International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.8 (2014), pp.407-418
http://dx.doi.org/10.14257/ijmue.2014.9.8.38

A Platform for Changing Legacy Application to Multi-tenant Model

Yangpeng Zhu

School of Economic and Management, Xi' an Shiyou University, Xi'an 710065, China
zyp_hello@126.com

Abstract

In order to easily convert existing application to multi-tenant Software as a Service model, a Java migration platform is proposed. Firstly, the existing application is embed into the conversion platform and the single-tenant database was transformed to multi-tenant database by database transformation function. Secondly, each tenant's operation and data access was isolated in business and database layer by tenant filter function. Thirdly, combined with the certification and configuration functions in the SaaS conversion platform, the original system was converted to support multi-tenant SaaS system based on cloud computing with fer resource code updates. At last, a restaurant management system was migrated and functions and performances tests were taken to the migrated SaaS system. The results showed that the transformation had a lower manual workload, a shorter transformation lifecycle and a higher utilization of server resources.

Keywords: *cloud computing, software as a service, multi-tenant, migration*

1. Introduction

As a new software service model, SaaS provider provides all the network infrastructure and software updates. In this model users need not buy software and hardware and hire IT professional workers. They can use the SaaS system by internet and pay for selected services [1]. Multi-tenant is the key characters of SaaS in which multiple enterprises or departments share one application instance. The growing numbers of tenants bring down the total costs of software provider and help them put more energy to develop high value business. For this reason, it is popular for medium and small enterprise to use SaaS system.

Software providers have two ways to enter SaaS market, one is to create new SaaS software productions and the other is to transform the existing single-tenant software to multi-tenant SaaS model. For creating a new SaaS application, SaaS application provider not only need be family with business process, but also need address the key technologies about SaaS model such as multi-tenant architecture, tenant isolation, customization configuration and application scalability. Migration an existing single-tenant application to a multi-tenant model makes less market risks than development a new one. Therefore, transformation an existing application to a multi-tenant SaaS model is a useful attempt for SaaS application provider.

Now, researchers pay more attention to the problems about creation a new SaaS application such as multi-tenant architecture [2-6], tenant data isolation and security[7,8,9, 11], user customization requirement [12-14], application scalability and SaaS application performance management [3,15]. Some authors propose multi-tenant reengineer suggestions. In [16] the authors propose an experience with reengineering an existing industrial, single-tenant software system into a multi-tenant one using a lightweight reengineering approach. The reengineering approach is designed for .net environment and lack performances test data. In [17] the authors present a method to

ISSN: 1975-0080 IJMUE

build a multi-tenant platform which can run unmodified LAMP applications in a flexible, secure and scalable manner, but it needs virtualization machine to isolate tenants which increase the consumption of server resources. In this paper, we proposed a migration platform which can transform existing single-tenant application to multi-tenant SaaS application with small amounts of source codes modifications.

The paper is organized as follows. Section 2 describes the difference between existing single-tenant ASP application and multi-tenant SaaS application and their deployment methods. Section 3 shows the architecture and working principle of migration platform. Section 4 describes the detail process to transform the restaurant application to SaaS application. Section 5 introduces the conversion result and the performance of the utilization of server resources. Finally, Section 6 provides concluding remarks and outlines directions for future work.

2. Compare Existing single-tenant to multi-tenant SaaS application

Multi-tenant is meet lots of enterprises to share one application instance in which each user can only access data belonging to his tenant. The main difference between SaaS model and traditional software is whether it supports multi tenant. As showed in Figure 1.(a), traditional Application Service Provider(ASP) deploys application instance for each user according to users' requirement, and meets different customer's individual requirements by providing different configurations. This model has a lower application development effort and a high operational cost.

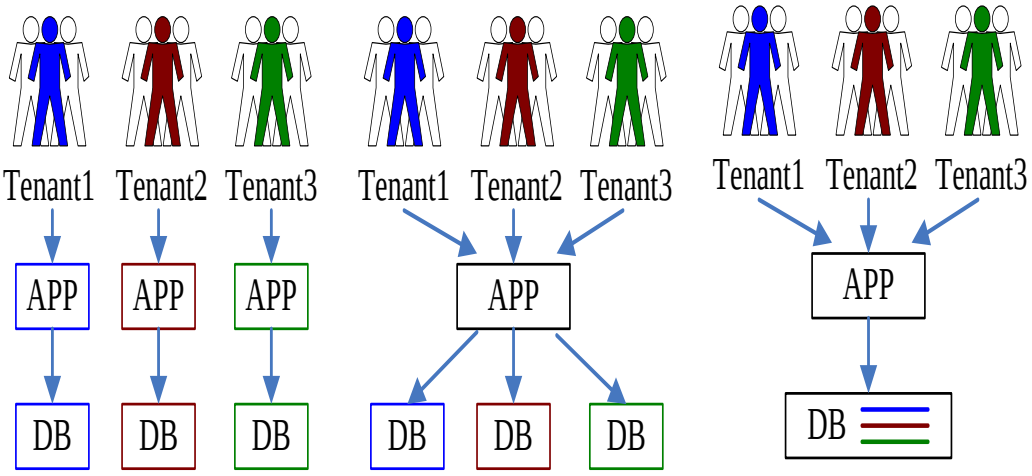

Figure 1.(a) ASP Model **Figure 1.(b)** Separation Database **Figure 1.(c)** Share Database

Multi-tenant application shares one application instance for all tenants and reduces cost by scale. In this context, every tenant feels he is using a customization system for himself as showed in Figure 1(b) and 1(c). In Figure 1(b), multiple tenants share one application instance and each of them has a separation database to isolate their business data. 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. In Figure 1(c), all tenants store data in a shared database schema. TenantID is used to distinguish tenants' data to others. This model has the highest data sharing and lowest data isolation and has lowest hardware and maintenance cost. The third model is a truly multi-tenant model that shares the resources, and keeps the tenants' access to data separate by application logic rather than by physically separating them. This leads to significant efficiencies of scale as more tenants are added.

In this paper, we transform the first high operational cost single-tenant ASP model application to the third lower operational cost multi-tenant model application.

3. Transformation Framework Architecture

Figure 2 provides an overview of the migration framework which includes database transformation module, authentication module, original web system, administration module and Infrastructure /Cloud. The authentication module is responsible for the registration of tenants and login authentication. Database conversion module is responsible for converting the original database to multi-tenant database and filtering user's operation to isolate tenant data. Platform management module is in charge of tenant management, rental management, resource monitoring, resource scheduling, *etc*. The conversion platform works as follow:

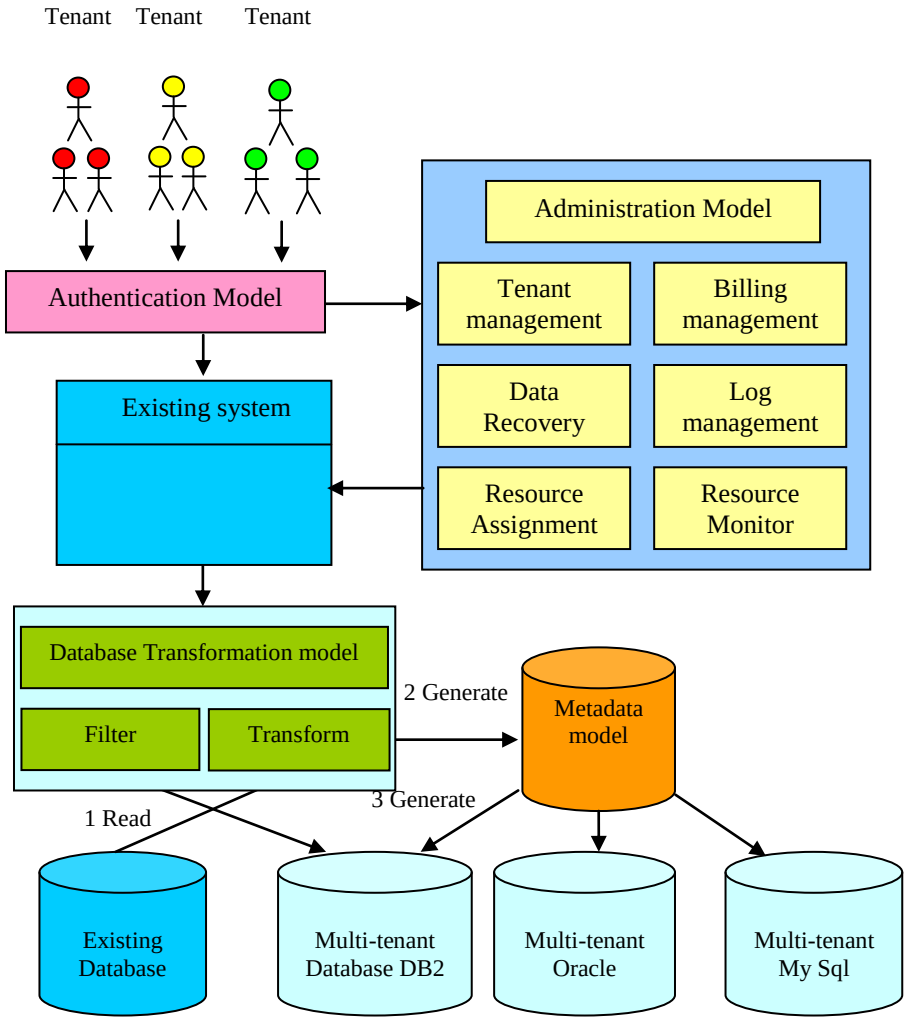

Figure 2. Migration Platform Architecture

(1) Embed the existing web application into the migration platform.

(2) Transform the single tenant database to multi-tenant database through the database conversion module.

(3) Update the database configuration file and make the new application connect to new multi-tenant database.

(4) Register tenant information through the authentication module.

(5) SaaS administrator manages the tenant application through the administration module.

(6) Tenants begin to use the multi-tenant application via the authentication module.

3.1 Database Transformation Module

Database transformation module (DTM) includes database transformation function and tenant filter function, the database transformation function is charged for transforming original system database into multi-tenant model and the tenant filter function is to filter user operation and data access.

In multi tenant SaaS system development, database is commonly designed for three models: separate database, shared database independent schema and shared database shared schema which have different methods to meet the SaaS application requirement [2, 4].

(1) Each tenant's data stores in an independent database and each tenant shares the same application. In this model, It is a simplest storage way and is convenient for user to backup and restore their own data and has a good data isolation, but the shortcoming is occupying too many recourses. This model adapt to users who need high security but not focus on cost, such as bank, hospital etc.

(2) All tenants store their data in a shared database, but each tenant has dependent schema, that means every tenant have a dependent set of table. Despite the advances in virtualization and cloud provisioning, there are inherent problems in the ability to scale up this model as more customers come on board.

(3) In this database model, all tenants store data in a same database schema. A column tenantID was added to each relation table to distinguish tenant to others. This model has the highest data sharing and lowest data isolation and has lowest hardware and maintenance cost.

In our transformation platform, we use the third database model. The DTM is inserted between presentation layer and database layer. Firstly, let us look up the Database transformation function.

(1) Extraction and Generation Meta database schema. DTM reads the database schema from original system and creates Meta database schema.

(2) Generation multi-tenant database. According to different user requirements, DTM can generate multi-tenant database with Meta database schema by adding corresponding management tables and inserting "TenantID" column to each table to distinguish various tenants' data. Because of only adding "TenantID" column to every table in multi-tenant database, the old web system can work correctly and needn't any changes in business process. Owing to the DTM using Meta database schema, user can select to generate various kinds of multi-tenant database such as Oracle, DB2, My Sql and SQL Server database.

(3) Modification database connection configuration. In order to make the original system connecting to new database correctly, we must modify the database connection configuration file.

(4) Operation filters. In SaaS software model, users can only access data belonging to their own tenant. In order to achieve this goal, "TenantID" column is used to distinguish each tenant's data. DTM can shield other tenant' data by data filter function when users access the application by operation such as inserting, updating, deleting and browsing. DTM obtains the SQL sentence submitting to the new multi-tenant database from the

application. It parses the SQL sentence and adds filter condition such as "TenantID=LoginTenantID" which will guarantee the data security in shared multi-tenant database and ensure the user can only access his own tenant's information.

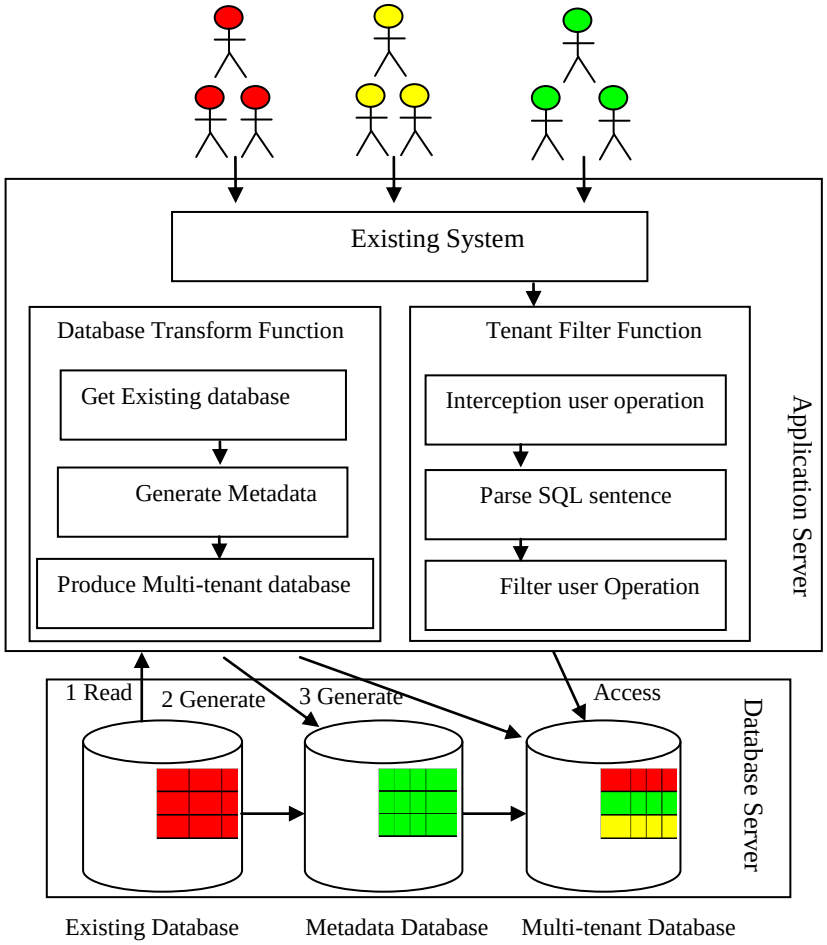

Figure 3. Database Transformation and Tenant Filter Process

3.2. Authentication module

The authentication module has two parts which includes tenant register process and user login process. The tenant register process is as flow:

(1) Tenant registers in SaaS application;

(2) SaaS administrator checks the tenant information. If the tenant information is passed, the saas application will generate an account for the tenant;

(3) The tenant tries to use SaaS application with the generated account and he can select the application functions according to his need;

(4) The tenant can create user accounts and assign permissions for his enterprise's users, who can use the generated account to access the application;

(5) If the tenant is satisfied with the SaaS application, he can pay for the application functions needed;

(6) As soon as the SaaS application receives the rent, the tenant will change to official users.

Certainly, apart from differences from register, the multi-tenant application is much different from typical web application in login process. The multi-tenant application login process is shown as flow:

(1) User inputs his username and password.

(2) SaaS application verifies the user's input.

(3) If the user passed the verification, the multi-tenant application can get the user's tenant information through the userID and store the userID and tenantID into session variable.

(4) Users access the SaaS application.

(5) Then, the use's role information is got.

(6) The user's permission is got through his role.

(7) User gets his using environment and uses the application.

From the above process we can see that authentication module is responsible for the tenant management and rights management for all tenants. According to user's TenantID, authentication module can navigate user to the appropriate tenant environment. In this module, tenant information management functions need to be added to migration platform and the original web login module need to do a small amount of code modification.

3.3 Migration Platform Management Module

The management functions of SaaS platform include security, tenant management, resources assigning and monitoring, etc. For transformation the original web application to SaaS application, we must combine the original application with the SaaS management functions and reform some old application. To make the multi-tenant application work well, the multi-tenant platform management module should include functions as shown follow:

- Tenant management function, administrator can check the tenant's register information and manage tenant status such as trial user, official user, suspend user or lacking fees user.
- Lease fees management function, Tenant needn't buy the software and who can pay lease fees by month or year. Platform can automatic remind tenant paying for fees according to their paid time.
- Tenant template settings function, for implementation tenant's personalized user interface, multi-tenant application provides templates for each tenant in which tenant can specify their own login pages, logo, font, color scheme and unique URL.
- Log Management function: User log is a good way to record user's operations and data changed history which ensures that user's behavior can not be faked, be destroyed, and deniable.
- Resource assign and monitor, Multi-tenant application can assign and monitor resources in Cloud platform according to users' indeed requirement which can reduce the user's cost and save energy.

4. Migration experiment and analysis

In this section, a single-tenant web restaurant application was migrated to a multi-tenant model. The transformation was described in detail and transformed multi-tenant restaurant application was tested.

4.1. Web Restaurant Application Introduction

The web restaurant application has functions such as ordering, select food, billing, food management, business querying and statistics which are well used in many restaurants. The application was developed in Service-Oriented Architecture model and in four tiers which include present layer, business layer, data persistent layer and database layer. SOA model designation is a good ways to develop software in which each layer only need achieve its functions and call the lower layer function and not to care the detail realization of lower function. The functions can be wrapped to Web Services which can be called from outer system and not care the implementation detail of the Web Service. In SOA model, we can easily combine the old web restaurant with the transformation platform.

4.2. Transformation Process

With the migration platform, the transformation process is as following:

(1) The web restaurant system is deployed in the migration platform firstly. In this step, the application is deployed in the application server and database is stored in database server.

(2)The database transformation module read the restaurant system database in meta data schema and transform the meta data schema into virious multi-tenant database according to user's selection.

(3)The platform administrator adjusts the database configuration files and makes the program to connect to new muli-tenant database.

(4)The platform administrator adapts the user authority program to meet the multi-tenant application.

As can be seen by the above conversion process, in addition to user logon and adjust the database connection file, the entire conversion process is very few source code changes. The workload of the system conversion is showed in Table 1. We can see that the increment files is mainly about the database transformation module and administration module and the file updates is in authentication module. The database transformation and administration modules are written only once and can be used in many programs without any modification.

Table 1. Compare the Workload of Resource Update

system status	*file number*	*code line*	*update file number*	*update code line*
Before migration	1256	185312	25	352
After migration	1734	260108		

5. Migration Platform Test

In order to test the conversion platform we need test the conversion system functions, tenant data isolation and platform performance.

5.1. Test environment

In order to compare the performance of single-tenant restaurant applicaion and multi-tenant converted restaurant application, we need two 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: LoadRunner 8.0, HP LoadRunner 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.

5.2. System functions and tenant data isolation test

In order to test the migrated multi-tenant restaurant application can weather meet the original user's business needs and the tenant data isolation, we need test the transformed multi-tenant restaurant application.

The migrated multi-tenant restaurant application was deployed on Server1 and ten tenants were creation. The ten tenants share the restaurant application and each tenant has ten users. Users from different tenants login the multi-tenant restaurant application and set user interface and enterprise Loge for his tenant. We can find as follow:

Each user can rightly access the functions of the migrated multi-tenant restaurant application.

Users can only access the data and information belongs to their tenant.

Users can customerize their tenants' functions, user inter interface and enterprise Logo.

The application does not change the users' operation habits.

From above test, we can see that multi-tenant conversion platform does not change the functions of the restaurant, so the system converted from single-tenant model to multi-tenant model does not affect the system's original business functions. Because of adding tenant data filtering function, the converted system strictly ensures the isolation of the tenants' data.

5.3. Migration platform performance testing

Performance testing of the platform includes server response time and resource utilization test. Resource utilization mainly considers the CPU and memory usage of the server.

5.3.1. Testing the relation of user number and system performance

Testing environment: The converted multi-tenant restaurant application was deployed at Server1 and the original single-tenant restaurant was deployed at Server2. We create one tenant in the multi-tenant application and create numbers of virtual user through LoadRunner to simulate real users' operation. The statistics of the average response time and resource utilization from different number of user's simultaneous access to the server is shown as picture 4 and picture 5.

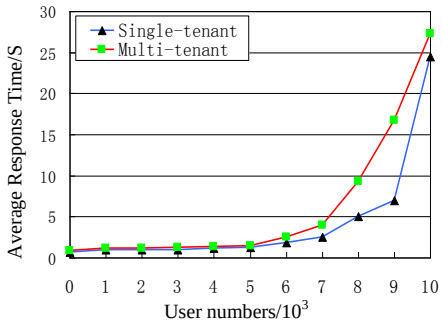

Figure 4. Relation between Response Time and User Numbers

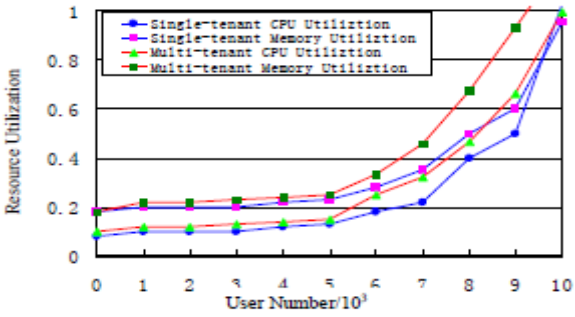

Figure 5. Relaton between Resource Utilization and User Numbers

From above charts, we can see that the server's response time increases with the increase of the user number. When the user number is less than 7000, the multi-tenant restaurant application and original restaurant application have a low response time (not more than 5 seconds). When user number exceeds 7000, the two servers' response time increase sharply. This is because when the application user numbers are few, the server resource consumes few and the server can have a timely response to clients' request. As the users number increase, the server resources consumption is increasing and response time increase. When the users numbers increase to a certain extent (when user numbers increase to 7000), the server resource reach the bottleneck and cannot have a response to client on time. At this time, more physical servers must be increased to meet the additional users access.

5.3.2. Testing the Relation between Tenants Number and Server Performance

Testing environment: The converted multi-tenant restaurant application was deployed at Server1 and the original single-tenant restaurant was deployed at Server2. We analyze the relation between the tenants number and server response time at the same user number. The testing process is shown as follow:

(1) Multiple tenants were created to simulate multiple enterprise operation process. Each tenant has 100 users to simulate multiple users and multiple tenants simultaneously access the application. The relation between tenants number and server response time was shown as Figure 6.

(2) Multiple instances of original restaurant application 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 as Figure 7.

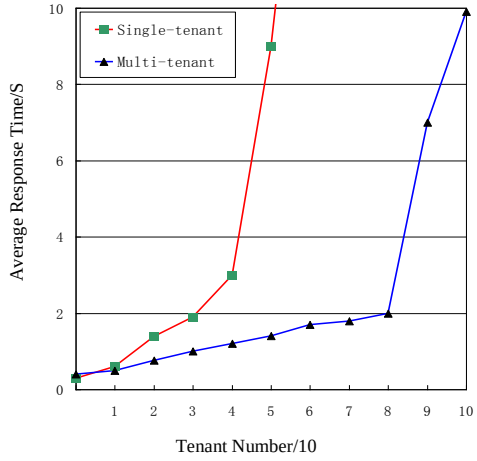

Figure 6. Relation between Tenants/Instance Number and Response Time

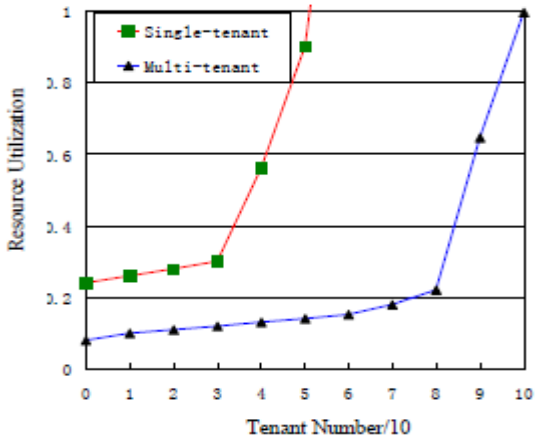

Figure 7. Relation between Tenants/Instance Number and Resource Utilization

In Figure 6, when server1 only has one tenant and server2 has one instance, the server2 response time is smaller than the server1. This is because at this time the server2 only has one instance, it consumes few resources than server1 which hosts the multi-tenant application. When 2-40 enterprises access the application, multi-tenant application response time on server1 is significantly less than the single-tenant restaurant application on server2. This is because when more enterprises need access the single-tenant application, more application and database instances need to be deployed on server2 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 server1. In single-tenant application, when the application instance number is more than 40, the server2 response time is rapid growth. This is because when more than 40 application instances are deployed in server2, the server2 resource has reached a bottleneck. When the tenant number reaches more than 80, the server1 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 has a shorter response time.

6. Conclusion

In this paper, a Java based SaaS migration platform was designed which could transform existing single-tenant ASP application to multi-tenant model. With a restaurant management application migration as example, the migration principle and process was described. The migration results showed that the conversion platform could quickly complete the multi-tenant system migration and have a small workload of artificial code modification. Finally, a performance test was taken and the performance test proved that the transformed multi-tenant application could improve the utilization of server resources.

Our future work will focus on improving the customization capabilities of the migration platform, so that the converted multi-tenant SaaS system not only can customize the user interface with application template, but also can complete tenant personalization business process through migration platform interface.

Acknowledgment

This research is supported by 2013 scientific research plan projects of Shaanxi Education Department grant number 13JK0178. We thank to reviewers for their valuable comments.

References

[1] Christof Ebert, "A Brief History of Software Technology", IEEE Software, vol. 26, no. 6, pp.22-25, 2008.
[2] Frederick C, Gianpaolo C, "Architecture Strategies for Catching the Long Tail", http://msdn.microsoft.com/en-us/library/aa479069.aspx, 2006-4.
[3] Markku Sääksjärvi, Aki Lassila, Henry Nordström, "Evaluating the software as a service business model: From CPU time-sharing to online innovation sharing", In Proceedings of the IADIS International Conference e-Society, pp.177-186, 2005.
[4] Frederick C, Gianpaolo C, Roger W, "Multi-Tenant Data Architecture", http://msdn.microsoft.com/en-us/library/aa479086.aspx, 2006-6.
[5] I-Ching Hsu, "Web 2.0-based SaaS for Community Resource Sharing", JDCTA: International Journal of Digital Content Technology and its Applications, Vol. 5, No. 5, pp. 132-141, 2011.
[6] Joseph Cosmas Mushi, Guan-zheng Tan, Felix Musau,Cheruiyot Wilson, "Performance Analysis of Recharging Scheme of M-SaaS through M-banking", JCIT: Journal of Convergence Information Technology, Vol. 6, No. 7, pp. 140-153, 2011.
[7] Ernesto Damiani, S.De CapitanidiVimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, Pierangela Samarati, "Key management for multi-user encrypted databases", In Proceedings of the 2005 ACM workshop on Storage security and survivability, pp.74-83, 2005.
[8] Amihai Motro, Francesco Parisi-Presicce, "Blind Custodians: A Database Service Architecture That Supports Privacy without Encryption", Lecture Notes in Computer Science, pp.338-352, 2005.
[9] Zhang Kun, Li Qingzhong, Shi Yuliang, "Research on Data Combination Privacy Preservation Mechanism for SaaS", Chinese Journal of Computers, Vol. 33, No. 11, pp. 2044-2054, 2010.
[10] Kong Lanju, Li Qingzhong, Shi Yuliang, et al. "Research on Index of Multi-Tenant Based on Key-Values for SaaS Application", Chinese Journal of Computer, Vol. 33, No. 12, pp. 2239-2247, 2010.
[11] Yu-Hui Wang, "The effect of Privacy, Security Reliability in SaaS Continuance Use", JCIT: Journal of Convergence Information Technology, Vol. 7, No. 7, pp. 283-291, 2012.
[12] Ralph Mietzner, Frank Leymann, "Generation of BPEL Customization Processes for SaaS Applications from Variability Descriptors", In Proceedings of the 2008 IEEE International Conference on Services Computing, IEEE Computer Society Washington, DC, USA, pp. 359-366, 2008.
[13] Ralph Mietzner, Andreas Metzger, Frank Leymann, Klaus Pohl, "Variability modeling to support custommization and deployment of multi-tenant-aware Software as a Service application", In Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems, IEEE Computer Society Washington, DC, USA, pp.18-25, 2009.
[14] Shi Yuliang, Luan Shuai, Li Qingzhong, et al. "TLA Based Customization and Verification Mechamism of Business Process for SaaS", Chinese Journal of Computer, Vol. 33, No. 11, pp. 2054-2067, 2010.
[15] Lin Hailue, Han Yanbo. "Performance Management for Multi-Tenant Web Application", Chinese Journal of Computers, Vol. 33, No. 10, pp. 1881-1895, 2010.
[16] Cor-Paul Bezemer, Andy Zaidman, Bart Plazbeecher, Toine Hurkmans, Aad't Hart, "Enabling multi-tenancy: An industrial experience report", In Proceedings of 2010 IEEE International Conference on Software Maintenance, pp.1-8, 2010.
[17] Buddhika Siddhisena, Lakmal Warusawithana, Mithila Mendis, "Next generation multi-tenant virtualization cloud computing platform", In Proceedings of 2011 13th International Conference on Advanced Communication Technology, pp.405-410, 2011.

