A Model of Cloud Based Application Environment for Software Testing

T. Vengattaraman, P. Dhavachelvan
Department of Computer Science
Pondicherry University
Puducherry, India
vengat.mailbox, dhavachelvan@gmail.com

R. Baskaran
Department of Computer Science and Engineering
Anna University
Chennai, India
baaski@cs.annauniv.edu

Abstract— Cloud computing is an emerging platform of service computing designed for swift and dynamic delivery of assured computing resources. Cloud computing provide Service-Level Agreements (SLAs) for guaranteed uptime availability for enabling convenient and on-demand network access to the distributed and shared computing resources. Though the cloud computing paradigm holds its potential status in the field of distributed computing, cloud platforms are not yet to the attention of majority of the researchers and practitioners. More specifically, still the researchers and practitioners community has fragmented and imperfect knowledge on cloud computing principles and techniques. In this context, one of the primary motivations of the work presented in this paper is to reveal the versatile merits of cloud computing paradigm and hence the objective of this work is defined to bring out the remarkable significances of cloud computing paradigm through an application environment. In this work, a cloud computing model for software testing is developed.

Keywords-component; Cloud Computing; Software Testing; Web Services

I. INTRODUCTION

Cloud computing is a recent evolution of distributed computing paradigm which can support on-demand service sharing with higher level of flexibility and dynamic scalability. Flexibility and scalability in cloud computing environment can be accomplished through load balancing of application instances running separately on a variety of operating systems and connected through Web services [1]. Cloud computing provide Service-Level Agreements (SLAs) for guaranteed uptime availability. Furthermore, cloud computing make use of a model for enabling convenient and on-demand network access to the distributed and sharable computing resources which are configurable [2] [4]. The greatest advantage of the cloud computing is the user need not have knowledge of the technology and control over the infrastructure in the cloud environment.

Though the cloud paradigm holds its potential status in the field of distributed computing, cloud platforms aren’t yet at the center of most people’s attention. More specifically, still the researchers and practitioners community has fragmented and imperfect knowledge on the principles and techniques of cloud computing paradigm. Due to these, other paradigms are seems to be better, but this scenario will be changed shortly on behalf of the real attractions of cloud based computing, including scalability and cost effective services [2-3]. So it is the responsibility of the researchers to make the practitioners to realize the flavor of cloud computing environment. In this context, one of the primary motivations of the work presented in this paper is to reveal the versatile merits of cloud computing paradigm and hence the objective of this work is defined to bring out the remarkable significances of cloud computing paradigm through an application environment. This development is aimed to a broad extent and its significances are realized in a limited scope, which can be enhanced to its appropriate extent.

Once a decision has made to implement a cloud environment, the designer is supposed to adopt a standard procedure for implementing the same. But at present there is no such standard procedure or guidelines are available for building the cloud computing environment and hence here the services are implemented using web service concepts. The technology of web service based systems has generated lots of interest among the user and developer in the recent years through its ability as a new paradigm for conceptualizing, designing, and implementing software systems. This characteristic is particularly attractive for creating software that operates in environments that are distributed and open, such as the internet and this property increases the need for web service systems that consist of services that communicate in a peer-to-peer fashion is becoming evident. Central to the design and effective operation of such web services are a core set of issues and research questions that have been studied over the years by the distributed community [10-12]. They are large-scale systems and the research community aims at web service collaboration to achieve their functions in a highly flexible manner. But unfortunately, in the existing systems, too few are described with sufficient details to adopt them for real world problems [8-9]. The solution is to construct and use objective specific frameworks that are best suited for the given problem space. From this perspective here, an objective specific web service based cloud framework for software testing is developed.
From the cloud management point of view, a critical challenge in creating effective clouds is making them robust against potential failures. In few cases, system failures may cause the poor performance of the application though it has been well developed and deployed. Thus, in addition to the functionality features, the application should be able to coordinate with the system that is handling the particular application. On the other hand, the cloud should also be implemented with such capabilities to coordinate with the applications. So, appropriate failure or exception handling in the cloud framework are not only to improve the quality of the system in terms of availability, also to improve the performance of the application domain. In this view, the possible set of exceptions and exception handling mechanism are also described in order to assist in management of clouds presented in this paper. The mechanisms can also be considered as generic and may be applied wherever they are required.

The organization of paper is as follows: Section 2 defines the proposed enhanced version of web service for software testing as an application domain. Section 3 explains the experimentation methodology to validate the proposed system in two perspectives; performance assessment of the framework w.r.t. the application environment at the service level and competence assessment of the framework w.r.t. the computing environment management in terms of cloud computing performance attributes. Section 4 presents the quantitative statistical analysis over the results obtained from the experiments and the Section 5 offer the concluding remarks and directions of future enhancements.

II. PROPOSED SYSTEM

A. Introduction of Cloud

Cloud computing is a recent evolution of distributed computing paradigm that uses the internet and central remote servers to maintain data and applications and thereby allows the users to access technology-enabled services from the Internet without knowledge of, expertise with, or control over the technology infrastructure that supports them. Dynamism and resource sharing are the key architectural indicators of cloud computing and this technology allows much more efficient computing by centralizing storage, memory, processing and bandwidth. In general, the cloud services can be grouped into three broad categories [1][3-7]:

Software as a Service (SaaS): It is a kind of application runs exclusively in the cloud. The services are typically hosted and managed in their own data center and make it available over the Web.

Platform as a Service (Paas): These categories of services are the on-premises applications that are to be built on shared platform in the cloud. This system facilitates development and deployment of applications without the cost and complexity of buying and managing the underlying infrastructure and enhances the on-premises application functionalities by accessing the application specific services provided in the cloud. They can be viewed as the set of attached services useable only by a particular application.

Infrastructure as a Service (IaaS): Infrastructure as a Service (IaaS) is the delivery of hardware and associated software as a service. A cloud platform provides cloud-based services for creating applications rather than building their own custom foundation, they can be built on a cloud platform.

The general interaction between the on-premises applications and cloud services [1] are described in the Fig 1. The work presented in the paper is based on the Paas category, which includes the application services of on-premises and cloud environments.

B. Proposed Model Development

In modeling the proposed System, it will be easier to understand the model by expressing it in the product based definition form. In this form, the model is used to define different sets of Testing Services with respect to the product, which is to be tested. In all the cases, the second category is the subset of the former one. From the product perspective, the proposed system can be defined in terms of product specific services.
Definition-1: Let ‘A’ be the set of Clouds needed for testing the product ‘P’, and with respect to P, ‘A’ can be defined as in the Equation (1)

\[
A_p = \left\{ \left( sm_{ij}, tc_{i1}, tc_{i2}, \ldots, tc_{iy} \right) \right\}
\]

\[
tc_{ij} = \left( ts_{i(1)}, ts_{i(2)}, \ldots, ts_{i(K_j)} \right),
\]

\[
tc_{ij} = \left( ts_{i(21)}, ts_{i(22)}, \ldots, ts_{i(2K_j)} \right),
\]

\[
tc_{ij} = \left( ts_{i(31)}, ts_{i(32)}, \ldots, ts_{i(3K_j)} \right)
\]

where,

- ‘sm’ is the Service Manager of the product ‘P’ belongs to the On-Premises Application Services and ‘H’ is the number of products to be tested simultaneously and then 0 < i ≤ H.

- ‘tc’ is the one of the Testing Cloud of ‘A’ and ‘tc_{ij(u)}’ is the one of the Testing Services of ‘tc’ and then 0 < j ≤ y , 0 < u ≤ K_j - 1.

- ‘y’ is the number of Clouds and also the number of different Testing Clouds required for the product ‘P’.

- ‘K’ refers to the maximum number of Testing Services of ‘a’ and ‘K’ refers to the total number of Testing Services available in the particular Testing Clouds ‘j’ of the product ‘P’.

Since this cloud framework provides, scalar type testing environment, at any instant, \( A_{i1} \cap A_{i2} = \emptyset \), where, 0 < i1, i2 ≤ H and i1 ≠ i2. i.e. at any specific service duration, neither a single Testing Cloud nor a Testing Services can be shared by more than one product simultaneously. The logical layout of the proposed cloud framework for software testing is shown in the Fig 2.

Generally there are three sets of components as Service Manager \( \left\{ sm_1, sm_2, \ldots, sm_y \right\} \), Testing Clouds \( \left\{ tc_1, tc_2, \ldots, tc_x \right\} \) and the clones of Testing Services \( \left\{ ts_{11}, ts_{12}, \ldots, ts_{1K_1}, \ldots, ts_{x1}, ts_{x2}, \ldots, ts_{xK_x} \right\} \) each of which runs locally on different machines in a network. The logical links between the services refers to the service level dependency among the services in the framework. Such a dependency can represent the fact that one service depends on another for a goal to be fulfilled, task to be performed, or a resource to be made available.

In specifying the proposed system, it has been found that it is highly desirable to explain the model in a layered approach; there are two types (levels) of services: On-Premises Application Services and Cloud Application Services. The cloud application services layer is the dynamic one, which will be generated at the time of the testing processes. The overall support is composed of units, which specialize in supporting different testing services.

The Service Manager is responsible for supervision and coordination of the core activities of respective layer entities of the overall environment. After getting the assignment from the user, the Service Manager (SM) will form the Testing Clouds and can define the set \( \{ tc_1, tc_2, \ldots, tc_x \} \). The process of defining the set of required clouds depends on the testing techniques required for any product ‘P’. Then, based on the load, each Testing Cloud will select one or more additional Testing Services of its respective techniques and define the sets of Testing Services. These additional services are to be referred to as secondary services of Service Manager.
Each cloud is responsible for providing a specific output standard to which the task output must conform. The output standard depends on the task type and the required output properties such as number of test cases, defects found, time spent for testing, etc. The input to the Consumer Service ‘CS’ is from the tester and it includes the set of the testing product, time specification for testing, defect detector estimations and the specification about the required testing techniques. This is transferred to the Service Manager ‘SM’ of ‘CS’. The Service Reception receives the input from the CS and analyze the specification about the required testing techniques. Based on the testing service specifications, the appropriate set of Clouds can be defined and identified through the negotiation process in the Cloud Application Services. Upon the agreement of the negotiation process ‘K’, Clouds are formed.

Then the assignments will be distributed to the identified set of Testing Cloud and their outputs can be obtained for integration. The Environmental Test Reports from the identified testing Clouds will be integrated in the testing management block and then passed to the external world (CS) through the service reception of SM. The Service Manager is responsible for all types of co-ordination activities in this system. The initial input to the Testing Cloud ‘tcj’ is from the Testing Application Management. It includes the set of the testing techniques, time specification for testing, defect detector estimations and the estimated complexity of the product. This is transferred to the Testing Clouds ‘tcj’ with a set of estimated values on average size of the modules of the product and the predicted values of total number of test cases to be built by tcj, average size of the test case and average time required for generating and executing an unit test case.

Service Manager will define the mode of load distribution and it is based on the input from the Consumer Service. This is also responsible for defining the number of Testing Services that are needed to generate. Moreover processes of registration, load distribution and collection of results from the Testing Services are to be done. At the same time it will distribute the load to the Testing Clouds of same tcj. The results from the Testing Clouds i.e. Environmental Partial Test Reports (EPTRs) from the Testing Services will be processed in the Testing Application Management. Then the generated Environmental Test Report (ETR) of tcj will be transferred to the Consumer Service CS.

III. CONCLUSION

In this work, a model of cloud computing environment for software testing has been developed with different types of clouds. This work is one of the very few examples of cloud frameworks constructed for problem solving in software engineering field, which is being explicitly integrated as a service in a distributed network environment and validated with multiple folds; framework assessment with respect to the application environment and framework assessment with respect to the computing environment management. From the software engineering perspective, it demonstrated how a new web service paradigm enables legacy software to be incorporated in a relatively straightforward manner under any kind of circumstances.

REFERENCES

[1] David Cheppal et. al, “A Short Introduction Cloud Platforms – An Enterprise Oriented View”, Technical Report, August 2008.
[2] Khrushd Ahmed, Al-Nayyem Abdullah and Gupta Indranil, “Performance Evaluation of the Illinois Cloud Computing Testbed”, Technical Report, June 2009. http://hdl.handle.net/2142/12983.
[3] Rajkumar Buyya, Chee S. Yeo, Srikumar Venugopal, James Broberg, Isuna Brandic, “Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility”, Future Generation Computer Systems, Vol. 25, No. 6, pp. 599-616, June 2009.
[4] Judith M. Myerson, “Cloud computing versus grid computing”, White Paper IBM, March 2009.
[5] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica and Matei Zaharia, “Above the Clouds: A Berkeley View of Cloud Computing Export”, No. UCB/EECS-2009-28, White Paper, February 2009.
[6] Introduction to Cloud Computing Architecture, White Paper, Sun Microsystems, 1st Edition, June 2009.
[7] Architectural Strategies for Cloud Computing, Oracle White Paper, August 2009.
[8] Dhaavachelvan P and Uma, G.V, “Multi-agent Framework Construction for Intra Class Testing of Object-Oriented Software”, Lecture Notes in Computer Science (LNCS), Vol.2869, pp. 992-999, Springer Verlag, September-2003.
[9] Dhaavachelvan P and Uma, G.V, “Reliability Enhancement in Software Testing: An Agent-Based Approach for Complex Systems “, Lecture Notes in Computer Science (LNCS), Vol. 3356, pp. 282-291, Springer Verlag, December 2004.
[10] Bethea, W.; Cole, R.; Harshavardhana, P., “Automated discovery of information services in heterogeneous distributed networks”, Military Communications Conference, MILCOM 2008, Page(s): 1 – 6, 2008.
[11] Chou, Wu; Li, Li; Liu, Feng, “Web Services for Service-Oriented Communication”, International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2006, Page(s): 1 – 8, 2006.
[12] Feng Liu; Gesan Wang; Li Li; Wu Chou, “Web Service for Distributed Communication Systems”, IEEE International Conference on Service Operations and Logistics, and Informatics, SOLO ’06, Page(s): 1030 – 1035, 2006.