ISSN 2395-1621

Compatibility-Aware Cloud Service Composition under Fuzzy Preferences and weightage of services

[#1]Sushant Barne, [#2]Shraddha Gade, [#3]Amita Malusare, [#4]Nilesh Pawar.

[1]sushantbarne@gmail.com
[2]shraddhagade93@gmail.com
[3]amitamalusare54@gmail.com
[4]nnnpawar7@gmail.com

[#1234]Department of computer Engineering, Shri Chatrapari Shivajiraje College of Engineering, Dhangwadi, Savitribai Phule Pune University, Maharashtra, India.

ABSTRACT

Many times user needs more than one service to fulfill their needs because one single service can't satisfy all the requirements of the user that's why cloud service composition is required. In this service compositions there are more than one services are present which are preferable for user. Selecting more than one service can be difficult task for non-technical user and also it is time consuming and tedious task. If system provides the combination of frequently used services to the user then task of selecting the more than one service for non-technical user will become very easy task/process. We develop an ontology based approach to analyze Cloud service compatibility by applying reasoning on the expert knowledge. In addition, to minimize effort of users in expressing their preferences, we apply combination of evolutionary algorithms and fuzzy logic for composition optimization. This lets users express their needs in linguistics terms which brings a great comfort to them compared to systems that force users to assign exact weights for all preferences.

Keywords- Cloud Computing, Cloud Service Composition, Fuzzy Logic, Ontology, Weightage of Services.

ARTICLE INFO

Article History

Received :28th May 2016

Received in revised form :

28th May 2016

Accepted : 1st June , 2016

Published online :

3rd June 2016

I. INTRODUCTION

Opposed Cloud Computing is emerging as a new processing paradigm based on out- sourcing infrastructure on a pay-as-you-go basis, accommodating services ranging from private data processing to public website hosting. Despite much attention from the research community, research and development of Cloud Computing systems, applications, services and resource management are still in their infancy.

In order to deliver their solutions, application service providers can either utilize the Platform as- a-Service (PaaS) offerings such as Google App Engine and Open Shift or develop their own hosting environments by leasing virtual machines from Infrastructure-as-a-Service (IaaS) providers like Amazon EC2 or GoGrid.

However, most PaaS services have restrictions on the programming language, development platform, and databases that can be used to develop applications. Such restrictions encourage service providers to build their own platforms using IaaS service offerings.

One of the key challenges in building a platform for deploying applications is to automatically compose, configure, and deploy the necessary application that consists of a number of different components. If we consider the deployment requirements of a web application service provider, it will include security devices (e.g. firewall), load balancers, web servers, application servers, database servers, and storage devices. Setting up such a complex combination of appliances is costly and error prone even in traditional hosting environments let alone in Clouds. Virtual appliances provide an elegant solution for this problem. They are built and configured with a necessary operating system and software packages to meet software requirements of a user. In IBM smart Cloud, Amazon EC2, GoGrid, Rackspace, and other key players in the IaaS market, users first have to select their software solution (asset catalogs, images, etc.) and then select the proper virtual machine configuration (e.g. instance type) to host the software solution. Furthermore, a

user may require more than one virtual appliance and machine, and a composition of them that can meet all the requirements of users is required. However, the selection of the best composition is a complex task and none of the providers provide any ranking system to choose the best instance type and software solution for the deployment. In addition, the best choices found for individual appliances cannot be simply put together as they may not be compatible with the hosting environment. In this paper, to simplify the process of selecting the best virtual appliance and unit (computing instance) composition, a novel framework is presented.

An example of such systems is the one which require users to assign weights to their objectives. In this case, users have to find a way to prioritize their preferences and then map them to weights. After that, the system has to find out how precise users have gone through the process of weight assignment. To tackle this issue, a major objective of this research is to offer ranking system for Cloud service (i.e. virtual appliance and machine) composition that let users express their preferences conveniently using high-level linguistic rules. Our system then utilizes multi-objective evolutionary approaches and fuzzy inference system to precisely capture the entered preferences for ranking purpose.

II. PREVIOUS WORK

The proposed architecture offers a unified solution that uniquely applies state of the art technologies of semantic services, agent negotiation, and multi objective and constraints optimization to satisfy the requirements of whole service deployment life cycle. The main goal of the architecture is to provide: ease of use for non-experts, semantic interoperability, more precise discovery and selection, more reliable Service Level Agreement (SLA) monitoring, and automatic negotiation strategy. The main components are described below:

1) User Portal: All services provided by the system are presented via the web portal to clients. This component provides graphical interfaces to capture users' requirements such as software, hardware, QoS requirements, firewall, and scaling settings. In addition, it transforms user requirements to WSMO format in the form of goals which are then used for Cloud service discovery and composition. For more details regarding the format of goals, readers can refer to our previous work. Moreover, it contains an account manager, which is responsible for user management.

3) Cloud Service Repositories: They are represented by appliance and virtual unit service repositories and allow IaaS providers to advertise their services. For example, an advertisement of a computing instance can contain descriptions of its features, costs, and the validity time of the advertisement. From standardization perspective, a common meta model that describes IaaS provider's services has to be created. However, due to the lack of standards, we developed our own meta model based on previous works.

4) Discovery and Negotiation Service: This component maps user's requirements to resources using the ontology-based discovery technique. It acts in user's interest to satisfy quality of service (QoS) requirements by selecting the set of eligible IaaS providers. The negotiation service uses a time-dependent negotiation strategy that captures preferences of users on QoS criteria to maximize their utility functions while only accepting reliable offers. These negotiation strategies are described in our previous work.

5) Composition Optimizer: Once the negotiation completed and eligible candidates are identified, the composition component, which is the focus of this paper, builds the possible compositions and uses the compatibility checking component to eliminate incompatible candidates. Then Composition Optimizer evaluates the composition candidates using the users' QoS preferences. The Composition Optimizer takes advantage of multi-objective algorithm and fuzzy logic to let users express their preference conveniently and efficiently selects the closest candidates to users' interests. Throughout the paper we show how ontology-based compatibility checking, multiobjective algorithms, and fuzzy logic techniques can work in harmony to provide an elegant solution to the composition problem.

6) Planning: The planning component determines the order of appliance deployment on the selected IaaS providers and plans for the deployment in the quickest possible manner.

7) Image Packaging: The Packaging component builds the discovered virtual appliances and the relevant meta-data into deployable packages, such as Amazon Machine Image (AMI) or Open Virtualization Format (OVF) packages. Then the packages are deployed to the selected IaaS provider using the deployment component.

8) Deployment Component: It configures and sets up the virtual appliances and computing instances with the necessary configurations such as firewall and scaling settings. For example in a web application, specific connection details about the database server need to be configured.

10) Appliance Administration Service: After the deployment phase, this component helps end users to manage their appliances (for example starting, stopping, or redeploying them). It uses the Deployment Descriptor to manage the deployed services.

11) Failure Recovery: It automatically backs up virtual appliance data and redeploys them in the event of Cloud service failure.

The first step in service composition is modelling the Cloud services (based on cost, size, functionality, etc.) and QoS requirements of users. This step not only allows appliance and virtual unit providers to advertise QoS of their services, but also provides a way for end users to express their QoS preferences. In this work, WSMO is extended and used for appliance and virtual unit QoS modelling. This allows us to model Compatibility and legal constraints required to build a valid composition. However, the process of converting the Cloud service advertisements to WSML is time-consuming and error-prone if it is carried out manually.

III. MATHEMATICAL MODEL

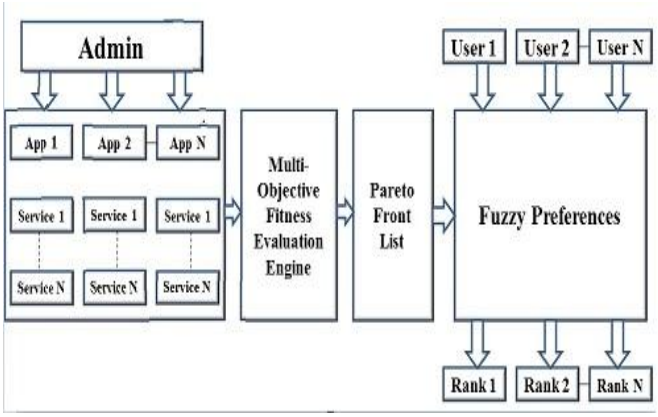
Let us consider a set S
where,
S= {U, R, SER, A, S, Fuzzy Cmean()}
Here,

S: System which includes,
U: Set of Users Where U= {U1, U2, U3 …, Un}
R: Set of Requests.
Where R= {R1, R2, R3……., Rn}.
SER: Server.
A: Set of applications,
Where A= {A1, A2, A3, An}
S: Are services.
Success Conditions: If the user preferable composition is created successfully then we get success state. Failure Conditions: If the user preferable composition is not created successfully then we get failure state.

IV. SYSTEM IMPLEMENTATION

In our problem we consider three user objectives, the lowest cost, quickest deployment time, and the highest reliability. This makes it infeasible to find an optimal composition as these objectives can conflict with each other. One way to address this problem is to convert the appliance composition problem to a single-objective problem by asking users to give weights for all the objectives.

However, this approach is error-prone and impractical, as not all the users have the knowledge to accurately assign weights to objectives. Furthermore, since the composition solutions will depend on the capability of users to assign proper weight to the objectives, additionally we have to find a way to evaluate the knowledge of users about each objective to ensure the accuracy of the approach.



V. CONCLUSION

Thus, our system helps non-expert users with limited or no knowledge on legal and image format compatibility issues to deploy their services faultlessly. In addition, we proposed a technique to optimize the service composition based on user preferences such as deployment time, cost, and reliability.

REFRENCES

[1] Amir Vahid Dastjerdi, Member, IEEE and Rajkumar Buyya, Senior Member, IEEE, "Compatibility-aware Cloud Service Composition Under Fuzzy Preferences of Users" ,2014
[2] P. Wang, "QoS-aware web services selection with intuitionistic fuzzy set under consumer's vague perception," Expert Systems with Applications, vol. 36, no. 3, pp. 4460–4466, 2009.
[3] Nouha Khediri and Montaceur Zaghdoud, "Survey of uncertainty handling in cloud Service discovery and composition" , 2014
[4] A. Dastjerdi and R. Buyya, "An autonomous reliability-aware negotiation strategy for cloud computing environments," in Proceedings of 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE, 2012.
[5] Pavithra.M. ME, S.K.Mahalingam ME,Ph.D., "Trust Driven Workflow Scheduling By Composition of Cloud Services under Fuzzy Preferences of Users", 2015
[6] A. Dastjerdi, S. Tabatabaei, and R. Buyya, "A dependency aware ontology-based approach for deploying service level agreement monitoring services in cloud," Software: Practice and Experience, vol. 42, no. 4, pp. 501–518, 2011.
[7] A. Dastjerdi, S. Tabatabaei, and R. Buyya, "An effective architecture for automated appliance management system applying ontology-based cloud discovery," in Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010.
[8] P. Cingolani and J. Alcala-Fdez, "jfuzzylogic: a robust and flexible fuzzy-logic inference system language implementation," in Proceedings of International Conference on Fuzzy Systems, 2012.
[9] F. Rosenberg, M. Muller, P. Leitner, A. Michlmayr, A. Bouguettaya,and S. Dustdar, "Metaheuristic optimization of large-scale QoS-aware service compositions," in Proceedings of IEEE International Conference on Services Computing, 2010.
[10] T. Pham, H. Truong, and S. Dustdar, "Elastic high performance applications–a composition framework," in Proceedings of IEEE Asia-Pacific Services Computing Conference, 2011.

