The SaaS Applied in Oilfield Software

Sun Lei, Li Zhuo
College of Geophysics and Information Engineering
China University of Petroleum
Beijing, China
e-mail: sunlei@cup.edu.cn, lizhuocup@gmail.com

***Abstract*— With the rapid development of the software industry in the 21st century, the traditional software is increasingly constrained by the costs and software and hardware environment. As a result, the SaaS emerged and quickly gained public acceptance. In order to use SaaS services, the users don't have to cost a lot of money to purchase entire software, only need to pay for the corresponding modules they ordered. By analysis of current oilfield software, we present a software layout mode which combines C/S and B/S model, and solve a series of problems in data storage and interaction, finally implement a SaaS framework in oilfield software industry.**

Keywords- oilfield software, SaaS, on-demand order, service

I. INTRODUCTION

Since the beginning of the 21st century, the Internet technology development has exploded. With the rapid development in internet software and hardware technology, application software technology is becoming riper. In this background, SaaS (Software as a Service) emerged as a completely new type of software application mode [1]. It brings an impact to the traditional software mode, brings fresh blood to the software world, leads a new direction for the future development of software, and makes cloud computing a hot topic.

Behind the hot debate of SaaS and cloud computing, we notice that at present the most applications of SaaS are ERP（Enterprise Resource Planning） and CRM (Customer Relationship Management) software which are used by a large number companies and people and have strong universality. In these applications, customers' functional requirements are more or less the same; just have some small differences in the number of modules and the relationship between modules. Thus we can know that if a kind of software can be transformed into SaaS service, it must have some generality and the services it provides must be able to meet the needs of many users [2]. In the oilfield software industry, we meet the same condition. We can construct a public platform providing these general services, and integrate the service modules into one platform software. Users can choose modules which they want and obtain service through the internet, and they only need pay for what they choose.

II. MAIN FRAMEWORK

With the rapid development of internet, SaaS owns the hardware foundation of being widely spread. Our client can completely utilize the ubiquitous Internet, to access the resources we need anytime, anywhere. Using LAN (Local area Network) to deploy software which is traditional is being impacted by the Internet technology. Our SaaS implementation uses the Internet which data transmission and operation based on. That is, the whole process that users use services of the SaaS depends on the Internet.

The data transmission uses Microsoft's distributed processing way, Remoting [3]. Remoting is an upgrade of DCOM (Distributed Component Object Model) and have many promotions. It can work well with the .NET platform. Remoting technology supplies a framework which allows object interacting with another object through the application domain. Through the framework, we can access the objects in the application domain, whether objects are in one process or not, even in separate computer. In this way, the client can access server's application interface remotely through .NET Remoting technology.

In our system, the customers use client to connect with server, and the data flows in the internet through Remoting technology. The customer data is stored in the SaaS provider's database server so they don't have to worry about the problems of the database, server, etc. When customer wants to get services provided by SaaS, they can easily purchase via the order system and obtain the functions immediately. The whole order process is easy, convenient and quick, saves a lot complex links, improves the user experience greatly, achieves the on-demand order and pay, and saves the customer's cost significantly.

Fig. 1 shows the main framework of SaaS in oilfield software, and we can see that Internet plays an important role in it.

III. SOFTWARE LAYOUT MODE

The software layout of SaaS system program is a very important issue. After several oil software design and development, considering the complexity of the oilfield software and the high operability of the software interface, we realize that the B/S (Browser/Server) structure is hard to achieve a satisfying performance. Although the B/S technology is growing fast and RIA (Rich Internet Application) has made great breakthrough, compared to the

Published by Atlantis Press, Paris, France.

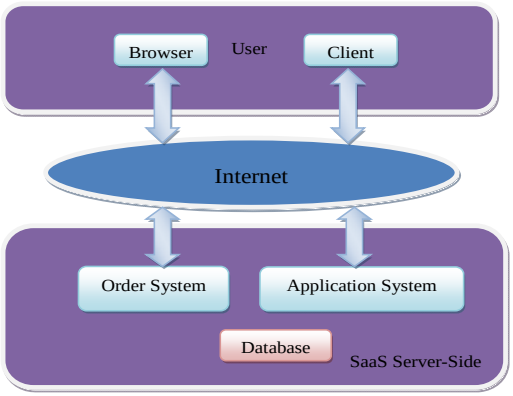

Figure 1. Main Framework of SaaS in oilfield software

traditional C/S (Client/Server) system, there are still a lot of inadequacies in B/S about the aesthetics of the interface, operability and fluency. Thus, in our design of SaaS, considering the particularity of oilfield software, we use C/S structure for the layout of our application system and B/S structure for our order system.

C/S structure is a well-known software architecture. In this mode software modules are distributed to Client and Server side appropriately in order to make full use of hardware and software resources, and nearly all the network software system use this layout before internet technology exploding. B/S structure takes the advantages of Internet and web browser, combines various scripting languages (like JavaScript, VBScript) and ActiveX technology [4-5], is a product of the rapid development of Internet technology.

In our SaaS of oilfield software, we combine B/S and C/S structure together instead of use a single model, adopt different software layout model for different target system. Fig. 2 shows the SaaS software layout.

The two systems are developed separately since the using of two different software layouts. But they use the same database, and only when customer orders the corresponding modules in the order system can they have the corresponding access in the application system. The data sharing between the two systems ensures effective access control.

IV. DATA STORAGE AND INTERACTION

Data is the values obtained by enterprise design, experimental and productive process. And it can be used for scientific research, to guide the design, to sum up experience and decision-making. For modern enterprises, data is of crucial importance. It is the lifeblood of the enterprise, and it can fully reflect the past, present and future of the enterprise.

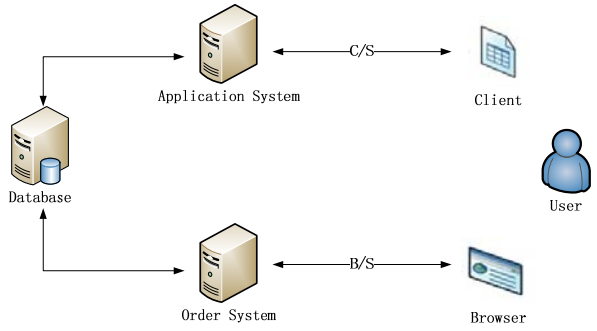

Figure 2. SaaS software layout

Data storage is an essential issue to our SaaS. How to make user data a better security, how to isolate data between different users, as well as how to get higher read/write performance, these are the problems we need to consider. We will explore these problems bellow.

A. *The relative isolation of the multi-customer data*

The SaaS in oilfield software needs to be able to serve for multiple customers at the same time. Relative to the common single customer system, its database need to be improved to isolate customer data, in order to make data between customers invisible to each other. From the customer's point of view, their data belong to themselves entirely and just themselves.

Generally, the isolation of multi-customer data can be implemented in three ways [6]:

1) Independent database, which creates separate database for each customer and there is no connection between the databases. It has the highest data isolation and security level but cost is higher.

2) Shared database, in which every customer uses the same database, but own isolated data tables. Actually in this method, database is not fully separated. But one database can support multiple customers.

3) Shared data table, in which the customers' data is completely mixed together. This method costs much lower than the first two methods, and every database can serve a large number of customers. But we must add appropriate security restrictions in the development of program, because the isolation level was reduced.

The first method has the highest isolation level, and customers' data is completely separated. On the other hand, it costs the highest. The third has a lower costs and easy to maintain, but it also has a lower isolation level. In fact, isolation level depends on the market position of the product and the customer's acceptance to the data security. We need to select specific data isolation to specific products and customers.

To save cost, we will choose the third way to isolate data between customers. In this way, we should add a common field "comCode" to identify the data's owner. Fig. 3 shows the design of data table.

B. *data security and encryption*

After the isolation of customers' data, actually customers will also be concerned about the data leakage, which may have extremely serious consequences. The SaaS service provider must be able to put user's data in a state of relative safety, encrypt customers' important data (such as password, real name, major production indicators). Through these measures, even if the database is hacked, the hacker can't get useful information but messy gibberish.

Data encryption refers to the plaintext, key and ciphertext. Plaintext is the original customer data; ciphertext is the

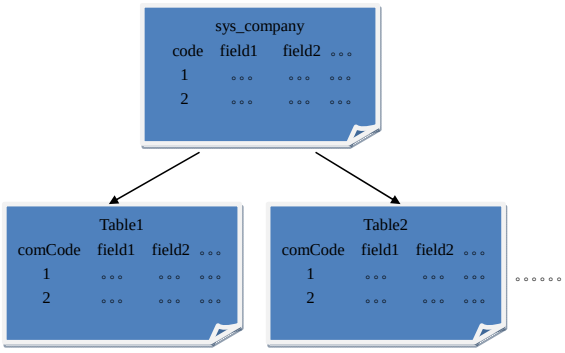

Figure 3. The data table design

encrypted content that will be stored in the database. Actually when design database, we can take out the important data tables, and isolate them physically so as to reduce the risk of data leakage.

In the data encryption process, we can use the irreversible encryption algorithm [7] such as MD5 to encrypt the data like customer passwords, because they do not need to reconstruct the data. In this way, we can fully guarantee the security of customer's data. For the data such as production data, because we must be able to decrypt the data with decryption algorithm, we use the reversible encryption algorithms, such as DES, RSA, base64, etc. according to the requirement of security level.

After data leakage is prevented by data encryption, there is another issue we need to talk: non-normal data loss, such as database server crash and natural disaster, etc. For these challenges, SaaS provider can use real-time backup technology on the database. Every database server has server keeps synchronization with it, and one will continue to work even if the other comes down. This is a mature technology but just costs a bit high. In addition, specific to natural disasters such as flood, fire, earthquake, and so on, the SaaS provider can deploy synchronization servers in different districts. Two servers may be thousands of miles apart, but they are synchronized, which can avoid data security problems brought by natural disasters.

C. Distributed storage of large amounts of data

With the number of SaaS customers increasing, customer data will be increasing too, so that ordinary single database server will gradually not be able to meet the demand. At this time, we need to design database clusters which can be expanded, and use distributed databases to store our data.

We will design it in two aspects while considering the scalability of the database server. Firstly, to split according to the data table and each server stores only certain specified data tables. In this way the database server will be more unitary, and the amount of data will be substantially reduced. Secondly, to split according to the customer and each server only stores the specific customer data, the amount of stored data will be substantially reduced as well.

By analyzing the above two ways, it can be easily found that the second approach is significantly more scalable than the first approach. In the first approach, although our data table can be split in a very fine way, even one database server only stores one data table, but with the increasing of the customers' number and their data, the amount of data of each data table will eventually be unthinkably huge, so that the first way will also arrive to the dead end. The second approach compared to the first one will split data according to the customer and all data of one customer will be stored in one database server which is specified. When customer number increases, we can increase the server number at the same time, so that its scalability can be enhanced. Of course, there are also limiting cases of the second approach, that is, when the customer data is too large to a server, the second way will encounter expansion bottleneck. In this case, we can combine the first and second approach to solve the problem, which means that we only store one customer's data table in one database server in a subdivided layer and with further data split.

D. Client and server-side data interaction

The data is stored in the SaaS server and any user of customers can connect to the database server through the Internet, to access and operate on the required data when uses the client. This is the data interaction between users' client and server-side. Just as the Remoting technology which is introduced before, in the oilfield software SaaS service platform we designed, our data interaction will use Remoting technology, so that the data can be passed interactively on the Internet.

The data requested by the user is in the form of a binary stream for transmission on the Internet, so that it can enhance the transmission speed and security, and the transmission protocols in use are HTTP, TCP/IP which we are all familiar with. Those protocols have sure been installed in the user's client machine, and so we will not require users to install other software, which will be convenient to users who will use the client.

E. Order and permissions verification

Before the interaction between user client and server-side, the first thing to do is to verify the identity of the user and only when pass the verification, the user can use the function module and continue to interact with server-side. As user authenticating in the oilfield software SaaS, in addition to the ordinary user name and password authentication, there are two types of validations. The first one is order verification. Because SaaS service will charge, each customer should order or renew the corresponding service before they can use it. And the second is permissions validation. For limiting user's operation, each customer has one administrator user and this user can set permission for all subordinate users of the customer, and only the user who has permission can operate the corresponding function modules. These two types of verification, coupled with the user name, and password authentication composes the user authentication system of SaaS, and each user who logins SaaS applications system needs to go through these verifications.

V. IMPLEMENTATION OF APPLICATION SYSTEM

Order system is multi-customer oriented and used by administrator users, and every enterprise customer registration will automatically generate an administrator user, the administrator user can do operations such as order, renewal, user management and rights management, etc. After registering and the first logging in order system, the administrator user should choose the service modules which they need, set the number of users and duration time of service, then the order system will figure out the total cost. After payment, the administrator user can create departments, roles and users, and grant the appropriate privileges to roles and users. The current customer's order status will display in the order system, and the renewal process is basically the same as order process.

Application system is customer-oriented and used for all users. The customers can create and manage their subordinate user accounts in the SaaS order system, so that the users can then use these accounts to login and operate the application system.

Application system uses C/S model which has the server and client. As described in the previous section, the server-side and client will exchange data through Remoting. The server-side provides the necessary support (basically data support) to the client, the client will get the support through the Internet by connecting to the server-side, so that the entire application system will be up and running and the server-side must remain online during the process.

The user login authentication (including the user name and password verification, order verification and permissions validation) must be processed while using the SaaS applications. After a successful login, the client will load the appropriate function module according to the customer's order as well as the user's permissions, and then generates the corresponding function menu according to the functional modules.

After the user's login, the client will generate a static instance to store user's information such as user name, user's permissions, enterprise and department information as well as a local object service proxy which has been initialized, so that, in the SaaS various functional modules can freely call these public properties and objects.

Every functional module, which exists independently to each other, contains one service provided by the SaaS, and is encapsulated in separate DLL file. The client loads modules by reading the DLL files. We read the DLL file through reflection mechanism in C #.

VI. CONCLUSION

The concept of SaaS which has been proposed over many years [8], developing until now, has a lot of achievements, and arrives in an enterprise-scale, finally becomes an successful business mode of the hot cloud computing. This article is summarized on the basis of a certain oilfield software projects. Guided by the SaaS concept and technology, we design a SaaS system solution in the oilfield software. By using the popular technical means, we have solved the problems appears in the design and implementation process. This solution breaks the traditional deployment mode of oilfield software, enables customers to order the on-demand services freely, and saves the customer's purchase costs.

In the same time, there are some deficiencies exist in the SaaS solution summarized in this paper, such as: user rights management is not detailed and flexible enough; the online payment system is not considered. We will improve them in our future work.

REFERENCES

[1] Frederiek Chong, Gianpaolo Carraro, Architecture Strategies for Catching the long Tail, Microsoft Corporation, April 2006.

[2] Mark Turner, David Budgen, Pearl Brereton, "Turning Software into a Service," IEEE Computer Society,vol. 36, 2003, pp. 38-44.

[3] P. Obermeyer, J. Hawkins, .NET Remoting: A Technical Overview, Microsift, MSDN Library, 2001.

[4] Kris Hadlock. Ajax for Web Application Developers, Beijing: Tsinghua University Press, 2007.

[5] Lei Lei,Chen Jun, "Implmentation of Web communication by ExtJS and pushlet," Software Guide, vol. 2, Sept. 2010, pp. 118-120.

[6] Ye Wei, The Software Revolution in Internet Age: SaaS Architecture Design, Beijing: Electronic Industry Press, 2009.

[7] Atul Kahate, Cryptography and network security, McGraw-Hill Education (Asia) Co., 2005.

[8] Keith Bennett, Paul Layzell, David Budgen and Pearl Brereton, Service-Based Software: The Future for Flexible software, UK: University of Durham,1999.