"Multi-tenant SaaS Cloud"

Gurudatt Kulkarni[1], Rupali Shelke[2], Rajnikant Palwe[3],
1, 2,3Lecturer in Marathwada Mitra Mandal's Polytechnic, Pune

Prasad Khatawkar[4]
4 Principal, Vidya Pratishtan's Polytechnic, Indapur

Sadanand Bhuse [5]
Hemant Bankar[6]
5,6 Lecturer in Vidya Pratishtan's Polytechnic, Indapur

Abstract-Multi-tenant applications are usually cloud based software services which can serve different users at the same time. This is done using single instance of applications by sharing hardware, infrastructure, data storage and virtualization. To achieve multi tenancy different approaches are there at every layer (Application, Data, hardware). Software-as-as-Service (SaaS) is a new approach for developing software, and it is characterized by its multi-tenancy architecture and its ability to provide flexible customization to individual tenant. SaaS (Software as a Service) is a modern approach to deliver large scalable enterprise software as a service on Internet. Cloud computing platform provides the scalability, availability and utility computing for services on Internet. There are many technical challenges involved in SaaS development. One of them is multi-tenancy, which allows single instance of software to serve multiple organizations by accommodating their unique requirements through configuration at the same time. The target SaaS platform is composed of the key components that supports SaaS Application execution environment that serves multiple tenant using a single service instance. An example scenario of SaaS software lifecycle is also described to explain how the target platform would be operated from development and deployment of SaaS application to configuration by tenants.

Keywords- configuration, Multi-tenant, SaaS, platform

I. INTRODUCTION

Pure SaaS is a complete business model; it's more than just a check box on an RFP. Realizing the true business benefits the Cloud offers means choosing a vendor who can do more than over promise and under deliver. After all, it's all about delivery. Multi-tenancy means that while the service shares a single solution behind the scenes, they've been designed and implemented with the frontlines in mind and are highly configurable to meet specific business needs. In the true multi-tenant SaaS model, no two implementations of a SaaS product look exactly alike even though they're sharing the same centrally hosted system. SaaS (also known as On-demand solutions or hosted software solution) is a software application delivery model where a vendor develops software and hosts/operates (independently or through a 3rd-party) over the internet. Software users pay the vendors on a "pay as you go" basis, usually a monthly or yearly subscription fee. In this model, the software users can access the software from anywhere over the Internet. This hosted software solution is adopted in line of business services like CRM, HR, Payroll processing, etc. Consumer related services like Google Docs, Web-based-mails, and online banking are also on-demand solutions. SaaS has gained acceptance not only among the non-critical business processes like above, but some of the software products that service critical business functions (e.g. Hospital management systems) are also considering SaaS more actively for some of their product lines.

II. MULTI TENANCY

Multi Tenancy is the ability to handle multiple organizational data and applications in a single instance. Four kinds of maturity models are available while building a SaaS based product. Out of the four models, two of them are single tenant or multi instance models. The rest of the two are Multi-tenant models. Moreover, there are variations within the single-tenant model as well as the multi-tenant models too. The idea of multi-tenancy, or many tenants sharing resources, is fundamental to cloud computing. Service providers are able to build network infrastructures and data architectures that are computationally very efficient, highly scalable, and easily incremented to serve the many customers that share them. Multi-tenancy spans the layers at which services are provided.

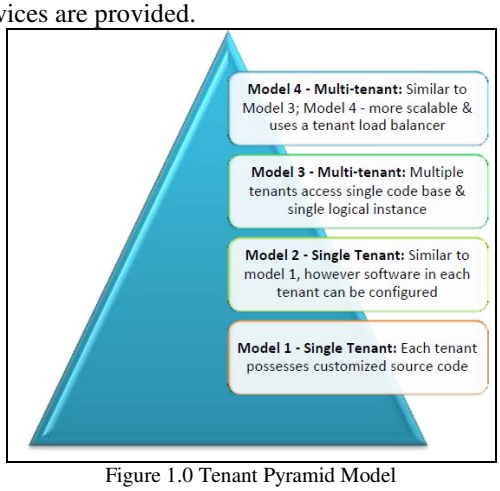

Figure 1.0 Tenant Pyramid Model

In IaaS, tenants share infrastructure resources like hardware, computer servers, and data storage devices. With SaaS, tenants are sourcing the same application (e.g., Salesforce.com), which means that data of multiple tenants is likely stored in the same database and may even share the same tables. When it comes to security, the risks with multi-tenancy must be

addressed at all layers. The next few sections examine how this can be accomplished for shared hardware and application infrastructure.

III. MULTI-TENANT SAAS

Multi-tenant SaaS architectures are becoming more and more prominently used among SaaS providers. In a multi-tenant environment, all clients and their users consume the service from the same technology platform, sharing all components in the technology stack including the data model, servers, and database layers. In our experience, the benefits of using a multi-tenant architecture translate to significant benefits for our customers, including:

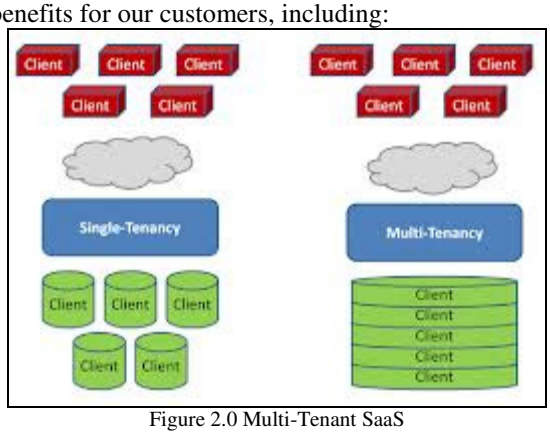

Figure 2.0 Multi-Tenant SaaS

A. *Hardware Resource Sharing*

In traditional single-tenant software development, tenants usually have their own (virtual) server. This set-up is similar to the traditional Application Service Provider (ASP) model. However, in the SME segment, server utilization in such a model is low. By placing several tenants on the same server, the server utilization can be improved. While this can also be achieved through virtualization, virtualization imposes a much lower limit on the number of tenants per server due to the high memory requirements for every virtual server [10]. Higher utilization of the existing servers will result in lower overall costs of the application, as the total amount of hardware required is lower. The concept of multi-tenancy comes in different flavours, and depending on which flavour is implemented, the utilization rate of the underlying hardware can be maximized.

B. *Data Protection*

A SaaS architect is responsible for building adequate data protection as well as multiple defense level that complement each other to counter both internal and external threats. Data protection can be implemented through filters or firewalls, access control lists and encryption.

i. Proxy Filters: The objective is to create a filter or cache between the tenants and users and the actual data source. One common approach to manage the access to data is to monitoring the TCP packet for tempering, impersonation, incorrect source IP. The filtering mechanism can be implemented in a device outside the databases or within a view. A proxy can be setup to convert authentication information such as user name, password, database and roles to nondescriptive credential prior a connection is made to the actual database. A proxy has also the advantage to hide the actual IP/hostname and port the database engine is listening to.

ii. Access Control Lists: This commonly used technique relies on a trusted 3rd party subsystem which is already authenticated to access a database. Systems such as Kerberos and Active Directory monitor the actual credential of the principle or trusted client to the database. The combination of the user, tenant credentials as well as the trusted principal constitutes the security context.

iii. Data Encryption: Cryptology can be applied to both the tenant's credential with the protocol used to access the database and the stored data. Symmetric encryption/decryption which relies on a single key (i.e. common licensing mechanism) is not as safe as Asymmetric algorithms but requires a lot less computing power. Those asymmetric algorithms rely on a public/private keys paring such as SSL.

C. *Scalability*

A multi-tenant infrastructure makes it easy to increase capacity when more horsepower is required. When adding new hardware to the platform, the total capacity of the entire environment increases, becoming more scalable for not just a single customer, but for our entire client base. Scaling a well-architecture multi-tenant SaaS platform is, in many cases, a matter of simply plugging more hardware into the different elements contained within the technology stack. Because all tenants share the same application and data store, scalability is more of an issue than in single-tenant applications. We assume a tenant does not require more than one application and database server, which is usually the case in the SME segment. In the multi-tenant situation this assumption cannot help us, as such a limitation does not exist when placing multiple tenants on one server. In addition, tenants from a wide variety of countries may use an application, which can have impact on scalability requirements.

i. Dynamic Provisioning relies on a single master replication , in which only the original can be written to, is much easier to manage than multi-master replication, in

which some or all of the copies can be written to and some kind of synchronization mechanism is used to reconcile changes between different copies of the data.

ii. Partitioning involves pruning subsets of the data from a database and moving the pruned data to other databases or other tables in the same database. It means that the database is divided into several smaller databases using the same schema and structure, but with few rows in each table. An alternative is to divide the table into smaller tables with the same number of rows, but with each table containing a subset of the columns from the original. Not all the data should be partitioned. For instance, identity data or index table should be preserved as single database object.

iii. A mix model is usually implemented when the condition that trigger the provisioning of the database cluster are dynamic or the traffic patterns between the different tenants is not consistent.

D. Shared Infrastructure:

Multi-tenant allows multiple customers to run on the same infrastructure. Most people think of servers as the main infrastructure cost, but even more costly are the cost of database licenses and other infrastructure software, the 20%-per-year maintenance costs for all infrastructure components, and (most importantly) the operations personnel costs: DBAs, system administrators, network administrators, etc. Fifty customers, each buying and building their own infrastructure, would together spend an order or magnitude more cost than a shared-infrastructure SaaS solution. The customer gains from the SaaS vendor passing along a dramatically lower cost of ownership, and the SaaS provider can also provide capabilities that a single customer couldn't ever justify, such a hot-backup mirror sites, world-class scalability and performance, etc.

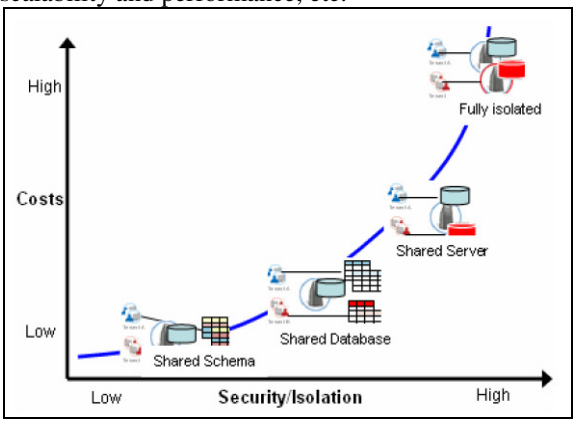

Figure 3 Comparing Multi-tenant Deployment Modes using Costs and Security

E. High Availability

One last critical element of database deployment in a SaaS multi-tenant environment is the guarantee of availability of the data. Each database object should be replicated in timely fashion and primary servers and database should have fail-over mechanism. Replication can be event-driven or scheduled. A significant drawback of the separate-schema approach is that tenant data is harder to restore in case of an outage as restoring the entire database would mean overwriting the data of every tenant on the same database with backup data for all the tenants.

F. Functionality

Functionality of software is significantly greater for single-tenant systems than for multi-tenant systems. In the multi-tenant option, modifications to the software are limited because multiple customers are running the same instance of the software and because their data is being housed in a pre-configured database format. This is not to say that the functionality isn't good. In fact, the opposite is generally true. For many companies with more basic database and information needs, configurability of software isn't a necessity. Multi-tenant SaaS providers generally do a very good job of anticipating the needs of current and prospective customers and the standardized functionality is often all that is needed by a company. For that reason, some vendors actually provide customers with a choice between single and multi-tenant options. By comparison, single-tenant systems use the provider's software as the base of the final application, and open the doors to configuration of the software to meet the specific needs of the user. Changes in the appearance of screens, additions to individual modules, conduits to different internal databases or external partner databases, etc. can all be accommodated in the single-tenant environment. For many companies, these configurations make all the difference between a system that works synergistically and that creates a seamless flow of data to authorized users, and one that does not.

IV. CONCLUSION

In cloud-based architectures, multi-tenancy means that customers, organizations, and consumers are sharing infrastructure and databases in order to take advantage of price and performance advantages that come with economies of scale. Tenants may share hardware on which their virtual machines or servers run, or they may share database tables where the data

of customer A is on one row and that of customer B is on another. Many cloud service customers are comprised of both types of tenants. It is very important to implement SAAS multitenant architecture for cloud either public or private by many of different ways. So here, SaaS Multitenancy for private cloud is implemented and event log, metering and service scheduling features are implemented. So that customer can use the best software as service at single instance to save time and money with quality features by getting license. In this way, conclusion of this paper is to implement SaaS multi tenancy for cloud at single instance and to provide different services quickly and then give license of that software to that customer. The identification of a new architectural style helps developer in creating future multitenant software applications. While the emerging PaaS environment are well-suited for implementing such applications, there are still many design decisions at the application tier and the database tier that have to be made for each application.

V. REFERENCES

1. Zhi Hu Wang, Chang Jie Guo, Bo Gao, Wei Sun, Zhen Zhang, and Wen Hao An. A study and performance evaluation of the multi-tenant data tier design patterns for service oriented computing. In *Proc. Of the Int. Conf. on e-Business Engineering (ICEBE)*, pages 94–101. IEEE, 2008.
2. Ralph Mietzner, Andreas Metzger, Frank Leymann, and Klaus Pohl. Variability modeling to support customization and deployment of multi-tenant-aware software as a service applications. In *Proc. of the ICSE Workshop on Principles of Eng. Service Oriented Systems (PESOS)*, pages 18–25. IEEE, 2009.
3. Bob Warfield. Multitenancy can have a 16:1 cost advantage over single-tenant. http://smoothspan.wordpress.com/2007/10/28/ multitenancy-can-have-a-161-cost -advantage-over-single-tenant/ (last visited on June 2nd, 2010), October 2007.
4. Ralph Mietzner, Tobias Unger, Robert Titze,Frank Leymann, "Combining Different Multi-tenancy Patternsin Service- Oriented Applications," edoc,pp.131-140, 2009 IEEE International Enterprise Distributed Object Computing Conference(edoc2009)
5. 2013 IEEE 7th International Symposium on Service Oriented System Engineering (SOSE), “SaaS Multi-tenant Application Customization”, Tsai, Wei-Tek ; Sun, Xin.
6. Annual SRII Global Conference (SRII), 2012 , “Model Driven Provisioning in Multi-tenant Clouds”, Gohad, A. ; Ponnalagu, K. ; Narendra, N.C.
7. IEEE 5th International Conference on Cloud Computing (CLOUD) 2012, , “A Multi-tenant Web Application Framework for SaaS”. Wonjae Lee ; Min Choi.