Abstract — Today’s businesses can acquire applications on demand using cloud computing. Multi-tenancy is an important feature of cloud computing in which a single application is shared among multiple tenants. Multi-tenancy offers variety of advantages including cost savings, version control and more. Currently more applications are moved from organization’s internal infrastructure, to data centers using shared infrastructure that provide Software as a Service (SaaS) to large number of organizations. However, lack of simulators that portrays the architectural concepts averts researchers and enterprises from using SaaS Multi Tenant Applications (MTA). In this paper we propose AppSIM an application simulator (AppSIM) that models multi-tenant SaaS application’s features. Our evaluation and discussion demonstrate the usefulness of the tool and shows AppSIM can be effectively used for evaluating MTA non-functional requirements.

Keywords — Cloud Computing, SaaS, Multi-Tenancy, Software Simulation.

I. INTRODUCTION

Today application development and deployment is possible from anywhere in the globe using Software-as-a-Service (SaaS). Traditionally applications were only developed in-house or purchased ready-made (Off-the shelf) and hosted in organization owned and maintained servers, later the concept of ASP (Application Service Provider) came into existence where organization developed applications in-house or purchased it ready-made but hosted the application using a ASP. Cloud computing makes software more attractive by offering it as a service [12].

In SaaS the software design, development, hosting, and maintenance is carried out by the SaaS service provider. SaaS brings various advantages including low cost, usage based pay as you go model, and provide features to build and deploy applications instantly anywhere globally, Fig 1 shows a typical multi-tenant SaaS application architecture. However, well defined simulation models are required for organization to evaluate the performance of the application before implementing it in SaaS. There is a lack of tools that facilitates developers to assess requirements of large scale cloud applications [4]. The tenant-specific data maintenance and diverse customization that required to be done at run-time make the simulation models more challenging in MTA [10].

SaaS simulation model needs to reproduce the actual characteristics of a typical datacenter and be customizable to fulfill the various functional and quality requirements of individual tenants. Moreover, provide organization an opportunity to analyze the outcome of the application before switching to the new software service.

In this paper we propose AppSIM an Application Simulator that models a customizable simulation framework for multitenant SaaS application. The contributions of this paper include a SaaS simulator that incorporates key elements of multi-tenant SaaS architecture; examine various run-time application features of MTA and calibrate the simulation continuously to represent an actual datacenter. Our evaluation and discussions demonstrate the usefulness of the simulator and shows the applications scaling under different static and dynamic number of users attached to the tenant.

The paper is structured as follows: Section II discusses related work, III. Motivation & Problem definition is presented in Section III; background information on Multitenant Architecture is presented in Section IV. Our proposed simulator APPSIM is described in Section V, the evaluation of our proposed simulator and discussions are described in Section VI and finally, the conclusion and future work are in Section VII.

M.D. Samrajesh is Research Scholar at Research and Development Centre, Bharathiar University, Coimbatore- 641046., Tamil Nadu, India. (e-mail: samrajesh.md@gmail.com).

N.P. Gopalan, is Professor at National Institute of Technology, Tiruchirappalli, Tamil Nadu, India (e-mail: npgopalan@nitt.edu).
II. RELATED WORK

A discrete event simulation based evaluation of cloud application’s response time, based on the number of VM’s and data centers is proposed in [4]. The data transfer cost policy is based on the Amazon EC2, experiment results is based on a social networking site and the response time for requests and the cost of infrastructure is also evaluated.

Emulation and simulation to model network behavior is presented in [6]. Communication of the real application is forwarded to the simulation engine. The main goal of simulation is use real-world distributed applications and network services to model network emulation, and to apply it simultaneously in a real-time simulator. However this concentrates more on the network rather than application.

A platform that supports testing of distributed applications is proposed in [7]. The model can be scaled with simulated nodes, links, and Internet traffic, allowing tests in larger scales than what is supported by the actual cluster nodes. The goal of simulation and emulation here is to increase scalability of the system. However it does not support modeling the application behavior.

A performance prediction setting for large scale machines is presented in [8], it supports simulation and emulation of parallel machines, emulation is used to execute actual applications, whereas simulation is used to model network latency and execute models of the application obtained during emulation. However, the simulation is not specific to SaaS applications.

An application service emulator for workflow applications is presented in [9]. It receives tasks from workflow engines and, instead of executing them in actual resources; it emulates execution of the application using information available about it. WORKEM’s goal is helping in designing, planning, and debugging of workflow applications, whereas the goal of emulation in EMUSIM is supplying an accurate representation of the application for a simulator. However, this does not consider multi-tenant nature of SaaS application.

Modeling and simulation of cloud computing environments is presented in [10]. It also simulated multiple data centers to enable study policies of cloud including migration of VMs for dependability and scaling. Our simulator considers the multi-tenancy of SaaS application in modeling and evaluation.

III. MOTIVATION & PROBLEM DEFINITION

Evaluating algorithms that have dynamic resource requirement on the data centre are of unique challenge to researchers. Even, if we assume that a huge infrastructure is available for testing, the implementation and evaluation of algorithm would be un-manageable and extremely time-consuming [11].

Testing for accuracy and reliability are the challenges in the development and deployment of cloud computing platforms due to large amount of hardware required [5]. There is lack of tools that allow developers to evaluate large scale cloud applications in terms of distribution of both computing and workloads [4].

Multi-tenant applications due to its native multi-tendency and tenant-specific data maintenance have made simulation comparatively complex [1]. Hence, there is a need for a well defined simulation models that can be used to evaluate large scale cloud applications for testing their accuracy and dependability.

IV. MULTITENANT ARCHITECTURE

A. Software as a Service (SaaS)

Software as a service (or SaaS) is a way of delivering software applications over the Internet as a service. Instead of installing and maintaining software, organization simply access the required software via the Internet; organization need not worry about the complex software and hardware management. In traditional packaged software customer managed the application source code its updates and individual fixes. In Software as a Service (SaaS) provider managed the source code, its updates, and a fix for one, fixes all. Cloud computing is emerging as a model in support of X-as-a-Service (XaaS) [19] where x-refers to everything.

NIST [4] defines three service models for cloud computing. The Infrastructure-as-a-Service (IaaS) model leverages virtualization to share hardware resources among customers. The Platform-as-a-Service (PaaS) model hosts applications of different customers within one middleware instance. Software-as-a-Service (SaaS) is a model which provides a ready to run, hosted application. Software as a Service (SaaS) is a delivery model where software applications are maintained and run by a SaaS provider instead of running it at the client location [13]. SaaS provides customers a centralized, access to data at reduced cost and overhead. Isolating cloud customers in terms of the performance they experience is an important concern in each of these scenarios. Software as a Service (SaaS) has emerged as a new application delivery model for software [14].

B. Multi-tenancy

The general definition of tenant refers one who has the occupation or temporary possession of lands or tenements of another; specifically: one who rents or leases (as a house) from a landlord. Multi-tenancy is a software architecture where multiple client organizations (tenants) are served by a single application running on a server as shown in Fig.1. A multi-tenant software application is highly scalable and hosted software, in which the application and its infrastructure are shared among multiple tenants to save development and
maintenance costs [15]. In multi-tenancy all tenants share the same source code, maintaining common code benefits all tenants as updates can be applied centrally.

A Multi-tenant application is designed stateless to ensure scalability. User-specific data is stored either on the client side or in the database. MTA allows I/O (Input/Output) operations to be carried out asynchronously. Moreover, unlike single-tenant system, tenant-specific implementations of the application logic are created and maintained in each application. The MTA queries the database for necessary tenant-specific business logic and creates a customized implementation at run-time. Multitenant applications are dynamic or polymorphic in nature to fulfill the individual customer requirements [16]. Tenants get a highly customized application that makes them feel as if the application is exclusively designed for them [17]. Fig.2 shows the various platforms for developing multi-tenant SaaS applications.

V. APPSIM: ARCHITECTURE AND IMPLEMENTATION

A. Architecture

The architecture of the simulator consists of two major components, namely the Component Load Generator (CLG) and Virtual Component Distributor (VCD) as shown in Fig 3. The Load generator (CLG) generates the load based on the Tenants Configuration (TC), the number of users attached to the Tenant (TU), and Server Configuration (SYS). The server configuration load generated represents a typical datacenter application load. VCD monitors components load and when the application component access load exceeds a predefined threshold logs are created.

The server load is generated using [20], Box-Muller methods takes two samples from a uniform distribution of interval (0, 1) and points them to normally distributed samples. The number of users attached to each tenant follows a Poisson distribution.

B. Assumptions & Initial Setup

The assumptions include, at any point of time tenants have one or more users attached to them. Each tenant has access to all core components of the application by default. Add-on components are randomly allotted using normal distribution to tenants. Tenant user’s component access can be either static or dynamic. In case of static access all users attached to a tenant have access only to predefined components assigned to them, whereas in the case of dynamic component access can be varying at run-time. The initial total number of Tenant is 5000. The number of users attached to each tenant follows a Poisson distribution. The add-on components accessed by tenant users are randomly distributed. The thresholds are set based on [18] to 85%.

C. Tenant Management

Tenants attached to the application can be customized each tenant has ‘n’ number of users attached. During run-time the number of users attached to the tenant may be either static or dynamic but within the maximum number of users predefined. The server load varies based on the number of users attached to the tenant and the application components accessed by the tenants.

VI. EVALUATION AND DISCUSSIONS

A. Application load

Two different scenarios of application load is evaluated, in scenario 1 the fixed number of users attached to the tenants and in scenario 2 dynamically varying number of users attached to the tenants using on MTA application is simulated. Fig. 4 shows the application load pattern of the simulator.

B. Application Scaling

Application performance is optimized by effective scaling of application’s component instances. Scaling can be done either time-based and/or based of application load metrics. In time based scaling application is scaled based on a know pattern of load during a time period such that the tenants have optimum performance. This is done using a trace driven approach that represent peak and off-peak duration. Next, in the case of load based scaling is dynamic and scales the application as the load spike occurs. However, load-based have certain overhead in detecting the increased load, and overhead related to the time taken to scale.
C. Discussions

A case study of a multi-tenant SaaS insurance Customer Relationship Management (CRM) application that have the core modules and add on components such as automobile, life, machinery, property, travel etc. Fig. 5 shows the application load under static and dynamic number of users attached to the tenant. The pattern of user access time and the number of users attached to the tenants have a significant impact on the application performance, hence the application scaling for this type of application can be time-based scaling.

VII. CONCLUSIONS AND FUTURE WORK

Today cloud computing is widely popular; however not all organization are ready to switch their application in SaaS. Hence, there is an increased demand for tools to study and examine the benefits application deployment in cloud. In this paper we presented a multi-tenant SaaS application simulator AppSIM. We have demonstrated how the simulator can be used effectively to simulate a multitenant SaaS environment with static and dynamic tenant users. Furthermore, we used a multi-tenant SaaS insurance Customer Relationship Management (CRM) application and illustrated how application scaling can be accomplished for optimum performance.

Our future research goals are targeted at developing more features for the simulator and consider the energy efficiency aspect of the applications in SaaS.

REFERENCES


