2014 IEEE 34th International Conference on Distributed Computing Systems

Towards Online, Accurate, and Scalable QoS Prediction for Runtime Service Adaptation

Jieming Zhu, Pinjia He, Zibin Zheng, Michael R. Lyu
Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China
Ministry of Education Key Laboratory of High Confidence Software Technologies (CUHK Sub-Lab)
Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong
{jmzhu, pjhe, zbzheng, lyu}@cse.cuhk.edu.hk

***Abstract*—Service-based cloud applications are typically built on component services to fulfill certain application logic. To meet quality-of-service (QoS) guarantees, these applications have to become resilient against the QoS variations of their component services. Runtime service adaptation has been recognized as a key solution to achieve this goal. To make timely and accurate adaptation decisions, effective QoS prediction is desired to obtain the QoS values of component services. However, current research has focused mostly on QoS prediction of the working services that are being used by a cloud application, but little on QoS prediction of candidate services that are also important for making adaptation decisions. To bridge this gap, in this paper, we propose a novel QoS prediction approach, namely adaptive matrix factorization (AMF), which is inspired from the collaborative filtering model used in recommender systems. Specifically, our AMF approach extends conventional matrix factorization into an online, accurate, and scalable model by employing techniques of data transformation, online learning, and adaptive weights. Comprehensive experiments have been conducted based on a real-world large-scale QoS dataset of Web services to evaluate our approach. The evaluation results provide good demonstration for our approach in achieving accuracy, efficiency, and scalability.**

Keywords*—*Service adaptation; QoS prediction; online learning; adaptive matrix factorization

I. Introduction

Cloud computing has gained increasing prevalence in recent years for providing a promising paradigm to host and deliver various online applications over the Internet. However, as these applications scale up, for example, spanning across multiple geographically distributed data centers [1], a significant challenge faced by application designers is how to engineer their applications with self-adaptation capabilities in response to the constantly changing operational environments, whereby the quality of service (QoS) can be guaranteed.

Many cloud applications have employed service-oriented architecture (SOA) as a mechanism for achieving self-adaptation [2], where component services are composed in a loosely-coupled way to fulfill complex application logic. For example, Amazon's e-commerce platform is built on SOA by composing hundreds of component services hosted world-wide to deliver functionalities ranging from item recommendation to order fulfillment to fraud detection [3]. The features of SOA such as loose coupling and dynamic binding enable applications to switch component services without going offline, and thus make it particularly amenable to the introduction of service adaptation [4]. On the other hand, with the proliferation of cloud computing, many service providers begin to offer more and more services in the cloud that provide equivalent (or similar) functionalities through a well-defined interface (*e.g.*, Web service) [5]. For example, both providers CDYNE.COM and WebserviceX.NET offer similar Web services for querying global weather information. Such redundant services can thus be utilized for service adaptation by replacing the current working services with the corresponding candidate services in response to unexpected QoS changes (e.g., unacceptable response time). To achieve so, knowledge about QoS values of the services is required to make timely and accurate adaptation decisions, such as when to trigger an adaptation action, which working services to be replaced, and which candidate services to employ. Note that, in this paper, we refer to working services as the services that are being used by a cloud application, and candidate services as the alternative services that have equivalent functionalities.

For a cloud application, the working services are frequently invoked, thus their QoS values can be collected via monitoring. In recent literature, existing QoS prediction approaches (*e.g.*, [6], [7], [8]) for service adaptation focus mostly on monitoring (or predicting) QoS values of the working services, which can help determine when to trigger an adaptation action and which working services to be replaced. However, to the best of our knowledge, there is no work explicitly addressing the problem of QoS prediction on candidate services for service adaptation, thus making it difficult in determining which candidate services to employ for an adaptation action. It is challenging to obtain QoS values of the candidate services due to the prohibitive overhead for actively measuring a large number of candidate services at runtime. Besides, some service invocations may be charged, which further increases the cost of service users (Hereinafter, we refer to cloud applications that invoke the services as "service users"). Therefore, it is highly desired to employ QoS prediction approaches to accurately estimate the QoS values of candidate services without requiring direct invocations, which is exactly the goal of our work. In particular, effective QoS prediction on candidate services needs to fulfill the following requirements.

1) ***Online***: The changing and evolving cloud environment introduces a high degree of variability and uncertainty on user-perceived service quality. For instance, due to the impact of dynamic network conditions and varying server workload, the QoS values may vary significantly during different time periods. Therefore, in order to identify high-quality candidate services for service adaptation, QoS prediction needs to be performed in an online fashion.

1063-6927/14 $31.00 © 2014 IEEE
DOI 10.1109/ICDCS.2014.40

2) ***Accurate***: Ensuring the accuracy of QoS prediction is fundamental for service adaptation. Inaccurate predictions may lead to the execution of improper adaptations or missed adaptation opportunities. For example, a working service may be wrongly replaced by a low-quality service. Consequently, we need accurate QoS prediction approaches, as well as proper metrics to evaluate the prediction accuracy.
3) ***Scalable***: In the dynamic cloud environment, new services with different QoS may become available, and existing services may be discontinued by their providers. Likewise, service users may often join or leave the environment. In face of the high churning rate of users and services, QoS prediction approaches need to scale well to new services and users, and perform robustly to make accurate predictions.

To achieve these goals, in this paper, we propose a novel QoS prediction approach, which is inspired from the collaborative filtering model used in recommender systems, to estimate the QoS values of candidate services by leveraging historical QoS data collaboratively from different users. The insight is that different users may use a common set of services and some users may observe similar QoS on the same service. However, different from the conventional matrix factorization (MF) model applied in recommender systems, our problem is more specific to the QoS prediction problem due to the aforementioned stringent requirements. As a result, we extend the conventional MF model into an online, accurate, and scalable QoS prediction approach, namely adaptive matrix factorization (AMF), by employing techniques of data transformation, online learning, and adaptive weights. To evaluate our AMF approach, experiments including accuracy comparison, efficiency analysis, and scalability analysis are conducted based on a real-world large-scale Web service QoS dataset, which consists of response time and throughput data between 142 users and 4,500 services over 64 continuous time slices (at an interval of 15 minutes). The evaluation results provide good demonstration for our approach in achieving accuracy, efficiency, and scalability. For reproducibility, we release our source code and dataset online[1].

In summary, our paper makes the following contributions:

- This is the first work to address the problem of QoS prediction on candidate services to guide candidate service selection for runtime service adaptation.
- A novel QoS prediction approach, adaptive matrix factorization (AMF), is proposed by employing techniques of data transformation, online learning, and adaptive weights.
- Comprehensive experiments are conducted based on a real-world large-scale QoS dataset of Web services to evaluate our proposed approach in terms of accuracy, efficiency, and scalability.

Paper organization. Section II overviews some background and related work. Section III presents the framework of QoS-driven service adaptation. Then we describe our AMF approach for QoS prediction in Section IV, and report the evaluation results in Section V. Finally, we conclude in Section VI.

[1]http://rmblab.github.io/icdcs2014_AMF.html

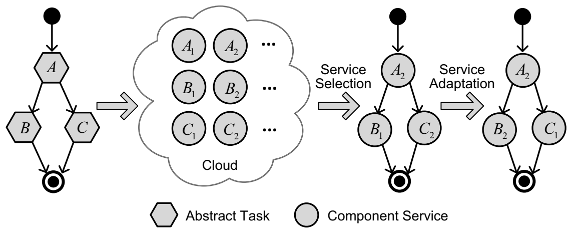


Fig. 1. An Illustrative Example of Service Adaptation

II. Background and Related Work

In this section, we review the background and related work from three aspects: service adaptation, QoS attributes, and QoS prediction.

A. Service Adaptation

Self-adaptation is a key solution for cloud applications to cope with the changing operational environments [9]. In contrast to the well-studied traditional adaptive software systems [10], the dynamic cloud environment imposes a number of new challenges to the adaptation of cloud applications. In service-based cloud applications, the application logic is typically expressed as a workflow with a set of abstract tasks, as shown in the leftmost panel in Fig. 1. These abstract tasks (*e.g.*, A, B, C) are then implemented by invocations to the underlying component services (*e.g.*, A_2, B_1, C_2) provided in the cloud. It is expected that the proliferation of cloud computing will bring substantial deployment of services into the cloud, so that for each abstract task there are a set of functionally-equivalent candidate services. Conventional service composition approaches (*e.g.*, [11]) focus on how to make optimal service selection from those candidate services at design time. However, due to the dynamic nature of cloud environment, original services may become unavailable, new services may emerge, and the QoS values of services may change from time to time, thus leading to violations of SLA (service-level agreement). In such a setting, QoS-driven service adaptation is desired. Fig. 1 presents such an illustrative example, where services B_1, C_2 are replaced with services B_2, C_1 respectively in an adaptation action, in the cases that the invocation to B_1 fails and the QoS of C_2 degrades.

To achieve this goal, a large body of research work has been conducted in recent literature. For example, the work [4] and [12] extend BPEL (Business Process Execution Language) engines with an interception and adaptation layer to enable monitoring and recovery of services. The work [13] employs autonomic configuration of performance parameter settings to achieve self-adaptation for online Web applications. Some other work, such as [2] and [9], provides feasible adaptation mechanisms (*e.g.,* replacing the component services, or re-structuring the workflows) to support QoS-driven service adaptation. While most of these studies focus on adaptation mechanism design, our work targets at another key challenge, namely online QoS prediction [14], which is also fundamental for service adaptation.

B. QoS Attributes

QoS attributes are widely-used metrics to evaluate the non-functional properties of services, including response time,

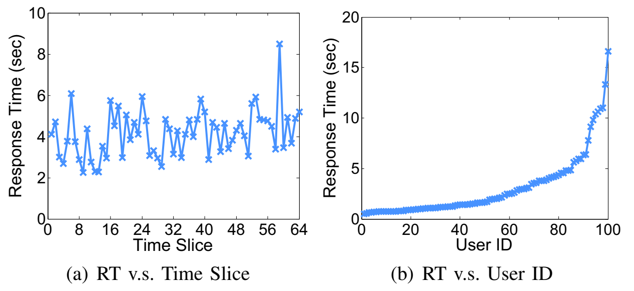


Fig. 2. Real-world Response Time (RT) Observations

throughput, failure probability, availability, etc. [11]. Ideally, the QoS values of services can be directly specified in the SLA by service providers. However, it is infeasible in most cases due to the following characteristics of QoS attributes.

Dynamic: Most of QoS attributes (*e.g.*, response time and throughput) are time-varying. For instance, due to the impact of varying server workload and dynamic network conditions, QoS delivered to users may vary widely during different time periods. Fig. 2(a) depicts a real-world example of the response times of a user at Pittsburgh (IP: 12.108.127.138) invoking a Web service located at Iran (http://profiles.roshd.ir/security.asmx?WSDL) over 64 consecutive time slices (at 15-minute interval), where the data is extracted from our QoS dataset in Section V-A. The curve confirms that the user-perceived response time fluctuates around an average QoS value along the time. Therefore, some QoS attributes should be evaluated at runtime.

User-specific: Services hosted in the cloud may be located across many data centers world-wide (*e.g.*, the shopping cart service for Amazon.com [3]). With the increase of geographic distribution of services, the impact of the network on user-perceived QoS becomes non-negligible. Thus, users from different locations may observe different QoS values even on the same service. Fig. 2(b) confirms such observation by a real-world example, which presents the response times (sorted in ascending order) perceived by 100 randomly-selected users that invoke the same service. The large variation of the curve implies that QoS attributes like response time are user-specific and should be evaluated independently from each user side.

As a result, it is extremely challenging to obtain QoS values of component services without incurring much overhead. QoS prediction has been emerged as a key solution to estimate the unknown QoS values by employing the historical usage data, while requiring no additional service invocations.

C. QoS Prediction

Accurate QoS prediction is fundamental for QoS-driven service adaptation. The predicted QoS values directly impact the service adaptation decisions. For example, inaccurate predictions may cause improper adaptations and thus lead to SLA violations. For this purpose, online monitoring and prediction approaches, as presented in [6], [8], have been proposed to detect service failures and QoS deviations of the working services, but QoS prediction on candidate services is still not well explored. The approach introduced in [15] proposes to collect QoS values by sampling and invoking the candidate services, which is heavily limited by the incurred overhead. Thus, our work is motivated to address the QoS prediction problem for candidate services.

To drive our study, in the following, we briefly introduce ***collaborative filtering (CF)***. CF techniques [16] are widely used in commercial recommender systems. The basic idea of CF is to exploit and model the observed data for predicting the unknown values. In recent literature, CF has been introduced as a promising technique for system engineering tasks, including service recommendation [17], [18], system reliability prediction [19], and QoS-aware datacenter scheduling [20]. In particular, as with rating prediction in recommender systems [16], the user-perceived QoS values on service invocations can also produce a user-service QoS matrix, which can be further modelled as a CF problem for making user-specific QoS predictions.

Matrix factorization (MF), as one of the most promising models for collaborative filtering, has been widely studied in recent years [21]. In our previous work [22], we made use of the MF model to guide dynamic request routing for online cloud applications. Some recent work (*e.g.*, [18], [23]) introduces MF into the QoS prediction problem for service selection at design time. However, their approaches primarily work offline on the collected QoS data, and thus fail to meet the online requirement for runtime service adaptation. Besides, these existing approaches cannot easily scale to new users and services, because of the prohibitive overhead for constant re-training. Our work thus aims to address these problems.

III. Framework of QoS-driven Service Adaptation

To build high-quality cloud applications, we propose a basic framework for QoS-driven service adaptation, as illustrated in Fig. 3. In this framework, two modules are incorporated to support QoS-driven service adaptation.

Execution middleware: A service-based cloud application typically comprises a workflow specified in BPEL and runs on a BPEL engine, like *Apache ODE*[2]. In order to support QoS-driven service adaptation actions (*e.g.*, replacing component services or re-structuring workflows), BPEL engines can be enriched with sophisticated functionalities like QoS manager, candidate service manager, and user-specified adaptation polices. Concretely, candidate service manager discovers all available candidate services that match their needs, while QoS manager monitors the QoS values of service invocations, uploads the observed QoS data, and then obtains the related QoS prediction results through the interface of *QoS prediction service*. Based on the QoS prediction results, various adaptation polices (*e.g.*, when to trigger an adaptation action and, if necessary, which candidate services to employ) can be plugged in and executed automatically without causing any downtime of the overall application.

QoS prediction service: This module is designed as a service working by collaboratively collecting the observed QoS data from different users and then providing accurate QoS prediction results for these users transparently through a standard interface. More specifically, the QoS prediction service works as follows: 1) *Input handling*: The observed QoS data are collected and processed as formatted stream

[2]http://ode.apache.org/

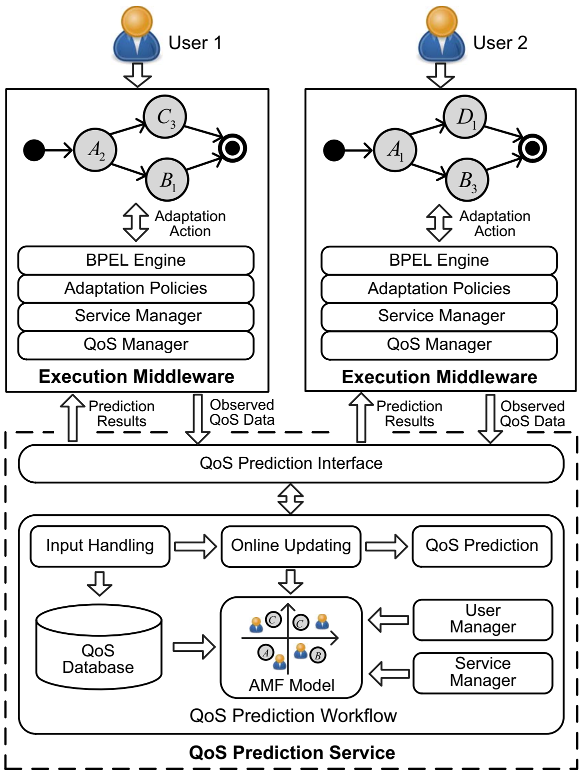


Fig. 3. Framework of QoS-driven Service Adaptation

data. The QoS database can be updated accordingly. 2) *Online updating*: The AMF model can be updated online by using the sequentially observed QoS data. 3) *QoS prediction*: The QoS prediction results by our AMF model can be provided to users on demand through the QoS prediction interface. Additionally, a service manager is desired to provide utilities like service discovery and service management of available services. Likewise, a user manager is set up to manage the joining or leaving activities of users.

The framework shows that effective QoS prediction is fundamental for successful service adaptation executions in service-based cloud applications, because the performance of service adaptation is heavily influenced by the QoS prediction results. Thus, QoS prediction is the main focus of our study.

IV. QoS Prediction

In this section, we first introduce the problem of QoS prediction on candidate services. Then we describe the conventional matrix factorization model and discuss its potential limitations on service adaptation. At last, we illustrate how to address these limitations by extending the MF model into our AMF model to achieve accuracy, efficiency, and scalability for service adaptation.

A. Problem Description

In the previous work, QoS prediction approaches (e.g., [6], [8]) focus primarily on QoS prediction for working services (that are being used by a cloud application) by employing techniques such as time series analysis on historical QoS data. According to the prediction results, potential SLA violations can be detected, thereby facilitating adaptation decisions such as when to trigger an adaptation action and which component services to be replaced. In contrary, our work focuses on QoS prediction for candidate services to help determine which candidate services to employ for an adaptation action.

Specifically, as with rating prediction in recommender systems, historical service invocations can produce a user-service QoS matrix with respect to each QoS attribute (*e.g.*, response time). This QoS matrix can be collected from user side in the form of user collaboration through our framework. In this matrix, each row denotes a service user (*i.e.*, a cloud application), each column denotes a candidate service in the cloud, and each entry denotes the QoS value observed by the a user when invoking a service. In practice, the QoS matrix is very sparse, since each user usually only invokes a handful of services. As in Fig. 4(b), values in grey entries are observed QoS data from the user-service invocation graph in Fig. 4(a), and the blank entries are unknown QoS values to be predicted. For example, the response time between user u_1 and service s_1 is $1.4s$, while the response time between user u_1 and service s_2 is unknown because u_1 has never invoked s_2.

Our goal of QoS prediction is to employ the observed QoS data to estimate the other unknown values. Formally, suppose there are n users and m services, we can obtain a sparse QoS matrix $R \in \mathbb{R}^{n\times m}$ with respect to each QoS attribute, where R_{ij} denotes the QoS value between user u_i and service s_j. As such, the QoS prediction problem can be modelled as a collaborative filtering problem that approximately reconstructs the unknown values from a small number of observed entries [16]. In addition, the QoS prediction approach should be performed in an online, accurate, and scalable manner.

B. Matrix Factorization and Its Limitations

Matrix factorization [21] is a classic model to address the above collaborative filtering problem, which constrains the rank of the QoS matrix, *i.e.*, $rank(R) = d$. The low-rank assumption is based on the fact that the entries of R are largely correlated, thereby resulting in a low effective rank in R. For instance, close users may have similar network conditions, and thus experience similar QoS on the same service. Fig. 4 illustrates an example that makes use of matrix factorization for QoS prediction. Concretely, factoring a matrix is to map both users and services into a joint latent factor space of a low dimensionality d (*e.g.*, $d = 2$ in Fig. 4(c)), such that values of the user-service QoS matrix can be captured as inner products of latent factors in that space. Then the latent factors can be employed for further prediction on unknown QoS values. For example, as shown in Fig. 4(d), the predicted response time value between user u_1 and s_2 is $0.8s$.

Formally, latent user factors are denoted as $U \in \mathbb{R}^{d\times n}$ and latent service factors as $S \in \mathbb{R}^{d\times m}$, which are used to fit the QoS matrix R, *i.e.*, $R \approx U^T S$. To avoid overfitting, regularization terms that penalize the norms of the solutions (*i.e.*, U and S) are added. Thus we aim to minimize the following loss function:

$$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(R_{ij} - U_i^T S_j)^2 + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_S}{2}\|S\|_F^2, \quad (1)$$

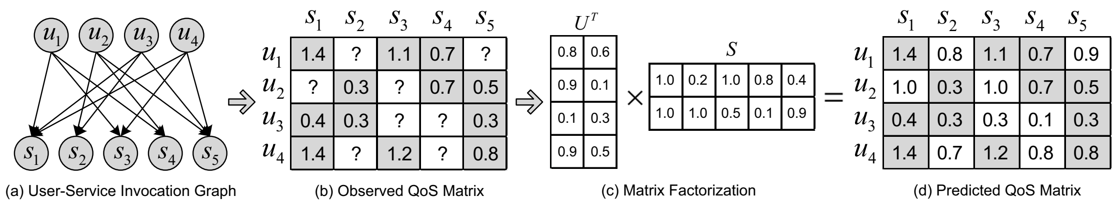


Fig. 4. An Example of QoS Prediction by Matrix Factorization

where I_{ij} acts as an indicator that equals to 1 if R_{ij} is observed, and 0 otherwise (*e.g.*, $I_{11} = 1$ and $I_{12} = 0$ in Fig. 4(b)). $\|\cdot\|_F$ denotes the *Frobenius norm* [21], and λ_U, λ_S are two parameters to control the extent of regularization. *Gradient descent* [21] is usually employed to derive the solutions U and S, by iterating in the following form until convergence:

$$U_i \leftarrow U_i - \eta \frac{\partial \mathcal{L}}{\partial U_i}, \quad S_j \leftarrow S_j - \eta \frac{\partial \mathcal{L}}{\partial S_j}, \tag{2}$$

where η is the learning rate controlling how much change to make at each iteration. After obtaining the latent factors U and S, the unknown QoS values can then be predicted by their corresponding inner products: $\hat{R}_{ij} = U_i^T S_j$, where $I_{i,j} = 0$.

Although this conventional matrix factorization model performs well for rating prediction problem in recommender systems, it is insufficient to address our QoS prediction problem for service adaptation, due to the following limitations:

Limitation 1: Due to our observation on a real-world QoS dataset, we find that different from the coherent value range of ratings (*e.g.*, 1∼5) in recommender systems, the QoS values vary widely (*e.g.*, $0\sim20s$ for response time and $0\sim7000kbps$ for throughput). Moreover, the distributions of QoS data are highly skewed with large variances (as shown in Fig. 7) compared with the rating distribution, which mismatches with the probabilistic assumption for matrix factorization [21]. Consequently, directly applying the original MF model to QoS data may significantly degrade its prediction accuracy.

Limitation 2: Our QoS prediction problem differs from recommender systems mainly in that QoS values are time-varying while rating values keep unchanged once being rated. In other words, existing QoS values will be continuously updated with newly observed values, or become expired after a time period without updating. However, conventional MF model primarily works offline on collected data. Therefore, to adapt to a newly observed QoS value, the MF model has to be entirely retrained, which will incur large computation overhead and make it infeasible to be performed online.

Limitation 3: Due to the dynamic nature of cloud environment, both users and services may continuously join or leave the environment (*i.e.*, churn occurs). However, the MF model focuses on the user-service QoS matrix with a fixed size (*w.r.t.* users and services), thus is not easily scalable to handle new users and new services without retraining the whole model.

C. Adaptive Matrix Factorization

To address the above limitations, we propose our new QoS prediction approach, adaptive matrix factorization (AMF), which aims to be online, accurate, and scalable. To achieve this goal, our AMF approach integrates three techniques: data transformation, online learning, and adaptive weights.

1) Data Transformation

To address *Limitation 1* (*i.e.*, skewed QoS value distributions), we apply a classic data transformation method, *Box-Cox* transformation [24], to QoS data. This technique is used to stabilize data variance and make the data more normal distribution-like in order to fit the matrix factorization assumption. The transformation is rank-preserving and performed by using a continuous power function defined as follows:

$$boxcox(x) = \begin{cases} (x^\alpha - 1)/\alpha & \text{if } \alpha \neq 0 \ , \\ \log(x) & \text{if } \alpha = 0, \end{cases} \tag{3}$$

where the parameter α controls the extent of the transformation. For simplicity, we denote $\tilde{R}_{ij} = boxcox(R_{ij})$. Note that $\tilde{R}_{max} = boxcox(R_{max})$ and $\tilde{R}_{min} = boxcox(R_{min})$ due to its monotonously nondecreasing property of Box-Cox transformation. R_{max}, R_{min} are the maximal and minimal QoS values respectively, which can be specified by users (e.g., $R_{max} = 20$s and $R_{min} = 0$ for response time in our experiments). Similarly, $\tilde{R}_{max}$ and $\tilde{R}_{min}$ are the maximal and minimal values after data transformation. Then we map the data into the range $[0, 1]$ by linear normalization,

$$r_{ij} = (\tilde{R}_{ij} - \tilde{R}_{min}) \big/ (\tilde{R}_{max} - \tilde{R}_{min}). \tag{4}$$

Especially, when $\alpha = 1$, the data transformation is relaxed to a linear normalization, where the effect of Box-Cox transformation is masked.

To fit the normalized QoS data r_{ij}, we employ the sigmoid function $g(x) = 1/(1 + e^{-x})$ to map the value $U_i^T S_j$ into the range of [0, 1], as described in [21]. Therefore, the loss function in Equation 1 can be transferred to:

$$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(r_{ij} - g_{ij})^2 + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_S}{2}\|S\|_F^2, \tag{5}$$

where g_{ij} denotes $g(U_i^T S_j)$ for simplicity.

However, conventional matrix factorization model minimizes the sum of squared errors and employs the absolute error metrics (*e.g.*, MAE as defined in Section V-B) to evaluate the prediction results. In practice, absolute error metrics are not suitable for evaluation of QoS prediction due to the large value range of QoS values. For instance, given two services with QoS values $s_1 = 1$ and $s_2 = 100$, the corresponding thresholds for adaptation action are set to $s_1 > 5$ and $s_2 < 90$. Suppose there are two sets of prediction results: (a) $s_1 = 8$ and $s_2 = 99$, (b) $s_1 = 0.9$ and $s_2 = 92$, we would choose (a) with smaller MAE if using the MAE metric. However, prediction (a) will cause a wrong adaptation action due to $s_1 > 5$, while prediction (b) is more reasonable. Consequently, we propose to employ relative error to evaluate the prediction results, where the corresponding loss function is derived as follows:

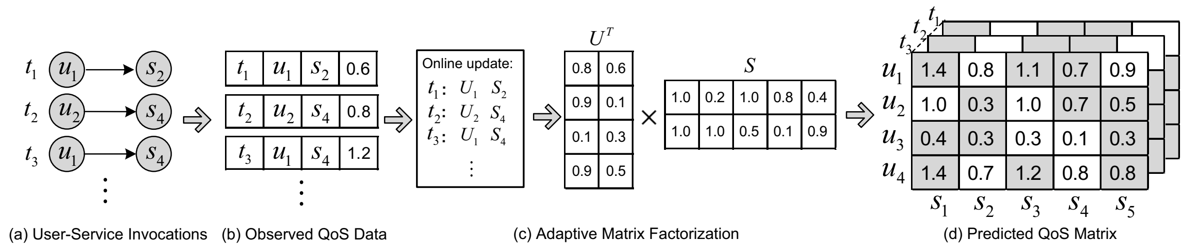


Fig. 5. An Example of QoS Prediction by Adaptive Matrix Factorization

$$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}\left(\frac{r_{ij}-g_{ij}}{r_{ij}}\right)^2 + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_S}{2}\|S\|_F^2, \quad (6)$$

2) Online Learning

To address *Limitation 2* (*i.e.*, time-varying QoS values), online learning algorithms are required to keep continuous and incremental updating using the sequentially observed QoS data. For this purpose, we employ a classic online learning algorithm, stochastic gradient descent (SGD) [25] to train our AMF model. For each QoS value observed by user u_i for invoking service s_j, we have the following pairwise loss function:

$$\ell(U_i, S_j) = \frac{1}{2}\left(\frac{r_{ij}-g_{ij}}{r_{ij}}\right)^2 + \frac{\lambda_u}{2}\|U_i\|_2^2 + \frac{\lambda_s}{2}\|S_j\|_2^2, \quad (7)$$

such that $\mathcal{L} = \sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}\ell(U_i, S_j)$. $\|\cdot\|_2$ denotes the *Euclidean norm*.

Instead of directly minimizing $\mathcal{L}$, SGD relaxes to minimize the pairwise loss function $\ell(U_i, S_j)$. By replacing $\mathcal{L}$ with ℓ in Equation 2, we can derive the following update equations regarding each data sample (u_i, s_j, R_{ij}):

$$U_i \leftarrow U_i - \eta((g_{ij}-r_{ij})g'_{ij}S_j/r_{ij}^2 + \lambda_u U_i), \quad (8)$$
$$S_j \leftarrow S_j - \eta((g_{ij}-r_{ij})g'_{ij}U_i/r_{ij}^2 + \lambda_s S_j), \quad (9)$$

where g'_{ij} denotes $g'(U_i^T S_j)$, and $g'(x) = e^x/(e^x+1)^2$ is the derivative of $g(x)$. η is the learning rate.

As illustrated in Fig. 5(a)(b), every time when a new data sample is observed, online updating can be performed on its corresponding factors using Equation 8 and 9. In other words, at each iteration, user u_i can take a small change on feature vector U_i and service s_j can have a small change on feature vector S_j, given a newly observed data sample (u_i, s_j, R_{ij}) after user u_i invoke service s_j.

3) Adaptive Weights

To address *Limitation 3* (*i.e.*, scalability on new users and services), we make use of the above online learning algorithm, which can update the feature vectors incrementally without retraining the whole model. However, the above online learning algorithm may not perform well under the high churning rate of users and services (*i.e.*, continuously joining or leaving the environment). The convergence is controlled by the learning rate η, but a fixed η will lead to problems for new users and services. For example, for a new user u_1, if its feature vector U_1 is at its initial position, larger η is needed to help it move quickly to its correct position. But for an existing service s_2 that user u_1 invokes, its feature vector S_2 may have already been converged. Adjusting the feature vector (S_2) of service s_2 according to the user u_1 is likely to increase prediction error rather than to decrease it, since user u_1 itself has large prediction error with its initial feature vector (U_1) not converged. Thus, our approach, if performed online, need to be robust towards the churning of users and services.

To achieve this goal, we propose to employ adaptive weights in our AMF model. Although the weighted matrix factorization has also been studied in [26], our approach differs from it in that we use adaptive weights instead of fixed weights in the iteration process. Specifically, we design an adaptive weight to control the step size at each iteration, depending on the accuracy achieved by the corresponding user or service. The goal is to mitigate the impact of new users or services that have high errors with their feature vectors not converged. Intuitively, an accurate user should not move much according to an inaccurate service while an inaccurate user need to move a lot with respect to an accurate service, and vice versa. For example, if service s_1 has an inaccuracy of 10% and service s_2 with inaccuracy 1%, when a user invokes both s_1 and s_2, it should move less for its feature vector to service s_1 compared with service s_2. As a result, we have two weights w_{u_i} and w_{s_j} for user u_i and service s_j respectively. Then we derive the following loss functions corresponding to U_i and S_j:

$$\ell_w(U_i) = \frac{1}{2}w_{u_i}\left(\frac{r_{ij}-g_{ij}}{r_{ij}}\right)^2 + \frac{\lambda_u}{2}\|U_i\|_2^2, \quad (10)$$
$$\ell_w(S_j) = \frac{1}{2}w_{s_j}\left(\frac{r_{ij}-g_{ij}}{r_{ij}}\right)^2 + \frac{\lambda_s}{2}\|S_j\|_2^2, \quad (11)$$

where $w_{u_i} + w_{s_j} = 1$, such that $\ell(U_i, S_j) = \ell_w(U_i) + \ell_w(S_j)$.

We denote the average error of user u_i as e_{u_i} and the average error of service s_j as e_{s_j}. Then we compute the weights w_{u_i}, w_{s_j} to control the credence between each other, as follows:

$$w_{u_i} = e_{u_i}/(e_{u_i}+e_{s_j}), \quad w_{s_j} = e_{s_j}/(e_{u_i}+e_{s_j}). \quad (12)$$

To update e_{u_i}, e_{s_j}, we compute the exponential moving average [27] at each iteration, which is a weighted average with more weight (controlling by β) given to the latest data.

$$e_{u_i} = \beta w_{u_i} e_{ij} + (1-\beta w_{u_i})e_{u_i}, \quad (13)$$
$$e_{s_j} = \beta w_{s_j} e_{ij} + (1-\beta w_{s_j})e_{s_j}, \quad (14)$$

where e_{ij} denotes the relative error between g_{ij} and r_{ij}:

$$e_{ij} = |r_{ij} - g_{ij}|/r_{ij}. \quad (15)$$

We also find that similar weights have been used for controlling the credence of node in network coordinate system [28], but our approach is the first to incorporate such weights into matrix factorization. After obtaining the updated weights w_{u_i} and w_{s_j} at each iteration, we can derive the following updating equations by computing the gradients in Equation 10 and 11:

$$U_i \leftarrow U_i - \eta w_{u_i}((g_{ij}-r_{ij})g'_{ij}S_j/r_{ij}^2 + \lambda_u U_i), \quad (16)$$
$$S_j \leftarrow S_j - \eta w_{s_j}((g_{ij}-r_{ij})g'_{ij}U_i/r_{ij}^2 + \lambda_s S_j), \quad (17)$$

Algorithm 1: Adaptive Matrix Factorization Algorithm

```
Input: Sequentially observed QoS data samples: $(t_{ij}, u_i, s_j, R_{ij})$,
       and all the model parameters.
Output: QoS prediction results: $\hat{R}_{ij} \leftarrow (U_i, S_j)$, where $I_{ij} = 0$.
1  repeat    /* Continuous and incremental updating */
2      Collect newly observed QoS data;
3      if receive a new data sample $(t_{ij}, u_i, s_j, R_{ij})$ then
4          $I_{ij} \leftarrow 1$;
5          if $u_i$ is a new user or $s_j$ is a new service then
6              Randomly initialize $U_i \in \mathbb{R}^d$, or $S_j \in \mathbb{R}^d$;
7              Initialize $e_{u_i} \leftarrow 1$, or $e_{s_j} \leftarrow 1$;
8          Update $t_{ij}, R_{ij}$ corresponding to $u_i, s_j$;
9          OnlineUpdate$(t_{ij}, u_i, s_j, R_{ij})$;
10     else
11         Randomly pick an existing data sample $(t_{ij}, u_i, s_j, R_{ij})$;
12         if $t_{now} - t_{ij} < TimeInterval$ then
13             OnlineUpdate$(t_{ij}, u_i, s_j, R_{ij})$;
14         else
15             Existing data sample is obsolete: set $I_{ij} \leftarrow 0$;
16     if converged then
17         Wait until observing new QoS data;
18 until forever;

19 OnlineUpdate$(t_{ij}, u_i, s_j, R_{ij})$:        /* Function */
20 Normalize $R_{ij}$ by Equation 3 and 4: $r_{ij} \leftarrow R_{ij}$;
21 Update $w_{u_i}, w_{s_j}$ by Equation 12:
   $w_{u_i} \leftarrow (e_{u_i}, e_{s_j})$,  $w_{s_j} \leftarrow (e_{u_i}, e_{s_j})$;
22 Compute $e_{ij}$ by Equation 15: $e_{ij} \leftarrow (r_{ij}, g_{ij})$;
23 Update $e_{u_i}, e_{s_j}$ by Equation 13 and 14:
   $e_{u_i} \leftarrow (w_{u_i}, e_{ij}, e_{u_i})$,   $e_{s_j} \leftarrow (w_{s_j}, e_{ij}, e_{s_j})$;
24 Update $U_i, S_j$ simutaneously by Equation 16 and 17;
```

With U_i and S_j, we can predict the unknown QoS value R_{ij} (where $I_{ij} = 0$) for the service invocation between user u_i and service s_j. Finally, a backward data transformation of $g(U_i^T S_j)$ is required, which can be computed according to the inverse functions for Equation 3 and 4.

4) AMF Algorithm

After analyzing the ingredients of our AMF model, we can have a big picture of the algorithm. Fig. 5 presents an illustrative example for QoS prediction by using AMF. Different with MF in Fig. 4, our AMF approach collects each observed QoS value in a stream way (Fig. 5(a)(b)), and keeps online updating accordingly (Fig. 5(c)). Then the current QoS valuse can be predicted using the updated model (Fig. 5(d)). The pseudo code of our online updating algorithm for AMF is provided in Algorithm 1. Specifically, at each iteration, the newly observed QoS data are collected to update the model (Line 2 $\sim$ 9), or else existing data are randomly selected for model updating (Line 11 $\sim$ 15) until convergence. Especially, the online updating operations are defined as a function $OnlineUpdate(t_{ij}, u_i, s_j, R_{ij})$ given a data sample $(t_{ij}, u_i, s_j, R_{ij})$, according to the steps described in 1) $\sim$ 3). Note that for a newly observed data sample, we first check whether the corresponding user or service is new, so that we can add it to our model (Line 5 $\sim$ 7) and keep updating its feature vector using more observed data on this user or service (Line 8 $\sim$ 9). As such, our model can scale to new users and services without retraining the whole model. Another important point is that we check whether an existing QoS value has become expired (Line 12), and if so, discard this value (*i.e.*, in Line 15, set $I_{ij} = 0$). In our experiment, for example, we set the expiration time interval to 15 minutes.

Statistics	Values
#Users	142
#Services	4,500
#Time slices	64
#Time interval	15*min*
RT range	$0 \sim 20s$
RT average	$1.33s$
TP range	$0 \sim 7000kbps$
TP average	$11.35kbps$

Fig. 6. Data Statistics

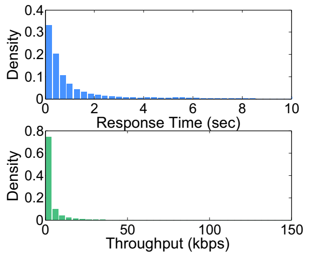


Fig. 7. Data Distribution

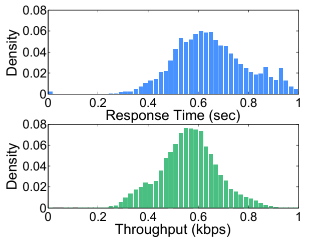


Fig. 8.Transformed Data Distribution

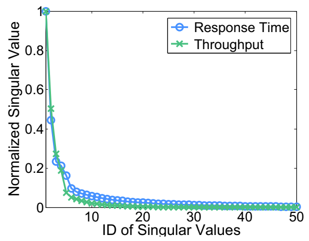


Fig. 9. Sorted Singular Values

V. Evaluation

In this section, we conduct a set of experiments based on a real-world Web service QoS dataset to evaluate our AMF approach from various aspects, including accuracy comparison, impact of parameters, efficiency analysis, and scalability analysis. All the experiments were conducted on a machine with a 3.2 GHz Intel CPU and 4 GB RAM, running Win7.

A. Data Description

In our experiments, we focus primarily on two QoS attributes: response time (RT) and throughput (TP). Response time stands for the time duration between user sending out a request and receiving a response, while throughput denotes the data transmission rate (*e.g.*, $kbps$) of a user invoking a service.

The data used in our experiment are extracted from a real-world Web service QoS dataset [29], including both response time and throughput values. These QoS values are collected by 142 users invoking 4,500 Web services for 64 consecutive time slices, at an interval of 15 minutes. The users are 142 machines (PlanetLab nodes) located in 22 countries, and the services are 4,500 publicly available real-world Web services from 57 countries [29]. The table in Fig. 6 provides some basic statistics of our data. For example, the range of response time is 0$\sim$20s, and the throughput 0$\sim$7000$kbps$. Furthermore, we plot the data distributions of response time and throughput in Fig. 7. For better visualization, we cut off the response time beyond 10s and the throughput more than 150$kbps$. It is shown that the data distributions are highly skewed. In contrast, as shown in Fig. 8, we obtain more normal data distributions through our data transformation in Section IV-C.

In addition, we investigate the singular values of the data matrices of response time and throughput between users and services. The singular values are computed by a singular value decomposition (SVD) [30] and then normalized so that the largest singular value is equal to 1, as illustrated in Fig. 9. We can observe that except the first few largest singular values, most of them are close to 0. This observation indicates that

TABLE I. ACCURACY COMPARISON (A SMALLER MAE, MRE OR NPRE VALUE MEANS BETTER ACCURACY)

QoS	Approach	Density = 10%			Density = 20%			Density = 30%			Density = 40%			Density = 50%		
		MAE	MRE	NPRE	MAE	MRE	NPRE	MAE	MRE	NPRE	MAE	MRE	NPRE	MAE	MRE	NPRE
RT	UPCC	1.224	0.769	7.842	1.076	0.611	5.893	1.006	0.557	4.943	0.967	0.529	4.547	0.940	0.511	4.332
	IPCC	1.273	0.776	6.650	1.218	0.779	6.354	1.144	0.736	5.768	1.070	0.680	5.192	1.020	0.647	4.826
	UIPCC	1.215	0.764	7.489	1.076	0.610	5.889	1.005	0.558	4.977	0.962	0.530	4.571	0.932	0.524	4.383
	PMF	1.104	0.593	3.017	1.030	0.596	3.414	0.982	0.581	3.390	**0.948**	0.564	3.294	0.928	0.546	3.198
	AMF	**1.076**	**0.478**	**1.765**	**1.007**	**0.386**	**1.080**	**0.974**	**0.356**	**0.968**	0.950	**0.344**	**0.929**	**0.921**	**0.334**	**0.914**
	Improve.(%)	2.5%	19.4%	41.5%	2.2%	35.2%	68.4%	0.8%	38.7%	71.5%	-0.2%	39.0%	71.8%	0.8%	38.8%	71.4%
TP	UPCC	9.019	2.179	26.176	8.237	1.900	25.091	7.691	1.697	23.830	7.382	1.624	24.134	7.131	1.537	23.850
	IPCC	8.744	0.833	12.816	8.434	0.832	12.750	7.960	0.789	11.830	7.452	0.729	10.438	7.127	0.699	9.748
	UIPCC	8.596	1.534	17.982	8.048	1.842	21.250	7.501	1.694	20.499	7.074	1.511	18.804	6.764	1.390	17.667
	PMF	6.894	0.567	2.899	6.474	0.525	2.929	6.235	0.488	2.847	5.960	0.459	2.764	5.668	0.436	2.657
	AMF	**6.303**	**0.513**	**2.148**	**5.920**	**0.414**	**1.424**	**5.742**	**0.385**	**1.170**	**5.694**	**0.368**	**1.042**	**5.621**	**0.356**	**0.983**
	Improve.(%)	8.6%	9.5%	25.9%	8.6%	21.1%	51.4%	7.9%	21.1%	58.9%	4.5%	19.8%	62.3%	0.8%	18.6%	63.0%

both data matrices are approximately low-rank, which conforms to the our low-rank assumption of matrix factorization. In our experiment, we set rank $d = 10$ (*i.e.*, the dimensionality of U_i, S_j).

B. Evaluation Metrics

We evaluate the prediction accuracy of our proposed approach in comparison with other existing approaches by using the following metrics.

- **MAE** (Mean Absolute Error). MAE is widely employed to measure the average prediction accuracy in recommender systems, defined as follows:

$$MAE = \sum_{I_{ij}=0} \left| \hat{R}_{ij} - R_{ij} \right| \Big/ N, \qquad (18)$$

  where R_{ij} is the measured value and $\hat{R}_{ij}$ is the corresponding predicted value. N is the number of samples that satisfy $I_{ij} = 0$.
- **MRE** (Median Relative Error). MRE takes the median value of all the pairwise relative errors:

$$MRE = \underset{I_{ij}=0}{median} \left| \hat{R}_{ij} - R_{ij} \right| \Big/ R_{ij}. \qquad (19)$$

- **NPRE** (Ninety-Percentile Relative Error). NPRE takes the 90th percentile of all the pairwise relative errors.

Due to the large variance of QoS values, in this paper, we focus more on relative error metrics, *i.e.*, MRE and NPRE, which are more appropriate for QoS prediction evaluation. Many papers report on MAE, so it is also included for comparison purpose. Nevertheless, our optimization efforts are not focused on MAE.

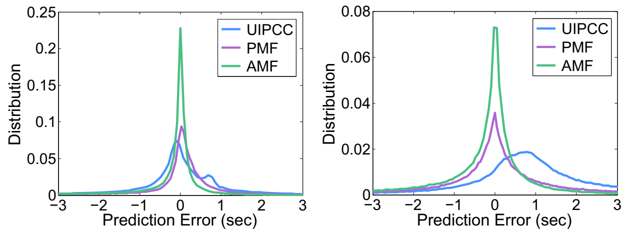
C. Accuracy Comparison

In order to evaluate the prediction accuracy, we compare our AMF approach with the following approaches that have been introduced for QoS prediction [17], [23]. It is worth noting that although these approaches are included for comparison purpose, they cannot be directly used for runtime service adaptation in practice, due to the aforementioned limitations.

- **UPCC**: This is a user-based collaborative filtering approach [17] that employs the similarity between users to predict the QoS values.
- **IPCC**: This is an item-based collaborative filtering approach [17] that employs the similarity between services to predict the QoS values.
- **UIPCC**: This is a hybrid approach proposed in [17], by combing both UPCC and IPCC approaches to make full use of the similarity between users and the similarity between services for QoS prediction.
- **PMF**: This is a widely-used implementation of matrix factorization model [21], which we have introduced in Section IV-B.

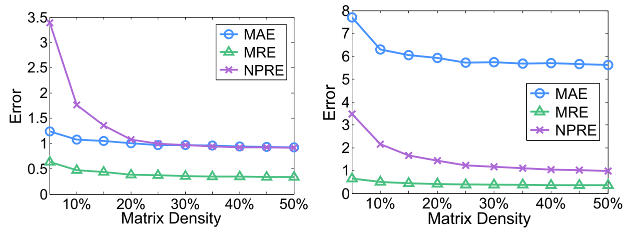
As we mentioned before, the available QoS data matrix is sparse in practice, because each user typically only uses a small number of candidate services out of all of them. To simulate the sparse situation, we randomly remove entries from the data matrix at each time slice so that each user only keeps a few available historical values. In this way, we vary the matrix density from 10% to 50% at a step increase of 10%. Matrix density = 10%, for example, indicates that each user invokes 10% (*i.e.* about 450) of the services, and each service is invoked by 10% (*i.e.* about 14) of the users. For AMF approach, the preserved data entries are randomized as a QoS data stream for training. Then the removed entries are used as the testing data to evaluate the prediction accuracy. In the sequel, for simplicity, we set $\lambda_u = \lambda_s = \lambda$ for AMF. Specifically, in this experiment, we set $d = 10$, $\lambda = 0.001$, $\beta = 0.3$, $\eta = 0.8$, $\alpha = -0.007$ for RT, and $\alpha = -0.05$ for TP. Note that the parameters of the other approaches are also optimized accordingly to achieve their optimal accuracy.

At each time slice, each approach is performed 20 times for each matrix density (with different random seeds). Then the results on average prediction accuracy over the first time slice are reported (The full results over all the time slices are reported in the supplementary report [31]). Table I provides the comparison results over three metrics, but we focus more on relative error metrics, *i.e.*, MRE and NPRE. As we can observe, our AMF approach significantly outperforms the other approaches over MRE and NPRE, while still achieving comparable (or best) results on MAE. Concretely, for response time (RT) data, AMF achieves 19.4%$\sim$39.0% improvement on MRE and 41.5%$\sim$71.8% improvement on NPRE at different matrix densities. Similarly, for throughput (TP) data, AMF has 9.5%$\sim$21.1% MRE improvement and 25.9%$\sim$63.0% NPRE improvement. Note that all improvements are computed as the percentage of how much AMF outperforms the other most competitive approach. We also find that although UIPCC



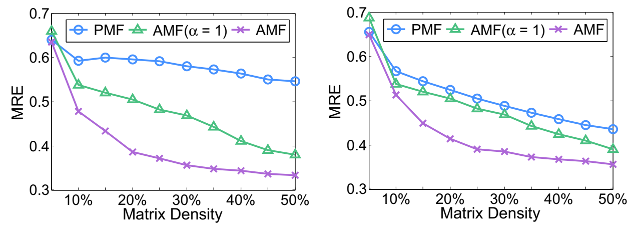
(a) Response Time (b) Throughput

Fig. 10. Distribution of Prediction Errors



(a) Response Time (b) Throughput

Fig. 12. Impact of Matrix Density



(a) Response Time (b) Throughput

Fig. 11. Impact of Data Transformation

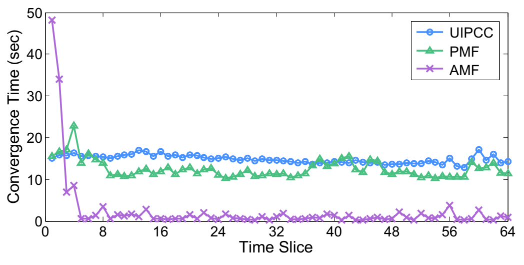


Fig. 13. Efficiency Result

achieves higher accuracy over MAE than UPCC and IPCC as reported in [17], and PMF achieves better performance compared with the first three approaches as reported in [23], all these approaches have large errors over MRE and NPRE. Thus, only minimizing the absolute error may lead to large relative error, which is not suitable for QoS prediction problem.

To further analyze the benefit of our AMF approach, we plot the distributions of prediction errors in Fig. 10. We can observe that AMF achieves denser distribution around the center 0, while UIPCC and PMF have flat error distributions, which indicates the better performance of AMF.

D. Impact of Data Transformation

The effect of data transformation on data distributions has been illustrated in Fig. 8. To further evaluate the impact of data transformation on prediction accuracy, we compare the prediction accuracy among three approaches, including PMF, AMF($\alpha = 1$), and AMF. In AMF($\alpha = 1$), $\alpha = 1$ indicates that the data transformation is relaxed to a linear normalization procedure, since the effect of the function $boxcox(x)$ is masked. In contrast, AMF is our approach with a well-tuned α (*e.g.*, $\alpha = -0.007$ for response time and $\alpha = -0.05$ for throughput). In this experiment, we also vary the matrix density and then compute the corresponding MRE values. The results are illustrated in Fig. 11. We can observe that the data transformation method has a significant impact on improving prediction accuracy over MRE. Especially, the PMF approach aggressively minimizes the absolute error, resulting in large MRE as shown in Fig. 11. Besides, AMF improves a lot in MRE compared with AMF($\alpha = 1$) due to the effect of Box-Cox transformation on QoS data distributions.

E. Impact of Matrix Density

To present a comprehensive evaluation on the impact of the matrix density, we vary the matrix density from 5% to 50% at a step increase of 5%. Besides, we set the other parameters as in Section V-C. Fig. 12 illustrates the evaluation results. We can observe that as the matrix density increases, better prediction accuracy can be achieved. In particular, the error decreases dramatically with the increase of matrix density, when the QoS matrix is excessively sparse (*e.g.*, matrix density = 5%). It shows that the model can fall into the overfitting problem due to data sparsity. With more data collected, the overfitting problem can be alleviated, thus further improving QoS prediction accuracy.

F. Efficiency Analysis

To evaluate the efficiency of our approach, we compare the convergence time of AMF with two other approaches, UIPCC and PMF. As we can see in Fig. 13, despite the long convergence time for the first time slice, our AMF approach becomes quite fast in the following time slices because AMF incrementally updates the model by online learning using sequentially observed data samples. In contrast, UIPCC and PMF are more computationally expensive, since they need to re-train the whole model at each time slice, which incurs high computational overhead compared to our online algorithm. Thus, they are more appropriate for one-time training as used in traditional recommender system.

G. Scalability Analysis

To analyze the scalability of our AMF model on new users and services, we evaluate the prediction accuracy on these new users and services, as well as the robustness of the prediction results. For this purpose, we simulate the new users and services from our dataset. Specifically, we randomly select 80% of users and services from our dataset at time slice 1 as existing users and services, and then train the AMF model using their data. After the model converges, we add the remaining 20% of users and services into the model at time $t = 400s$. Ideally, by using our algorithm 1, AMF can scale well to the new users and services, and perform robustly by keeping updating the feature vectors of existing users and

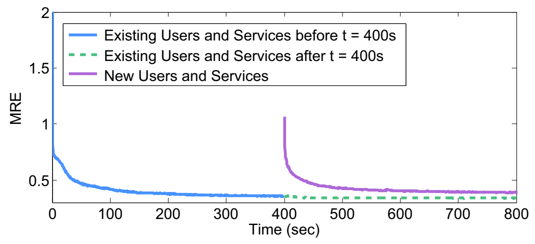


Fig. 14. Scalability Result

services with small weights, and the feature vectors of new users and services with large weights. Fig. 14 presents the results, where we can see that the MRE for the new users and services rapidly decreases after their joining. However, the MRE for existing users and services still keep stable, which indicates the robustness of our model under the churning of users and services. Therefore, our AMF approach shows good scalability on new users and services.

VI. Conclusion

This is the first work to study the problem of QoS prediction on candidate services for service adaptation. Towards this end, we propose adaptive matrix factorization (AMF) to address the online QoS prediction problem that is fundamental for runtime service adaptation. AMF formulates the QoS prediction problem as a collaborative filtering problem inspired from recommender systems, and extends the traditional matrix factorization model with techniques of data transformation, online learning, and adaptive weights, in order to address the unique challenges faced in runtime service adaptation. Comprehensive experiments based on a real-world QoS dataset have been conducted to evaluate our AMF approach, which demonstrates its good performance in achieving accuracy, efficiency, and scalability.

Acknowledgment

The work described in this paper was supported by the National Basic Research Program of China (973 Project No. 2011CB302603), the National Natural Science Foundation of China (Project No. 61332010 and Project No. 61100078), and the Research Grants Council of the Hong Kong Special Administrative Region, China (No. N_CUHK405/11 of the NSFC/RGC Joint Research Scheme and No. 415212 of the General Research Fund).

References

[1] Q. Zhang, Q. Zhu, M. F. Zhani, and R. Boutaba, "Dynamic service placement in geographically distributed clouds," in *Proc. of IEEE ICDCS*, 2012, pp. 526–535.

[2] V. Nallur and R. Bahsoon, "A decentralized self-adaptation mechanism for service-based applications in the cloud," *IEEE Trans. Software Eng.*, vol. 39, no. 5, pp. 591–612, 2013.

[3] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," in *Proc. of ACM SOSP*, 2007, pp. 205–220.

[4] O. Moser, F. Rosenberg, and S. Dustdar, "Non-intrusive monitoring and service adaptation for ws-bpel," in *Proc. of ACM WWW*, 2008.

[5] L. Zhang, J. Zhang, and H. Cai, *Services Computing: Core Enabling Technology of the Modern Services Industry*. Tsinghua University Press, 2007.

[6] C. Wang and J.-L. Pazat, "A two-phase online prediction approach for accurate and timely adaptation decision," in *Proc. of IEEE SCC*, 2012, pp. 218–225.

[7] P. Leitner, A. Michlmayr, F. Rosenberg, and S. Dustdar, "Monitoring, prediction and prevention of SLA violations in composite services," in *Proc. of IEEE ICWS*, 2010, pp. 369–376.

[8] A. Amin, L. Grunske, and A. Colman, "An automated approach to forecasting qos attributes based on linear and non-linear time series modeling," in *Proc. of IEEE/ACM ASE*, 2012, pp. 130–139.

[9] V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. L. Presti, and R. Mirandola, "MOSES: A framework for QoS driven runtime adaptation of service-oriented systems," *IEEE Trans. Software Eng.*, vol. 38, no. 5, pp. 1138–1159, 2012.

[10] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM Trans. Auton. Adapt. Syst.*, vol. 4, no. 2, pp. 14:1–14:42, 2009.

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

[12] L. Baresi and S. Guinea, "Self-supervising bpel processes," *IEEE Trans. Software Eng.*, vol. 37, no. 2, pp. 247–263, 2011.

[13] X. Bu, J. Rao, and C.-Z. Xu, "A reinforcement learning approach to online web systems auto-configuration," in *Proc. of IEEE ICDCS*, 2009, pp. 2–11.

[14] A. Metzger, C.-H. Chi, Y. Engel, and A. Marconi, "Research challenges on online service quality prediction for proactive adaptation," in *Proc. of the 2012 Workshop on European Software Services and Systems Research - Results and Challenges (S-Cube)*, 2012, pp. 51–57.

[15] B. Jiang, W. K. Chan, Z. Zhang, and T. H. Tse, "Where to adapt dynamic service compositions," in *Proc. of ACM WWW*, 2009.

[16] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. Artificial Intellegence*, 2009.

[17] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Qos-aware web service recommendation by collaborative filtering," *IEEE T. Services Computing*, vol. 4, no. 2, pp. 140–152, 2011.

[18] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "An extended matrix factorization approach for qos prediction in service selection," in *Proc. of IEEE SCC*, 2012.

[19] Z. Zheng and M. R. Lyu, "Collaborative reliability prediction of service-oriented systems," in *Proc. of ACM/IEEE ICSE*, 2010, pp. 35–44.

[20] C. Delimitrou and C. Kozyrakis, "Paragon: Qos-aware scheduling for heterogeneous datacenters," in *Proc. of ACM ASPLOS*, 2013.

[21] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. of NIPS*, 2007.

[22] J. Zhu, Z. Zheng, and M. R. Lyu, "DR^2: Dynamic request routing for tolerating latency variability in online cloud applications," in *Proc. of IEEE CLOUD*, 2013, pp. 589–596.

[23] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service qos prediction via neighborhood integrated matrix factorization," *IEEE T. Services Computing*, vol. 6, no. 3, pp. 289–299, 2013.

[24] R. M. Sakia, "The box-cox transformation technique: A review," *Journal of the Royal Statistical Society. Series D (The Statistician)*, vol. 41, no. 2, pp. 169–178, 1992.

[25] A. Shapiro and Y. Wardi, "Convergence analysis of gradient descent stochastic algorithms," *Journal of Optimization Theory and Applications*, pp. 45–4, 1996.

[26] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *Proc. of ICML*, 2003, pp. 720–727.

[27] "Exponential moving average," http://en.wikipedia.org/wiki/Moving_average, [Accessed: 5-Apr-2014].

[28] F. Dabek, R. Cox, M. F. Kaashoek, and R. Morris, "Vivaldi: a decentralized network coordinate system," in *Proc. of ACM SIGCOMM*, 2004, pp. 15–26.

[29] Y. Zhang, Z. Zheng, and M. R. Lyu, "WSPred: A time-aware personalized qos prediction framework for web services," in *Proc. of IEEE ISSRE*, 2011.

[30] "Singular value decomposition (SVD)," http://en.wikipedia.org/wiki/Singular_value_decomposition, [Accessed: 5-Apr-2014].

[31] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "Towards online, accurate, and scalable QoS prediction for runtime service adaptation (supplementary)," in *Supplementary Report*, 2014, [Available at: http://rmblab.github.io/icdcs2014_AMF.html].