INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 13 (15): 2489-2495, 2014
ISSN 1812-5638 / DOI: 10.3923/itj.2014.2489.2495

Construction of SaaS-based Restaurant Management System

Yangpeng Zhu
Institute of Economics and Management, Xi'an Shiyou University, Xi'an, 710065, China

Abstract: Software as a service is becoming a popular research field in software development for its feature of low costing entry, easy implementation and zero infrastructures. SaaS is a multi-tenant model which is different from traditional software in user data security, software development and deployment. In this study, we analyze the SaaS application architecture and developed a SaaS-based restaurant management system. The result shows that the SaaS-based restaurant management can meet the requirement of tenant isolation and reduce the costs for users and services providers. The SaaS-based restaurant management system provides references to other software provider to develop SaaS-based application.

Key words: Software as a service, multi-tenant, restaurant management system

INTRODUCTION

As a new software service model, SaaS provider provides all the network infrastructure and software updates. In this model users need not to buy software and hardware and hire IT professional workers. They can use the SaaS system by internet and pay for selected services (Kwok *et al*., 2008). The growing number of tenants brings down the total costs of software provider and help the user put more energy to develop high value business. For this reason, it is popular for medium and small enterprise to use SaaS system (Xu *et al*., 2013). Having multi-tenant as main features, SaaS architecture is quite different from traditional software at technology lever and has many challenges (Saaksjarvi *et al*., 2005) these include:

- Multi-tenant architecture, lots of tenants share a single instance is different from traditional model of one tenant one application
- User data security, in multi-tenant architecture, every tenant shares one software instance and each tenant data store together. How to protect each tenant data in a shared storage is a challenge problem (Mietzner and Leymann, 2008)
- With the growing number of users, how to improve system performance is an important problem

Therefore SaaS system has many key issues to be addressed and a wide range of research space (Li *et al*., 2009).

DIFFERENCE TO TRADITIONAL DEVELOPMENT TECHNOLOGY

SaaS means software as a service which is a new software model along with the development of internet and web applications in 21st century. The SaaS model provides software through internet and it begins a new business model in which software providers deploy application on their own server while users can buy software service which they need through internet and pay to the service according to service content and service time (Ferraiolo *et al*., 1999). SaaS model has some advantages and difference to traditional software.

Support multi-tenant: The main difference between SaaS model and traditional software is whether it supports multi-tenant. Traditional custom-built software deploys application instance for each user according to his requirement and meets different customer's individual requirements by providing different configurations. SaaS model is a multi-tenant one instance application in which all tenants share one instance and reduces cost by more users.

Software license: In traditional software, user buys software license which means the software belongs to the user. But in SaaS model, users rent for selected service instead of buying it and can abandon on it easily by not to rent at any time if he does not want to use it any more. This reduces the risk of user investment.

Reduce the user's investment: In traditional model, user needs lots of money to buy hardware such as Router, Hub and Servers. Even more, development custom-built software is either a hight risk investment. If the software can't meet the user's needs, developer needs to change the software or even to redesign it which increases the user and developer's investment risk. But in SaaS model, user need not to buy any hardware and can try to use the software first, if it can meet the user's demand, user pays just for rent.

Easy to implement: In traditional model, provider needs to deploy applications for user for a long time, but in SaaS model, user needs not to install any software in his own computer, he can easily access the application through internet.

Convenient to update: In traditional model, user needs frequently update application for more function or software bug. In SaaS model, software provider updates the application and users need not do any more.

Zero maintenance: In traditional model, user needs hire professional engineer to maintain server, computer system and to backup the user data. In SaaS model, user needn't afford for the maintenance fee (Gandhimathi and Jayakumar, 2014).

SAAS ARCHITECTURE

In SaaS model, lots of tenants access the system together and each tenant has no relation to others. Tenant can create needed users for himself and assign right to his users. SaaS is much similar to other application in architecture, as show in Fig. 1.

UI layer is responsible for displaying information to user and process layer is the main part of the model which processes the enterprise business and users can custom his business process or arrange the service at this layer. Service layer focuses on providing services for an atom function for user and database layer is for storing user data (Zhang *et al*., 2011).

The maturity model is to judge whether a SaaS system is expansible, configurable and stable. SaaS application is classed for four grade maturity (Chong *et al*., 2006), as shows in Fig. 2. The model in Fig. 2a is similar to traditional web system in which each tenant has a single application instance. Model 2 is similar to model 1 and difference of them is user can configure application through the management console in model 2

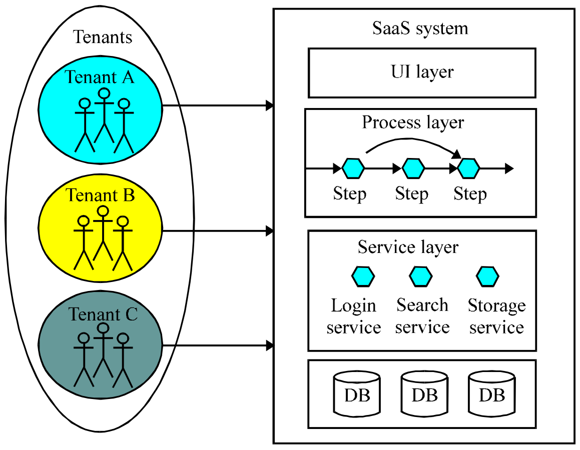

Fig. 1: SaaS architecture

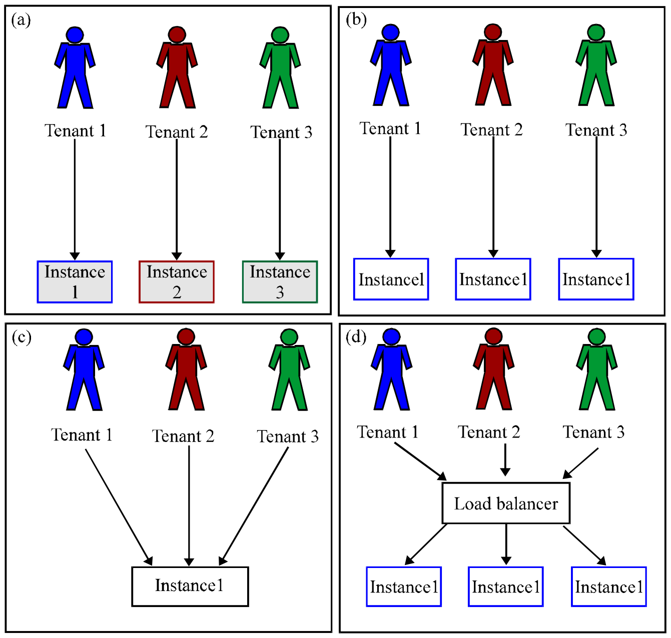

Fig. 2(a-d): Multi-tenant 4 mature architecture models, Model (a) 1, (b) 2, (c) 3 and (d) 4

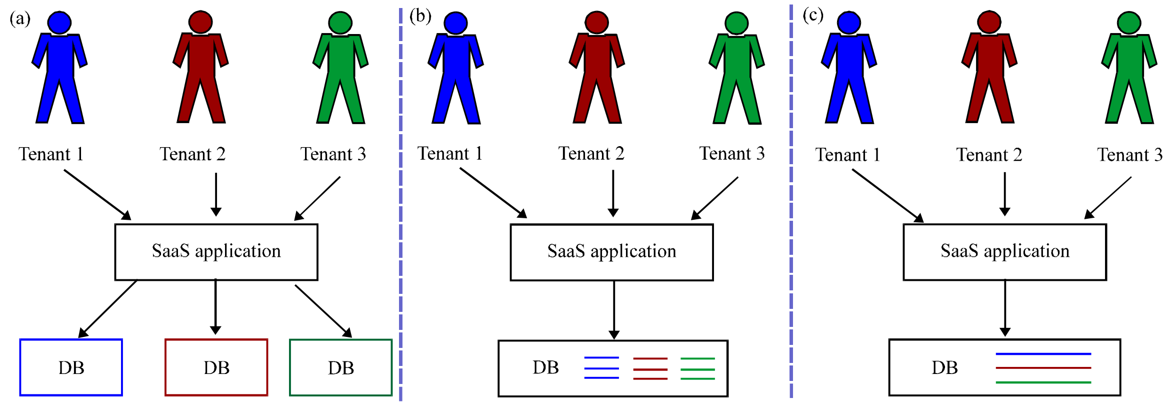

Fig. 3(a-c): Multi-tenant application database design for model (a) 1, (b) 2 and (c) 3

as shown in Fig. 2b. Model 3 is a real multi-tenant SaaS application in which all tenants share a single application instance which can save resources for services providers as shown in Fig. 2c. Model 4 is a multi-tenant multi instance model which has a high maturity. In this model, with the increase tenants number, tenant load instance assign tenants to different instance according to the load of each instance. Through multi instance, system can extend to unlimited number of hardware servers as shown in Fig. 2d.

In multi-tenant SaaS system development, database is commonly designed for three models (Aulbach *et al*., 2008; Chong and Carraro, 2006) as shown in Fig. 3.

Independent database: Each tenant's data stores in a dependent database and each tenant shares the public application. It is a simplest storage way and is convenient for user to backup and restore their own data. This model adapt to users who need high security but not focus on cost, such as bank, hospital etc as shown in Fig. 3a.

Shared database and independent schema: All tenants store their data in one database, but each tenant has dependent schema, that means every tenant have a dependent set of table as shown in Fig. 3b.

Shared database and shared schema: All tenants store data in a database and schema. We use a column "tenantID" to distinguish tenant to others. This model has the highest data sharing and lowest data isolation and has lowest hardware and maintenance cost. It has a high number of tenants per server. But all tenants use one common database. This model has low isolation and security which adapts to those systems has few tenants as shown in Fig. 3c.

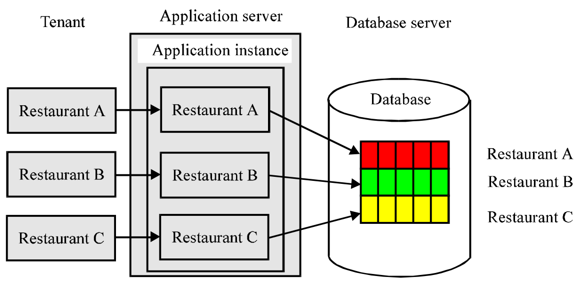

Fig. 4: SaaS-based restaurant management system architecture

CONSTRUCTION SAAS-BASED RESTAURANT MANAGEMENT SYSTEM

In this study, restaurant management system was developed which can provide some services for restaurant management. These services include user login, vegetables purchasing, order management, user cashier and a series of consumer information management. The restaurant manager can view the company sales, cost accouning, inventory manage and online payment at anytime, anywhere via the Internet. Different resraurant can choose different functional modules according to their needs, thereby improving their restaurant management level.

SaaS-based restaurant management system architecture: In this study, we developed the SaaS-based restaurant manegement system with the third mature model and shared database shared schema database model. The system architecture is show in Fig. 4.

In this system model, all tenants share one application and store their data in a share database which can save resources for Application server and Database server. In this database model, all the tenants' data store

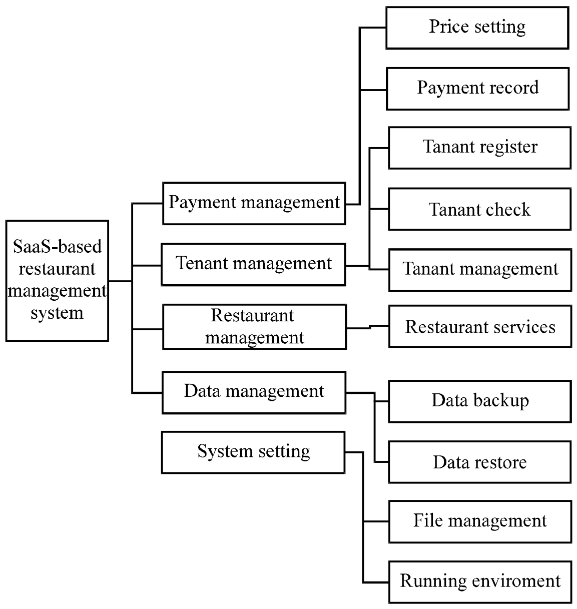

Fig. 5: Restaurant management system functions

in one database and each table has a column "tenant-id", application uses tenant-ID to distinguish tenants' data according tenants' login information. So, every tenant can only access his own data.

System function diagram: In this study, we developed a restaurant management system which includes functions like user login, vegetables purchase, order management, cashier and a series of consumer information management. The restaurant manager can view the company sales, cost accouning, inventory manage and online payment at anytime, anywhere via the Internet. Different resraurant can choose different functional modules according to their needs, thereby improving their restaurant management level. The resraurant management system has two layer, the top layer is mainly used by system administrator includes tenants management, payments management, data process, system enviroment management, basic setting and restaurant management, as shown in Fig. 5.

System business process diagram: SaaS-based applications are different from traditional applications in registering and user login. The user register process is as shown in Fig. 6.

When restaurants want to use a SaaS-based restaurant management system, they can register their enterprise information on the system website.

If their register information pass the administrator' check, the SaaS-based restaurant management system can

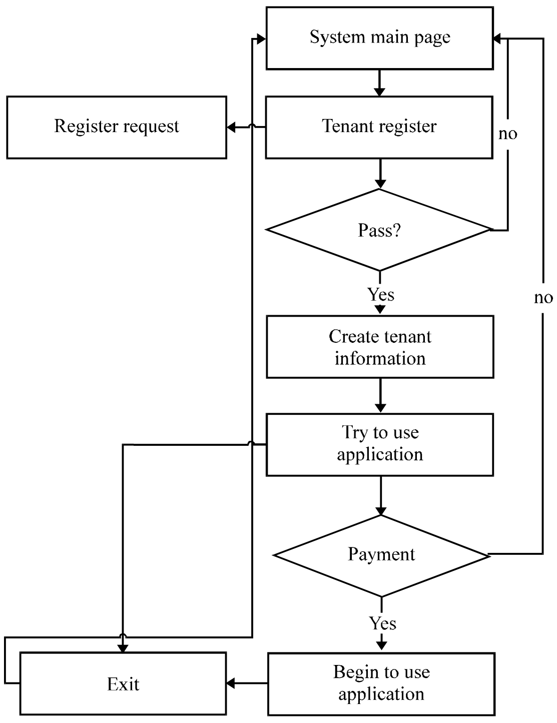

Fig. 6: Tenant register process

create Tenant register information for tenant users. Users can try to use the restaurant management system to judge whether the system can meet their enterprise' business management request for one month. If the restaurant

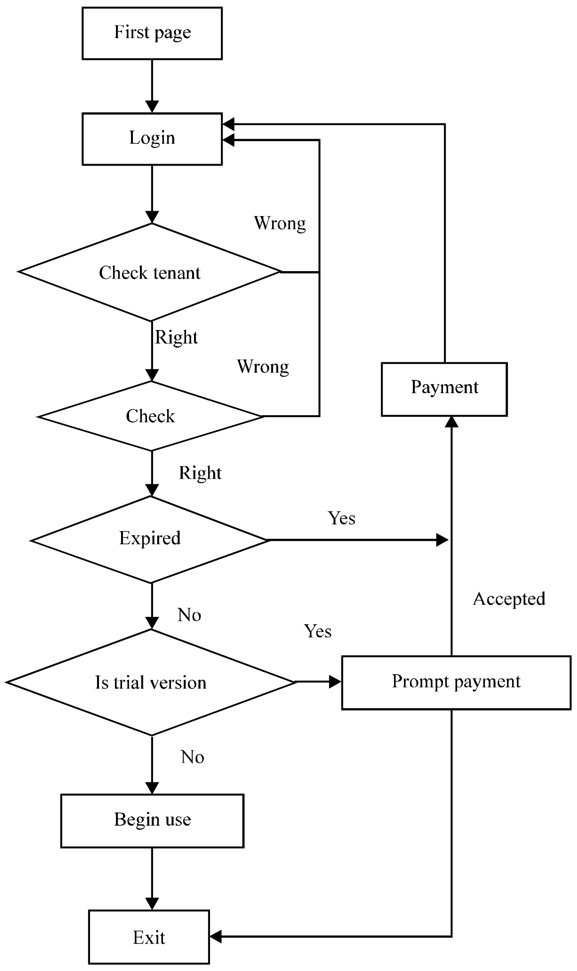

Fig. 7: Tenant login process

management system can meet their request, the tenant can pay system rent for use the restaurant management system service for about half a year.

In SaaS-based restaurant management system, the tenant login process is as showing in Fig. 7. When a user login the system, the system checks the user login information and looks for which tenant the user belong to according to the user login information. If the tenant account is expired or is a trial version, the SaaS-based restaurant management system will prompt user for payment. If the user succeeds in login process, the system will direct the user to the right tenant data and operate the system services.

SYSTEM PERFORMANCE TESTING

In order to compare the performance of single-tenant restaurant management system and multi-tenant SaaS-based restaurant management system, we need two

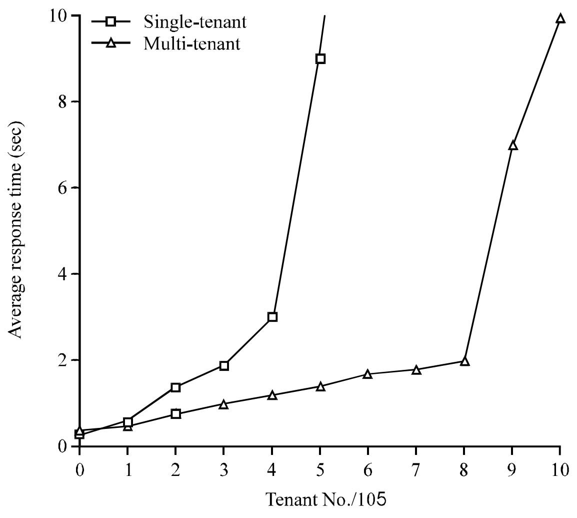

Fig. 8: Relation between tenant/instance number and response time

same configuration servers whose configuration is as follows:

- **Hardware:** Intel Core-i3-2100T 2.5GHZ 3M L3 cache, dual four thread CPU, 4G memory, hard disk of 1 T
- **Software:** Windows Server 2008 operation system, the web server using the Apache Tomcat 7.0.22, MySQL database 5.5 and JDK6

Test tools: Load Runner 8.0, HP Load Runner is an automated performance and testing product from Hewlett-Packard for examining system behaviour and performance while generating actual load. LoadRunner can simulate hundreds or thousands of concurrent users to put the application through the rigors of real-life user loads while collecting information from key infrastructure components (Web servers, database servers etc.). The results can then be analyzed in detail, to find out the reasons for particular behavior.

Testing environment: The multi-tenant SaaS-based restaurant management system was deployed at server 1 and the single-tenant restaurnant management system was deployed at server 2. We analyze the relation between the tenants number and server response time at the same user number. The testing process is shown as follow:

- Multiple tenants were created to simulate multiple restaurant opearation process. Each tenant has 100 users to simulate multiple users and multiple tenants simutaneously access the application. The relation between tenants number and server response time (unit: seconds) was shown in Fig. 8.

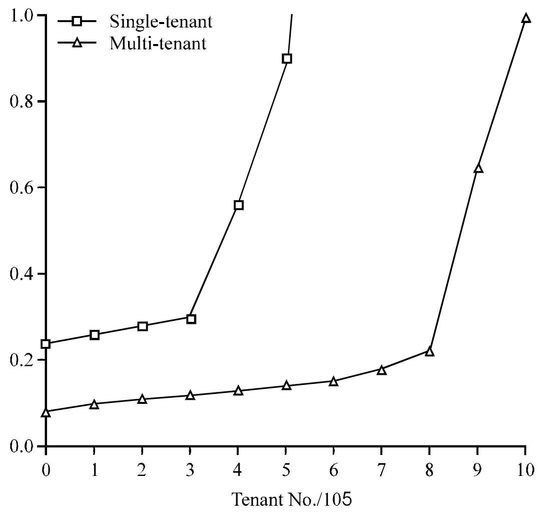

Fig. 9: Relation between tenant/instance number and resource utilization

- Multiple instances of single-tenant restaurant applications were deployed at server 2 which included multiple application instance and database instance. Each restaurant enterprise can use one application instance to manage its business. Each instance can have 100 users to simulate multiple users access the application instance. The relation between application instances number and server response time was shown in Fig. 9

In Fig. 8 when serverl only has one tenant and server 2 has one instance, the server 2 response time is smaller than the server 1. This is because at this time the server 2 only has one instance, it consumes few resources than serverl which hosts the multi-tenant application. When 2-40 enterprises access the application, multi-tenant application response time on serverl is significantly less than the single-tenant restaurant application on server 2. This is because when more enterprises need access the single-tenant application, more application and database instances need to be deployed on server 2 which will increase the consumption of server resources. For multi-tenant application, more enterprises only need to add some additional data in multi-database and consume less server resources on serverl. In single-tenant application, when the application instance numer is more than 40, the server 2 response time is rapid growth. This is because when more than 40 application instances are deployed in server 2, the server 2 resource has reached a bottleneck. When the tenant number reaches more than 80, the server 1 response time and resource consumption are also a substantial increase and need more servers. In this experiment, the multi-tenant application improves the utilization of server resources and have a shorter response time.

CONCLUSIONS

In this study, the multi-tenant SaaS system mature architecture and database model and talk about the differences between multi-tenant SaaS system and traditional systems were studied. We develop a third mature SaaS-based restaurant management system which is in shared database and shared schema model. In this database model, all tenants share one same application instance and store their data in one database. Users can only access the data belongs to his tenant according to the column “tenantID” each table. With the incensement of tenant number, the multi-tenant SaaS-base restaurant management system can reduce operational costs for users and service providers.

ACKNOWLEDGMENT

This study is supported by 2013 Scientific Research Plan Projects of Shaanxi Education Department grant number 13JK0178. We thank to reviewers for their valuable comments.

REFERENCES

Aulbach, S., T. Grust, D. Jacobs, A. Kemper and J. Rittinger, 2008. Multi-tenant databases for software as a service: Schema-mapping techniques. Proceedings of the ACM SIGMOD International Conference on Management of Data, June 9-12, 2008, Vancouver, BC., Canada, pp: 1195-1201.

Chong, F. and G. Carraro, 2006. Architecture strategies for catching the long tail. Microsoft Corporation, April 2006. http://msdn.microsoft.com/en-us/library/aa479069.aspx.

Chong, F., G. Carraro and R. Wolter, 2006. Multi-tenant data architecture. Microsoft Corporation, June 2006. http://msdn.microsoft.com/en-us/ library / aa479086.aspx.

Ferraiolo, D.F., J.F. Barkley and D.R. Kuhn, 1999. A role-based access control model and reference implementation within a corporate intranet. ACM Trans. Inform. Syst. Secur., 2: 34-64.

Gandhimathi, G. and S. Jayakumar, 2014. Speech enhancement using an artificial bandwidth extension algorithm in multicast conferencing through cloud services. Inform. Technol. J., 13: 1953-1960.

Kwok, T., T. Nguyen and L. Lam, 2008. A software as a service with multi-tenancy support for an electronic contract management application. Proceedings of the IEEE International Conference on Services Computing, Volume 2, July 7-11, 2008, Honolulu, HI., USA., pp: 179-186.

Li, H., Y. Shi and Q. Li, 2009. A multi-granularity customization relationship model for SaaS. Proceedings of the International Conference on Web Information Systems and Mining, November 7-8, 2009, Shanghai, China, pp: 611-615.

Mietzner, R. and F. Leymann, 2008. Generation of BPEL customization processes for SaaS applications from variability descriptors. Proceedings of the IEEE International Conference on Services Computing, Volume 2, July 7-11, 2008, Honolulu, HI., USA., pp: 359-366.

Saaksjarvi, M., A. Lassila and H. Nordstrom, 2005. Evaluating the software as a service business model: From CPU time-sharing to online innovation sharing. Proceedings of the IADIS International Conference E-Society, June 27-30, 2005, Qawra, Malta, pp: 177-186.

Xu, J., Z. Ping, L. Shu-Qin and H. Yu, 2013. Development for rural E-government SaaS application. Inform. Technol. J., 12: 3566-3570.

Zhang, S., Z. Li and X. Chen, 2011. Research of configuration on multi-tenant. Inform. Technol. J., 10: 2154-2160.