An Optimization of QoS-Aware Web Service Selection Method Using PSO

Dr. Srikanth Thodeti

1HOD, Department of Computer Science, N.S.V. Degree College, Jagityal, Telangana, India.

Abstract—Cloud computing is a web primarily based computing and is turning into popular. QoS scores offer treasured data for making finest cloud provider selection from a hard and fast of functionally equal carrier candidates. To attain QoS values, actual-global invocations at the service candidates are commonly required. In this take a look at, we advocate a effective QoS-conscious webservice selection technique. This method adopts a particle swarm optimization set of rules choose the most service with clientsQoS necessities. Investigational results show that our method can find good appropriate services with low time value in web service selection.

Index Terms—QoS, Web service, Discovery, semantic web technologies

I. INTRODUCTION

Cloud computing is Internet-based computing, whereby shared configurable resources (e.g., infrastructure, platform, and software) are provided to computers and other devices as services. Strongly promoted by the leading industrial companies (e.g., Amazon, Google, Microsoft, IBM, etc.), cloud computing is quickly becoming popular in recent years. Applications deployed in the cloud environment are typically largescale and complex. With the rising popularity of cloud computing, how to build high-quality cloud applications becomes an urgently required research problem. Similar to traditional component-based systems, cloud applications typically involve multiple cloud components communicating with each other over application programming interfaces, such as through web services. On-functional performance of cloud services is usually described by quality-of-service (QoS). QoS is an important research topic in cloud computing. When making optimal cloud service selection from a set of functionally equivalent services, QoS values of cloud services provide valuable information to assist decision making. In traditional component-based systems, software components are invoked locally, while in cloud applications, cloud services are invoked remotely by Internet connections. Client-side performance of cloud services is thus greatly influenced by the unpredictable Internet connections. Therefore, different cloud applications may receive different levels of quality for the same cloud service. In other words, the QoS ranking of cloud services for a user cannot be transferred directly to another user since the locations of the cloud applications are quite different. Personalized cloud service QoS ranking is thus required for different cloud applications. The most straightforward approach of personalized cloud service QoS ranking is to evaluate all the candidate services at the user-side and rank the services based on the observed QoS values. However, this approach is impractical in reality, since invocations of cloud services may be charged. Even if the invocations are free, executing a large number of service invocations is time consuming and resource consuming, and some service invocations may produce irreversible effects in the real world. Moreover, when the number of candidate services is large, it is difficult for the cloud application designers to evaluate all the cloud services efficiently. To attack this critical challenge, we propose a personalized ranking prediction framework, named Cloud Rank, to predict the QoS ranking of a set of cloud services without requiring additional real-world service invocations from the intended users. Our approach takes advantage of the past usage experiences of other users for making personalized ranking prediction for the current user. Extended from its preliminary conference version, the contribution of this paper is twofold: This paper identifies the critical problem of personalized QoS ranking for cloud services and proposes a QoS ranking prediction framework to address the problem. To the best of our knowledge, Cloud Rank is the first personalized QoS ranking...
prediction framework for cloud services. Extensive real-world experiments are conducted to study the ranking prediction accuracy of our ranking prediction algorithms compared with other competing ranking algorithms. The experimental results show the effectiveness of our approach. We openly release our service QoS data set1 for future research, which makes our investigations reproducible.

Publishing, discovering and invoking Web services are the key functions that a Web services platform needs to support. A Standard Web Service Brokerage Model illustrated in Fig.1, which is a standard picture in most web service literature, the idea of bringing together web service providers and users via some form of brokerage service has been a core part of the web service vision from the start. Further, to provide a discovery technology that allows easy and precise service discovery by businesses or clients all over the Internet is among several basic issues the platform designer needs to consider. Discovery methods can range from manual to automatic [1] has summarized the discovery methods.

![Diagram](Fig 1: Web Service Discovery)

**II. RELATED WORKS**

In service composition, service candidates have a number of different QoS attributes, leading to large variation in their QoS attribute values or scopes. Considering global QoS constraints, computing or valuating QoS is different for each service candidate. So QoS utility function was proposed in [7]. The function maps the quality vector Qs into a single real value, enabling sorting and ranking of service candidates and simplifying choosing to satisfy QoS constraints of the service components. The QoS utility function in this paper is similar to [8].

Match making is considered as one of the crucial factors to ensure dynamic discovery and composition of web services. Current match making methods are in adequate given their inability to abstract and classify the underlying data of web services. Match making is considered as a search or discovery problem wherein service consumers attempt to locate required web services in order to accomplish their tasks. UDDI is developed inorder to advertise and discover web services [5]. Current match making techniques do not abstract and classify the underlying data associated with web services. Fuzzy logic enables data representation with linguistic variables and fuzzy values. The suggested fuzzy matchmaking framework aims to abstractly represent the underlying data of web services using fuzzy logic and semantic web technologies in order to optimize the discovery process.

One of the challenging problems that Web service technology faces is the ability to rapidly locate useful online services based on their capabilities. With the increasing popularity of Web services, a wide variety of services are offered that can satisfy the functional requirements of a special task, therefore there is need for mediation infrastructures able to support humans or agents in the eventual selection of appropriate services. To address this problem, a fuzzy multi-attribute decision making algorithm for Web services selection based on quality of service is proposed, which can select the most appropriate one with the highest degree of membership belonging to the positive ideal solution.

Web service [6] framework brings in a new revolution in traditional computing. Besides modeling the web service selection problem as FMCDM (Fuzzy Multiple Criteria Decision Making), we introduce a synthetic weight which combines both the subjective and objective weights. For subjective weights defined by human preference, we apply linguistic variables and fuzzy numbers. For objective weights, we investigate entropy concepts to improve the judgment consistency. The larger the entropy value, the lower the information express quantity, the entropy weighting is an effective measurement for the average essence of information quantity.
III. PROPOSED WORK

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.

Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called lbest when a particle takes all the population as its topological neighbors, the best value is a global best and is called gbest.

The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its pbest and lbest locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward pbest and lbest locations. In past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods.

Another reason that PSO is attractive is that there are few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement.

In this paper, we use PSO to discovery the best service candidate for each service class (i.e. the optimal particle). First, a suitable coding scheme should be designed for searching the optimal quality control line. We use integer array coding scheme on the basis of the characters of the best service. The number of items in the array denotes the number of service classes, and each element of the array denotes the index of service candidates. The maximum of the element is the number of service candidates, and the minimum is 1. The velocities of the particles are also encoded as n-dimensional arrays, and each element value of the velocity is an integer that is within $[V_{min}, V_{max}]$. In PSO, Every particle uses its own information, local best information, and global optimum to produce a new particle for better fitness, through parallel global search. When the algorithm stops, it obtains the best scheme. In this section, we evaluate the computation time of proposed approach. In Fig. 2, we evaluate the computation time of our approach with respect to the number of service candidates. In this experiment, the number of service classes is 10. In Fig. 2, we evaluate the computation time with respect to the number of service classes. In this experiment, the number of service candidates is 50.

From the Fig. 2, regardless of the number of service candidates, the computation time of our proposed approach is obviously short. In Fig. 2, when the number of service candidates is 50, the computation time is only 300.2 millisecond (msec). It is far shorter than 1 second. When the number of service candidates is 500, the computation time is 430.1 msec. The time cost is very short and also shorter than 1 second. Hence, according to its average computation time (351.1 msec), our proposed approach is very suitable to service selection for users with real-time requirement.

![Fig. 2 Computation time with the number of service candidates.](image)
From the Fig. 3, regardless of the number of service classes, the computation time of our proposed approach is obviously short. In Fig. 3, when the number of service classes is 5, the computation time is only 296.3 millisecond (msec). It is far shorter than 0.5 second. When the number of service candidates is 50, the computation time is 549.7 msec. The time cost is very short and also shorter than 1 second. Hence, according to its average computation time (415.6 msec), our proposed approach is very suitable to service selection for users with real-time requirement. Hence, according to the two experimental results of Figs. 2 and 3, our proposed approach is very effective for service selection.

IV. CONCLUSION

While the current studies often cause high time complexity and are out of the real-time requirements, in the paper, we suggest an methodology with particle swarm optimization. The comparison results with a existing scheme show that our proposed approach is efficient and can satisfy the real-time requirement for the fast selection of web services. However, there are some limitations in this scheme. For instance, there are no clear-cut guidelines on what would be the best choice to some parameters of particle swarm optimization such as w and c. Consequently, in order to endorse and deepen continuing researches in the future, it is worthwhile to investigate more studies to uncover the valuable new study issues.

REFERENCES


Author:

Dr. Srikanth Thodeti, working as HoD, Department of Computer Science in N.S.V. Degree College, Jagityal, Telangana, India. I received Master of Computer Applications (M.C.A.) from Kakatiya University in April 2004 and received Ph.D. (Computer Science) from Dravidian University on Feb 25th 2017.