Personalized Quality Prediction for Dynamic Service Management based on Invocation Patterns

Li Zhang[1,2], Bin Zhang[1], Claus Pahl[2], Lei Xu[2], Zhiliang Zhu[1]

[1]Northeastern University, Shenyang, China
{zhangl,zhuzl}@swc.neu.edu.cn, zhangbin@ise.neu.edu.cn
[2]Dublin City University, Dublin, Ireland
{cpahl,lxu}@computing.dcu.ie

Abstract. Recent service management needs, e.g., in the cloud, require services to be managed dynamically. Services might need to be selected or replaced at runtime. For services with similar functionality, one approach is to identify the most suitable services for a user based on an evaluation of the quality (QoS) of these services. In environments like the cloud, further personalisation is also paramount. We propose a personalized QoS prediction method, which considers the impact of the network, server environment and user input. It analyses previous user behaviour and extracts invocation patterns from monitored QoS data through pattern mining to predict QoS based on invocation QoS patterns and user invocation features. Experimental results show that the proposed method can significantly improve the accuracy of the QoS prediction.

Keywords: Service Quality, Web and Cloud Services, QoS Prediction, Invocation Pattern Mining, Collaborative Filtering, Personalized Recommendation.

1 Introduction

Service QoS (Quality of Service) is the basis of Web and Cloud service discovery [1-4], selection [5,6] and composition [7-9]. For services located in open environments such as the Web or the Cloud, QoS may vary depending on the network, the service execution environment and user requirements. Additionally, a *personalized QoS evaluation* of services for different service users is necessary in particular in these open environments, as users more and more expect the customisation of publically provided services used by them. Generally, service QoS information is derived in three ways: delivered by services providers, evaluated based on user feedback and predicted based on monitoring information. Prediction based on monitoring is more objective and reliable in untrusted Web and Cloud contexts and more suitable for these *dynamically changing environments*. There are two types of prediction based on monitoring: one is based on statistical, the other is personalized prediction. Many implementations [1,5,7,9,10] adopt the statistical approach for usage (QoS) prediction. The statistical method is simple and easy to implement, e.g., response-time is usually calculated based on the average response time. This method ignores the users' personalized requirements, network conditions and execution features. For example,

for an on-demand cloud-based movie/video processing service, the size of the video has a significant influence on the response time. Different users, accessing the service through the cloud, may experience different response times. In [11,12,13], collaborative filtering methods are proposed, predicting QoS for a user by referring to past information of similar users. The influence of the user environment and input can also be considered to provide a user with a more personalized QoS prediction.

Moreover, even the same user does not experience the same QoS values for different invocations at different times and with different invocation parameters – which is something that current cloud services, whether multimedia on-demand for end users or commercial applications in the cloud, highlight as a problem. If a user invokes a service many times, then the QoS cannot be determined by collaborative filtering, which is inefficient for QoS prediction for every invocation. A Bayesian network-based QoS assessment model for web services is proposed in [13]. It predicts the service performance level depending on the user requirements level, but how to define the performance level is still a problem. Most QoS prediction methods of services do not consider the impact on service performance by environmental factors. The *prediction performance* becomes a critical aspect in dynamically managed service and cloud environments, where monitored QoS data is taken into account.

Our experiments and analyses show that user inputs, network conditions and Web server performance impact on QoS significantly. Assume three services s_1, s_2 and s_3 with similar functions. Take input data size, network throughput and CPU utilization as representatives of input, network and Web server characteristics. Table 1 shows an invocation log of these services. It records information for every invocation: network throughput (MB), data size (MB), Web server CPU utilization and response-time(s).

Table 1. Services Usage Information

Service name	1st invocation	2nd invocation	3rd invocation	4th invocation	5th invocation
s_1	<2, 10, 0.2, 0.5>	<1.5,20, 0.5, 2>	<2.5,10,0.1,0.2>	<2,30, 0.3, 0.8>	<2, 8,0.2, ?>
s_2	<1.5,10,0.3, 0.3>	<2,20, 0.4, 1.8>			
s_3	<2, 20, 0.3, 3 >	<1, 20, 0.2, 6 >	<1.5,20, 0.3, 4 >	<2,15,0.2, 2.4 >	

In the third invocation of s_1, network throughput is 2.5MB, data size is 10MB, Web server CPU utilization is 0.1 and response time is 0.2sec. Now, if there is another user wanting to invoke s_1, the network throughput is 2MB, data size is 8MB and CPU utilization is 0.2. Then, predicting the response time for this user depends on history information (cf. Table 1). The three services were invoked 5 times, 2 times or 4 times. The average response times are 0.875s, 1.05s and 3.85s. This is independent of invocation parameters. No matter what the situation of the next invocation, the traditional prediction results will be the same, but according to Table 1, the real result is dependent on input, network and Web server factors. The prediction in our previous work [15,16] is based on collaborative filtering. It predicts QoS through calculating the similarity of invocation parameters and parameters in past invocations. The prediction is more accurate than an averaging method, but needs to calculate the similarity of

target invocation and all past invocations, resulting in too many repeated calculations and low efficiency, which needs to be addressed for dynamic contexts like the cloud.

An important observation is that most services have relatively fixed *service invocation patterns* (SIPs). A SIP consists of ranges of input characteristics, network characteristics and Web server characteristics and reflects relatively stable, acceptable variations. The service QoS keeps steady under a SIP. If we can abstract the SIP from service usage, the prediction can be based on usage information for the matched pattern. If there is no usage information for the matched pattern, the prediction needs to be calculated using past log information of other similar services. We propose constructing SIPs by analysing user input, network environment and server status factors. We *adapt collaborative filtering prediction to be based on SIPs and pattern mining*, improving prediction accuracy and performance. Thus, our contributions are:

- Firstly, we propose the novel concept of Service Invocation Pattern and an aligned method for *mining and constructing SIPs* (Sections 2 and 3). It considers the influence of environmental characteristics on the quality of a Web service.
- Secondly, we propose a collaborative filtering QoS prediction algorithm based on SIPs (Sections 4 and 5). This approach can *predict QoS based on personalized user requirements*. It improves the prediction accuracy and computational performance.

2 Service Invocation Pattern SIP

Services QoS characteristics are related to user input, network status and server performance. It means that a certain range of input, network status and server status determines a relatively fixed service invocation pattern. A SIP reflects that the QoS remains steady under this pattern, i.e., predicting QoS this way is beneficial. We analyse the characteristics which impact Web service execution and define the SIP.

Definition. 1. *Service Invocation Characteristic* (SIC). $\boldsymbol{C}$ = < ***Input***, ***Network***, ***Server***> is the characteristic model of one invocation. ***Input***, ***Network*** and ***Server*** represent user input characteristics, network characteristics and Web server characteristics, respectively. We take input data size, network throughput and CPU utilization as examples. In invocation characteristic *<30, 1.5, 0.2>*, the input data size is 30MB, throughput between server and user is 1.5MB and server CPU utilization is 0.2.

Definition. 2. *Input Characteristic* (IC). ***Input*** $=< In^1, In^2, \ldots, In^P >$ is the input characteristics vector. It describes the input characteristics that have an influence on QoS. I^k $(1 \leq k \leq p)$ is the *k-th* input characteristic.

Definition. 3. *Network Characteristic* (NC). ***Network*** $=< net^1, net^2, \ldots, net^r >$ is the network characteristics vector. It describes the network characteristics that have an influence on QoS. n^k $(1 <= k \leq r)$ represents the *k-th* network characteristic.

Definition. 4. *Web Server Characteristic* (WSC). ***Server*** $=< se^1, se^2, \ldots, se^q >$ is the server characteristics vector. It describes the Web server characteristics that have an influence on service QoS. se^k $(1 \leq k \leq q)$ represents the *k-th* server characteristic.

Definition. 5. *Service Invocation Pattern* (SIP). A SIP is a group of service invocation characteristics SIC. In a SIP, the value of invocation characteristics is a range.

The QoS is meant to be steady under a SIP. We describe it as $\boldsymbol{M}$ = < $\boldsymbol{Input_{low}} \sim \boldsymbol{Input_{high}}$, $\boldsymbol{Network_{low}} \sim \boldsymbol{Network_{high}}$, $\boldsymbol{Server_{low}} \sim \boldsymbol{Server_{high}}$ >, ***Input***, ***Network*** and ***Server*** are input characteristics, network characteristics and Web server characteristics.

Definition. 6. *Invocation Pattern-QoS matrix*. If the QoS of services keeps steady or have a fixed relation to a SIP, then this relation can be expressed as a matrix ***MS***:

$$\boldsymbol{MS} = \begin{array}{c} \\ \boldsymbol{M_1} \\ \boldsymbol{M_2} \\ \cdots \\ \boldsymbol{M_l} \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & \cdots & s_m \end{array} \\ \begin{bmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,m} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,m} \\ \cdots & \cdots & \cdots & \cdots \\ q_{l,1} & q_{l,2} & \cdots & q_{l,m} \end{bmatrix} \end{array}$$

The matrix ***MS*** shows the QoS information of all the services *s* under all the patterns ***M***. q_{ij} ($1 \le j \le l$, $1 \le i \le m$) is the QoS of service s_j under the pattern $\boldsymbol{M_i}$.

with

$$q_{i,j} = \begin{cases} \phi & \text{Service } s_j \text{ has no invocation history under pattern } \boldsymbol{m}_i . \\ low_{i,j} \sim high_{i,j} & \text{Service } s_j \text{ has invocation history under pattern } \boldsymbol{m}_i \text{ with range '}\sim\text{'}. \end{cases}$$

If a pattern is <20-30*MB*, 0.5-0.6, 0.2-0.4, 30-40*MB*>, then the input data size is 20-30*MB*, CPU utilization is 0.5-0.6, memory utilization is 0.2-0.4 and server throughput is 30-40*MB*. We can search for the QoS of a service based on information related to this pattern. If there is corresponding information and the value keeps steady in a range, then it is returned to the user. If the value is not consecutive, it means the service is not only affected by the characteristics of the invocation pattern. It then needs further calculation based on history information. If there is no invocation history, this value will be null. In that case, prediction is done for a user invocation requirement. Below is an example of an Invocation Pattern-QoS Matrix. There are 4 invocation patterns. We introduce how to abstract/mine SIPs and predict in Sect. 3.

$$\begin{array}{c} \\ \boldsymbol{M_1} \\ \boldsymbol{M_2} \\ \boldsymbol{M_3} \\ \boldsymbol{M_4} \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & s_3 & s_4 \end{array} \\ \begin{bmatrix} 0.2 \sim 0.5s & & 1 \sim 1.3s & \\ 0.8 \sim 1.1s & 1.1 \sim 1.5s & & \\ 0.4 \sim 0.5s & & 2 \sim 2.4s & 0.3 \sim 0.5s \\ & 3 \sim 4s & & \end{bmatrix} \end{array}$$

Some services have no usage information within a pattern range – e.g., since s1 has an invocation history for pattern M1, it returns this range of values, but as s2 has no invocation history, it needs collaborative filtering for prediction.

3 Service Invocation Pattern Abstraction and Mining

The values of user invocation characteristics are spread across a certain range. Obtaining these value ranges significantly helps QoS prediction, but the number of SIPs that reflect these cannot be decided in advance and all usage information is multi-dimensional. Density-based spatial clustering of applications is used to achieve this. DBSCAN (density-based spatial clustering of application with noise) [17] is a densi-

ty-based clustering algorithm. It analyses the density of data and allocates them into a cluster if the spatial density is greater than a threshold. DBSCAN can find clusters of any shape. The DBSCAN algorithm has two parameters: ε and *MinPts*. If the distance between two points is less than the threshold ε, they can be in the same cluster. The minimum number of points in a cluster must be greater than *MinPts*. DBSCAN clusters the points through spatial density. The main steps of DBSCAN:

1. Select any object *p* from the object set and find the objects set *D* in which the object is density-reachable from object *p* with respect to ε and *MinPts*.
2. Choose another object without cluster and repeat the first step.

A SIP Extraction Algorithm based on DBSCAN shall now be introduced. A SIP is composed of user input, network and server characteristics. For these aspects, we take throughput, input size and CPU utilization as representatives, respectively. We consider the execution time as the representative of QoS here.

- An execution log records the input data size and execution QoS.
- A monitoring log records the network status and Web server status.

We reorganize these two files to find the SIP under which QoS keeps steady. A SIP extraction algorithm is shown in Alg. 1 (see also the SIP format in Definition 5).

Algorithm 1: SIP Extraction Algorithm based on DBSCAN
Input: Service Usage Information ***InforSet*** (execution+monitoring log), ε, *MinPts*.
Output: SIP Database ***PatternBase***, Pattern-QoS information ***PatternQoS***.

```
1 for ( Infor_i<DataSize, CPU, ThroughPut, time> ∈ InforSet )
2 {
3   if ( Infor_i does not belong to any exist cluster ) {
4     P_j= newPattern(Infor_i) // create a new pattern withInfor_i as seed.
5     Add( P_j, PatternBase )
6     InforSet = InforSet − Infor_i
7     SimInfor = SimilarInfor(InforSet, Infor_i, ε) // SimInfor is the infor-
8          mation set which includes all the similar usage information of
9          Infor_i. Differences between the information in SimInfor and
10         Infor_i on the charac-teristics value except execution time are
11         less than ε. n is the number of information items in SimInfor.
12    InforSet = InforSet − SimInfor
13    if ( n>MinPts ) {  // MinPts is min number of exec inform in cluster.
14      (S_1, S_2, … ,S_m) = Divide(SimInfor) // Divide SimInfor into different
15         groups. Group S_1 includes all information of servs_1.
16      for(k=1; k≤m; k++){
17        for(Infor_j∈S_k) {
18          SimInfor = SimilarInfor(InforSet, Infor_j, time, MinPts, ε)
19           // Search similar info of S_k in execution information set. If
20             the number of similar information item is less than MinPts,
21             then the density will turn low and  top the  loop.
22          S_k = S_k + SimInfor
23          InforSet = InforSet − SimInfor
24        }
25         PatternCharacteristics(S_k)  // Organizes the information in the
26              cluster and statistics for the ranges of characteristics.
27              Completes the pattern-QoS matrix.
28      }
29    }
30  }
31}
```

The distance calculation between two objects in this algorithm is different from the traditional DBSCAN. It includes two types of distance:

- Firstly, when we initialize a cluster, we randomly select an object without cluster. We take it as the seed to find the cluster it belongs to. In this cluster, the response time of different services may differ, but the performance of different invocations of the same service keeps steady. The distance between the other information and seed information is computed based on all characteristics except response time.
- Secondly, when the cluster has been constructed, we need to check whether the information does not belong to any cluster or belongs to the given cluster. We need to compare this information with others of the same service in the cluster and calculate the distance of this information with the cluster. Then, the distance computation is dependent on all the characteristics of the two information items.

4 The QoS Prediction based on SIP

This section will introduce the Web Service QoS prediction approach. It uses Service Invocation Patterns and the Invocation Pattern-QoS Matrix to carry out the prediction. It fully considers the requirements of every invocation.

4.1 The QoS Prediction Procedure based on SIP

In order carry out the prediction, we assume that the SIP database has been created.

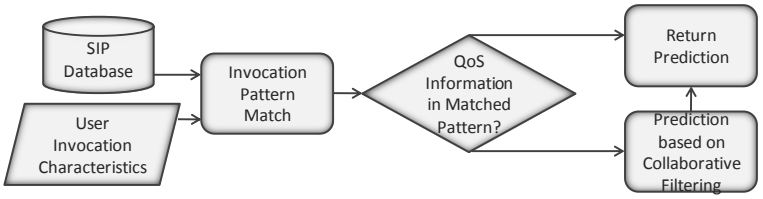

Fig. 1. QoS Prediction Procedure

The steps of the prediction procedure in Fig. 1 are as follows. Firstly, we match the target invocation characteristics with the SIPs in the database. We match the characteristics of target service s_j with the characteristics of stored patterns. If there is a pattern that can be matched directly, then we return it. Otherwise, we employ the Gray Relevance Analysis to get the matched pattern. Assume that the matched pattern is $\boldsymbol{m_i}$. Then, we search information about a matched pattern in the QoS matrix. If there is QoS information of the target service in pattern $\boldsymbol{m_i}$, then we return it directly. Finally, if there is no related QoS information, then we predict QoS by collaborative filtering.

4.2 Matching User Invocation Characteristics with Patterns

A characteristics vector of a user invocation is $\boldsymbol{C}$ = < ***Input***, ***Network***, ***Server***>. Here *Input*, *Network* and *Server* represent input, network and server characteristics,

respectively. The Service Invocation Pattern is defined as $M = < Input_{low}\sim Input_{high}, Network_{low}\sim Network_{high}, Server_{low}\sim Server_{high} >$. During the matching process, we compare the user invocation characteristics and the respective component in the pattern. Matching is successful if $Input_{low} \le Input \le Input_{high}$, $Network_{low} \le Network \le Network_{high}$ and $Server_{low} \le Server \le Server_{high}$. Assume the matched pattern is m_i.

If there is no matched pattern, adopt the *Gray Relevance Analysis* method to calculate the association degree between QoS and invocation characteristics to *a)* find the ordering of characteristics that have greater impact on QoS and *b)* match the pattern based on the order. Table 2 shows the *n* times invocation information of service *s*.

Table 2. Usage Information of Service *s*

Features	1	……	*I*	……	*n*
Response Time	T_1	……	T_i	……	T_n
Input Datasize	$Data_1$	……	$Data_i$	……	$Data_n$
Throughput	TP_1	……	TP_i	……	TP_n
CPU utilization	CPU_1	……	CPU_i	……	CPU_n

1. Take response time as the reference sequence $x_0(k)$, $k = 1,\ldots, n$, and other characteristics as comparative sequences. Calculate the association degree of the other characteristics with response time. First, take the characteristics of an invocation as standard and carry out normalization of the other characteristics. The reference sequence and comparative sequence are handled dimensionless. Assuming the standardized sequence $y_i(k)$, $i=1,\ldots,4$, $k=1,\ldots,n$, Table 3 shows the result matrix.

Table 3. Normalized Usage Informaiton

Features	1	……	*I*	……	*n*
Response Time	1	……	$y_1(i)$	……	$y_1(n)$
Input Datasize	1	……	$y_2(i)$	……	$y_2(n)$
Throughput	1	……	$y_3(i)$	……	$y_3(n)$
CPU utilization	1	……	$y_4(i)$	……	$y_4(n)$

2. Calculate absolute differences for Table 3 using $\Delta_{0i}(k) = |y_0(k) - y_i(k)|$. The resulting absolute difference sequence is:

$$\Delta_{01} = (0, y_{01}(1), \cdots, y_{01}(n)),\ \Delta_{02} = (0, y_{02}(1), \cdots, y_{02}(n)),\ \Delta_{03} = (0, y_{03}(1), \cdots, y_{03}(n))$$

3. Calculate a correlation coefficient between reference and comparative sequence:

$\zeta_{0i}(k) = \dfrac{\Delta_{\min} + \rho\Delta_{\max}}{\Delta_{0i}(k) + \rho\Delta_{\max}}$ is the correlation coefficient of the Gray Relevance.

Here $\Delta_{0i}(k) = | y_0(k) - y_i(k) |$ is the absolute difference and $\Delta_{\min} = \min_i \min_k \Delta_{0i}(k)$ is the minimum difference value between two poles, and $\Delta_{\max} = \max_i \max_k \Delta_{0i}(k)$ is the maximum difference value. $\rho \in (0,1)$ is the distinguishing factor.

4. Calculate the correlation degree: Use $r_{0i} = \frac{1}{n}\sum_{k=1}^{n}\zeta_{01}(k)$ to calculate the correlation degree between characteristics. Then, sort the characteristics based on the correlation degree. If r_0 is the largest, it has the greatest impact on response time and will be matched prior to others. Assume usage information of s as in Table 4.

5 QoS Prediction Based on Collaborative Filtering

If there is no related QoS within matched patterns, we need to predict QoS based on collaborative filtering. In the Invocation Pattern-QoS Matrix, there are a usually a number of null values. The prediction accuracy will be affected if we ignore these null values. We need to fill the null values for the information items of similar services.

5.1 QoS Prediction Process based on Collaborative Filtering

Assume that the target service is s_j, and the matched pattern is $\boldsymbol{m_i}$. When service s_j has no QoS information in pattern $\boldsymbol{m_i}$, the prediction process is as follows:

1. For any service s_v, $v \neq j$, if there is information of s_v under pattern $\boldsymbol{m_i}$. then calculate the similarity between service s_j and service s_v.
2. Get the k neighbouring services of service s_j through the similarity calculated in step 1. The set of these k services is $S = \{s_1', s_2', \cdots, s_k'\}$. We fill the null QoS values for the target invocation using the information in this set.
3. Using the information in S, calculate the similarity of $\boldsymbol{m_i}$ with other patterns that have the information for target service s_j.
4. Choose the most similar k' patterns of $\boldsymbol{m_i}$, and use the information across the k' patterns and S to predict the QoS of service s_j .

5.2 Service Similarity Computation

Assume that $\boldsymbol{m_i}$ is the matched pattern and s_j is the target service. If there is no information of s_j in pattern $\boldsymbol{m_i}$, we need to predict the response time $q_{i,j}$ for s_j. Firstly, calculate the similarity of s_j and services which have information within pattern $\boldsymbol{m_i}$ ranges. For a service $s_v \in I_i$ where I_i is the set of services that have usage information within pattern $\boldsymbol{m_i}$, calculate the similarity of s_j and s_v. Vector similarity calculation commonly adopts cosine similarity, correlation similarity or correction cosine similarity. However, these 3 methods do not consider the impact of user environment differences, i.e., the methods are not suited for service similarity computation directly. We need to improve the similarity calculation. We define service similarity as follows:

Definition. 7. The similarity of two services s_j and s_v is defined by

$$sim(s_v, s_j) = \alpha \cdot sim_{sum}(s_v, s_j) + \beta \cdot sim_{data}(s_v, s_j) \quad (1)$$

where

- $sim_{sum}(s_v,s_j)$ is the similarity of the numbers of invocation patterns which are invoked by services s_v and s_j together. Two services are more similar if they have more used invocation patterns in common.
- $sim_{data}(s_v,s_j)$ is the similarity of the usage information of services s_j and s_v. Two services are more similar if their usage information is more similar.
- α and β are adjustable balance parameters. They can be changed based on different user requirements.

For services s_j and s_v, $P(s_j/s_v)$ is the probability of the coexistence of services s_j and s_v within a pattern. This probability can be used to measure the similarity of s_j and s_v:

$$sim_{sum}(s_v, s_j) = \frac{num(s_v s_j)}{num(s_j)} \tag{2}$$

Here, $num(s_v, s_j)$ is the number of the common pattern-based invocations by two services. $num(s_j)$ is the number of pattern-based invocation by service s_j. Based on formula (1), $sim_{sum}(s_v, s_j)$ is between 0 and 1.

Our definition of the similarity of invocation information adopts the correction cosine similarity method. It is shown in formula (3). M_{vj} is the set of invocation pattern models which have the usage information of s_v and s_j.

$$sim_{data}(s_v, s_j) = \frac{\sum_{\boldsymbol{m}_c \in M_{vj}} (q_{c,v} - \overline{q_v})(q_{c,j} - \overline{q_j})}{\sqrt{\sum_{\boldsymbol{m}_c \in Mvj} (q_{c,v} - \overline{q_v})^2} \sqrt{\sum_{\boldsymbol{m}_c \in Mvj} (q_{c,j} - \overline{q_j})^2}} \tag{3}$$

Here, $\overline{q_v}$ is the average of the usage information for service s_v, $\overline{q_j}$ is the average of the usage information for service s_j.

From formula (3), we can obtain all similarities between s_j and others services which have usage information within pattern $\boldsymbol{m_i}$. The more similar the service is to s_j, the more valuable the data of it is. Formulas (2) and (3) are two aspects of service similarity. Formula (1) provides the sum of these two different similarities.

5.3 Predicting Missing Data

Missing data will have a negative impact on the accuracy of QoS prediction. We calculate the similarity between two services and get the k neighbouring services. Then, we establish the k neighbours matrix $\boldsymbol{T_{sim}}$ and fill the missing data in $\boldsymbol{T_{sim}}$.

Assume the k neighbouring services form the set $S = \{s_1', s_2', \cdots, s_k'\}$. Here, s_1' has the highest similarity with service s_j and so on. Then, these k services are more valuable and their usage information is defined as follows in matrix (4) below. Matrix $\boldsymbol{T_{sim}}$ shows the usage information of the k neighbouring services of s_j within all invocation patterns. The data space is reduced to k columns and the computational effort required is consequently also reduced. In this matrix, there are still many missing data items $t_{i,j}$. We need to fill these empty spaces before prediction. Firstly, we fill the missing data references to the services similarity.

$$\boldsymbol{T}_{sim} = \begin{array}{c} \\ \boldsymbol{M_1} \\ \cdots \\ \boldsymbol{M_i} \\ \cdots \\ \boldsymbol{M_l} \end{array} \begin{array}{c} \begin{array}{ccccc} s_j & s_1^{'} & s_2^{'} & \cdots & s_k^{'} \end{array} \\ \begin{bmatrix} t_{1,j} & t_{1,1}^{'} & t_{1,2}^{'} & \cdots & t_{1,k}^{'} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \Phi & t_{i,1}^{'} & t_{i,2}^{'} & \cdots & t_{i,k}^{'} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ t_{l,j} & t_{l,1}^{'} & t_{l,2}^{'} & \cdots & t_{l,k}^{'} \end{bmatrix} \end{array} \tag{4}$$

We fill $P_{i,p}^{ser}$, which is the data of service s_p under pattern m_i. The method is:

$$P_{i,p}^{ser} = \bar{t}_p^{'} + \frac{\sum_{n \in S'} sim_{n,p} \times (t_{i,n}^{'} - \bar{t}_n^{'})}{\sum_{n \in S'} (|\, sim_{n,p}\,|)}, \tag{5}$$

Here $\bar{t}_p^{'}$ is the average QoS of service s_p, and $sim_{n,p}$ is the similarity between service s_n and s_p. For any service $p \in S'$, every service has usage information within all the pattern ranges in m_i after this process.

5.4 Calculating the Pattern Similarity and Prediction

There is QoS information of k neighbouring services of s_j in matrix $\boldsymbol{T}_{sim}$. Some of them are prediction values. We can calculate the similarity of pattern $\boldsymbol{m_i}$ and other patterns using the correction cosine similarity method:

$$sim_{\mathrm{model}}(m_i, m_j) = \frac{\sum_{s_k \in S} (t_{i,k}^{'} - \overline{t_i^{'}})(t_{j,k}^{'} - \overline{t_j^{'}})}{\sqrt{\sum_{s_k \in S} (t_{i,k}^{'} - \overline{t_i^{'}})^2} \sqrt{\sum_{s_k \in S} (t_{j,k}^{'} - \overline{t_j^{'}})^2}}, \tag{6}$$

After determining the pattern similarity, the data of patterns with low similarity are removed from $\boldsymbol{T}_{sim}$. The set of the first k patterns is $\mathbb{M}' = \{\boldsymbol{M}_1^{'}, \boldsymbol{M}_2^{'}, \cdots \boldsymbol{M}_k^{'}\}$. The data of these patterns are retained for prediction.

As described above, if $p_{i,j}$ is the data to be predicted as the usage data of service s_j within pattern $\boldsymbol{M_i}$, it is calculated as:

$$p_{i,j} = \bar{t}_i^{'} + \frac{\sum_{\boldsymbol{n} \in \boldsymbol{M}'} sim_{n,i} \times (t_{n,j}^{'} - \bar{t}_n^{'})}{\sum_{\boldsymbol{n} \in \boldsymbol{M}'} (|\, sim_{n,i}\,|)} \tag{7}$$

Here $\bar{t}_i^{'}$ is the average QoS of the data related to pattern $\boldsymbol{m_i}$ and $sim_{n,i}$ is the similarity between patterns $\boldsymbol{M_n}$ and $\boldsymbol{M_p}$.

6 Experimental Analysis

We have designed a simulation environment to evaluate the efficiency and accuracy of the approach proposed. First, we implemented 100 Web services. These services belong to 3 categories, which are sensitive to data size, network throughput and CPU utilization separately. They are distributed over different network environments. All Web servers provide an open SNMP service and we installed a monitoring pro-

gram for network monitoring. We gathered user input data size, server CPU utilization and server port throughput. The monitor submits environment information to the monitoring log recorder, which is responsible for cleaning the monitor log and storing data in the database. We generated a 200*100 invocation pattern-QoS matrix, restricted to the response time characteristics. Fig. 2 shows the experimentation architecture.

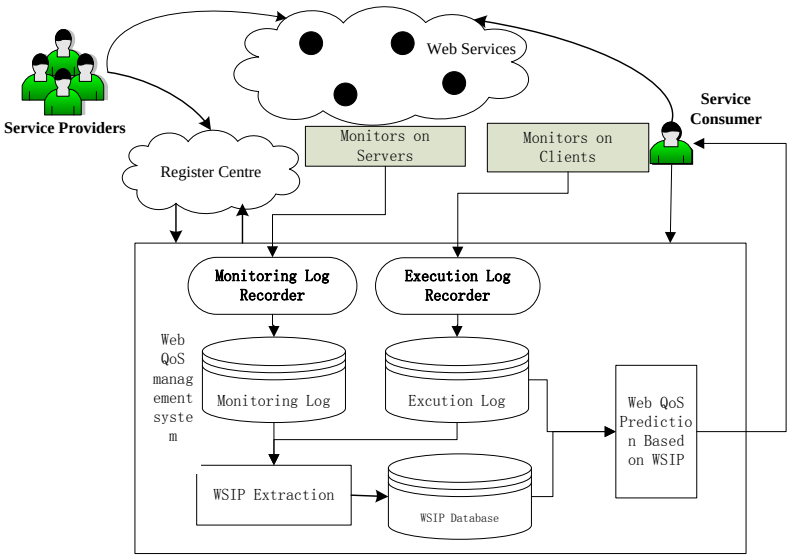

Fig. 2. Experimentation Architecture

Service providers register their Web service with a registry centre. Monitors for server and client are responsible for submitting the monitoring data to the Monitoring Log Recorder and the Execution Log Recorder. The Service Invocation Pattern Extraction module is responsible for extracting the service invocation patterns from the monitoring log and the execution log. When user requirements need to be processed, the QoS Management System will predict service QoS for a user according to their requirements. Then, the user can decide to invoke this service or not.

Accuracy Analysis. *MAE* (Mean Absolute Error) is the normal standard to measure the prediction accuracy. Here *MAE* is the mean absolute error between prediction and real response time. The smaller the *MAE*, the more accurate is the prediction. Assuming p_{ij} is the prediction value and t_{ij} is the real value, then *MAE* can be calculated as follows, where N is the total number of predictions:

$$MAE = \frac{\sum_{i,j} | t_{ij} - p_{ij} |}{N} \tag{8}$$

Different characteristics of QoS have different ranges. Consequently, we use *NMAE* (Normalized Mean Absolute Error) instead of *MAE*. The smaller the *NMAE*, the more accurate is the prediction. *NMAE* is the normalized *MAE*:

$$NMAE = \frac{MAE}{\sum_{i,j} t_{i,j} / N} \tag{9}$$

The accuracy of the prediction is important. Web QoS prediction algorithms usually are statistics-based and collaboration method-based. Average-based methods do not consider the users' personalized requirements and the impact of the network. Thus, they calculate the same prediction for all users. Collaboration-based methods need to use all historic data, i.e., the computation takes too long. We analysed these three approaches and tried different settings of k, α and β to assess the result.

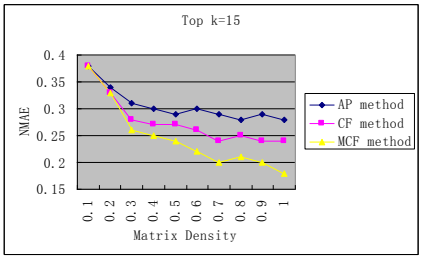

Fig. 3. NMAE of k=15

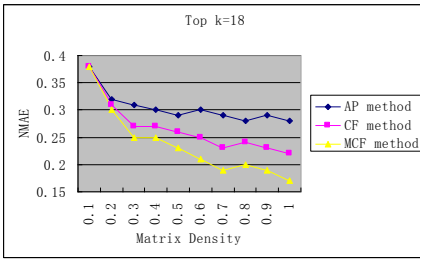

Fig. 4. NMAE of k=18

Different ks have different impacts on the result. If k is too large, there will be too much unnecessary information. The prediction result will be affected. However, if k is too small, useful information will be ignored and the data will not be sufficiently large enough for prediction. The similarity of the first k patterns maybe different under different data condition. Thus, a fixed k is not the objective. We tested different numbers of neighbouring patterns. We took the square root of the number of patterns first. Then, considering the pattern similarity, we fixed 0.5 as the critical value of similarity. If similarities between the target pattern and all other patterns exceed 0.5, then we increase k, otherwise decrease k. After testing, when k is 15 or 18, the performance is better in our environment. α and β in Formula (1) have also different impacts in different datasets. For our dataset, the performance is best when α is 0.2. We use AP to represent the average method. CF is the abbreviation of the collaboration-based algorithm. MCF is the abbreviation of the approach in this paper. As indicated in Figures 3 and 4, an increase of the dataset size improves the accuracy significantly.

Efficiency Analysis. If the target invocation can be matched in the service invocation pattern database and if there is QoS of the target service within the matched pattern, we can predict QoS directly. Only if there is no related data, collaborative computation is needed. The dataset for collaborative computation is related to service invocation patterns, but the number of patterns is far less than the number of usage information items. We used DBSCAN to obtain the service invocation patterns. We determined 150 invocation patterns from 2400 usage recordings. Compared to work in [11,12], the matrix for collaborative computation is reduced from 2400*100 to 150*100. Here, only when the matched pattern has no information of the target service, the calculation for prediction is required. Thus, the computation effort is decreased to a large extent. We tested the algorithm on many datasets. For each dataset,

50 predictions were taken and we averaged the response time. The comparison between the methods is shown in Figure 5. When the size of the dataset grows, time consumption in normal collaborative cases increases quickly. Our approach (MCF) is not much affected by data size.

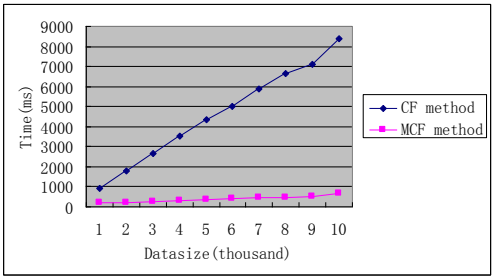

Fig. 5. Efficiency Analysis

7 Related Work

Different types of Web or cloud services [26] usually have different QoS characteristics [1-12]. The normally used ones are response time, execution cost, reliability, availability, security and reputation. There are many factors that impact on QoS [18]. Some factors are static, some are run-time static, the others are totally dynamic. Run-time static and dynamic factors are uncertain. They are client load, server load, network channel bandwidth or network channel delay. Most factors can be obtained by monitoring, but not all. Then, their impact cannot be calculated.

QoS-based service selection has been widely covered [1-10]. Many service prediction methods are proposed. There are three categories of prediction. The first one is statistic, which is normally adopted [1,2,7,8,9]. This method is simple and easy to implement. The second category is based on user feedback and reputation [19,20]. It can avoid malicious feedback, but these methods do not consider the impact of user requirements and the environment and cannot personalize prediction for users. The third category is based on collaborative filtering [11-14]. Collaborative filtering is a widely adopted recommendation method [21-24,28]. Zeng [22] summarizes the application of collaborative filtering in book, movie and music recommendation. In this paper, collaborative filtering is combined with service invocation patterns, user requirement and preferences. This considers different user preferences and makes prediction personalized, while maintaining good performance results.

Some works integrate user preferences into QoS prediction [11-15], e.g. [11-13] propose prediction algorithms based on collaborative filtering. They calculate the similarity between users by their usage data and predict QoS based on user similarity. This method avoids the influence of the environment factor on prediction. Even the same user will have different QoS experiences over time or with different input data, but these works do not consider user requirements and generally show low efficiency.

The proposed method in this paper takes full account of user requirements, the network and server factors. It abstracts the service invocation pattern to keep the ser-

vice QoS steady. When user requirements are known, prediction can be done based on matched patterns. This approach is efficient and reduces the computational overhead.

8 Conclusion

Service management in Web and Cloud environments [26,27], e.g. public clouds, requires service-level agreements (SLA) for individual users to be managed continuously, based on monitored QoS data. (Cloud) service managers take care of this for the users. Dynamic, personalised prediction of QoS is an essential component of reliable service provisioning that makes service lifecycle management more reliable. The need to personalise services dynamically is highlighted by e.g. cloud requirements for efficient service quality management adapted to user-specific requirements and situations across a range of end-user and business solutions offered as cloud services.

This paper proposes a service QoS prediction technique to satisfy personalized requirements. It considers not only the impact of the network, but also the Web server environment, and especially the individual user requirements. Based on historic information, we can abstract past user invocation pattern (mined from monitored log data) in order to predict future QoS of potential services to be utilised. The pattern approach provides independent reliability for the prediction of SLA-relevant aspects. When there is no information about the target pattern, we utilize collaborative filtering to predict according the data of other patterns. The results show that this approach is more accurate and personalized, and also demonstrates good prediction performance, which allows for dynamic utilisation of the technique.

Acknowledgement. This research has been supported by the National Natural Science Foundation of China (grant 61073062), the *Technology Project of LiaoNing Province* (2011216027) and the *Irish Centre for Cloud Computing and Commerce*, an Irish national Technology Centre funded by Enterprise Ireland and the Irish Industrial Development Authority.

References

1. Cardoso J., Sheth A., Miller J., Arnold J., and Kochut K.: Quality of Service for Workflows and Web Service Processes. Journal of Web Semantics,1(3), 281-308 (2004)
2. Kritikos, K., Plexousakis, D.: Requirements for QoS-based Web service description and discovery. IEEE Transactions on Services Computing, 2(4), 320-337 (2009)
3. Zheng, K., Xiong, H.: Semantic Web service discovery method based on user preference and QoS. Intl Conf on Consumer Electr, Comms and Netw CECNet'12, 3502-3506 (2012)
4. Ali, R. J. A., Rana, O.F. Walker, D. W.: G-QoSM: Grid service discovery using QoS properties. Computing and Informatics, 21(4), 363-382 (2012)
5. Wang, P.: QoS-aware web services selection with intuitionistic fuzzy set under consumer's vague perception. Expert Systems with Applications, 36(3), 4460-4466 (2009)
6. Huang, A. F. M., Lan, C. W., Yang, S. J. H.: An optimal QoS-based Web service selection scheme. Information Sciences, 179(19): 3309-3322 (2009)

7. Ye, Z., Bouguettaya, A., Zhou, X.: QoS-Aware Cloud Service Composition based on Economic Models. Service-Oriented Computing, Springer, 111-126 (2012)
8. Alrifai, M., Skoutas, D., Risse, T.: Selecting skyline services for QoS-based web service composition. Proc. Intl Conf on World Wide Web, ACM, 11-20 (2010)
9. Zeng, L., Benatallah, B., Ngu, A. H. H., et al.: QoS-Aware middleware for Web services composition. IEEE Trans on Software Engineering, 30(5), 311-327 (2004)
10. Yu, T., Lin, K. J.: Service Selection Algorithms for Web Services with End-to-end QoS constraints. Information Systems and E-Business Management, 3(2):103-126 (2005)
11. Shao, L., Zhang, J., Wei, Y., et al.: Personalized QoS prediction for Web services via collaborative filtering. IEEE Intl Conference on Web Services ICWS 2007, 439-446 (2007)
12. Zheng, Z., Ma, L. M. R., et al.: Qos-aware web service recommendation by collaborative filtering. IEEE Transactions on Services Computing, 4(2) ,140-152 (2011)
13. Zheng, Z., Ma, H.: WSRec: A Collaborative Filtering Based Web Service Recommender System. Proc IEEE Intl Conference on Web Services, 437 – 444 (2009)
14. Wu, G., Wei, J., Qiao, X., et al.: A Bayesian network based QoS assessment model for web services. Proc IEEE Intl Conference on Service Computing, 498-505 (2007)
15. Li, Z., Bin, Z., Ying, L., et al. A Web Service QoS Prediction Approach Based on Collaborative Filtering. IEEE Asia-Pacific Services Computing Conf APSCC'10, 725-731(2010)
16. Li, Z., Bin, Z., Jun, N., et al.: An Approach for Web Service QoS prediction based on service using information. Intl Conference on Service Sciences ICSS'2010. 324-328 (2010)
17. Ester, M., Kriegel, H. P., Sander, J., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. Proc. Intl Conf on Knowledge Discovery in Databases and Data Mining (KDD-96). AAAI Press, 226-232 (1996)
18. Lelli, F., Maron, G., Orlando, S.: Client Side Estimation of a Remote Service Execution, IEEE International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS) (2007)
19. Vu, L. H., Hauswirth, M., Aberer, K.: QoS-based Service Selection and Ranking with Trust and Reputation Management. Computer Science, 3760(2005) , 466-483 (2005)
20. Yan, L., Minghui, Z., Duanchao, L., et al.: Service selection approach considering the trustworthiness of QoS data. Journal of Software, 19(10), 2620-2627 (2008)
21. Sarwar, B., Karypis, G., Konstan, J., et al.: Item-based collaborative filtering recommendation algorithms. Proc 10th Int'l World Wide web Conf. ACM Press, 285-295 (2001)
22. Chun, Z., Chunxiao, X., Lizhu, Z.: A Survey of Personalization Technology. Journal of Software, 13(10), 1852-1861 (2002)
23. Hailing, X., Xiao, W., Xiaodong, W., Baoping, Y.: Comparison study of Internet recommendation system. Journal of Software, 20(2):350-362 (2009)
24. Ailing, D., Yangyong, Z., Bole, S.: A Collaborative Filtering Recommendation Algorithm Based on Item Rating Prediction. Journal of Software, 14(9):1621-1628 (2003)
25. Balke, W. T., Matthias, W.: Towards personalized selection of Web services. Proc. Intl World Wide Web Conf. New York: ACM Press, 20-24 (2003)
26. Pahl, C., Xiong, H., Walshe, R.: A Comparison of On-premise to Cloud Migration Approaches. European Conference on Service-Oriented and Cloud Computing ESOCC 2013. Springer LNCS (2013)
27. Pahl, C., Xiong, H.: Migration to PaaS Clouds - Migration Process and Architectural Concerns. IEEE 7th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems MESOCA'2013. IEEE (2013)
28. Huang, A.F., Lan, C.W., Yang, S.J.: An optimal QoS-based Web service selection scheme. Information Sciences, 179(19), 3309 -3322 (2009)