Implementation of Service Selection in Multi-Cloud Environment Using Cloud Sim

Prof. Shilpa Veerabhadrappa

Lecturer

BCA Department

KLE's JT College, Gadag.

Abstract : Cloud computing is one of the emerging field in the world of technology. In previous years, the cloud computing emerged with several problems which included vendor lock-in and interoperability issues across the distinct cloud platforms. And hence, the users are finding the problem in choosing the best service provider that meets their requirements. The Cloud service broker (CSB) is used to find the optimal service that satisfies the user's requirements. In this paper, we provide an essential support for simulation using the CloudSim toolkit. Given the various service offerings, we propose the architecture for providing an optimal service. This relies on the Rank-Selection algorithm which aims at selecting the best service provider based on the requirements of the user.

***Keywords*—Quality-of-Service, Rank-Selection Algorithm, Service Provider, Cloudsim, OCCI.**

I. INTRODUCTION

Cloud computing [1] is one of the emerging field in the world of technology. It supplies the on-demand computing of services to both the individuals as well as organizations in the form of various types of services. They manifest an interest in providing the services that increases the growth of the business and reduces the operational cost. And hence increases the desire on the services that may meet their functional and non-functional requirements. Cloud offers the hosted services through the internet. Many organizations have started migrating themselves from their IT infrastructure to the cloud. Several companies such as Amazon, Microsoft, and IBM are offering their own cloud solutions. Applications that are provided by means of cloud computing is called as cloud services. In terms of cloud services, the three main models are: Infrastructure-as-a-Service (IaaS), Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS).

An end-user using the cloud through the web-based applications can access hi/her documents, files whenever he/she wants and wherever he/she is. The end user can store the large amount data in the cloud than on personal computer systems. Documents and files are stored in the cloud permanently, no matter what happens to the user's PC. In addition to this, cloud computing encourages the group collaboration between the users from the different physical locations can collaborate by sharing files efficiently at lower costs. As the number of services offered by the service providers increases exponentially, the users may find difficulty in choosing the right service provider that fulfills user's requirements. It becomes more tedious job for the end user to find the each service present under each service provider. And this type of a discovery may prevent the user from choosing the best and optimal service. Thus, cloud service providers loose their confidence in offering the variety of services. The success of each service providers depends on the cloud users satisfaction based of QoS (Quality-of-Services). Suppose if the services are not delivered properly as expected by the users, this may affect the service provider's belief. Consider a scenario, where the user searches for a particular service for example compute.

The number of web pages is returned for the compute service. It becomes very difficult for the user to find the right service provider and a right service. And also becomes difficult to make the combinations between the functional and non-functional requirements which the normal search engines don't provide According to the proposed model, there is a user, service provider and a mediator. The request from the user will be sent to the broker that uses the Rank-selection algorithm to find an optimized solution for that request based on user's requirement. The cloud computing has become a very large marketplace in the world of business. Now, this area mainly focuses on the research and experimentation. Once the services are found, before deploying these services to the cloud tests must be conducted using the simulation tools. Testing is a must for any organizations that are providing the cloud services, which in turn reduces the risk during the deployment of the cloud Various simulation tools are available for testing, one of them is CloudSim[2]. It is a toolkit that allows the customers to model as well as simulate the cloud and also run the programs. The structure of this paper is organized as follows. The next section of the paper we discuss the works related to cloud service broker frameworks. The algorithm for selecting the best service provider is discussed in section 3. Then it is followed by the implementation in section 4.The evaluation results are drafted in section 5. Then we concluded with the brief summary and also describe the future work section 6.

II. BACKGROUND

CloudAnalyst[3] is a graphical simulation tool constructed on the uppermost part of the CloudSim. It was developed by CLOUDS (Cloud Computing and Distributed Systems) labs at University of Melbourne. CloudAnalyst is used to model and examine methodically the behavior of social network applications. The traffic routing between the remote users and datacenters are controlled by this tool. According to several routing policies, each service provider will resolve that which datacenter must attend to the requests from the end users.The OPTIMIS[4] project is a toolkit, that is proposed for flexible multi-cloud architecture. The broker provides decision making that takes into account many aspects such as trust, risk and cost. This particular tool is not implemented; they conducted experiments using the cost and risk aspects as the policies in decision making. Comparing the above mentioned approaches, their implementation on real Clouds is not implemented completely and their current validations and evaluations are mostly based on simulation methodologies. The framework that is presented in this paper is also implemented based on the simulation approach.

III. FRAMEWORK OVERVIEW

The components used in the system designed below are

1. Service Provider
2. Rank-Selection algorithm or CSB
3. Database
4. Service Consumer

The architectural design figure 1 shows the conceptual model of the system. Here we have considered a central Database which is directly connected to Rank-selection algorithm. Every service provider present on the cloud bind their data or services to the central data base. When user enter some request through web browser. The request first goes to Rank-selection algorithm, discover module search the availability of the service in the central database. If it is present the functional and non functional module asks the user to enter the desire requirement. Based on the user requirement Rank-selection algorithm will calculate the utility function[6] on both service provider side and consumer side. If any service provider's utility value exceed the user's minimum threshold value then it is acceptable by user.

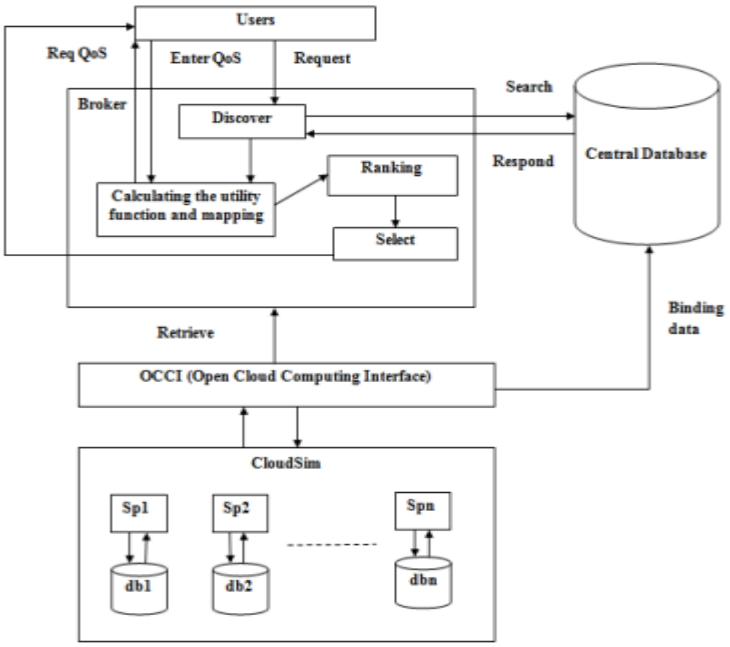


Figure 1: Architecture Design for the Rank-Selection Algorithm.

A. Algorithm

- Let P={X1, X2… Xn} be the vector of QoS parameters (availability, cost, speed) considered in the system.

- Let M={m1,m2…..mn} be the vector of minimal requirements that the consumer tolerates for the required type of the service, with 0 ≤ mini≤ 1.
- Let Si={SP1,SP2….SPk} be the set of service providers who provide services to the service consumers.
- Let Qi = {q1i,q2i….. qni} be the QoS provided by each service provider.

Rank-Selection Algorithm()

Input: *Enter the request req and the requirements that are needed for the service.*

Output*: Displays the best service provider.*

1. if (req=database(data)) then

2. Forward to the requirements entry page.

3. Else display the error message and go to 1.

4. Calculate the utility function for user side (US).

5. Based on the weight entered by the user in the nonfunctional requirements page, calculate the utility function for server side (SS).

6. Loop:

if(SS>=US) then store them and rank all service providers in descending order of their utility function value.

End loop

7. Loop:

if(SS[0]==SS[i]) then Count the number of service providers and print their respective URL

else print the first service provider's URL.

End loop

Utility function is given by,

$$U = w_1 x_1^{\beta 1} + w_2 x_2^{\beta 2} + \cdots + w_n x_n^{\beta n}|$$

Where,

$x_1 \ldots\ldots x_n$ is the number of QoS parameters and $W_1 \ldots\ldots W_n$ are the weights assigned to each of these parameters, such that $0 \leq X \leq 1$ and $0 \leq W \leq 1$. β_i is a measure of the service consumer sensitivity to the QoS attribute $\bar{x}_i$ As β_i increases, the service consumer is expressing increasing concern about x_i.

IV. IMPLEMENTATION

We implemented the simulation environment for the architecture presented in the previous section. The implementation details are discussed in this section.

A. CloudSim

CloudSim is a toolkit that provides the features such as modeling and simulation for infrastructures including data centers, brokers, hosts and virtual machines (VMs) on a single host. It also provides allocation and scheduling policies in the simulation made the CloudSim attractive tool. In literature, authors in paper [5] provide similar architecture but only for infrastructure services. In our simulation environment CloudSim is used to simulate the IaaS, SaaS, and PaaS present in the multi-cloud environment. The simulation is built on the top of CloudSim 2.2.1 toolkit which is depicted in the figure 2.

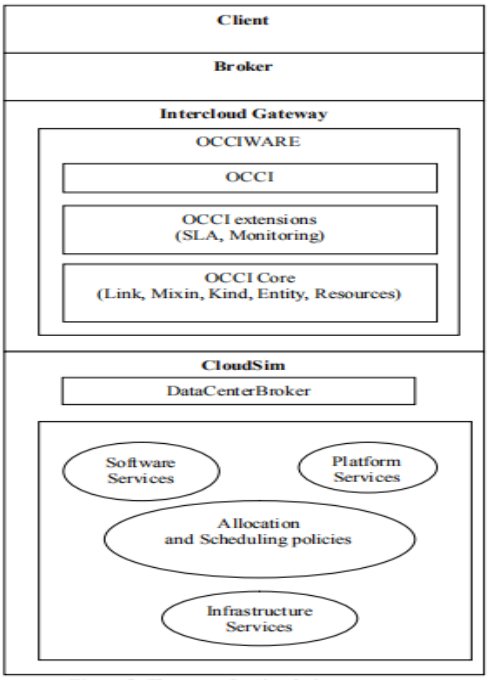


Figure 2: The setup for simulation.

The above figure 2 shows the simulation environment setup where it includes CloudSim, occi-based intercloud gateway, broker and the client. Initially the CloudSim was setup using the Eclipse IDE, the steps for this is as follows:

Step 1: Create the new java project in the IDE.

Step2: Select the project, under library add the cloudsim package on "Add Jars/Folders"

Step3: Import the org package into the source package.

S**tep4**: To check whether it is working properly, run the project.

B. OCCI (Open Cloud Computing Interface)

In order to simulate the Intercloud Gateway component, we implemented, based on the open source Java implementation for OCCI[7][8] called OCCI4JAVA. The communication between broker and providers is forwarded to the CloudSim DatacenterBroker class through standard OCCI-interfaces. Next step after installing the cloudsim is setting up the OCCI; here we need to configure the Eclipse IDE as follows:

(i) In eclipse: install the following tools from http://download.eclipse.org/releases/mars: Mylyn WikiText (for Wiki syntax support, normally pre-installed/in-built eclipse mars vers), Eclipse Modeling Framework (EMF SDK & EMF COMPARE), Acceleo Core SDK, Xtext Complete SDK, Sirius Specification Environment (check that Sirius Runtime is included), OCL Examples and Editors SDK.

(ii) (ii) In the workspace directory: git-clone the main project https://github.com/occiware/ecore (containing all projects).

(iii) In eclipse: import the all projects from "/workspace/ecore" directory (File > Import > Existing Project into Workspace).

(iv) Verify that there are no errors on compiled projects. If errors are marked and the projects can't compile, close projects that are in "trash" and in "xtext" path (including occi2ecore, clouddesigner.occi.runtime). Also check the Java (JRE) version (right-click on project > properties).

(v) Using the using Acceleo (on Obeo's CloudDesigner) create a new .occie file.

After configuring the eclipse the next step is to implement the broker that makes use of Rank-Selection Algorithm that is

covered in the next section of this paper. The requirements required to setup the above simulation environment is

- Operating System : Windows 8.1 Pro,7, XP
- Processor : Intel(R) Pentium(R) CPU N3520 @ 2.16GHz
- RAM : 2.00 GB
- System Type : 32-bit Operating System, x64-based processor

V. RESULTS

As a proof of the above concept, we create a scenario where we considered few QoS parameters such as cost, speed and availability. Table 1 shows the minimum QoS requirements of an end user.

	Availability	Cost	Speed
w_i	0.35	0.2	0.3
m_i	0.4	0.5	0.6
β_i	1	1	1

Table 1: Minimum QoS requirements of the user

The minimum requirement found for the user after computing the utility function is 0.42. Assume there are three service providers, SP1, SP2, and SP3. And the QoS offerings that they provide is listed in table 2.

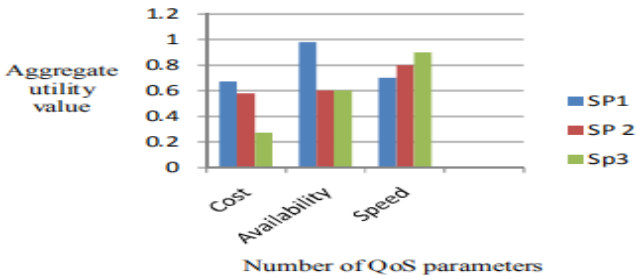
	Availability	Cost	Speed
Sp_1	0.5	0.5	0.8
Sp_2	0.7	0.6	0.7
Sp_3	0.1	0.4	0.5

Table 2: QoS offering of three service providers.

Calculating the utility function to each service provider and the values obtained are as follows and arrange it from highest to lowest.

SP1= 0.67, SP2= 0.58, SP3 = 0.27

This means that SP3 do not meet the minimum QoS requirements of the user.



VI. CONCLUSION

By following the cloud computing technology, and the growing number of service providers, service consumer will face the challenge of finding appropriate service providers that can satisfy user's functional and non-functional requirements. Initially the simulation environment is been setup using the CloudSim and an OCCI interface which is used to communicate between the broker and the service providers. When the user enters the request based on his requirements the optimal and best service provider is displayed to him. The results presented in this paper shows that the Rank-Selection algorithm selects the best service provider in the given simulation setup. Hence, as future work we define the composition of service based on the QoS requirements.

VII. REFERENCES

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, Special Publication 800-145, 2011.

[2] R. N. Calheiros, Rajiv Ranjan, Anton Beloglazov, C.A.F.De Rose, Rajkumar Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software Practice and Experience, Wiley publishers, 2010.

[3]B.Wickremasinghe, R. N. Calheiros and R. Buyya, "CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications," in AINA 2010, 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 446– 452, April 2010.

[4] A. J. Ferrer, et al., "OPTIMIS: A Holistic Approach to Cloud Service Provisioning," in Future Generation Computer Systems, vol. 28, pp. 66–77, January 2012.

[5] Foued Jrad, Jie Tao and Achim Streit " Simulation-based Evaluation of an Intercloud Service Broker" The Third International Conference on Cloud Computing, GRIDs, and Virtualization, CLOUD COMPUTING 2012 [6] A. AuYoung, L. Grit, J. Wiener, and J. Wilkes, ―Service contracts and aggregate utility functions,‖ In Proc. of the IEEE Symposium on High Performance Distributed Computing, pp. 119–131, 2006.

[7] S. Yangui, M. Mohamed, M. Sellami, and S. Tata. Open Cloud Computing Interface - Platform, May 2013.

[8] M. Sellami , S. Yangui, M. Mohamed and S. Tata. Open Cloud Computing Interface - Application, May 2013.