2016 3rd International Conference on Informative and Cybernetics for Computational Social Systems (ICCSS)

Architecture Strategies and Data Models of Software as a Service: A Review

Huixin Chen
Library
Party School of the Shandong Provincial Committee of the CPC
Jinan, Shandong, China
Email: chhx8397@126.com

***Abstract*—Cloud computing is a new paradigm that changes the way our industry works. Software as a service (SaaS), as the top model of cloud computing, has assisted the industry to develop different multi-tenant applications. It has brought new challenges of software architecture, data storage, and IT infrastructure. In this paper, we have reviewed current state-of-art of cloud computing, including the challenges, characteristics, architecture strategies, and data models of Software as a Service.**

***Index Terms*—cloud computing; Software as a service; architecture; data model**

I. Introduction

Software as a Service (SaaS) plays an important part in transforming the way information technology (IT) departments [1]. It creates more opportunities for industry to to change their focus on installing and deploying applications. The way changes from purchasing a software to renting a software [2]. That means SaaS is going to have a major impact on the software industry [3], and reduces the cost of developing applications [4].

A. Characteristics of Software as a Service

There is the tendency to convert legacy applications to SaaS applications by adding SaaS to your IT infrastructure. Not all the SaaS applications share the same characteristics, the features below are common in most SaaS applications. [5].

- Customization. Although SaaS applications share the data, they have some requirements on customization. Different tenants will has different requirements on data and business.
- Requirement change. SaaS applications have more requirements on frequent upgrades in many cases.
- Integration requirement. SaaS applications have the requirements on integration with third-party applicatons. The integration protocols are based on HTTP, REST, SOAP and JSON.

B. Organization

The remainder of the paper is organized as follows. The architecture strategies of software as a service is discussed in Section II. Section III introduces the storage strategies of multi-tenant data, and multi-tenant data models. Brief conclusions are outlined in the last section.

II. Architecture Strategies of Software as a Service

A. Software as a Service Maturity Model

In this section, we show the maturity model of SaaS. Generaaly, the maturity model ranges from separating to sharing. We can use a maturity model to describe four stages of SaaS levels, where each level is different from the previous stage. Scalability, configuration, and multi-tenancy are added to describe the SaaS maturity model shown in Figure 1.

1) The first level of maturity is called Ad Hoc/Custom. In this level, each tenant exclusively shares different instances and has its own customized instance. This model dates back to traditional application service provider (ASP) model. This architecture is similar to single instance application. Some conversional client/server applications can be converted into this category with some extra development work easily.
2) The second level of maturity is called Configurable. In this level, each tenant exclusively shares the same instance and and has its own customized instance. Each instance remains isolated from all the others. The disadvantage of this level is requiring the service providers provide sufficient hardware and storage resources to support large-scale instances.
3) The third level of maturity is called Multi-Tenant-Efficient. In this level, all the tenants share the same and only one instance. Configuration is supported to satisfy customization. The advantage of this level is to reduce the instances, and the disadvantage is that diminishing returns make it impossible to add more power cost-effectively.
4) The fourth level of maturity is called Scalable and Multi-Tenant-Efficient. In this level, each tenant connects to the same load balancing middleware, and this middleware connects to different instances. The service providers hosts multiple tenants. The advantage of this level is scalable because the middleware scales up without requiring additional re-architecting of the application

978-1-5090-3367-6/16/$31.00 ©2016 IEEE

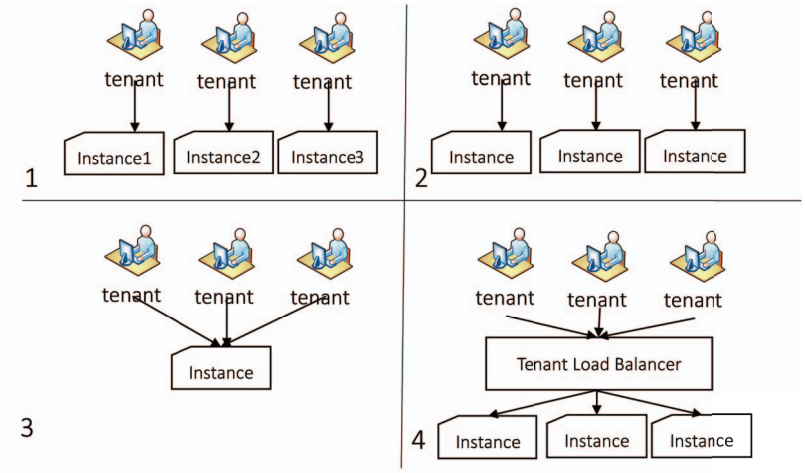

Fig. 1. Four-level Saas maturity model.

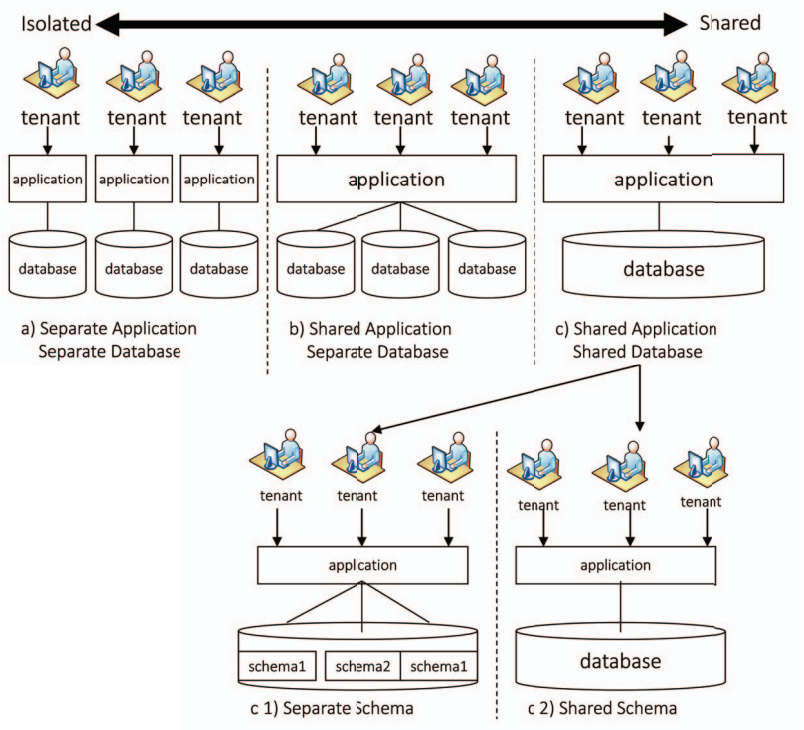

Fig. 2. Current SaaS Solutions.

III. Multi-tenant Data Model

A. Storage Strategies of Multi-tenant Data

There are three storage strategies of multi-tenant data from the balance between isolation and sharing [6]. Current SaaS data models are shown in Figure 2.

1) Separate application and separate database: Separate application and separate database is similar to single instance. Each tenant uses its own application and database. They are isolated from each other in this simple model. The disadvantage of this model is generating higher costs for maintaining so many instances. In addition, the cost of backing up is inestimable.

2) Shared application and separate database: In this model, all the tenants share the same application, and the application uses different databases. All the databases are physicaly isolated. The application security prevents any tenant from invade its own logical application instance.

3) Shared application and separate database: In this model, all the tenants share the same application and database. We divide this model into separate schema and shared schema [7].

1) Shared database and separate schema.

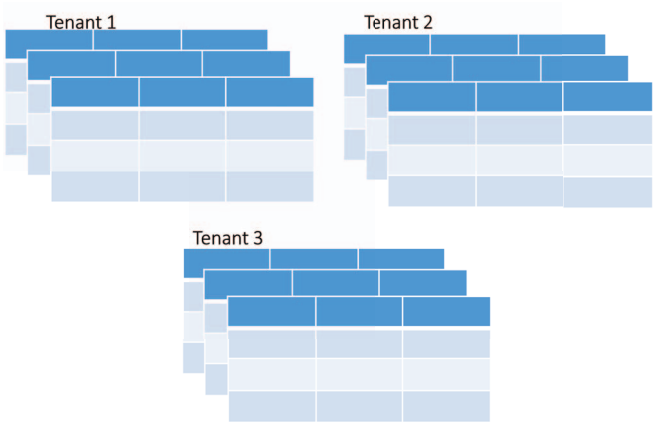

Fig. 3. Shared database and separate schema.

In this model shown in Figure 3, the database houses multiple schemas in the same database, where each tenant has its own schema (tables, views, stored procedures and triggers). The advantage of this approach is that separate schema is relatively easy to implement. But the disadvantage is the massive maintenance of schemas. To restore the database, the database administrator must restore all the schemas.

2) Shared database and shared schema.

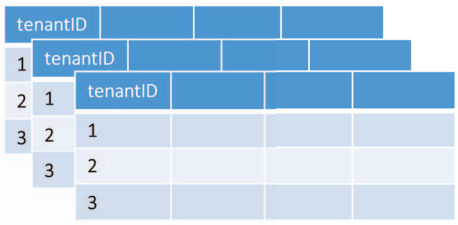

Fig. 4. Shared Database and Shared Schema.

The second model is shared database and shared schema. Compared with shared database and separate schema, all the tenants' data are stored in the same database and schema. Figure 4 shows this data model. A given table of the database includes the records of different tenants, and it is distinguished by the tenant ID.

Next, we made more analysis on the two models. The shared schema approach has the lowest hardware and backup costs, but has more maintenance and development work on distinguishing the tenant data. This is because it allows you to serve the largest number of tenants per database server. Generally, a multi-tenant data middleware is required to convert the data and minimize the development work to the utmost. In case of unexpected bugs or attacks, this approach may incur additional work.

B. Multi-tenant Data Models

This section outlines multi-tenant data storage models using private table, extension table, document store, and wide table. Figure 5 shows the metamodel of different data models. To describe the model, we give the following denotations. Column A and B are the customizing data, while other columns (C, D and E) are the personalized data.

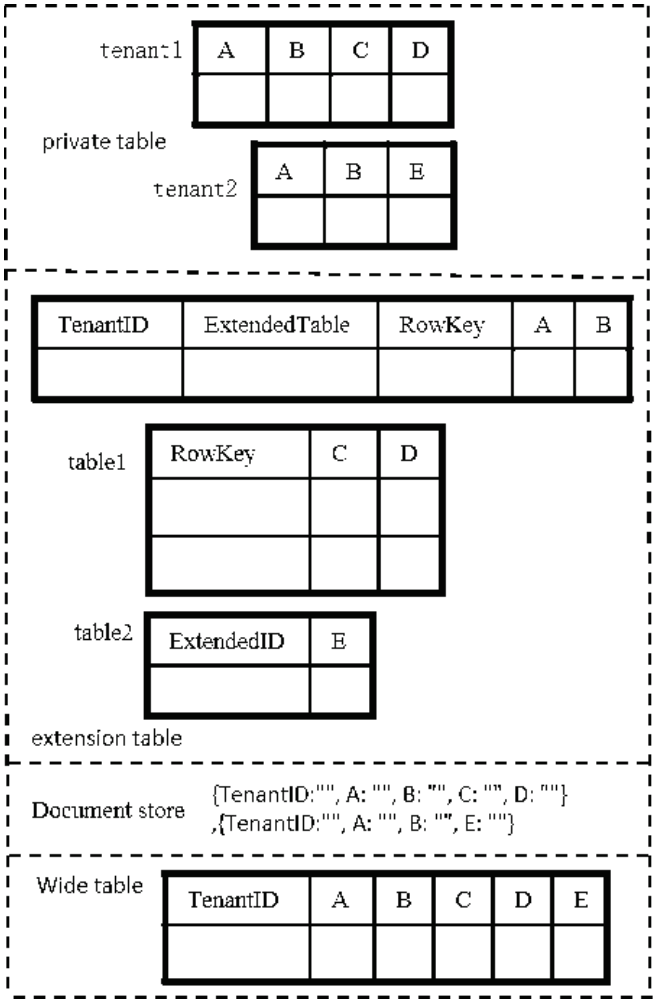

Fig. 5. Classical multi-tenant data models.

1) Private Table: Private table is affiliated to only one tenant. For example, tenant 1 has A, B, C, and D columns, and tenant 2 has A, B, and E columns. To distinguish tenant's data, the conversion layer needs renaming tables. The advantage of this model is better expansibility on the customization and isolation, but the disadvantage is massive maintenance of so many tables.

2) Extension Table: Extension table is the improvement of prviate table. Each tenant share the same basic table, and has different extension tables. The actural table is the connection of base table (sharing data) and extension table (separate data). Since base table and extension table are connected, an identity column of a tenant (tenantID) must be give to distingush the tenant data. The diadvantage is that this approach has not yet resolved the expansion problem. When it is used, the extra work is need to map both base table and extension table into one logical table.

3) Document Stores: Document store is a schema-free NoSQL key/value database that are intended for simple retrieval and appending operations. This model has significant read/search performance benefits in terms of latency and throughput. For example, document stores provide JSON-style with dynamic schemas to store data. The disadvantage is this solution has an amount of connective operations to reformulate a relational record with n columns.

4) Wide Table: 1) Single Wide Table.

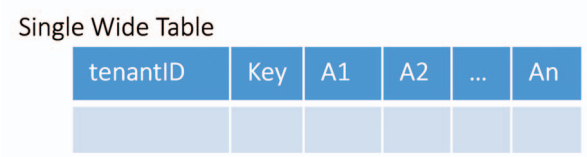

Fig. 6. Single wide table.

Single wide table is often called sparse table shown in Figure 6. In this model, we use a wide table to describe the sharing and separating tenant data. When queering, the statements are composed on only a small subset of the attributes [8]. The disadvantage is this model might cause schema null issue because this schema has too many null values. In addition, the cost is inestimable when the data is sparse.

2) Single Wide Table with Vertical Scalability.

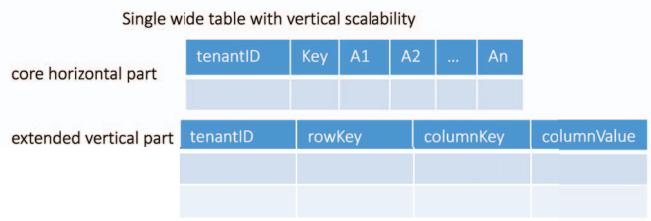

Fig. 7. Single wide table with vertical scalability.

To address schema null issue and reduce the waste of data resources efficiently, wide table is dessigned to support vertical scalability. We call this model single wide table with vertical scalability [8]. In this model, wide table is divided into basic sharing table and personalized data from wide table using extended vertical metadata. Each record in the extended vertical metadata is a key/value pair shown in Figure 7. If the the personalization of tenants is identical to each other, then the extended vertical metadata can be omitted.

3) Multiple wide tables with vertical scalability.

To reduce the cost of data migration, we divide wide tables with vertical scalability into different sizes shown in Figure 8. In this model, multiple wide tables with gradient distribution replace several single wide tables, which satified dynamic storage requirements of tenants. The core horizontal part and the extended vertical part formulate the tenants' customization.

IV. Conclusions

There is no doubt that cloud computing is a new paradigm that changes the way our industry works. Software as a service

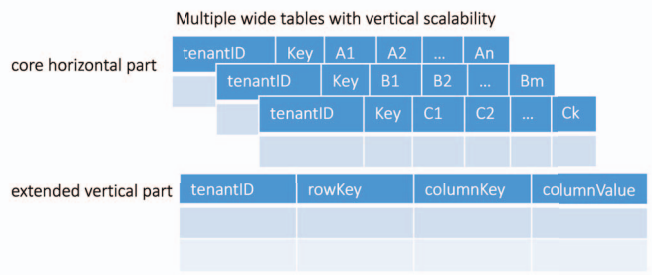

Fig. 8. Multiple wide tables with vertical scalability.

(SaaS), as the top model of cloud computing, has assisted the industry to develop different multi-tenant applications. It has brought new challenges of software architecture, data storage, and IT infrastructure. We have reviewed current state-of-art of architecture strategies of software as a service, storage strategies of multi-tenant data, and multi-tenant data models.

References

[1] G. Carraro and F. Chong, "Software as a service (saas): An enterprise perspective," *MSDN Solution Architecture Center*, 2006.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[3] D. Chantry, "Mapping applications to the cloud. microsoft corporation, january 2009."

[4] K. Ma and B. Yang, "Introducing extreme data storage middleware of schema-free document stores using mapreduce," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 20, no. 4, pp. 274–284, 2015.

[5] F. Chong and G. Carraro, "Architecture strategies for catching the long tail," *MSDN Library, Microsoft Corporation*, pp. 9–10, 2006.

[6] K. Ma, Z. Chen, A. Abraham, B. Yang, and R. Sun, "A transparent data middleware in support of multi-tenancy," in *Next Generation Web Services Practices (NWeSP), 2011 7th International Conference on*. IEEE, 2011, pp. 1–5.

[7] K. Ma, B. Yang, and A. Abraham, "A template-based model transformation approach for deriving multi-tenant saas applications," *Acta Polytechnica Hungarica*, vol. 9, no. 2, pp. 25–41, 2012.

[8] K. Ma and B. Yang, "Multiple wide tables with vertical scalability in multitenant sensor cloud systems," *International Journal of Distributed Sensor Networks*, vol. 2014, 2014.

[9] K. Ma, W. Zhang, and Z. Tang, "Toward fine-grained data-level access control model for multi-tenant applications," *International Journal of Grid and Distributed Computing*, vol. 7, no. 2, pp. 79–88, 2014.

[10] K. Ma and A. Abraham, "Toward lightweight transparent data middleware in support of document stores," in *Information and Communication Technologies (WICT), 2013 Third World Congress on*. IEEE, 2013, pp. 253–257.