Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 7308913, 9 pages
https://doi.org/10.1155/2018/7308913

Research Article

Method of Resource Estimation Based on QoS in Edge Computing

Guangshun Li, Jianrong Song, Junhua Wu, and Jiping Wang

School of Information Science and Engineering, Qufu Normal University, Rizhao 276800, China

Correspondence should be addressed to Guangshun Li; 30752585@qq.com

Received 13 October 2017; Accepted 21 December 2017; Published 22 January 2018

Academic Editor: Shangguang Wang

With the development of Internet of Things, the number of network devices is increasing, and the cloud data center load increases; some delay-sensitive services cannot be responded to timely, which results in a decreased quality of service (QoS). In this paper, we propose a method of resource estimation based on QoS in edge computing to solve this problem. Firstly, the resources are classified and matched according to the weighted Euclidean distance similarity. The penalty factor and Grey incidence matrix are introduced to correct the similarity matching function. Then, we use regression-Markov chain prediction method to analyze the change of the load state of the candidate resources and select the suitable resource. Finally, we analyze the precision and recall of the matching method through simulation experiment, validate the effectiveness of the matching method, and prove that regression-Markov chain prediction method can improve the prediction accuracy.

1. Introduction

With the development of Internet of Things (IoT) [1], more and more devices, especially mobile devices, constantly access the Internet. CISCO predicts that 50 billion devices will connect to the Internet by 2020 [2]. These devices will generate large amounts of data at the end of the network, which leads to the increment of the burden of cloud data center. Moreover, the remote distance between the mobile devices and cloud data center makes the transmission delay increase, which makes some delay-sensitive services can not get response and processing rapidly. IoT services, such as connected vehicle and video streaming, require high bandwidth and low latency content delivery to guarantee QoS. Edge computing [3] plays an important role by using network resources near the local network to provide a low latency service and improve QoS.

Edge computing is located between the end devices and cloud data center. It transfers data and computing power from the cloud to the edge of the network, which processes data locally [4]. Due to the fact that edge servers are close to end users, the latency between mobile devices and data center is reduced. Moreover, the resources are close to the user that can improve QoS to some extent.

In the IoT environment, the number of network devices is increasing constantly. The resources with the same service function are also increasing gradually. The complexity of user requirements makes resource management a challenge. Among the many available resources, selecting the suitable resources to meet the needs of users has become the main goal.

Although the latency is reduced, the resources in edge computing are limited compared to cloud computing. The availability and short-term prediction of resources load [5, 6] have some influence on task scheduling, application performance, and the QoS of users. The method of resource prediction can provide the appropriate resource for users by analyzing the load of the resource itself. Therefore, the estimation of Resource QoS is significant to user satisfaction and task allocation in edge computing.

In this paper, we first propose an edge computing framework. Then, we introduce the penalty factor and Grey incidence matrix to improve the accuracy of similarity matching and select resources which satisfy the needs of users. We use the regression-Markov chain prediction method for candidate resources to analyze the change of the load state of

the resource itself. Finally, we prove the effectiveness of the estimation method through simulation experiment.

The remainder of this paper is organized as follows. The related works are introduced in Section 2. We describe the architecture of the edge computing in Section 3. In Section 4, we describe the method of dynamic resource estimation. The experiment results are presented in Section 5. Section 6 concludes the paper.

2. Related Work

Edge computing is the extension of cloud computing to network edge. There is no standard architecture. The work of resource estimation has not been well addressed. Mao et al. [7] developed a joint radio and computational resource management algorithm for multiuser MEC systems, with the objective of minimizing the average weighted sum power consumption of the mobile devices and the MEC server. Shekhar and Gokhale [8] proposed Dynamic Data Driven Cloud and Edge Systems (D3CES) as a framework for adaptive resource management across cloud and edge resources to provide QoS guarantees for performance-sensitive applications. Wang et al. [9] developed an Edge NOde Resource Management (ENORM) framework to manage edge nodes. They proposed a mechanism for provisioning and autoscaling edge node resources. Their method reduced data transmission and communication frequency between the edge node and cloud.

Aazam and Huh [10] proposed a dynamic resource estimation and pricing model for IoT. Their work was mainly focused on considering different types of customers and devices, and the service-oriented relinquish probabilities. Based on previous studies, Aazam et al. [11] proposed a QoE-based resource estimation model to enhance QoS. They devised MeFoRE method on the basis of service relinquish probability and historical records. And they considered previous QoE and Net Promoter Score records to enhance QoS. Zuo et al. [12] proposed a resource estimation model based on entropy optimization and dynamic weighted method to accurately grasp the dynamic load and availability information of resources. They used the objective function and constraint condition of maximum entropy to select the resources which meet the QoS of user. This method achieved optimal scheduling and guaranteed the QoS of user. Dynamic weighted load evaluation was carried out on the filtered resources to realize load balancing and improve system utilization.

Zhao et al. [13] proposed a multidimensional resource model for dynamic resource matching in the IoT, with the multidimensional description of the IoT resources. They considered the ontology structure and ontology description to calculate the similarity by matching the hardware features and software feature between resources. Zhou et al. [14] proposed a multiple QoS resource selection and estimation algorithm. They used Multiple Attribute Decision Making and Analytic Hierarchy Process (AHP) algorithm. Their method mainly considered the preference information of users. Dong et al. [15] analyzed the QoS of a single resource node. They used a local optimization algorithm to select the appropriate cloud resources that meet the needs of users. Ding et al. [16] proposed QoS-aware resource matching and recommendation method in cloud computing. They described a resource matching algorithm that considers both functional requirements and QoS attributes and designed a resource recommendation method for cloud computing.

3. Edge Computing

Edge computing, located between the mobile end devices and cloud data center, provides computing, storage, and network services. Edge computing has the ability of decentralized computation and storage compared with cloud computing. The resources near the user can support mobile devices for real-time communication. The main goal of edge computing is to preprocess data, reduce the delay, and serve the applications of low delay and real-time response. Currently, the common edge computing architecture is a three-tier network architecture, as shown in Figure 1.

As the underlying node, the end devices not only consume data but also produce data. In particular, mobile devices will require more resource from edge servers rather than cloud data center. The end devices include all IoT devices, such as smart mobile phone, intelligent vehicle, and virtual sensor nodes. These devices are able to sense data, communicate with the upper layer through sensor networks, 3G, WiFi, and so on, and transmit the collected raw data. Compared with the servers in data centers, most of the IoT devices or sensors at the edge are somewhat limited in both computing power and battery capacity [17]. Edge devices include some traditional network devices (routers, switches, etc.) and some specially deployed devices (local servers). Edge devices have the ability of computing, processing, storage, and forwarding data to the cloud servers. Edge servers can be connected directly to nearby mobile devices via one-hop wireless link. Micro data center (MDC) resources include computing resources, network resources, storage resources, and software resources. On the one hand, resources come from the cached local resources; on the other hand, they come from the data obtained by the monitoring equipment. As the top layer, cloud computing layer includes data center (DC) and some cluster servers. The cloud servers can receive data sent by edge devices for processing and storage.

Figure 2 shows the request process of the service in the edge computing. The user first submits the request to the system administrator through the end devices. The system administrator stores the submitted information and transmits the statistical QoS requirements to the edge servers through the edge gateway. Monitoring equipment generates statistics log by tracking resource utilization and availability of sensor, applications, and services. The monitoring data is transmitted to the edge servers. The edge servers analyze the QoS of requested service for end users and process locally. The service is divided into several tasks which are processed locally as far as possible. Each task selects the best matching resource from the resource pool based on the estimation results. Then, the selected resource is allocated and scheduled to meet the QoS requirements.

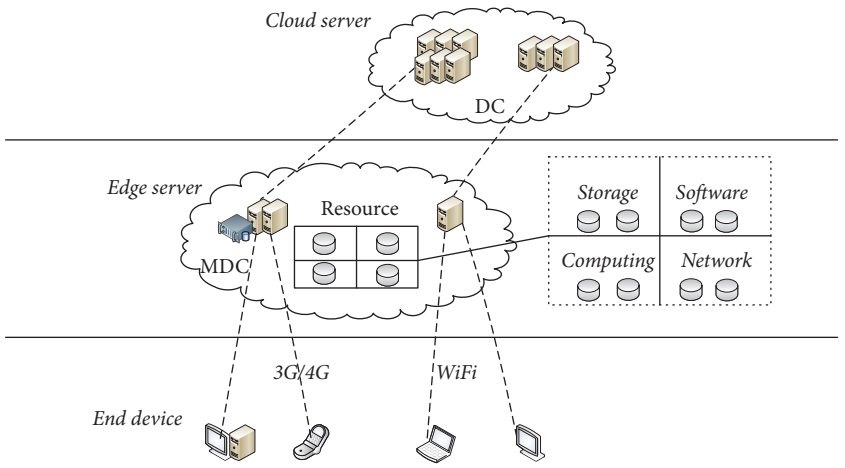

Figure 1: Edge computing architecture.

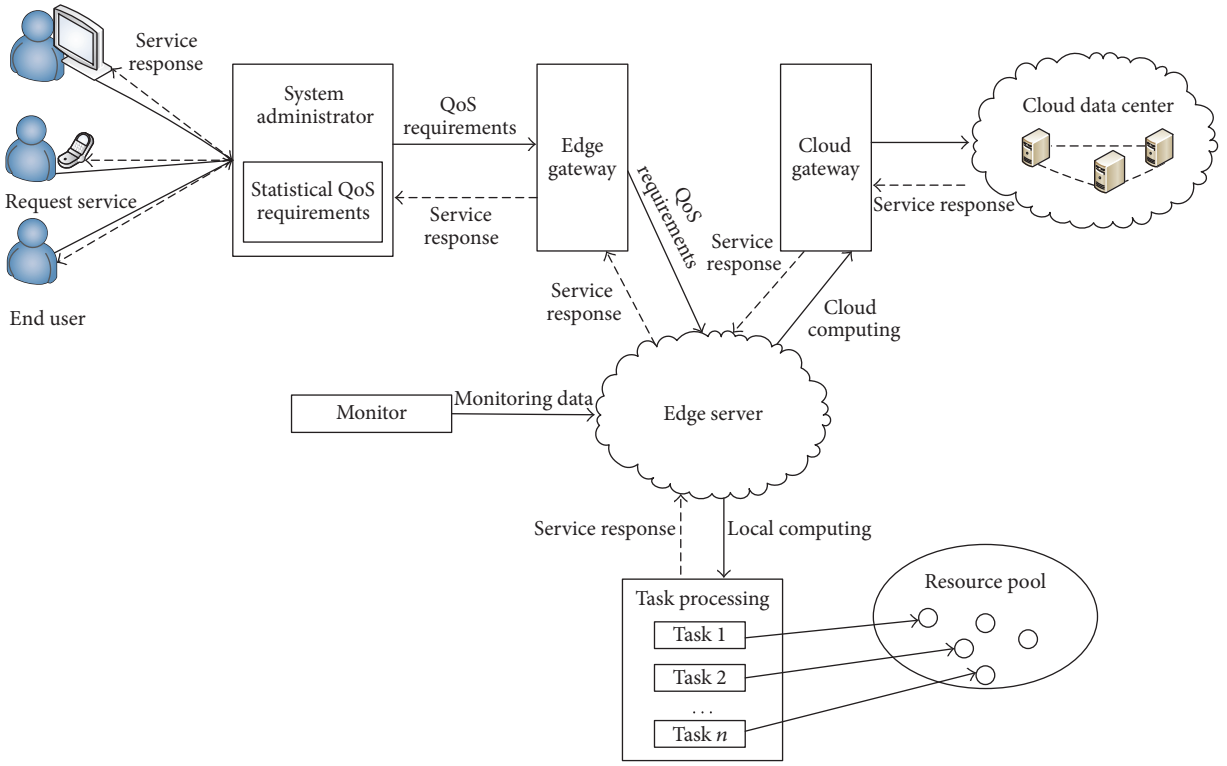

Figure 2: The request process of service in edge computing.

4. Dynamic Estimation Method

In this section, we describe the dynamic estimation method and estimate the resource with the same service function. The specific process is shown in Figure 3.

As shown in the figure, the dynamic estimation method includes two phases: similarity matching algorithm and regression-Markov chain prediction method. First, we establish a Resource QoS matrix to calculate the matching degree between QoS requirements of users and resources through similarity matching. Then, resources which satisfy the needs of the users are selected by threshold. Finally, we analyze the change of resource load and select the optimal resource through regression-Markov chain prediction method.

4.1. Similarity Matching Method

4.1.1. Resource Description. The resources in the edge nodes can be provided to meet the QoS requirements of users. In this paper, resource set which have the same service function is defined as follows:

$$Rs = \{r_1, r_2, \ldots, r_n\}. \quad (1)$$

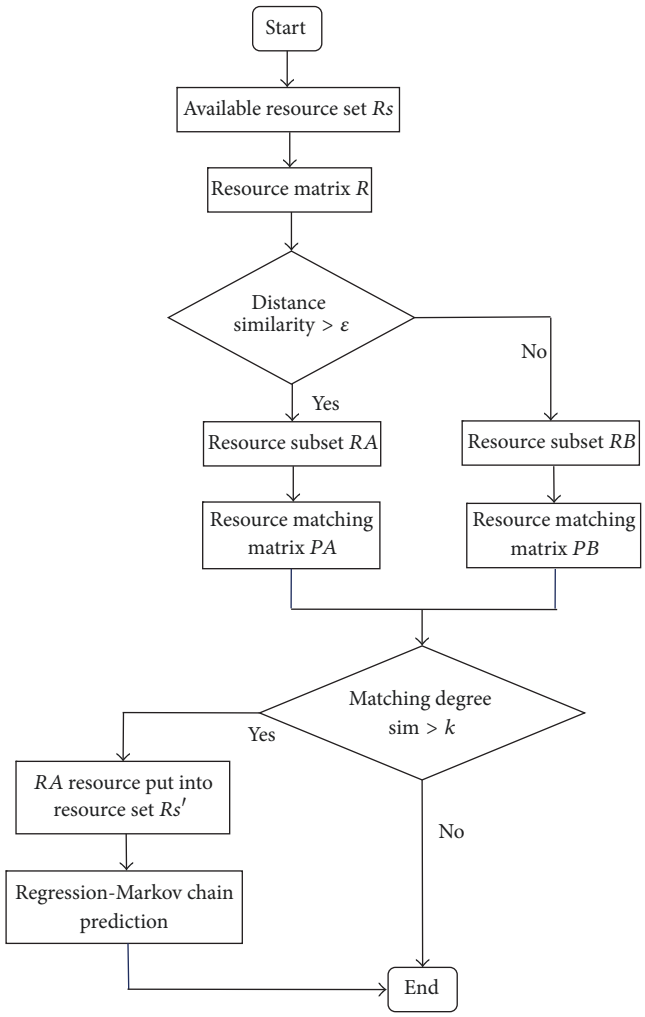

Figure 3: Flow chart of resource estimation method.

Each resource has different QoS attributes. According to these attributes, the resource is evaluated to determine the matching between the needs of users and resources. Resource set Rq_i denotes a collection of the QoS attributes of resource.

$$Rq_i = \{rq_1, rq_2, \ldots, rq_m\}, \tag{2}$$

where index m represents that each resource has m different QoS attributes, including response time, availability, reliability, and price. The price attribute in this paper is defined as follows:

$$p = U * b\frac{\mu}{\varphi} * D_{\text{dev}}, \tag{3}$$

where U is the basic price of service. μ is the number of service requests completed in unit time. φ is the number of service requests received in unit time. b is a price adjustment factor, determined by the service provider. D_{dev} represents the device type, which can generally be divided into static devices, small mobile devices, and large mobile devices. The relative reserved resources of each device are 1, 1.25, and 1.5 times, respectively [10].

The QoS attributes requested by the users are the same as the resources. The QoS attributes set is defined as follows:

$$uq = \{uq_1, uq_2, \ldots, uq_m\}. \tag{4}$$

In this paper, we construct a QoS attribute matrix of resources as the decision matrix.

$$R = \left(r_{ij}\right)_{n\times m} = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nm} \end{pmatrix}, \tag{5}$$

where r_{ij} is the jth QoS attribute value of ith resource.

Since the measurement units of the QoS attributes are different, it is meaningless to process the matrix directly. Therefore, according to the relationship between attributes and user satisfaction, we use the following formula to standardize processing:

$$z_{ij} = \begin{cases} \dfrac{r_{ij} - r_j^{\min}}{r_j^{\max} - r_j^{\min}}, & q \text{ is positive attribute} \\ \dfrac{r_j^{\max} - r_{ij}}{r_j^{\max} - r_j^{\min}}, & q \text{ is negative attribute.} \end{cases} \tag{6}$$

The above formula indicates two cases: if QoS attribute q is positive attribute, the greater the value of the attribute is, the higher the satisfaction of the users is. On the contrary, the negative attribute indicates that the smaller the attribute value is, the higher the user satisfaction is.

In order to ensure the objectivity of the estimation results, we use the entropy weight method to calculate entropy value e_j and entropy weight w_j for the QoS attribute of resources. The formula is as follows:

$$\begin{aligned} e_j &= -\frac{1}{\ln n}\sum_{i=1}^{n} f_{ij} \cdot \ln f_{ij} \\ w_j &= \frac{1 - e_j}{m - \sum_{j=1}^{m} e_j} \\ f_{ij} &= \frac{z_{ij}}{\sum_{i=1}^{n} z_{ij}} \\ \sum_{j=1}^{m} w_j &= 1. \end{aligned} \tag{7}$$

4.1.2. Similarity Matching Algorithm. In order to reduce the matching time and improve the matching efficiency, the resources are firstly classified before similarity matching. Based on the QoS attributes value requested by the users, each resource can be regarded as a point in the multidimensional space. The distance between the resources and the needs of users is measured by Euclidean distance. Since each user may have preferences for one attribute, or each attribute of the resources affects the result of the measurement differently,

therefore, we set objective weight for each attribute and use weighted method to calculate.

$$d(i, uq) = \sqrt{\sum_{j=1}^{m} w_j * \left(rq_j - uq_j\right)^2}$$
$$d_{\text{sim}}(i, uq) = \frac{1}{1 + d(i, uq)}, \tag{8}$$

where w_j represents the weight of resource attribute. $d(i, uq)$ indicates the distance between the ith resource node and the ideal node (the node indicates the QoS requirements of users) in the space. $d_{\text{sim}}(i, uq)$ is the degree of proximity, whose range is from 0 to 1. The resource is close to the ideal node when $d(i, uq)$ is smaller.

We set the proximity threshold ε by computing the proximity degree between the resource node and the ideal node. The range of threshold is from 0 to 1. It divides the resources into two sets.

$$R'(\varepsilon) = \left\{r_k \mid d_{\text{sim}}(k, uq) \geq \varepsilon\right\}. \tag{9}$$

On the basis of the resource classification, we establish the matching matrix between the requested QoS of users and the QoS of the candidate resources.

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ p_{(n+1)1} & p_{(n+1)2} & \cdots & p_{(n+1)m} \end{pmatrix}, \tag{10}$$

where the first n row represents the QoS attributes value of the n resources. The $n + 1$ row represents the QoS attribute value of the expectation of users. We use formula (6) to standardize the matrix and calculate the matching degree between the expectation resource of users and the resources.

In this paper, we use Pearson correlation coefficient to calculate the matching degree of resources.

$$\text{sim}(x, y) = \frac{\Sigma_{r \in R}\left(q_{r,x} - \overline{q}_x\right)\left(q_{r,y} - \overline{q}_y\right)}{\sqrt{\Sigma_{r \in R}\left(q_{r,x} - \overline{q}_x\right)^2}\sqrt{\Sigma_{r \in R}\left(q_{r,y} - \overline{q}_y\right)^2}}, \tag{11}$$

where $\text{sim}(x, y)$ is the similarity between attributes x and y. $q_{r,x}$ and $q_{r,y}$ represent the corresponding value of the QoS attributes x and y of all resources, respectively. $\overline{q}_x$ and $\overline{q}_y$ represent the average value of attributes x and y of all resources, respectively.

In order to ensure that the QoS attribute value of candidate resources is within the expected range of users, we introduce the penalty factor and Grey incidence matrix to analyze the gap between the resource attribute value. The penalty factor λ is set on the condition of resource constraint. The smaller the penalty factor is, the closer the user expectation range is, and the higher the correlation is. On the contrary, when the penalty factor is larger, the distance is more away from the expectation range of users, the lower the correlation is, and the lower the matching degree is.

$$\lambda_j = \begin{cases} 0 & r_j < \delta_j \\ \dfrac{r_j - \delta_j}{\gamma_j - \delta_j} & \delta_j \leq r_j \leq \gamma_j \\ 1 & r_j > \gamma_j, \end{cases} \tag{12}$$

where δ_j and γ_j represent corresponding minimum value and maximum value of the expectation range of users for each attribute of the resources, respectively.

We introduce the penalty factor into the Grey relational coefficient to calculate correlation degree and correct matching function. The attribute correlation coefficient is calculated as follows:

$$\xi_{ij} = \frac{\min + \eta \cdot \max}{\lambda_j \left|p_{n+1}(j) - p_i(j)\right| + \eta \cdot \max}$$
$$\min = \min \min \left|p_{n+1}(j) - p_i(j)\right| \tag{13}$$
$$\max = \max \max \left|p_{n+1}(j) - p_i(j)\right|.$$

The Grey incidence matrix is defined as follows:

$$\xi = \begin{bmatrix} \xi_{11} & \xi_{12} & \cdots & \xi_{1m} \\ \xi_{21} & \xi_{22} & \cdots & \xi_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \xi_{n1} & \xi_{n2} & \cdots & \xi_{nm} \end{bmatrix}. \tag{14}$$

ξ_{ij} is correlation coefficient of the jth attribute of the ith resource. η is the distinguishing coefficient, with a general value of 0.5. λ_j is a penalty factor of the jth attribute. min and max represent the maximum difference and the minimum difference. $p_i(j)$ and $p_{n+1}(j)$ are standardized values.

The modified matching function is

$$\text{sim}(r_i, uq) = \alpha * \text{sim}(rq_i, uq_i) + (1 - \alpha) * \frac{1}{m}\sum_{j=1}^{m} \xi_{ij}, \tag{15}$$

where α represents weight. When the matching degree reaches the threshold k, that is, $\text{sim}(r_i, uq) \geq k$, this indicates a successful match. The resources that successfully match are put into a candidate resource set Rs'.

The specific matching algorithm is illustrated in Algorithm 1.

4.2. Regression-Markov Chain Prediction Estimation. Through the above analysis, we select a candidate resource set which satisfy the QoS requirements of users. Because of the dynamic and uncertain nature of the resources, the amount and value of data collected of QoS will fluctuate. Moreover, the inherent mobility of the IoT makes mobile users have certain volatility when using resources. Therefore, we further select resources by analyzing the load changes of resources themselves.

Input: $Rs = \{r_1, r_2, \ldots, r_n\}, R, uq$
Output: Rs'
(1) Normalized matrix R
(2) $[n, m] = \text{size}(R)$
(3) **for** $j = 1 \rightarrow m$ **do**
(4) get the w_j
(5) **end for**
(6) $Q = [R; uq]$
(7) $[\text{row}, \text{col}] = \text{size}(Q)$
(8) $RA[] \leftarrow 0$
(9) $RB[] \leftarrow 0$
(10) $u \leftarrow 1$
(11) $v \leftarrow 1$
(12) **for** $i = 1 \rightarrow$ row-1 **do**
(13) get the $d(i, uq)$ and $d_{\text{sim}}(i, uq)$
(14) **if** $d_{\text{sim}}(i, uq) \geq \varepsilon$ **then**
(15) $RA[u] \leftarrow r_i$
(16) $u \leftarrow u + 1$
(17) **else**
(18) $RB[v] \leftarrow r_i$
(19) $v \leftarrow v + 1$
(20) **end if**
(21) **end for**
(22) $Rs'[] \leftarrow 0$
(23) $t \leftarrow 1$
(24) **while** resource $r_i \in RA$ **do**
(25) get the matrix PA
(26) calculate the λ_j and ξ_{ij}
(27) calculate the $\text{sim}(r_i, uq)$
(28) **if** $\text{sim}(r_i, uq) \geq k$ **then**
(29) $Rs'[] \leftarrow r_i$
(30) $t \leftarrow t + 1$
(31) **end if**
(32) **end while**
(33) **return** Rs'

Algorithm 1: Multiattribute QoS resource matching.

Due to the randomness and volatility of the load, the single resource prediction method is difficult to predict accurately. The load has the characteristics of frequent changes in the short term [18]. The future load will be affected by the current load, and the load at the next time can be predicted by current load value. Therefore, this paper adopts the regression-Markov chain prediction method to better reflect the changing trend and stochastic fluctuation characteristics of resources.

According to the difference of time t, the original data sequence of the resource load value is recorded as

$$\left\{X^{(0)}(t)\right\} = \left\{X^{(0)}(1), X^{(0)}(2), \ldots, X^{(0)}(p)\right\}. \tag{16}$$

Firstly, we use the linear regression method to generate the predicted sequence

$$\left\{\widehat{X}^{(0)}(t)\right\} = \left\{\widehat{X}^{(0)}(1), \widehat{X}^{(0)}(2), \ldots, \widehat{X}^{(0)}(p)\right\} \tag{17}$$

and calculate the relative residuals of the original sequence and the predicted sequence

$$\left\{X^{(1)}(t)\right\} = \left\{\widehat{X}^{(0)}(t) - X^{(0)}(t)\right\}. \tag{18}$$

Then, we divide the relative residual sequence $\{X^{(1)}(t)\}$ into s state intervals. According to the state distribution $S = (1, 2, \ldots, s)$, a one-step transition probability matrix is calculated.

$$M_{s\times s} = M(s_i)(s_j) = p_{ij}(X_{s+1} = j \mid X_s = i)$$

$$M = \begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1s} \\ M_{21} & M_{22} & \cdots & M_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ M_{s1} & M_{s2} & \cdots & M_{ss} \end{bmatrix}. \tag{19}$$

If matrix M satisfies the following conditions: $M_{ij} \geq 0$, $i, j \in S$, and $\sum M_{ij} = 1$, $i, j \in S$, M is a random matrix. $P^{(n)} = (p_{ij}^{(n)}) = P^n$ represents n step transfer matrix.

It is assumed that the Markov chain reaches a stable state after the n step transition. The stable state vector $x = [x_1, x_2, \ldots, x_s]$ satisfies

$$x = xM$$
$$\sum_{i=1}^{s} x_i = 1. \tag{20}$$

We set the initial state to obtain the probabilities $p_1, p_2, \ldots, p_s$ corresponding to the error state interval by solving the distribution probability of the stable state. According to the maximum probability principle, the state corresponding to the maximum probability is taken as the state of the next moment. The predictive value is brought into the corresponding state interval to solve the prediction interval value. On this basis, we predict the availability of resources over a period of time and rationally select the right resource.

5. Experimental Results

To test the performance of estimation method, we use the performance indicators in [13] as the estimation indicators. The performance indicators in this experiment are defined as follows.

Precision (Prec = $|A|/|B|$) is used to measure the accuracy of resource selection. Specifically, candidate resources that successfully match the user QoS make up the percentage of all resources, where A represents candidate resources that successfully match the user QoS and B represents all the resources.

Recall (Rec = $|A|/|A \cup C|$) is used to measure the effectiveness of resource selection. Specifically, the candidate resources that successfully match the user account for the percentage of all matching successful resources, where C represents the matching successful resources that are not within the candidate resources set.

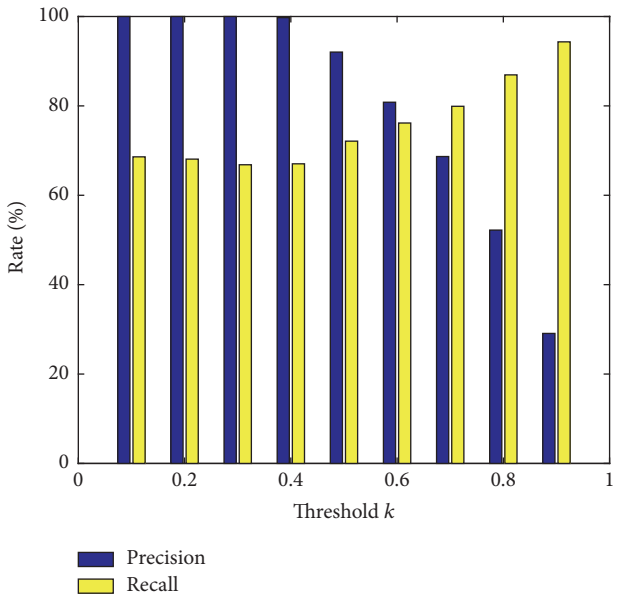

FIGURE 4: Variation of precision and recall under different matching threshold.

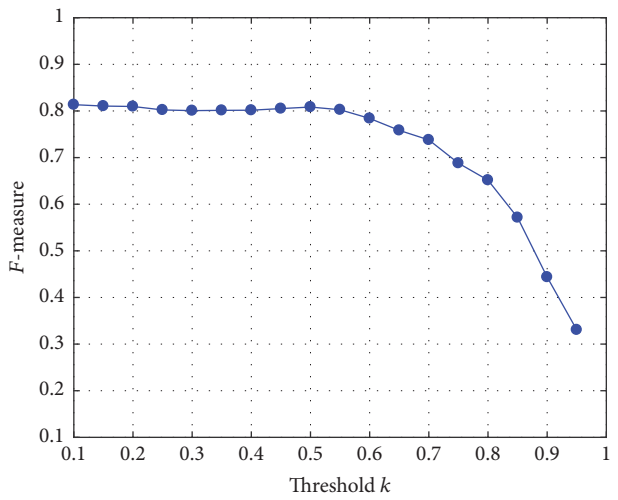

FIGURE 5: Variation of F-measure under different matching threshold.

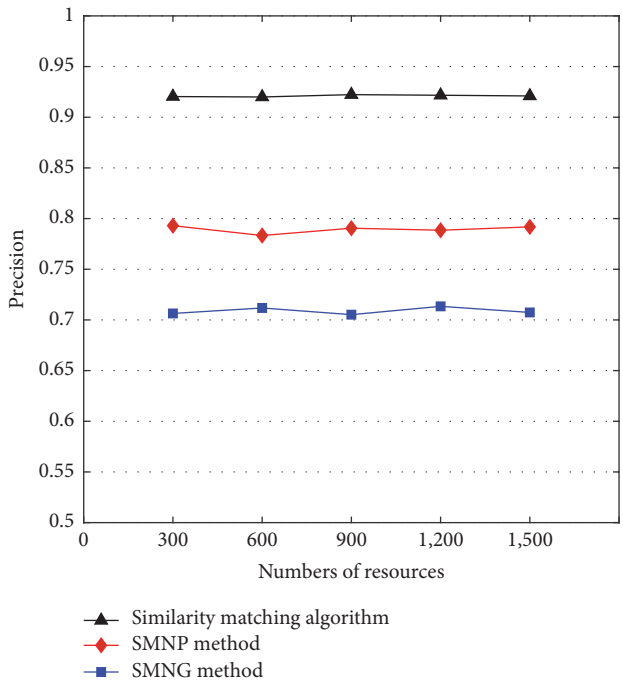

FIGURE 6: The comparison of precision of different methods under different numbers of resources.

F-measure (F-measure = (2 ∗ Prec ∗ Rec)/(Prec + Rec)) is the weighted average of precision and recall, which is used to reflect the overall performance. The more the F-measure tends to 1, the more effective the estimation method is.

The experimental platform adopts MATLAB to set up a different number of resource nodes randomly. Each resource node includes multiple QoS attribute information. The QoS attributes information is set by simulation of resources characteristics on edge servers, including response time, availability, and price. The proximity degree ε and the weight α are 0.5. In order to ensure the effectiveness of the result, we conduct the experiment 20 times and take the average value as the experimental result.

In order to obtain the best performance, we set up a matching threshold k to filter irrelevant resources by low similarity. Taking into account the contradictory relation between precision and recall, we decide to use F-measure as the initial standard to find the optimal threshold.

Figure 4 shows the variation of precision and recall under different matching threshold. As can be seen from Figure 4, the precision and recall are inversely related. When the matching degree is higher, the precision is lower, and the recall is higher. In order to ensure the precision and recall within acceptable limits, the threshold k should be between 0.5 and 0.6. Figure 5 shows the change of F-measure under different matching threshold. It can be seen that, with the increase of threshold, the F-measure is significantly decreased. When the k value is 0.5, the F-measure is in the higher position, so $k = 0.5$ is taken as the matching threshold.

Figures 6 and 7 show the precision and recall performance of the method, respectively. Our method is compared with two methods where the first does not consider the penalty factor (denoted as SMNP method) and the second does not consider the penalty factor and Grey relational grade (denoted as SMNG method), respectively. As we can see, our method is superior to other methods on precision and stays above 90%. Since the penalty factor improves the similarity of resources, it makes the resources meet the needs of users as much as possible. But the recall is lower than the other two methods and remains at around 72%, within the acceptable level. Although the method can avoid the constraint and relationship between the attributes of resources, it only matches the quantity attribute of the resources.

Figure 8 shows the F-measure under the different numbers of resources. It can be seen that our method is superior

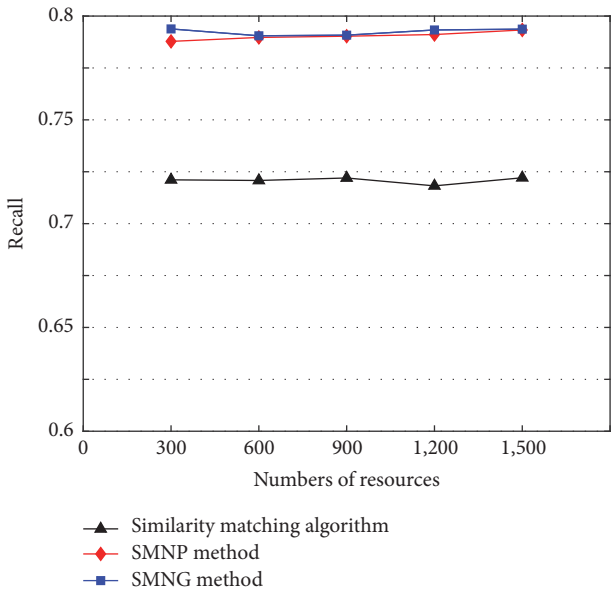

Figure 7: The comparison of recall of different methods under different numbers of resources.

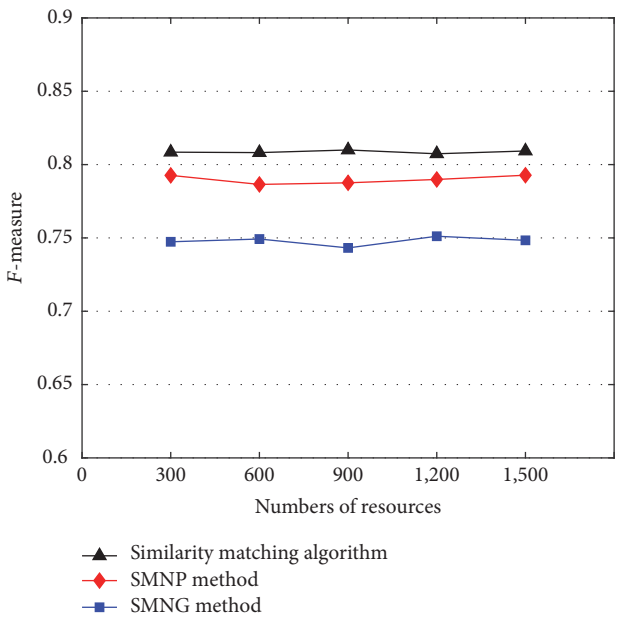

Figure 8: The comparison of F-measure of different methods under different numbers of resources.

Table 1: Residual state interval.

State	Residual interval
(1)	(−20%, −10%]
(2)	(−10%, −5%]
(3)	(−5%, 0%]
(4)	(0%, 5%]
(5)	(5%, 10%]
(6)	(10%, 20%]

to other methods. The F-measure is maintained above 80%, which further validates the effectiveness of our method.

In this paper, we take CPU utilization as resource load index to predict. We record CPU utilization of downloading files every 5 seconds to obtain time series data. We use the first 10 times' load data to make short-term prediction of the next 5 times' CPU utilization and prove the effectiveness of the method by the error values. The residual sequence is divided into the following 6 states by linear regression analysis, as shown in Table 1.

As the regression-Markov chain prediction method is interval prediction method, the prediction result is interval distribution. We select the state interval with the maximum probability as the prediction interval. We select (−5%, 0%] state interval as a result by analyzing the probability of each state interval in the stable state.

The error result of the regression-Markov chain prediction method is shown in Figure 9. It can be seen that the method has a higher prediction accuracy and less error compared with the linear regression prediction method. Due to the large fluctuation and randomness of the load, the single prediction method has a poor effect. The linear regression method can predict the changing trend of the load state, but it can not reflect the stochastic volatility. The Markov chain solves the stochastic volatility problem and improves the accuracy of the prediction method.

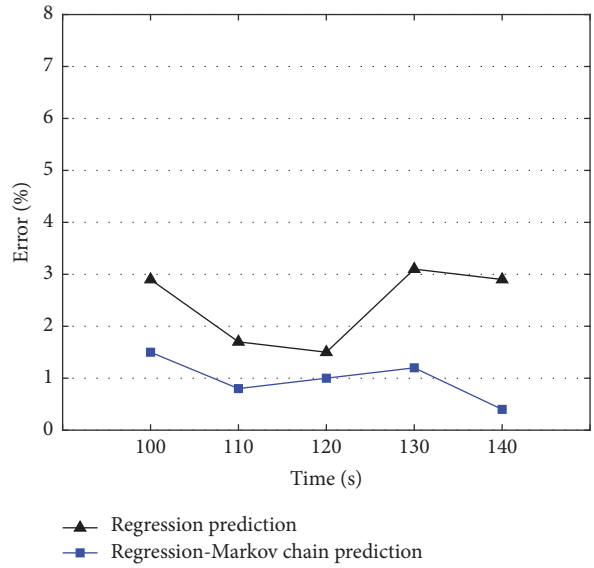

Figure 9: Comparison of errors between regression prediction and regression-Markov chain prediction in the future time.

6. Conclusion and Future Work

With the rapid development of Internet of Things and cloud computing, service QoS and user satisfaction become an important challenge. Local computing and storage capabilities of edge computing can reduce latency and improve user satisfaction. In this paper, we use weighted Euclidean distance similarity to classify multiple QoS attribute resources. We

select the appropriate resources by similarity matching and regression-Markov chain prediction method. Since the QoS attributes system is extensible and the user QoS requirements are dynamic, the estimation method has certain scalability. On the basis of the existing work, we can design a reasonable method of resource estimation to balance the satisfaction between users and service providers and improve the utilization of resources.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61672321, 61771289), the Shandong provincial Postgraduate Education Innovation Program (SDYY14052, SDYY15049), the Shandong Provincial Specialized Degree Postgraduate Teaching Case Library Construction Program, the Shandong Provincial Postgraduate Education Quality Curriculum Construction Program, the Shandong Provincial University Science and Technology Plan Project (J16LN15), and the Qufu Normal University Science and Technology Plan Project (xkj201525).

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: a survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] D. Evans, *The Internet of Things: How the Next Evolution of the Internet is Changing Everything*, Cisco Systems, 2011.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the Mcc Workshop on Mobile Cloud Computing*, pp. 13–16, ACM, August 2012.

[5] S. M. Hemam and O. Hioual, "Load balancing issue in cloud services selection by using MCDA and Markov Chain Model approaches," in *Proceedings of the International Conference on Cloud Computing Technologies and Applications*, pp. 163–169, IEEE, May 2016.

[6] M. M. Al-Sayed, S. Khattab, and F. A. Omara, "Prediction mechanisms for monitoring state of cloud resources using Markov chain model," *Journal of Parallel and Distributed Computing*, vol. 96, pp. 163–171, 2016.

[7] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[8] S. Shekhar and A. Gokhale, "Dynamic resource management across cloud-edge resources for performance-sensitive applications," in *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 707–710, IEEE Press, May 2017.

[9] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: a framework for edge node resource management," *IEEE Transactions on Services Computing*, 2017.

[10] M. Aazam and E.-N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT," in *Proceedings of the IEEE 29th International Conference on Advanced Information Networking and Applications (AINA '15)*, vol. 51, no. 5, pp. 687–694, IEEE Computer Society, March 2015.

[11] M. Aazam, M. St-Hilaire, C.-H. Lung, and I. Lambadaris, "MeFoRE: QoE based resource estimation at Fog to enhance QoS in IoT," in *Proceedings of the 23rd International Conference on Telecommunications (ICT '16)*, pp. 1–5, May 2016.

[12] L.-Y. Zuo, Z.-B. Cao, and S.-B. Dong, "Virtual resource evaluation model based on entropy optimized and dynamic weighted in cloud computing," *Journal of Software*, vol. 24, no. 8, pp. 1937–1946, 2013.

[13] S. S. Zhao, Y. Zhang, L. Yu, B. Cheng, Y. Ji, and J. Chen, "A multidimensional resource model for dynamic resource matching in internet of things," *Concurrency & Computation Practice & Experience*, vol. 27, no. 8, pp. 1819–1843, 2015.

[14] J. Zhou, M. Yan, X. Ye, and H. Lu, "An algorithm of resource evaluation and selection based on Multi-QoS constraints," in *Proceedings of the Web Information Systems & Applications Conference*, pp. 49–52, IEEE Computer Society, 2010.

[15] W. E. Dong, W. U. Nan, and L. I. Xu, "QoS-oriented monitoring model of cloud computing resources availability," in *Proceedings of the 5th International Conference on Computational and Information Sciences*, pp. 1537–1540, June 2013.

[16] S. Ding, C. Xia, Q. Cai, K. Zhou, and S. Yang, "QoS-aware resource matching and recommendation for cloud computing systems," *Applied Mathematics and Computation*, vol. 247, no. 15, pp. 941–950, 2014.

[17] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, 2017.

[18] L. Zhang and D. Y. Xu, "Multi-step optimized GM (1, 1) model-based short term resource load prediction in cloud computing," *Computer Engineering and Applications*, vol. 50, no. 10, pp. 65–71, 2014.