Building Virtual Instance Compatible with User's Web Application in Openstack Cloud Provider

C.Sumathi[1], Dr.B.Chandramouli[2]
[1]PG Student, [2]Professor
[1 2] Department of CSE, [1 2] Sri Ramanujar Engineering College, TamilNadu, India

Abstract-- When a single Cloud service (i.e., a software image and a virtual machine), on its own, cannot satisfy all the user requirements, a composition of Cloud services is required. Cloud service composition, which includes several tasks such as discovery, compatibility checking, selection, and deployment, is a complex process and users find it difficult to select the best one among the hundreds, if not thousands, of possible compositions available. Service composition in Cloud raises even new challenges caused by diversity of users with different expertise requiring their applications to be deployed across difference geographical locations with distinct legal constraints. The main difficulty lies in selecting a combination of virtual appliances(software images) and infrastructure services that are compatible and satisfy a user with vague preferences. Therefore, an algorithms which simplify Cloud service composition for unskilled users and develop an approach to analyze Cloud service compatibility by applying reasoning on the expert knowledge. In addition, to minimize effort of users in expressing their preferences . 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; Open Stack Cloud provider, PaaS IaaS*

I. INTRODUCTION

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 Go Grid. 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 [1], 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 t0o meet software requirements of a user. In IBM smart Cloud,

Amazon EC2, Go Grid, Rack space, 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. Moreover, there exist legal constraints imposed by countries such as the USA on importing and exporting of appliance from a provider to another. Dealing with all these complexities is costly and aggravating for unskilled users and encourages them to seek professional help. In this paper, to simplify the process of selecting the best virtual appliance and unit (computing instance) composition, a novel framework is presented. The framework proposes: An approach to help non-expert users with limited or no knowledge on legal and virtual appliance image format compatibility issues to deploy their services flawlessly. The knowledge base then is used for reasoning in an algorithm that identifies whether a set of Cloud services consisting of virtual appliance and units are compatible or not. A Cloud service composition optimization technique that allows non-expert Cloud users to set their preferences using high level if-then rules and get user friendly recommendations on the composition solution's prominence.

The majority of end users avoid systems that incur complexity in capturing their constraints, objectives and preferences. 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. RELATED WORK

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. 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.

A. OBJECTIVE OF THE PROJECT

This project is non- expert users with limited or no knowledge on legal and virtual appliance image format compatibility issues to deploy their services flawlessly and find the best combination of compatible virtual appliances and virtual machines

B. EXISTING SYSTEM

In Existing System, non-expert users are not aware of composition of cloud services that are compatible or not compatible . They are not aware of web applications dependencies and Configuration details. Their deployed applications can be misconfigured. Real time process of virtual instances is not supported.

Individual appliances cannot be simply put together as they may not be compatible with the hosting environment. Setting up complex combination of appliances is costly and error prone. None of the providers provide any ranking system to choose the best instance type and software solution for the deployment. The virtual instances are not supported for real time process. The provider can provide limited instances. Setting up such a complex combination of appliances is costly.

III. PROPOSED WORK

Proposed system uses Open Stack Cloud provider which is the best to build Virtual Instance. Cloud services consisting of virtual appliance and units are compatible with each other. Our system helps non-expert users with limited or no knowledge on legal and image format compatibility issues to deploy their services faultlessly. User can build their own instance based on his/her requirements. Web applications dependencies are available, the only thing is user has to select their web application dependencies based on his/her preferences. User can deploy application using Cygwin terminal. Real time process can be viewed in Open Stack Dashboard.

Merits of Proposed System:

To optimize the service composition based on user preferences such as deployment time , cost and reliability.Using open stack cloud service provider is easy to build a virtual instances.In the proposed system non-expert users with limited images and also users build their own instances based on requirements.

IV. SYSTEM REQUIREMENT SPECIFICATIONS

A. Purpose

The main aim of this project is check the compatibility of the Web Application with the Cloud Service composition and to build a Virtual Machine (Instance) for Deployment and various Preferences of Users.

B. Project Scope

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 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. 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.

C. Product Perspective

In Existing System, non-expert users are not aware of composition of cloud services that are compatible or not. They are not aware of web applications dependencies and Configuration details. Their deployed applications can be misconfigured. Real time process of virtual instances is not supported.

Individual appliances cannot be simply put together as they may not be compatible with the hosting environment. Setting up complex combination of appliances is costly and error prone. None of the providers provide any ranking system to choose the best instance type and software solution for the deployment.

D. Product Features

Proposed system uses open stack Cloud provider which is the best to build Virtual Instance. Cloud services consisting of virtual appliance and units are compatible with each other. The system helps non-expert users with limited or no knowledge on legal and image format compatibility issues to deploy their services faultlessly. User can build their own instance based on his/her requirements. Web applications dependencies are available, the only thing is user has to select their web application dependencies based on his/her preferences. User can deploy application using Cygwin terminal. Real time process can be viewed in open stack Dashboard.

E. User Classes and Characteristics

1) Cloud Service Composition

The user has to register his/her details. The server in turn stores the user information in its database. User can modify their information and updated information stored in database.

Advertisement of two default images of service provider is available in this module. Default images details stored in database. User buys images based on their requirements. At buy image stage, user redirects into GSVC Internet Banking. After successful transaction, the amount will be debited from his/her account. The user checks their bought instances in launching an Instance module. User owned instances details are stored in database.

2) Build and Configure Instance

User can create new instances based on his/her requirements. In this module, user can select their Web application dependency which includes Operating Systems, RAM, and Database etc. Sample web applications are available and then user can configure web application. Compatibility checking of cloud services is done in this module. If the cloud services are compatible, user redirects into GSVC Internet Banking. After successful transaction, the amount will be debited from his/her account. The user checks their newly created instances in launching an Instance module.

3) Launching an Instance and Deployment

In this module, instances owned by the user and running instances are displayed. Using jclouds, user can launch and stop their instances. Information about instances are displayed and stored in database. In this module, user can check their active instances in web page or in openstack Dashboard. Giving scp command in Cygwin Terminal, user can deploy their web applications. Real time process can be shown in openstack Dashboard.

F. System Features

Open stack Cloud provider is the best to build Virtual Instance. To build virtual instance, 4GB of storage is enough. Cloud services consisting of virtual appliance and units are compatible with each other. Our system helps non-expert users with limited or no knowledge on legal and image format compatibility issues to deploy their services faultlessly. User can build their own instance based on his/her requirements. Web applications dependencies are available, the only thing is user has to select their web application dependencies based on his/her preferences. User can deploy application using Cygwin terminal. Real time process can be viewed in open stack Dashboard.

V. ALGORITHM & ARCHITECTURE

When a user request is submitted through the cloud service repository, the site shows the advertisement of virtual image for relevant data and returns the results to the user. Specifically, the instance of the user is buy the image or else create the new instances. The process is done with available virtual image and the user's need is not satisfied the given domain create the own instance infrastructure-as-a service requirement will be provided the cloud service.

A. Data Flow Diagram (DFD)

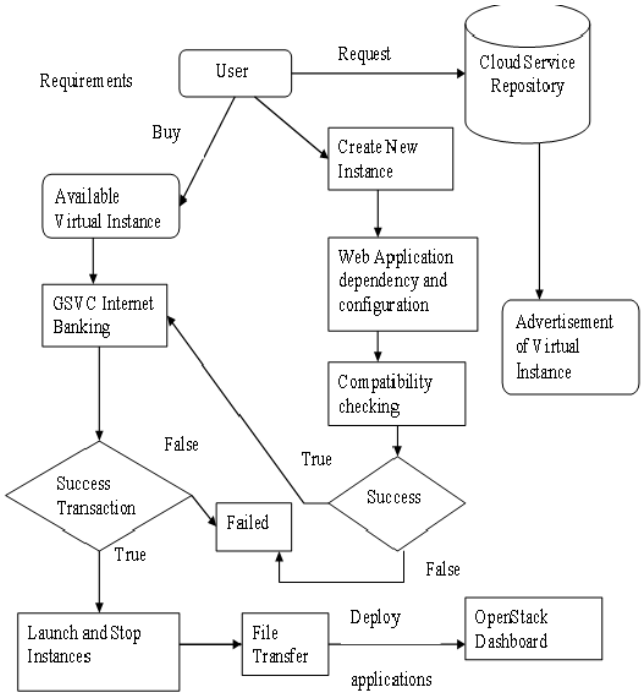

Fig System Architecture

Data Flow Diagram is a major tool available to analysis for communicating with user, it graphically represents the flow of data through system and services as a model of a system. It identifies the path that the data take in the processes to its final destination. Logical DFDs are also found to be easier since they show the sequence of transformation or conversion of data by different processes of the system. The following steps are followed to draw a logical DFD:

B. MODULES DESCRIPTION

1) Cloud Service Composition

The user has to register his/her details. The server in turn stores the user information in its database. User can modify their information and updated information stored in database. Advertisement of two default images of service provider is available in this module. Default images details stored in database. User buys images based on their requirements. At buy image stage, user redirects into GSVC Internet Banking. After successful transaction, the amount will be debited from his/her account. The user checks their bought instances in launching an Instance module. User owned instances details are stored in database.

2) Build and Configure Instance

User can create new instances based on his/her requirements. In this module, user can select their Web application dependency which includes Operating Systems, RAM, and Database etc. Sample web applications are available and then user can configure web application.

Compatibility checking of cloud services is done in this module. If the cloud services are compatible, user redirects into GSVC Internet Banking. After successful transaction, the amount will be debited from his/her account. The user checks their newly created instances in launching an Instance module.

3) Launching an Instance and Deployment

In this module, instances owned by the user and running instances are displayed. Using jclouds, user can launch and stop their instances. Information about instances are displayed and stored in database. In this module, user can check their active instances in web page or in Open Stack Dashboard. Giving scp command in Cygwin Terminal, user can deploy their web applications. Real time process can be shown in Open Stack Dashboard.

VI. CONCLUSIONS

In order to tackle Cloud service composition challenges, We presented an ontology-based approach to describe services and their qos properties, which helped us to build a composition with a set of compatible services. 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 preferencessuch as deployment time, cost, and reliability. we can effectively help an unskilled user to identify the appliance compositions which are closest to their preferences.. Integer Linear Programming (ILP), Pseudo-Boolean Optimisation (PBO), and Multiple Objective Ant Colony Optimization (MOACO) are all competitive algorithms, and with the best of our knowledge, for this particular problem, there is no in depth comparison between all three and OMOPSO. Therefore, providing such a comparison can be considered as another possible future work. Moreover, Cloud services have specific characteristics and QoS dimensions which have to be identified. Specifically, defining criteria which are able to model energy and carbon emission efficiency, reliability, and trust of a Cloud service are increasingly attractive to users. For example, methods to evaluate reliability and trust of providers from user feedbacks and monitoring services together can be further studied.

This consists of collecting required raw data from trusted sources and statistically analyzing and aggregating them. In addition, user experience is another important benchmark for Cloud service providers. Recently, crowd sourcing is being used to create collective knowledge to assess QoS. It requires an investigation on discovering the crowd that can evaluate a service efficiently, delegating evaluation tasks to crowds, and calculating the accuracy of the aggregated assessments. The current implementation of the translator component supports XML-based Cloud services aims at providing a standard way for describing Cloud resources. Therefore, investigating approaches that can semantically enrich these new formalisms can be taken as a future direction to further enhance the translator component.

REFERENCES

[1]. C. Sun, L. He, Q. Wang, and R. Willenborg, “Simplifying service deployment with virtual appliances,” in Proceedings of the IEEE International Conference on Services Computing (SCC), 2008.

[2]. J. De Bruijn, H. Lausen, A. Polleres, and D. Fensel, “The web service modeling language wsml: An overview,” in Proceedings of the 3rd European conference on The Semantic Web: research and applications, 2006.

[3]. A. Dastjerdi, S. Tabatabaei, and R. Buyya, “An effective architecture for automated appliance management system applying ontology-based cloud discovery,” 2010

[4]. A. Dastjerdi and R. Buyya, “An autonomous reliability-aware negotiation strategy for cloud computing environments,” 2012.

[5]. DMTF, “Open virtualization format,” http://www.dmtf.org/ standards/ovf.

[6]. A. Dastjerdi, S. Tabatabaei, and R. Buyya, “A dependency aware ontology-based approach for deploying service level agreement monitoring services in cloud,” 2011.

[7]. I. Foster, Y. Zhao, I. Raicu, and S. Lu, “Cloud computing and grid computing 360-degree compared,” in Proceedings of the Grid Computing Environments Workshop (GCE), 2008.

[8]. J. Kopeck` y, D. Roman, T. Vitvar, M. Moran, and A. Mocan, “Wsmo grounding. wsmo working draft v0. 1, 2007.”

[9]. D. Lambert, N. Benn, and J. Domingue, “Integrating heterogeneous web service styles with flexible semantic web services groundings,” 2010.

[10]. R. Barth and C. Smith, “International regulation of encryption: technology will drive policy,” Borders in Cyberspace: Information Policy and Global Information Infrastructure, pp. 283–299, 19