Work and Comparison Analyses. Cloud computing technology, on one hand, facilitates cloud users' sharing and use of various computing resources by providing an "easy-to-access" and "pay-per-use" resource provision manner and, on the other hand, brings a great challenge to minimize or optimize cloud users' service invocation cost. Many researchers have investigated this hot research topic and brought forth their respective resolutions [12, 14–17].

In [12], the authors divided the invocation cost of a cloud service into three categories: data-storage cost, CPU processing cost, and data-transfer cost. In order to evaluate and predict a cloud user's service invocation cost, *FL-FL* method was put forward in [16] by considering the service's past invocation costs; however, *FL-FL* method fails to generate an accurate service invocation cost as the latter is often influenced by some other factors. Work [17] analyzed the relationship between service invocation cost and user benefits and finally introduced a cost-benefit-aware cloud service scheduling method *Cost-plus*. However, one final optimization goal was to maximize user profits, not to minimize the service invocation cost. In order to minimize the service invocation cost, *FCFS* method was put forward in [15]. *FCFS* adopted the "First Come First Serve" rule so as to reduce the waiting time of user job and optimize the service invocation cost. However, *FCFS* did not consider the dynamic and varied time-aware pricing model in cloud environment. In view of this, work [14] took the time-dependent pricing model of cloud services into consideration and brought forth an invocation cost optimization method *CB*. However, *CB* method only considered service invocation time when optimizing the invocation cost, while neglecting some other important factors, for example, user job size and user's requested service quality level.

In view of the above shortcomings, a novel service invocation cost optimization method named *CS-COM* is put forward in this paper, which considers the multiple factors that influence the invocation cost in cloud environment. Experiment results show that *CS-COM* outperforms other related methods in terms of cost optimization.

5.2. Further Discussions. In this paper, we put forward a cost optimization method for web services based on multiple impact factors. Generally, the proposed multifactors-based optimization strategy can also be applied in other application domains with multiple factors, for example, performance optimization [18–23], feature analysis [24–29], quality evaluation [30–32], knowledge learning [33–37], and data mining [38–40]. However, several shortcomings are still present in our approach.

(1) Only three factors (i.e., *job size*, *service invocation time* and *service quality level*) are considered in our cost optimization method named *CS-COM*, which are not enough for real cloud service scheduling applications. Therefore, in the future, we will further improve our proposal by introducing more charging factors.

(2) Users' subjective preferences that play an important role in users' final service invocation decisions are not considered in our proposed *CS-COM* approach. In the future, we will refine our work by taking user preferences into consideration.

6. Conclusions

Cloud computing has provided an "easy-to-access" and "pay-per-use" resource delivery manner, to help users build their various complex business applications quickly and conveniently. However, due to the flexible pricing model of cloud services, a cloud service's invocation cost is often not fixed but varied, which brings a great challenge to optimize the service invocation cost when a cloud user requests a cloud service. In view of this challenge, we first analyze the multiple factors that may influence the invocation cost of a cloud service, for example, user job size, service invocation time, and service quality level. Afterwards, through considering the above multiple factors, a novel service invocation cost optimization method named *CS-COM* is put forward in this paper, to aid a cloud user to find the optimal service invocation start time as well as the lowest service invocation cost. Finally, through a set of simulated experiments deployed on *CloudSim* platform, we further demonstrate the feasibility and advantages of our proposed *CS-COM* method in terms of cost optimization.

In the future, we will further refine our proposed *CS-COM* method by introducing more charging factors, so as to make it more comprehensive and more applicable in real cloud service scheduling applications.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper is partially supported by Natural Science Foundation of China (no. 61402258, no. 61602253, no. 61373027, and no. 61672321) and Open Project of State Key Laboratory for Novel Software Technology (no. KFKT2016B22).

References

[1] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: leveraging mobile devices to provide cloud service at the edge," in *Proceedings of the 8th IEEE International Conference on Cloud Computing (CLOUD '15)*, pp. 9–16, July 2015.

[2] Z. Xia, X. Wang, X. Sun, Q. Liu, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2015.

[3] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Transactions on Communications*, vol. E98B, no. 1, pp. 190–200, 2015.

[4] Y. Kong, M. Zhang, and D. Ye, "A belief propagation-based method for task allocation in open and dynamic cloud environments," *Knowledge-Based Systems*, vol. 115, pp. 123–132, 2016.

[5] Z. Xia, X. Wang, L. Zhang, Z. Qin, X. Sun, and K. Ren, "A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2594–2608, 2016.

[6] L. Qi, W. Dou, and J. Chen, "Weighted principal component analysis-based service selection method for multimedia services in cloud," *Computing*, vol. 98, no. 1-2, pp. 195–214, 2016.

[7] D. M. Divakaran and M. Gurusamy, "Towards flexible guarantees in clouds: adaptive bandwidth allocation and pricing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1754–1764, 2015.

[8] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2546–2559, 2016.

[9] Z. Pan, Y. Zhang, and S. Kwong, "Efficient motion and disparity estimation optimization for low complexity multiview video coding," *IEEE Transactions on Broadcasting*, vol. 61, no. 2, pp. 166–176, 2015.

[10] Z. Pan, J. Lei, Y. Zhang, X. Sun, and S. Kwong, "Fast motion estimation based on content property for low-complexity H.265/HEVC encoder," *IEEE Transactions on Broadcasting*, vol. 62, no. 3, pp. 675–684, 2016.

[11] A. K. Talukder and L. Zimmerman, "Cloud economics: principles, costs, and benefits," in *Cloud Computing*, pp. 343–360, Springer, London, UK, 2010.

[12] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: the montage example," in *Proceedings of the ACM/IEEE Conference on Supercomputing*, pp. 1–12, IEEE Press, November 2008.

[13] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. de Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[14] C. Chawla and I. Chana, "Optimal time dependent pricing model for smart cloud with cost based scheduling," in *Proceedings of the 3rd International Symposium on Women in Computing and Informatics (WCI '15)*, pp. 522–526, August 2015.

[15] M. Li, D. Subhraveti, A. R. Butt, A. Khasymski, and P. Sarkar, "CAM: a topology aware minimum cost flow based resource manager for MapReduce applications in the cloud," in *Proceedings of the 21st ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC '12)*, pp. 211–222, June 2012.

[16] Q. Liu, W. Cai, J. Shen, Z. Fu, X. Liu, and N. Linge, "A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment," *Security and Communication Networks*, vol. 9, no. 17, pp. 4002–4012, 2016.

[17] W.-H. Choi and K.-S. Kang, "A Study on deciding optimal price of bioinformatics services," *Journal of the Korea Safety Management and Science*, vol. 18, no. 1, pp. 203–208, 2016.

[18] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2706–2716, 2016.

[19] Z. Pan, P. Jin, J. Lei, Y. Zhang, X. Sun, and S. Kwong, "Fast reference frame selection based on content similarity for low complexity HEVC encoder," *Journal of Visual Communication and Image Representation*, vol. 40, part B, pp. 516–524, 2016.

[20] Y. Xue, J. Jiang, B. Zhao, and T. Ma, "A self-adaptive articial bee colony algorithm based on global best for global optimization," *Soft Computing*, 2017.

[21] Y. Zhang, X. Sun, and W. Baowei, "Efficient algorithm for k-barrier coverage based on integer linear programming," *China Communications*, vol. 13, no. 7, pp. 16–23, 2016.

[22] X. Chen, S. Chen, and Y. Wu, "Coverless information hiding method based on the Chinese character encoding," *Journal of Internet Technology*, vol. 18, no. 2, pp. 91–98, 2017.

[23] C. Yuan, Z. Xia, and X. Sun, "Coverless image steganography based on SIFT and BOF," *Journal of Internet Technology*, vol. 18, no. 2, pp. 209–216, 2017.

[24] Z. Xia, X. Wang, X. Sun, and B. Wang, "Steganalysis of least significant bit matching using multi-order differences," *Security and Communication Networks*, vol. 7, no. 8, pp. 1283–1291, 2014.

[25] J. Li, X. Li, B. Yang, and X. Sun, "Segmentation-based image copy-move forgery detection scheme," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 507–518, 2015.

[26] Z. Xia, X. Wang, X. Sun, Q. Liu, and N. Xiong, "Steganalysis of LSB matching using differences between nonadjacent pixels," *Multimedia Tools and Applications*, vol. 75, no. 4, pp. 1947–1962, 2016.

[27] J. Wang, T. Li, Y.-Q. Shi, S. Lian, and J. Ye, "Forensics feature analysis in quaternion wavelet domain for distinguishing photographic images and computer graphics," *Multimedia Tools and Applications*, 2016.

[28] Z. Zhou, C.-N. Yang, B. Chen, X. Sun, Q. Liu, and Q. M. Jonathan Wu, "Effective and efficient image copy detection with resistance to arbitrary rotation," *IEICE Transactions on Information and Systems*, vol. E99-D, no. 6, pp. 1531–1540, 2016.

[29] C. Yuan, X. Sun, and R. Lv, "Fingerprint liveness detection based on multi-scale LPQ and PCA," *China Communications*, vol. 13, no. 7, pp. 60–65, 2016.

[30] Y. Chen, C. Hao, W. Wu, and E. Wu, "Robust dense reconstruction by range merging based on confidence estimation," *Science China Information Sciences*, vol. 59, no. 9, Article ID 092103, pp. 1–11, 2016.

[31] Z. Zhou, Y. Wang, Q. M. J. Wu, C.-N. Yang, and X. Sun, "Effective and effcient global context verifcation for image copy detection," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 48–63, 2017.

[32] Z. Fu, F. Huang, X. Sun, A. V. Vasilakos, and C.-N. Yang, "Enabling semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Transactions on Services Computing*, 2016.

[33] B. Gu and V. S. Sheng, "A robust regularization path algorithm for ν-support vector classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 5, pp. 1241–1248, 2016.

[34] B. Gu, X. Sun, and V. S. Sheng, "Structural minimax probability machine," *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[35] B. Gu, V. S. Sheng, K. Y. Tay, W. Romano, and S. Li, "Incremental support vector learning for ordinal regression," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 7, pp. 1403–1416, 2015.

[36] B. Gu, V. S. Sheng, Z. Wang, D. Ho, S. Osman, and S. Li, "Incremental learning for ν-support vector regression," *Neural Networks*, vol. 67, pp. 140–150, 2015.

[37] Q. Tian and S. Chen, "Cross-heterogeneous-database age estimation through correlation representation learning," *Neurocomputing*, vol. 238, pp. 286–295, 2017.

[38] Z. Qu, J. Keeney, S. Robitzsch, F. Zaman, and X. Wang, "Multilevel pattern mining architecture for automatic network monitoring in heterogeneous wireless communication networks," *China Communications*, vol. 13, no. 7, pp. 108–116, 2016.

[39] N. Zhang, J. Wang, and Y. Ma, "Mining Domain Knowledge on Service Goals From Textual Service Descriptions," *IEEE Transactions on Services Computing*, 2017.

[40] J. Wang, Z. Zhu, J. Liu, C. Wang, and Y. Xu, "An Approach of Role Updating in Context-Aware Role Mining," *International Journal of Web Services Research*, vol. 14, no. 2, pp. 24–44, 2017.