International Journal of Information Management 34 (2014) 319–328

Contents lists available at ScienceDirect

International Journal of Information Management

journal homepage: www.elsevier.com/locate/ijinfomgt

Towards cost efficient mobile service and information management in ubiquitous environment with cloud resource scheduling

Xin Li [a], Zhuzhong Qian [a], Ilsun You [b,*], Sanglu Lu [a]

[a] State Key Laboratory for Novel Software Technology, Nanjing University, China
[b] School of Information Science, Korean Bible University, South Korea

ARTICLE INFO

Article history:
Available online 26 December 2013

MSC:
68-06
68M01

Keywords:
Cloud computing
Cost efficient
Mobile service
Service composition

ABSTRACT

The past few years have witnessed an explosive popularity of mobile services, especially in the form of smart phone applications. To cope with the limited batteries and computational capacities of mobile devices, prior studies suggest to deploy service instances in clouds for accomplishing most of the computation-intensive tasks. Service composition, which compensates for the simplicity of single service, is an effective way to utilize the plentiful services on the clouds all over the world. In this paper, we focus on the problem of service instance selection with service instance replica limitation constraint. The objective is to select the optimal set of service instances, which composes the integrated service and brings out the optimal QoS (quality of service), in terms of service response time. To characterize the problem, we establish a new QoS model, which considers the comprehensive quality over all users, not just for any single user or service instance. We prove that the problem is NP-hard, since many functionally equivalent service instances spread all over the distributed clouds. To address the problem, we classify the problem into three cases, including two special cases and the general case. We present two effective heuristic algorithms to determine the service instances selection for the two special cases, which are still NP-hard. The two special cases provide empirical bounds for the general case. We propose an algorithm that simulates a vote procedure for the users in the general case. The selected service instances, which come from the vote procedure, can satisfy a majority of users. We conduct extensive simulations for all of the algorithms. The simulation results show that our algorithms work efficiently on service response time reduction.

1. Introduction

It is an attractive issue to offer all kinds of mobile services to the users, since mobile devices are ubiquitous in our real life. It is a big market and magnificent opportunity for the service providers. Technically, service is an effective method to take the advantage of computational utility and meet the users' demands. To overcome the limitation on computational capabilities and batteries of mobile devices, e.g. smart phones, services are widely deployed on clouds. With the development of cloud computing (Sultan, 2013), especially the development of SaaS (Software as a Service), the cloud infrastructure is the best option to deploy all kinds of services. Many services/applications are accessible in clouds, e.g. online social applications (Wang, Li, Sun, & Yang, 2012), and various services still need to be deployed on clouds (Changa, Waltersb, & Wills, 2013). This vision motivates us to study the service deployment problem in the cloud environment.

In this paper, we focus on the service deployment via employing a set of existing service instances and compose them to be an integrated service. We conduct this work on the basis of the following two reasons: (1) It is cost-consuming and time-consuming for small service providers to deploy services on clouds individually. This is because the service provide must deal with the issues of resource management in the clouds, e.g., dynamical resource requirement, workload prediction, resource monitor, etc. It will cause high cost for the service provider. (2) Single service is always insufficient to satisfy the users, because of its functional simplicity. Service composition is an effective method to utilize the plentiful services in clouds all over the world, and makes it possible to satisfy diverse user demands and overcome the complex computation context.

Fig. 1 shows an example of service composition with 3 basic simple services. The composite service can provide rich functions for the user, while any simple service $F_i (1 \le i \le 3)$ is insufficient to meet the user demands. To provide the composite service, the primary task for the service provider is to select concrete service instance(s) for each abstract task F_i. Because of the plenteousness of services,

* Corresponding author.
E-mail addresses: lixin@dislab.nju.edu.cn (X. Li), qzz@nju.edu.cn (Z. Qian), isyou@bible.ac.kr, kbu.isyou@gmail.com (I. You), sanglu@nju.edu.cn (S. Lu).

0268-4012/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.ijinfomgt.2013.11.007

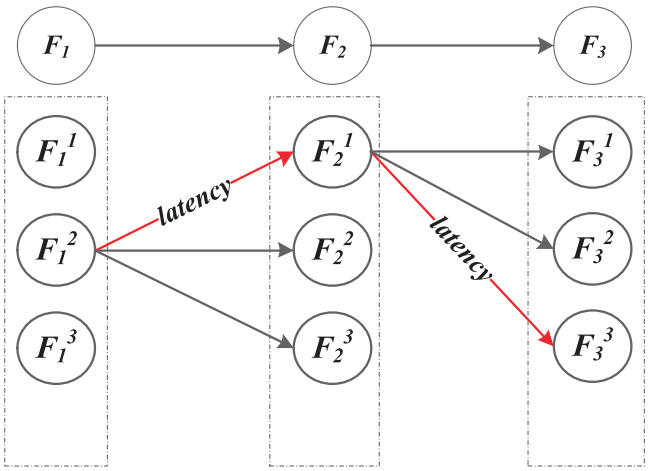

Fig. 1. An example of service composition with 3 basic simple services

there may be many functional equivalent service instances in the clouds with various QoS (quality of service) attributes (e.g. executing time, access latency). As shown in Fig. 1, we should select one or more concrete service instances for each abstract task F_i from the functional equivalent instances set $\{F_i^j\}(1 \le j \le \theta)$.

Hence, in this paper, we will study the service instance selection problem with the goal of minimizing the response time for the users. For example, in Fig. 1, F_1^2, F_2^1, and F_3^3 are selected to realize the composite service, which indicates that the path (red line), induced by F_1^2, F_2^1, and F_3^3, brings the least service response time for the user. In this case, one service instance is selected for each abstract task. Actually, we will also consider the case that multiple instances can be selected for each abstract task. We aim to select a service instance set to provide services with the best QoS for wide-area users in the long run. As more service instances are selected for each abstract service task, users can get better QoS. However, in the scenario of cloud computing, it is cost consuming to hire too many service instances. Hence, due to the cost constraint, there exists a quantitative limitation when selecting the service instances. The quantitative limitation and the diversity of service instances make it a challenging problem to select the optimal service instance set, which guarantees the best QoS. Here, QoS is defined as a metric for the entire service, not for any individual user.

We set out to investigate the service selection problem to optimize the total QoS, while being subjected to the service instance quantitative limitation constraint. We take the response time, the most concerning QoS attribute for the mobile users, as the metric to judge the quality of the selected service instance set. We aim to propose effective algorithms to select the optimal service set. The algorithms can act as guidance for the service providers. To calculate the response time for the users, we make the following settings. First, we let both user and service instance have a location, which is represented by a 2-dimensional coordinate system. User location implies where the service request starts from. Service instance location indicates where the service request is handled. There exists communication latency between any two locations. Second, we estimate the communication latency via the general network coordinate system, which is also adopted by Klein, Ishikawa, and Honiden (2012).

On the basis of the network and QoS model, we formulate the service instance selection problem, and prove the problem to be NP-hard. Then, we classify the problem into three cases according to the service composition settings. We first study a special case, that there is only one abstract task, and we can select multiple concrete service instances for it. We prove that this special still be NP-hard, and present a heuristic algorithm based on the distribution of service instances and users. For the second case, we let there is only one concrete service instance can be selected for each abstract task, while there are multiple abstract tasks. We present a greedy algorithm that selects the service instances step by step. In the third case, which is also the general case, where there are multiple abstract tasks and multiple concrete service instances can be selected for each abstract task. We present an algorithm that simulates a vote procedure for the users, and select the service instances which can satisfy a majority of users. For all of the algorithms, we evaluate them via extensive simulations, and the simulation results show their efficiency. Compared to the random selection algorithm, our approaches are better at improving the composite service QoS, the service response time can be reduced significantly.

The rest of the paper is organized as follows: we present the network model and QoS model in Section 2. In Section 3, we formalize the service instance selection problem and prove it to be NP-hard. The three cases of the problem are discussed in Section 4, and various algorithms are proposed. We evaluate the algorithms in Section 5. Then, we briefly present the related work in Section 6, and we make a conclusion in Section 7.

2. Service composition and QoS model

In this section, we present the preliminaries of the service instance selection problem. First, we formalize the service composition and introduce some notations in this paper. Then, we propose the network model to estimate the network latency. At the end, QoS model is introduced to measure the performance of the solutions.

2.1. Service composition

To eliminate the ambiguity of the notation of service composition, we define service composition as:

Definition 1 *((Service composition)).* To accomplish an integrated task, many basic processing units are needed to achieve the complicated job cooperatively, mainly by sequence without loop. The process of integrating basic processing units is known as **service composition**.

In this paper, we call the basic processing unit **service (functional) component**, while the final integrated service is named **composite service**.

Usually, a service composition task works based on a functional graph, which defines the logic execution sequence of service components and the possible composition patterns. Functional graph is defined as:

Definition 2 *((Functional graph)).* A functional graph is defined as $FG = \langle F, E, \lambda \rangle$, where F is the set of functional components represented by function name F, and E is the set of functional edges between these components. λ is the number of components.

There are two kinds of executing patterns in this paper, sequence pattern and conditional pattern; there is no loop pattern and parallel pattern. For example, in Fig. 2, F_1 and F_2 run in sequence pattern; after that, either F_3, F_4, or F_5 is selected to run (conditional pattern). No loop pattern or parallel pattern is considered.

Due to the conditional pattern, there may be many composition routes, which means a completed path from initial component to the terminal component. There may be many paths with different initial components and terminal components for the functional graph. To simplify the presentation, we let the paths of the functional graph share the same initial component and terminal component. Actually, it is easy to add a virtual source component acting as initial component if there are many initial components.