Computers and Electrical Engineering 41 (2015) 18–27

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

Ant colony optimization applied to web service compositions in cloud computing ☆

Qiang Yu [a], Ling Chen [a,b,*], Bin Li [a,b]

[a] Institute of Information Science and Technology, Yangzhou University, Yangzhou 225009, China
[b] National Key Lab of Novel Software Tech, Nanjing University, Nanjing 210093, China

ARTICLE INFO

Article history:
Received 28 February 2013
Received in revised form 2 December 2014
Accepted 4 December 2014
Available online 23 December 2014

Keywords:
Service composition
Multi-cloud base
Cloud combination
Ant colony optimization

ABSTRACT

Rapid developments in cloud computing technology mean that many different web services are now published and available in cloud data centers. There has recently been an increasing amount of interest in web service composition, because it is important in practical applications. However, most traditional service composition methods can only find service composition sequences in a single cloud, and cannot consider a multi-cloud service base. It is challenging to efficiently find a composition across multiple clouds, because it involves service compositions and combinatorial optimization. In this paper, we present a greedy algorithm called *Greedy-WSC* and an ant colony optimization based algorithm called *ACO-WSC*, which attempt to select cloud combinations that are feasible and use the minimum number of clouds. Our experimental results show that the proposed ant colony optimization method can effectively and efficiently find cloud combinations with a minimal number of clouds.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

Service oriented architecture (SOA) is a recent development in software architecture. The goal of SOA development is to build platform independent software components (called services), to improve system quality and productivity. Service rendering can be implemented using an SOA based method called service composition, and is typically implemented in a distributed environment on the internet [1]. Different kinds of web services have been published and made available for individual users and organizations. Cloud computing is becoming a prominent platform for providing web services. For instance, Microsoft has published its cloud computing platform, Windows Azure [2], which provides computing resources and cloud data storage centers. Other companies (like Amazon [3] and Salesforce [4]) provide similar cloud computing infrastructures. A huge amount of research has been devoted to areas such as service discovery. The resulting techniques are currently powerful enough to help web service requesters find standalone web services.

In many cases, the web service model consists of multiple service providers, and no single service can satisfy a user's requirements. Service components from different providers can be conveniently integrated into a composite service, regardless of their locations, platforms and/or execution speeds. Therefore, we need a service composition to combine

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Associate Editor Dr. Jian Li.

* Corresponding author at: Institute of Information Science and Technology, Yangzhou University, Yangzhou 225009, China. Tel.: +86 514 87978320.
E-mail address: lchen@yzu.edu.cn (L. Chen).

http://dx.doi.org/10.1016/j.compeleceng.2014.12.004
0045-7906/© 2014 Elsevier Ltd. All rights reserved.

several correlative web services together and fulfill the user's goal. Web service composition has received an increasing amount of research attention in the last few years.

Most existing service composition methods assume that all web services found in a composition sequence come from the same service repository. Instead of finding web services from multiple locations, these methods only discover composite web services that are stored in one service repository. However, increasing demands mean that many service providers have emerged (such as Windows Azure Platform [2] and Amazon S3 [3]). Many service providers have started to offer different QoS (quality of service) levels, to meet the needs of different user groups. In this case, traditional methods cannot find a web services composition from multiple service repositories.

A composite web service aggregates multiple atomic and composite web services, which interact according to a given process model. Each atomic web service is an indivisible software component and can implement a task of the business process underlying a composite web service. Given a request for a composite service, atomic services must be selected within a reasonably short time, especially for interactive and real-time applications. A speedy composition is required for urgent tasks and because a user may risk losing customers if they are frustrated by a long wait.

Communication between web services from different clouds is expensive and time consuming, so we need to find a valid composition that uses a minimum number of clouds. In this situation, however, it is hard to find an appropriate composition sequence. Therefore, effectively and efficiently choosing a service plan that minimizes the number of clouds is an open issue.

There have been recent advances in methods for delivering services over multi-vendor cloud resources [5–11]. Qi et al. [12] presented a QoS-aware composition method for supporting cross-platform service composition in a cloud environment. Lucas-Simarro et al. [13] presented a modular broker architecture that can work with different scheduling strategies to optimally deploy virtual services across multiple clouds based on different optimization criteria. Bastião Silva et al. [14] presented a platform that allows applications to interoperate with distinct cloud providers' services using a normalized interface. The proposed approach provides a common API that minimizes the present deficits of cloud APIs. Casalicchio et al. [15] analyzed the perspective of an application service provider using a cloud infrastructure, so as to scalabley provide its services with respect to QoS constraints. Zhang [16] proposed a cloud computing open architecture, and noted that virtualization and service-oriented architecture are two key techniques. Wang [17] introduced several enabling techniques for cloud computing. Zou et al. [18] proposed a service composition framework for multi-cloud base environments. They also presented three different cloud combination methods called *All Clouds*, *Base Cloud*, and *Smart Cloud*, which aim to select cloud combinations with a feasible composition sequence and a minimum number of clouds. In [19,20], methods were proposed for semantic web service composition using clustering and ant colony algorithms. Zhang et al. [21] proposed a QoS-based dynamic service composition method for web services based on ant colony optimization (ACO). They used ACO to solve the multi-objective optimal-path selection problem. Pejman et al. surveyed methods for web service composition in [22].

To evaluate and compare automatic cloud service composition methods, we must construct a test environment using some standards. The proposed composer can find the optimal composition length in each challenge set within a reasonable time. Mohammad et al. [23] gave a short survey that explored testing methods for cloud services.

In this work, we focused on web service composition methods that use heuristic optimization. We considered the web service composition as an optimization problem, and used ACO to search for the best service composition in multi-cloud base environments.

Because of the expensive communication costs when using web services from different clouds, the goal of our work was to effectively and efficiently minimize the number of clouds involved in a service composition sequence. This is a challenging problem, because we must consider the constraints provided by the services when optimizing the cloud selection. We present greedy and ACO algorithms for selecting a feasible cloud combination that also uses a minimum number of clouds.

ACO is an evolution simulation algorithm proposed by Dorigo et al. [24]. It was inspired by the behaviour of a real ant colony, after the authors noticed similarities between the ants' food-hunting activities and the travelling salesman problem (TSP). Dorigo et al. successfully solved the TSP problem using the same principle that ants use to find the shortest route to a food source, that is, communication and cooperation. ACO has been successfully applied to system fault detecting, job-shop scheduling, frequency assignment, network load balancing, graph coloring, robotics and other combinational optimization problems [24].

In ACO, artificial ants travel in a graph to search for optimal paths according to pheromone and problem-specific local heuristics information. Pheromone information is assigned to the edges of the graph, and is evaporated at a certain rate at each iteration. It is also updated according to the quality of the solutions containing this edge. ACO has some advantages such as allowing positive feedback, distributed computing, and a constructive greedy heuristic search.

Based on ACO, we present the *ACO-WSC* (web service composition) algorithm, which selects the cloud combination that contains a minimum number of clouds. Our experimental results show that the proposed ACO based method can effectively and efficiently find a desirable cloud combination for a web service composition.

The remainder of this paper is organized as follows. In Section 2, we present the problem formulation. In Section 3, we present a greedy algorithm (*Greedy-WSC*) for web service composition in a multi-cloud environment. An ACO based algorithm (*ACO-WSC*) for selecting a cloud combination is presented in Section 4. We give experimental results and our analysis in Section 5, and conclude the paper in Section 6.

Download English Version:

https://daneshyari.com/en/article/455290

Download Persian Version:

https://daneshyari.com/article/455290

Daneshyari.com