QoS Ranking Prediction for Cloud Services

Zibin Zheng, *Member, IEEE*, Xinmaio Wu, Yilei Zhang, *Student Member, IEEE*, Michael R. Lyu, *Fellow, IEEE, and Jianmin Wang*

Abstract—Cloud computing is becoming popular. Building high quality cloud applications is a critical research problem. QoS rankings provide valuable information for making optimal cloud service selection from a set of functionally equivalent service candidates. To obtain QoS values, real-world invocations on the service candidates are usually required. To avoid the time-consuming and expensive real-world service invocations, this paper proposes a QoS ranking prediction framework for cloud services by taking advantage of the past service usage experiences of other consumers. Our proposed framework requires no additional invocations of cloud services when making QoS ranking prediction. Two personalized QoS ranking prediction approaches are proposed to predict the QoS rankings directly. Comprehensive experiments are conducted employing real-world QoS data, including 300 distributed users and 500 real-world Web services all over the world. The experimental results show that our approaches outperform other competing approaches.

Index Terms—Quality-of-Service, cloud service, ranking prediction, personalization

1 Introduction

Cloud computing is Internet-based computing, whereby shared configurable resources (e.g., infrastructure, platform, and software) are provided to computers and other devices as services [1]. Strongly promoted by the leading industrial companies (e.g., Amazon, Google, Microsoft, IBM, etc.), cloud computing is quickly becoming popular in recent years. Applications deployed in the cloud environment (named *cloud applications* in this paper) are typically large-scale and complex. With the rising popularity of cloud computing, how to build high-quality cloud applications becomes an urgently-required research problem.

Similar to traditional component-based systems, cloud applications typically involve multiple cloud components communicating with each other over application programming interfaces, such as through Web services. Figure 1 shows an example of cloud applications. As shown in the figure, *Cloud application 1* is a tourism Website deployed in the cloud (e.g., Amazon EC2 http://aws.amazon.com/ec2), providing various types of tourism services to customers. The business process of this cloud application is composed by a number of software components, where each component fulfills a specified functionality. To outsource part of business to other companies, some of these components invoke other cloud services (e.g., airplane ticket services, car rental services, and hotel booking services in Figure 1). These cloud services (can be implemented as Web services) are provided and deployed in the cloud by other companies. These cloud services can also be employed by other cloud applications (e.g., *Cloud application 2* and *Cloud application 3* in Figure 1). Since there are a number of functionally equivalent services in the cloud, optimal service selection becomes important. In this paper, service users refer to cloud applications that use/invoke the cloud services. In the context of a service invocation, the user-side (or client-side) refers to the cloud applications and server-side refers to the cloud services.

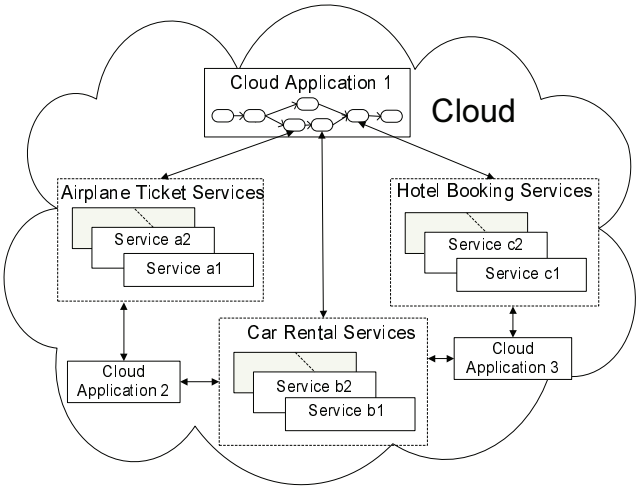


Fig. 1. Motivating Example

Non-functional performance of cloud services is usually described by Quality-of-Service (QoS). QoS is an important research topic in cloud computing. When making optimal cloud service selection from a set of functionally equivalent services, QoS values of cloud services provide valuable information to assist decision

• *Zibin Zheng, Yilei Zhang, and Michael R. Lyu are with the Shenzhen Research Institute, and Department of Computer Science & Engineering, The Chinese University of Hong Kong. Zibin Zheng is also with State key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications), Beijing, China. Michael R. Lyu is also with School of Computer Science, National University of Defence Technology, Hunan, China. Xinmiao Wu and Jianmin Wang are with the Department of Computer Science, Sun Yat-sen University, China. E-mail: zbzheng@cse.cuhk.edu.hk, oceanwxm@126.com, ylzhang@cse.cuhk.edu.hk, lyu@cse.cuhk.edu.hk, mcswjm@mail.sysu.edu.cn*
Corresponding Author: Jianmin Wang

Manuscript received March 01, 2012; revised Aug. 01, 2012;

Digital Object Indentifier 10.1109/TPDS.2012.285

making. In traditional component-based systems, software components are invoked locally, while in cloud applications, cloud services are invoked remotely by Internet connections. Client-side performance of cloud services is thus greatly influenced by the unpredictable Internet connections. Therefore, different cloud applications may receive different levels of quality for the same cloud service. In other words, the QoS ranking of cloud services for a user(e.g., *Cloud application 1*) cannot be transferred directly to another user (e.g., *Cloud application 2*), since the locations of the cloud applications are quite different. Personalized cloud service QoS ranking is thus required for different cloud applications.

The most straightforward approach of personalized cloud service QoS ranking is to evaluate all the candidate services at the user-side and rank the services based on the observed QoS values. However, this approach is impractical in reality, since invocations of cloud services may be charged. Even if the invocations are free, executing a large number of service invocations is time consuming and resource consuming, and some service invocations may produce irreversible effects in the real world. Moreover, when the number of candidate services is large, it is difficult for the cloud application designers to evaluate all the cloud services efficiently.

To attack this critical challenge, we propose a personalized ranking prediction framework, named *CloudRank*, to predict the QoS ranking of a set of cloud services without requiring additional real-world service invocations from the intended users. Our approach takes advantage of the past usage experiences of other users for making personalized ranking prediction for the current user. Extended from its preliminary conference version [24], the contribution of this paper is two-fold:

- This paper identifies the critical problem of personalized QoS ranking for cloud services and proposes a QoS ranking prediction framework to address the problem. To the best of our knowledge, CloudRank is the first personalized QoS ranking prediction framework for cloud services.
- Extensive real-world experiments are conducted to study the ranking prediction accuracy of our ranking prediction algorithms compared with other competing ranking algorithms. The experimental results show the effectiveness of our approach.
- We publicly release our service QoS dataset[1] for future research, which makes our experiments reproducible.

The rest of this paper is organized as follows: Section 2 introduces our system architecture. Section 3 describes the proposed CloudRank framework. Section 4 presents experiments. Section 5 discusses related work and Section 6 concludes the paper.

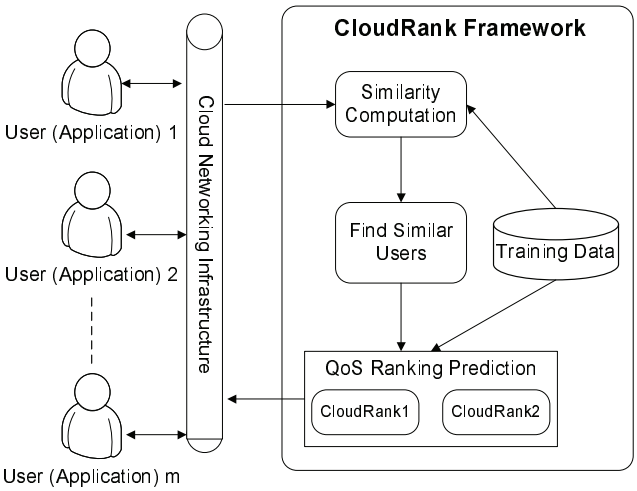


Fig. 2. System Architecture of CloudRank

2 System Architecture

Quality-of-Service (QoS) can be measured at the server-side or at the client-side. While server-side QoS properties provide good indications of the cloud service capacities, client-side QoS properties provide more realistic measurements of the user usage experience. The commonly-used client-side QoS properties include response-time, throughput, failure probability, etc. This paper mainly focuses on ranking prediction of client-side QoS properties, which likely have different values for different users (or user applications) of the same cloud service. Figure 2 shows the system architecture of our CloudRank framework, which provides personalized QoS ranking prediction for cloud services. The target users of the CloudRank framework are the cloud applications, which need personalized cloud service ranking for making optimal service selection. A user is called *active user* if he/she is requesting ranking prediction from the CloudRank framework. As shown in Figure 2, a user (e.g., *Cloud application 1* in Figure 1) can obtain service ranking prediction of all available cloud services from the CloudRank framework by providing observed QoS values of some cloud services. More accurate ranking prediction results can be achieved by providing QoS values on more cloud services, since the characteristic of the active user can be mined from the provided data.

Within the CloudRank framework, there are several modules. Firstly, based on the user-provided QoS values, similarities between the active user and training users can be calculated. Secondly, based on the similarity values, a set of similar users can be identified. After that, two algorithms are proposed (i.e., *CloudRank1* and *CloudRank2*) to make personalized service ranking by taking advantages of the past service usage experiences of similar users. Finally, the ranking prediction results are provided to the active user. The training data in the CloudRank framework can be obtained from: (1) the QoS

1. http://www.zibinzheng.com/tpds2012

values provided by other users; and (2) the QoS values collected by monitoring cloud services.

In our previous work [21], a user-collaborative mechanism is proposed for collecting client-side QoS values of Web services from different service users. The observed Web service QoS values can be contributed by users by running a client-side Web service evaluation application [21]. Different from service-oriented applications, the usage experiences of cloud services are much easier to be obtained in the cloud environment. The cloud applications can invoke and record the client-side QoS performance of the invoked cloud services easily by using monitoring infrastructure services provided by the cloud platform. The cloud provider can collect these client-side QoS values from different cloud applications easily with approval of application owners. The framework can be used at both design time and runtime. At runtime, the cloud application may obtain new QoS values on some cloud services. By providing these values to our CloudRank server, new QoS ranking prediction can be obtained. Based on the service QoS ranking, optimal system reconfiguration can be achieved.

3 QoS Ranking Prediction

This section presents our CloudRank QoS ranking prediction framework for cloud services. Section 3.1 calculates the similarity of the active user with training users based on their rankings on the commonly-invoked cloud services. Section 3.2 identifies a set of similar users. Section 3.3 presents two QoS ranking prediction algorithms, named *CloudRank1* and *CloudRank2*, respectively. Section 3.4 analyzes the computational complexity.

3.1 Similarity Computation

Ranking similarity computations compare users' QoS rankings on the commonly-invoked services. Suppose we have a set of three cloud services, on which two users have observed response-times (seconds) of {1, 2, 4} and {2, 4, 5}, respectively. The response-time values on these services observed by the two users are clearly different; nevertheless their rankings are very close as the services are ordered in the same way. Given two rankings on the same set of services, the Kendall Rank Correlation Coefficient (KRCC) [14] evaluates the degree of similarity by considering the number of inversions of service pairs which would be needed to transform one rank order into the other. The KRCC value of user u and user v can be calculated by :

$$Sim(u,v) = \frac{C-D}{N(N-1)/2}, \tag{1}$$

where N is the number of services, C is the number of concordant pairs between two lists, D is the number of discordant pairs, and there are totally $N(N-1)/2$ pairs for N cloud services. Since $C = N(N-1)/2 - D$, Eq. (1) is equal to $Sim(u,v) = 1 - \frac{4D}{N(N-1)}$. Employing KRCC, the similarity between two service rankings can be calculated by:

$$Sim(u,v) = 1 - \frac{4 \times \sum_{i,j \in I_u \cap I_v} \tilde{I}((q_{u,i} - q_{u,j})(q_{v,i} - q_{v,j}))}{|I_u \cap I_v| \times (|I_u \cap I_v| - 1)}, \tag{2}$$

where $I_u \cap I_v$ is the subset of cloud services commonly invoked by user u and user v, $q_{u,i}$ is the QoS value (e.g., response-time, throughput, etc.) of service i observed by user u, and $\tilde{I}(x)$ is an indicator function defined as:

$$\tilde{I}(x) = \begin{cases} 1 & \text{if x} < 0 \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

From above definition, the ranking similarity between two rankings is in the interval of [-1,1], where -1 is obtained when the order of user u is the exact reverse of user v, and 1 is obtained when order of user u is equal to the order of user v. Since KRCC compares service pairs, the intersection between two users has to be at least 2 ($|I_u \cap I_v| \geq 2$) for making similarity computation.

3.2 Find Similar Users

By calculating similarity values between the current active user with other training users, the similar users can be identified. Previous approaches [12], [18] usually employ information of all the users for making ranking prediction of the current user, which may include dissimilar users. However, employing QoS values of dissimilar users will greatly influence the prediction accuracy. To address this problem, we exclude the users with negative correlations (negative similarity values) and only employ the Top-K similar users for making QoS ranking prediction. In our approach, a set of similar users $S(u)$ is identified for the active user u by:

$$N(u) = \{v | v \in T_u, Sim(u,v) > 0, v \neq u\}, \tag{4}$$

where T_u is a set of the Top-K similar users to the user u and $Sim(u,v) > 0$ excludes the dissimilar users with negative similarity values. The value of $Sim(u,v)$ in Eq. (4) is calculated by Eq. (2).

3.3 QoS Ranking Prediction

Rating-oriented collaborative filtering approaches first predict the missing QoS values before making QoS ranking (details are provided in Appendix A). The target of rating-oriented approaches is to predict QoS values as accurate as possible. However, accurate QoS value prediction may not lead to accurate QoS ranking prediction.

For example, as shown in Figure 3, assume the expected response times of three services are 2, 3, and 5 seconds, respectively. There are two predictions using rating-oriented approaches: (3, 2, 4) and (1, 2, 3). Since rating-oriented approaches try to predict the QoS value as accurate as possible, *Prediction 1* is better than *Prediction 2*, since it has a smaller MAE value (MAE refers to Mean Absolute Error, which is an evaluation metric for

Expected Response time: (2, 3, 5) on (s1, s2, s3)
Prediction 1: (3, 2, 4); MAE = (|2−3| + |3−2| + |5−4|)/3 = 1
Prediction 2: (1, 2, 3); MAE = (|2−1| + |3−2| + |5−3|)/3 = 1.3

Expected Ranking: s1 < s2 < s3
Prediction 1: s2 < s1 < s3
Prediction 2: s1 < s2 < s3

Fig. 3. Rating-oriented vs. Ranking-oriented

rating-oriented prediction results. Details of MAE will be introduced in Section 4.2). However, from the ranking-oriented perspective, *Prediction 1* is worse than *Prediction 2* since the former leads to incorrect ranking based on the predicted QoS values.

To address this problem, we propose two ranking-oriented approaches, named as CloudRank1 and CloudRank2, in the following. Our ranking-oriented approaches predict the QoS ranking directly without predicting the corresponding QoS values.

3.3.1 CloudRank1

A user's preference on a pair of services can be modeled in the form of $\Psi : I \times I \rightarrow \mathbb{R}$, where $\Psi(i,j) > 0$ means that quality of service i is better than service j and is thus more preferable for the active user and vice versa. The value of the preference function $\Psi(i,j)$ indicates the strength of preference and a value of zero means that there is no preference between two services. The preference function $\Psi(i,j)$ is anti-symmetric, i.e. $\Psi(i,j) = -\Psi(j,i)$. We set $\Psi(i,i) = 0$ for all $i \in I$.

Given the user-observed QoS values on two cloud services, the preference between these two services can be easily derived by comparing the QoS values, where $\Psi(i,j) = q_i - q_j$. To obtain the preference values regarding pairs of services that have not been invoked or observed by the current user, the preference values of similar users $S(u)$ are employed. The basic idea is that the more often the similar users in $S(u)$ observe service i as higher quality than service j, the stronger the evidence is of $\Psi(i,j) > 0$ for the current user. This leads to the following formula for estimating the value of the preference function $\Psi(i,j)$, where service i and service j are not explicitly observed by the current user u:

$$\Psi(i,j) = \sum_{v \in N(u)^{ij}} w_v (q_{v,i} - q_{v,j}), \tag{5}$$

where v is a similar user of the current u, $N(u)^{ij}$ is a subset of similar users, who obtain QoS values of both services i and j, and w_v is a weighting factor of the similar user v, which can be calculated by:

$$w_v = \frac{Sim(u,v)}{\sum_{v \in N(u)^{ij}} Sim(u,v)}. \tag{6}$$

w_v makes sure that a similar user with higher similarity value has greater impact on the preference value prediction in Eq. (5).

Algorithm 1: CloudRank1

Input: an employed service set E, a full service set I, a preference function Ψ
Output: a service ranking $\hat{\rho}$

$F = E$;
while $F \neq \emptyset$ **do**
  $t = \arg\max_{i \in F} q_i$;
  $\rho_e(t) = |E| - |F| + 1$;
  $F = F - \{t\}$;
end
foreach $i \in I$ **do**
  $\pi(i) = \sum_{j \in I} \Psi(i,j)$;
end
$n = |I|$;
while $I \neq \emptyset$ **do**
  $t = \arg\max_{i \in I} \pi(i)$;
  $\hat{\rho}(t) = n - |I| + 1$;
  $I = I - \{t\}$;
  foreach $i \in I$ **do**
    $\pi(i) = \pi(i) - \Psi(i,t)$
  end
end
while $E \neq \emptyset$ **do**
  $e = \arg\min_{i \in E} \rho_e i$;
  $index = \min_{i \in E} \hat{\rho}(i)$;
  $\hat{\rho}(e) = index$;
  $E = E - \{e\}$;
end

With Eq. (5) and Eq. (6), the preference value between a pair of services can be obtained by taking advantage of the past usage experiences of similar users. Assuming there are n services to be ranked and QoS values of a services have already be observed by user u, the total number of service pairs that can be derived explicitly is $a(a-1)/2$, and the total number of pairs that need to be predicted from similar users is: $n(n-1)/2 - a(a-1)/2$.

Given a preference function Ψ which assigns a score to every pair of services $i, j \in I$, we want to choose a quality ranking of services in I that agrees with the pairwise preferences as much as possible. Let ρ be a ranking of services in I such that $\rho(i) > \rho(j)$ if and only if i is ranked higher than j in the ranking ρ. We can define a value function $V^{\Psi}(\rho)$ as follows, which measures the consistency of the ranking ρ with the preference function:

$$V^{\Psi}(\rho) = \sum_{i,j:\rho(i)>\rho(j)} \Psi(i,j). \tag{7}$$

Our goal is to produce a ranking ρ^* that maximizes the above objective value function. One possible solution is to search through the possible rankings and select the optimal ranking ρ^* that maximizes the value function defined in Eq. (7). However, there are $n!$ possible rankings for n services, and the optimal ranking search problem is NP-Complete [6]. To enhance the calculation efficiently, we propose a greedy-based algorithm in Algorithm 1 (named as *CloudRank1*) for finding an approximately optimal ranking. Algorithm 1 includes the following steps:

- Step 1 (lines 1 - 6): Rank the employed cloud services in E based on the observed QoS values. $\rho_e(t)$ stores the ranking, where t is a cloud service and the function $\rho_e(t)$ returns the corresponding order of this service. The values of $\rho_e(t)$ are in the range of $[1, |E|]$, where a smaller value indicates higher quality.
- Step 2 (lines 7 - 9): For each service in the full service set I, calculate the sum of preference values with all other services by $\pi(i) = \sum_{j \in I} \Psi(i, j)$. Since $\Psi(i, i) = 0$, including $\Psi(i, i)$ in the calculation does not influence the results. Larger $\pi(i)$ value indicates more services are less preferred than i (i.e., $\Psi(i, j) > 0$). In other words, service i should be ranked in a higher position.
- Step 3 (lines 10 - 18): Services are ranked from the highest position to the lowest position by picking the service t that has the maximum $\pi(t)$ value. The selected service is assigned a rank equal to $n-|I|+1$ so that it will be ranked above all the other remaining services in I. The ranks are in the range of $[1, n]$, where n is the number of services and a smaller value indicates higher quality. The selected service t is then removed from I and the preference sum values $\Psi(i)$ of the remaining services are updated to remove the effects of the selected service t.
- Step 4 (lines 19 - 24): Step 3 treats the employed services in E and the non-employed service in $I - E$ identically which may incorrectly rank the employed services. In this step, the initial service ranking $\hat{\rho}$ is updated by correcting the rankings of the employed services in E. By replacing the ranking results in $\hat{\rho}$ with the corresponding correct ranking of ρ_e, our approach makes sure that the employed services in E are correctly ranked.

Compared with the greedy algorithm in [6], our approach guarantees that the employed services are correctly ranked. As will be shown in the experiments in Section 4, our approach provides better ranking accuracy more consistently than the traditional greedy algorithm.

3.3.2 CloudRank2

The preference values $\Psi(i, j)$ in the CloudRank1 algorithm can be obtained explicitly or implicitly. When the active user has QoS values on both the service i and service j, the preference value is obtained explicitly. On the other hand, the preference value is obtained implicitly when employing QoS information of similar users. Assuming there are three cloud services a, b, and c. The active users have invoked service a and service b previously. The list below shows how the preference values of $\Psi(a, b)$, $\Psi(a, c)$, and $\Psi(b, c)$ can be obtained explicitly or implicitly:

- $\Psi(a, b)$: obtained explicitly.
- $\Psi(a, c)$: obtained implicitly by similar users with similarities of 0.1, 0.2, and 0.3.
- $\Psi(b, c)$: obtained implicitly by similar users with similarities of 0.7, 0.8, and 0.9.

Algorithm 2: CloudRank2

Input: an employed service set E, a full service set I, a preference function Ψ, confidence values C
Output: a service ranking $\hat{\rho}$

```
1  F = E;
2  while F ≠ ∅ do
3  |   t = arg max_{i∈F} q_i;
4  |   ρ_e(t) = |E| − |F| + 1;
5  |   F = F − {t};
6  end
7  foreach i ∈ I do
8  |   π(i) = Σ_{j∈I} C(i, j) × Ψ(i, j);
9  end
10 n = |I|;
11 while I ≠ ∅ do
12 |   t = arg max_{i∈I} π(i);
13 |   ρ̂(t) = n − |I| + 1;
14 |   I = I − {t};
15 |   foreach i ∈ I do
16 |   |   π(i) = π(i) − C(i, j) × Ψ(i, t)
17 |   end
18 end
19 while E ≠ ∅ do
20 |   e = arg min_{i∈E} ρ_e i;
21 |   index = min_{i∈E} ρ̂(i);
22 |   ρ̂(e) = index;
23 |   E = E − {e};
24 end
```

In the above example, we can see that different preference values have different confidence levels. It is clear that $C(a, b) > C(b, c) > C(a, c)$, where C represents the confidence values of different preference values. The confidence value of $\Psi(b, c)$ is higher than $\Psi(a, c)$, since the similar users of $\Psi(b, c)$ have higher similarities.

In the CloudRank1 algorithm, differences in preference values are treated equally, which may hurt the QoS ranking prediction accuracy. By considering the confidence values of different preference values, we propose a QoS ranking prediction algorithm, named *CloudRank2*, which uses the following rules to calculate the confidence values:

- If the user has QoS values of these two services i and j. The confidence of the preference value is 1.
- When employing similar users for the preference value prediction, the confidence is determined by similarities of similar users as follows:

$$C(i, j) = \sum_{v \in N(u)^{ij}} w_v Sim(u, v), \tag{8}$$

where v is a similar user of the current active user u, $N(u)^{ij}$ is a subset of similar users, who obtain QoS values of both services i and j, and w_v is a weighting factor of the similar user v, which can be calculated by:

$$w_v = \frac{Sim(u, v)}{\sum_{v \in N(u)^{ij}} Sim(u, v)}. \tag{9}$$

w_v makes sure that a similar user with higher similarity value has greater impact on the confidence calculation. Eq. (8) guarantees that similar users with

higher similarities will generate higher confidence values.

Algorithm 2 shows the details of the CloudRank2 algorithm, which considers the confidence values of different preference values when calculating the preference sum. In this way, more accurate ranking prediction can be achieved.

3.4 Computational Complexity Analysis

Assuming there are n cloud services and m users, this section analyzes the worst case computational complexity of the CloudRank1 and CloudRank2 algorithms, respectively.

In Section 3.1, the computational complexity of $Sim(a, u)$ by KRCC is $O(n^2)$, since there are at most $n(n-1)/2$ service pairs on the n commonly-invoked cloud services. To find similar users for an active user, we need to calculate the similarities between the active user with all the m training users. There are totally m times of similarity computations. Therefore, the total computational complexity of similarity computation is $O(n^2m)$ when using the KRCC similarity measure.

After the identification of similar users, as introduced in Section 3.3.1, we need to obtain the preference values between different pairs of cloud services. There are totally $n(n-1)/2$ service pairs. For each pair, in the worst case, we need to get QoS values from the Top-K similar users for making preference value estimation. Since there are at most m similar users, the total computational complexity of preference value computation of an active user is $O(n^2m)$.

Based on the preference values, the CloudRank1 algorithm and CloudRank2 algorithm make QoS ranking prediction. As shown in Algorithm 1 and Algorithm 2, the computational complexities of CloudRank1 and CloudRank2 are both equal to $O(n^2)$.

Based on the above analysis, the total computational complexity of the whole procedure (including similarity computation, preference value computation, and ranking prediction) is $O(n^2m) + O(n^2m) + O(n^2) = O(n^2m)$.

4 Experiments

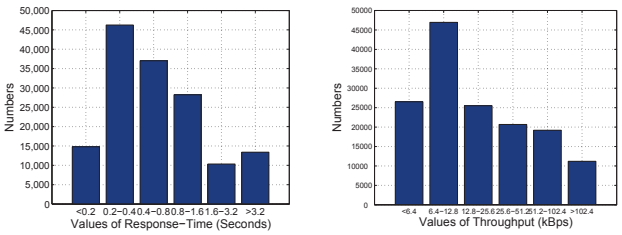
4.1 Dataset Description

To evaluate the QoS ranking prediction accuracy, we conduct a large-scale real-world Web service evaluation to collect QoS values on real-world Web services. We have collected addresses of 500 real-world Web services from the Internet. To collect QoS values of these Web services, firstly, we generated Web service invocation codes by Axis2[2], a Java-based open source package for Web services. Then, the invocation codes are deployed to 300 distributed computers in Planet-lab[3] to monitor these 500 real-world Web services. By this way, QoS values of the Web services can be obtained. In our experiment,

2. http://axis.apache.org/axis2
3. http://www.planet-lab.org

TABLE 1
Web Service QoS Dataset Descriptions

Statistics	Values
Num. of Web service invocations	150,000
Num. of service users	300
Num. of Web services	500
Minimum response-time value	0.005 s
Maximum response-time value	19.89 s
Mean of response-time	1.05 s
Standard deviation of response-time	2.14 s
Minimum throughput value	0.1 kBps
Maximum throughput value	1000 kBps
Mean of throughput	24.73 kBps
Standard deviation of throughput	40.84 kBps



(a) (b)

Fig. 4. **Value Distribution of User-Item Matrix**

each user invokes each Web service for one time. Totally 150,000 Web service invocations are conducted. The *response-time* and *throughput* values of each invocation are recorded. Response-time refers to the time duration between the user sending out a request to a service and receiving a response. Throughput represents the data transfer rate over the network. The detailed real-world QoS values are publicly released online[4], which makes our experimental evaluations reproducible.

The QoS values of the 500 Web services observed by the 300 service users can be presented as a 300×500 user-item matrix, where each entry in the matrix is the QoS value (e.g., response-time or throughput) of a Web service observed by a user. In the experiments, the QoS values of response-time and throughput are employed to rank the services independently. Table 1 shows descriptions of the obtained real-world Web service QoS values.

As shown in Table 1, the minimum and maximum values of response-time are 0.005 seconds and 19.89 seconds, respectively. The mean and standard deviation of all the 150,000 response-time values in the user-item matrix are 1.05 seconds and 2.14 seconds, respectively, indicating that the response-time values of different Web services observed by different users exhibit a great variation. The mean and standard deviation values of throughput are 24.73 kilo-Byte per second (kBps) and 40.84 kBps. The throughput values also exhibit a great variation. The distributions of the response-time and throughput values of the user-item matrix are shown in

4. http://www.zibinzheng.com/tpds2012

Figure 4(a) and Figure 4(b), respectively. Figure 4 shows that a large part of response-time values are between 0.2 seconds and 1.6 seconds, while most throughput values are between 6.4 kBps (kilo-Byte per second) and 102.4 kBps.

4.2 Evaluation Metric

Rating-oriented approaches must predict QoS values as accurate as possible. Therefore, differences between the predicted values and the true values are usually employed to evaluate the prediction accuracy. Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics are two widely adopted evaluation metrics for rating-oriented approaches. MAE is defined as:

$$MAE = \frac{\sum_{i,j} |r_{i,j} - \widehat{r}_{i,j}|}{N}, \tag{10}$$

and RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i,j} (r_{i,j} - \widehat{r}_{i,j})^2}{N}}, \tag{11}$$

where $r_{i,j}$ denotes the expected QoS value of service j observed by user i, $\widehat{r}_{i,j}$ is the predicted QoS value, and N is the number of predicted values.

However, since the object of this paper is to predict service QoS ranking instead of predicting QoS values, we employ the Normalized Discounted Cumulative Gain (NDCG) [2] metric, which is a popular metric for evaluating ranking results. Given an ideal service QoS ranking (used as ground truth) and a predicted QoS ranking, the NDCG value of the Top-K ranked services can be calculated by:

$$NDCG_k = \frac{DCG_k}{IDCG_k}, \tag{12}$$

where DCG_k and $IDCG_k$ are the discounted cumulative gain (DCG) values of the Top-K services of the predicted ranking and ideal ranking, respectively. The value of DCG_k can be calculated by:

$$DCG_k = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{log_2 i}, \tag{13}$$

where rel_i is the QoS value of the service at position i of the ranking. The premise of DCG is that high quality service appearing lower in a ranking list should be penalized as the QoS value is reduced logarithmically proportional to the position of the result via dividing by $log_2 i$. The DCG value is accumulated from the top of the ranking to the bottom with the gain of each result discounted at lower ranks. The ideal rank achieves the highest gain among different rankings. The $NDCG_k$ value is on the interval of 0 to 1, where larger value stands for better ranking accuracy, indicating that the predicted ranking is closer to the ideal ranking. The value of k is in the interval of 1 to n, where n is the total number of cloud services.

4.3 Performance Comparison

To study the personalized QoS ranking prediction performance, we compare nine methods. The first six approaches are rating-oriented methods, which rank the cloud services based on the predicted QoS values, while the last three methods are ranking-oriented approaches, which predict the QoS rankings directly.

- UVS (User-based collaborative filtering method using Vector Similarity): This method employs vector similarity for calculating the user similarities and engages the similar users for the QoS value prediction.
- IVS (Item-based collaborative filtering method using Vector Similarity): This method employs vector similarity for computing the item (cloud services) similarities when making QoS value prediction.
- UIVS (User-based and item-based collaborative filtering using Vector Similarity): This method combines the user-based and item-based collaborative filtering approaches and employs the vector similarity for the similarity computation for users and items.
- UPCC (User-based collaborative filtering method using Pearson Correlation Coefficient): This is a classical method. It employs PCC for calculating the user similarities and engages the similar users for the QoS value prediction [4].
- IPCC (Item-based collaborative filtering method using Pearson Correlation Coefficient): This method is widely used in industry company like Amazon. It employs PCC for the similarity computation and employs similar items (cloud services) for the QoS value prediction [15].
- UIPCC (User-based and item-based Collaborative filtering using Pearson Correlation Coefficient): This method combines the user-based and item-based collaborative filtering approaches and employs PCC for the similarity computation [23].
- Greedy: This method is proposed for ranking a set of items, which treats the explicitly rated items and the unrated items equally [6]. It does not guarantee that the explicitly rated items will be ranked correctly.
- CloudRank1: This method is proposed in Section 3.3.1. It calculates reference values between items and employs these values for making QoS ranking prediction.
- CloudRank2: This method is proposed in Section 3.3.2, which considers confidence levels of different preference values which help achieve better ranking accuracy. Both CloudRank1 and CloudRank2 employ KRCC (i.e., Eq. (2)) for the user similarity computation.

In real-world, the user-item matrices are usually very sparse since a user typically only employs a small number of cloud services. In order to conduct our experiments realistically, we randomly remove entries from the user-item matrix to make the matrix sparser with

TABLE 2
NDCG Comparison of Response Time (Larger value indicates better ranking accuracy)

	Matrix Density = 10%			Matrix Density = 30%			Matrix Density = 50%		
Methods	NDCG1	NDCG10	NDCG100	NDCG1	NDCG10	NDCG100	NDCG1	NDCG10	NDCG100
UVS	0.3653	0.3094	0.4940	0.5812	0.4828	0.6411	0.6777	0.6092	0.7389
IVS	0.3948	0.2964	0.4984	0.3318	0.3211	0.6548	0.3639	0.4323	0.7336
UIVS	**0.4214**	**0.3355**	0.5523	0.6116	0.5060	0.6948	0.6777	0.6110	0.7653
UPCC	0.3649	0.3139	0.5058	0.6043	0.4916	0.6606	0.6966	0.6306	0.7471
IPCC	0.3603	0.2882	0.4937	0.2906	0.2780	0.6067	0.3426	0.4352	0.7640
UIPCC	0.3868	0.3335	**0.5536**	**0.6240**	**0.5162**	**0.6964**	**0.6966**	**0.6332**	**0.7961**
Greedy	0.4400	0.4299	0.6874	0.7106	0.7072	0.8148	0.7923	0.7808	0.8734
CloudRank1	0.4703	0.4419	0.6992	0.7313	0.7240	0.8292	0.8230	0.8023	0.8849
CloudRank2	**0.4892**	**0.4465**	**0.7036**	**0.7346**	**0.7297**	**0.8314**	**0.8273**	**0.8170**	**0.8884**

TABLE 3
NDCG Performance Comparison of Throughput (Larger value indicates better ranking accuracy)

	Matrix Density = 10%			Matrix Density = 30%			Matrix Density = 50%		
Methods	NDCG1	NDCG10	NDCG100	NDCG1	NDCG10	NDCG100	NDCG1	NDCG10	NDCG100
UVS	0.3236	0.2262	0.4721	0.5106	0.4376	0.6278	0.6227	0.6058	0.7290
IVS	0.2890	0.1954	0.4379	0.1477	0.1432	0.5208	0.1666	0.1511	0.5900
UIVS	**0.3785**	0.2591	**0.5121**	**0.5315**	**0.4535**	**0.6525**	**0.6394**	**0.6229**	**0.7462**
UPCC	0.3390	0.2172	0.4631	0.5117	0.4326	0.6218	0.6103	0.5886	0.7271
IPCC	0.3135	0.2072	0.4323	0.1499	0.1317	0.4638	0.1363	0.1296	0.5335
UIPCC	0.3783	**0.2617**	0.5049	0.5117	0.4359	0.6363	0.6214	0.5911	0.7402
Greedy	0.5683	0.7039	0.7446	0.6068	0.7567	0.8103	0.6234	0.7939	0.8636
CloudRank1	0.5787	0.7126	0.7559	0.6536	0.7851	0.8356	0.7133	0.8417	0.8921
CloudRank2	**0.5834**	**0.7146**	**0.7567**	**0.6607**	**0.7895**	**0.8375**	**0.7204**	**0.8470**	**0.8943**

different densities. Matrix density (i.e., proportion of nonzero entries) 10%, for example, means that we randomly select 10% of the QoS entries to predict the QoS rankings of users. The rankings based on the original full matrix are employed as ideal rankings to study the ranking prediction accuracy. The above nine prediction methods are employed for making personalized QoS ranking prediction for each user in the user-item matrix employing the available QoS information in the matrix. We set Top-K to 5 in the prediction methods in this experiment.

Table 2 and Table 3 show the NDCG performance of response-time and throughput, respectively when employing 10%, 30% and 50% density user-item matrices. In the second row of the table, NDCG10 indicates that the ranking accuracy of the top 10 items is investigated. The value of NDCG10 can be calculated by Eq. (12). The first six methods in the table are rating-oriented methods, while the last three methods are ranking-oriented methods. For each column in the Tables, we highlight the best performer among the rating-oriented methods and the ranking-oriented methods.

Table 2 and Table 3 show that:

- Among all QoS ranking prediction methods, the CloudRank2 approach obtains the best prediction accuracy (largest NDCG values) for both response-time and throughput under all the experimental settings consistently, since CloudRank2 predicts the ranking directly and considers confidence levels of different preference values.
- Compared with the rating-oriented methods, the ranking-oriented methods achieve better prediction accuracy, since instead of predicting the QoS values as accurate as possible, the ranking-oriented methods attempt to directly predict the QoS rankings as accurate as possible.
- Compared with the Greedy approach, the CloudRank1 and CloudRank2 methods consistently achieve better ranking accuracy. We note that the Greedy method does not guarantee that the employed services will be ranked correctly. Our CloudRank1 and CloudRank2 algorithms ensure that the employed services are correctly ranked.
- When the density of the user-item matrix is increased from 10% to 50%, the ranking accuracy (in terms of NDCG values) is also enhanced, since denser user-item matrix provides more information for the ranking prediction.
- The approaches that combine user-based and item-based approaches (UIVS and UIPCC) outperform the user-based approaches (UVS and UPCC) and item-based approaches (IVS and IPCC) under most experimental settings. This observation indicates that by systematically fusing the information from similar users and similar items (cloud services), better QoS ranking prediction accuracy can be achieved.

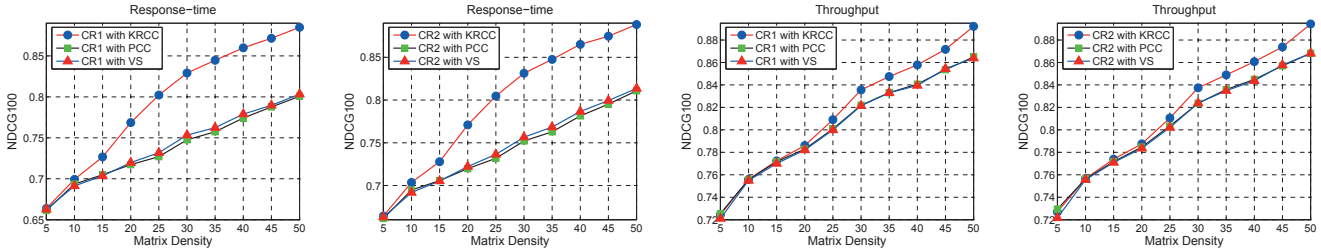


Fig. 5. **Impact of Similarity Computation**

4.4 Impact of Similarity Computation

There are different types of similarity computation methods. Rating similarity computation methods compare the QoS values of the commonly-invoked cloud services for the computation, while ranking similarity computation methods employ QoS rankings of services for calculating the similarities. Well-known rating similarity computation methods include VS and PCC, while well-known ranking similarity computation methods include KRCC.

To compare the performance of different similarity computation methods, we implement three versions of our CloudRank1 and CloudRank2 algorithms, using the KRCC, PCC, and VS similarity computation methods, respectively. We change the matrix density from 5% to 50% with a step value of 5%. We set Top-K to 5 in this experiment. CloudRank1 and CloudRank2 algorithms with different similarity computation methods are compared in this experiment.

Figure 5 shows the experimental results, where Figure 5(a) and Figure 5(b) are the NDCG100 results of response-time of CloudRank1 (labeled as CR1 in the Figure) and CloudRank2 (labeled as CR2 in the Figure), respectively. Figure 5(c) and Figure 5(d) are the NDCG100 results of throughput of CloudRank1 and CloudRank2, respectively. Figure 5 shows that:

- The ranking prediction accuracies of employing KRCC are better than PCC and VS in all the experimental settings, since KRCC computes the user similarity based on the QoS rankings instead of QoS values. In the ranking-oriented scenarios, KRCC provides better similarity computation accuracy.
- With the increase of matrix density, the improvements of *CR1 with KRCC* and *CR2 with KRCC* become greater compared with PCC and VS. When the matrix is sparse, the similarity computation methods do not have enough information for making accurate calculation. The advantages of KRCC is thus not obvious.
- The performance of PCC and VS is similar in this experiment, since these two similarity computation methods are similar with each other and are both rating-oriented.

Due to space limitation, more experimental investigations of impacts of parameters (e.g., matrix density, Top-K value, etc) are provided in Appendix B.

5 Related Work and Discussion

Cloud computing is becoming popular. A number of works have been carried out on cloud computing [8], [10], including performance analysis, market-oriented cloud computing, management tool, workload balance, dynamic selection, etc. Quality-of-Service (QoS) has been widely employed for presenting the non-functional characteristics of the software systems and services [19]. QoS of cloud services can be measured from either the client-side (e.g., response-time, throughput, etc.) or at the server-side (e.g., price, availability, etc.). Based on the service QoS measures, various approaches have been proposed for service selection [3], [19], [20], which enables optimal service to be identified from a set of functionally similar or equivalent candidates. To provide QoS ranking information for the service selection approaches, this paper focuses on predicting QoS ranking of cloud services.

Collaborative filtering methods are widely adopted in recommender systems [5], [15]. memory-based approach is one type of the most widely studied collaborative filtering approaches. The most analyzed examples of memory-based collaborative filtering include user-based approaches [4], [9], item-based approaches [7], [11], [16], and their fusion [13], [16], [17], [22], [23]. User-based and item-based approaches often use the vector similarity method [4] and the PCC method [15] as the similarity computation methods. Compared with vector similarity, PCC considers the differences in the user rating style when calculating the similarity.

The rating-based collaborative filtering approaches try to predict the missing QoS values in the user-item matrix as accurately as possible. However, in the ranking-oriented scenarios, accurate missing value prediction may not lead to accuracy ranking prediction. Therefore, ranking-oriented collaborative filtering approaches are becoming more attractive. Liu et al. [12] propose a ranking-oriented collaborative filtering approach to rank movies. Yang et al. [18] propose another ranking-oriented approach for ranking books in digital libraries. Different from these previous approaches [12], [18], our work provides a comprehensive study of how to provide accurate QoS ranking for cloud services, which is a new and urgently-required research problem.

Currently, our CloudRank framework is mainly de-

signed for cloud applications, because: (1) client-side QoS values of different users can be easily obtained in the cloud environment; and (2) there are a lot of redundant services abundantly available in the cloud, QoS ranking of candidate services becomes important when building cloud applications. The CloudRank framework can also be extended to other component-based applications, in case that the components are used by a number of users, and the past usage experiences of different users can be obtained.

6 Conclusion and Future Work

In this paper, we propose a personalized QoS ranking prediction framework for cloud services, which requires no additional service invocations when making QoS ranking. By taking advantage of the past usage experiences of other users, our ranking approach identifies and aggregates the preferences between pairs of services to produce a ranking of services. We propose two ranking prediction algorithms for computing the service ranking based on the cloud application designer's preferences. Experimental results show that our approaches outperform existing rating-based approaches and the traditional greedy method.

For future work, we would like to improve the ranking accuracy of our approaches by exploiting additional techniques (e.g., data smoothing, random walk, matrix factorization, utilizing content information, etc.). When a user has multiple invocations of a cloud service at different time, we will explore time-aware QoS ranking prediction approaches for cloud services by employing information of service users, cloud services, and time. As our current approaches only rank different QoS properties independently, we will conduct more investigations on the correlations and combinations of different QoS properties. We will also investigate the combination of rating-based approaches and ranking-based approaches, so that the users can obtain QoS ranking prediction as well as detailed QoS value prediction. Moreover, we will study how to detect and exclude malicious QoS values provided by users.

Acknowledgment

The work described in this paper was fully supported by the National Basic Research Program of China (973 Project No. 2011CB302603), the National Natural Science Foundation of China (Project No. 61100078, 61073132), and the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 415311 of General Research Fund and Project No. N_CUHK405/11 of the NSFC/RGC Joint Research Scheme).

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," *Technical Report, EECS-2009-28*, University of California, Berkeley, 2009.

[2] K. J. arvelin and J. Kekalainen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422–446, 2002.

[3] P. A. Bonatti and P. Festa, "On optimal service selection," in *Proc. 14th Int'l Conf. World Wide Web (WWW'05)*, 2005, pp. 530–538.

[4] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. 14th Annual Conf. Uncertainty in Artificial Intelligence (UAI'98)*, 1998, pp. 43–52.

[5] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[6] W. W. Cohen, R. E. Schapire, and Y. Singer, "Learning to order things," *Journal of Artificial Intelligent Research*, vol. 10, no. 1, pp. 243–270, 1999.

[7] M. Deshpande and G. Karypis, "Item-based top-n recommendation," *ACM Trans. Information System*, vol. 22, no. 1, pp. 143–177, 2004.

[8] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Trans. Parallel Distributed System*, vol. 22, pp. 931–945, June 2011.

[9] R. Jin, J. Y. Chai, and L. Si, "An automatic weighting scheme for collaborative filtering," in *Proc. 27th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'04)*, 2004, pp. 337–344.

[10] H. Khazaei, J. Misic, and V. B. Misic, "Performance analysis of cloud computing centers using m/g/m/m+r queuing systems," *IEEE Trans. Parallel Distributed System*, vol. 23, no. 5, pp. 936–943, May 2012.

[11] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[12] N. N. Liu and Q. Yang, "Eigenrank: a ranking-oriented approach to collaborative filtering," in *Proc. 31st Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'08)*, 2008, pp. 83–90.

[13] H. Ma, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *Proc. 30th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'07)*, 2007, pp. 39–46.

[14] J. Marden, *Analyzing and Modeling Ranking Data*. Chapman & Hall, New York, 1995.

[15] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Proc. of ACM Conf. Computer Supported Cooperative Work*, 1994, pp. 175–186.

[16] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int'l Conf. World Wide Web (WWW'01)*, 2001, pp. 285–295.

[17] J. Wu, L. Chen, Y. Feng, Z. Zheng, M. Zhou, and Z. Wu, "Predicting qos for service selection by neighborhood-based collaborative filtering," *IEEE Transactions on System, Man, and Cybernetics, Part A*, to appear.

[18] C. Yang, B. Wei, J. Wu, Y. Zhang, and L. Zhang, "Cares: a ranking-oriented cadal recommender system," in *Proc. 9th ACM/IEEE-CS joint conference on Digital libraries (JCDL'09)*, 2009, pp. 203–212.

[19] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for Web services selection with end-to-end QoS constraints," *ACM Trans. the Web*, vol. 1, no. 1, pp. 1–26, 2007.

[20] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for Web services composition," *IEEE Trans. Software Engineering*, vol. 30, no. 5, pp. 311–327, 2004.

[21] Z. Zheng and M. R. Lyu, "WS-DREAM: A distributed reliability assessment mechanism for Web services," in *Proc. 38th Int'l Conf. Dependable Systems and Networks (DSN'08)*, 2008, pp. 392–397.

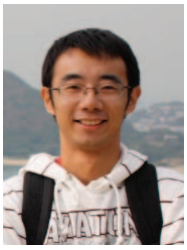
[22] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "WSRec: A collaborative filtering based Web service recommender system," in *Proc. 7th Int'l Conf. Web Services (ICWS'09)*, 2009, pp. 437–444.

[23] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-aware Web service recommendation by collaborative filtering," *IEEE Transactions on Service Computing*, vol. 4, no. 2, pp. 140–152, 2011.

[24] Z. Zheng, Y. Zhang, and M. R. Lyu, "CloudRank: A QoS-driven component ranking framework for cloud computing," in *Proc. Int'l Symposium Reliable Distributed Systems (SRDS'10)*, 2010, pp. 184–193.

Zibin Zheng is an associate research fellow at the Shenzhen research institute, The Chinese University of Hong Kong. He received his Ph.D. degree from department of Computer Science and Engineering, The Chinese University of Hong Kong in 2010. He received Outstanding Thesis Award of CUHK at 2012, ACM SIGSOFT Distinguished Paper Award at ICSE2010, Best Student Paper Award at ICWS2010, and IBM Ph.D. Fellowship Award 2010-2011. He served as program committee member of IEEE CLOUD2009, SCC2011, SCC2012, ICSOC2012, etc. His research interests include cloud computing, service computing, and software engineering.

Xinmiao Wu is currently a master student at Sun Yat-Sen University, GuangZhou, China. He received his B.Eng. degree in Network Engineering from Sun Yat-Sen University, China, in 2010. He received National Scholarship of China in 2007 and 2008, respectively. His research interests include services computing, cloud computing, and data mining.



Yilei Zhang received his B.Sc. degree in Computer Science from the University of Science and Technology of China, Hefei, China, in 2009. He is currently a Ph.D. candidate in the department of Computer Science and Engineering, The Chinese University of Hong Kong. He is an IEEE student member, an ACM student member, and a Hong Kong Computer Society student member. He also served as reviewer for international journals as well as conferences including TSE, TSC, ISF, WWW, WSDM, KDD, SCC, etc. His research interests include service computing, cloud computing, distributed systems, software reliability engineering.

Michael R. Lyu received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1981; the M.S. degree in computer engineering from University of California, Santa Barbara, in 1985; and the Ph.D. degree in computer science from the University of California, Los Angeles, in 1988. He is currently a Professor in the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, China. His research interests include software reliability engineering, distributed systems, fault-tolerant computing, mobile networks, Web technologies, multimedia information processing, and E-commerce systems. Dr. Lyu is an IEEE Fellow, an AAAS Fellow, and a Croucher Senior Research Fellow for his contributions to software reliability engineering and software fault tolerance. He received IEEE Reliability Society 2010 Engineering of the Year Award.

Jianmin Wang graduated with B.S in Computational Mathematics in 1996 from Nankai University, M.S. and Ph.D from Sun Yat-sen University in 1999 and 2003. With more than 30 scientific publications and 40 patents application, he is a staff of the Institute of computer application, and fellow of Engineering Research Center of Digital Life, Ministry of Education of China. He has won a number of awards on research papers and scholarships including National Science and Technical Progress Award of China(second level), and three times Science and Technical Progress Award of Ministry of Education of China(first level). His research interests cover network analysis, embedded media technology, Interaction design. Dr. Wang's teaching specialism are in embedded HCI, software architecture design, ubiquitous computing.