Building multi-tenant SaaS applications on Windows Azure
Abstract

Based on the Windows Azure platform, companies can easily build Web applications. In reality, most traditional Web applications would migrate to the cloud, or re-build as a multi-tenant SaaS application hosted on Windows Azure. Different enterprises have different approaches to implementing multi-tenant SaaS applications. This document aims to provide an introduction to multi-tenant solutions on Windows Azure.
Introduction

Traditional Web applications, such as Asp.net, Java, and PHP websites can easily be migrated to Microsoft’s Azure cloud platform. SaaS (Software as a Service) has become a common delivery model for many business applications. There are numerous design considerations that must be taken into account for a simple Web application (including its front page and backend database) that offers multi-tenant support after migration to the cloud: scalability of entire structure, cost and manageability of backend system. Different designs will lead to different results.

In the first scenario - migrating traditional Web applications that have a single-tenant structure to the cloud - this would require a change from single-tenant to multi-tenant architecture. From a system architecture perspective, architects need to consider the following concerns:

- How to adapt the backend database for multi-tenant usage
- How to adapt the application layer to support multi-tenant usage
- Whether the tenant can share computing and database resources
- Application version changes and database upgrade.

In the second scenario - migrating traditional Web applications that have a multi-tenant structure to the cloud - in theory, only minor adjustments are needed for migration to the Windows Azure platform. However, architects still need to consider the following issues:
• How to make use of the Azure platform’s storage and computing resources
• Whether the SQL Azure Database should be shared among multiple tenants, with regard to database resources
• How to minimize costs on Azure
• Administrator responsibilities for every tenant’s usage of resource management

Whether we are constructing based on the existing application, or constructing an entirely new multi-tenant application based on Windows Azure, we need to account for the following questions. For each different application, we have different areas of focus upon which we must pay attention. For example, video applications need more storage and computing, while CRM or ERP applications require more database and computing power.

Currently, there is an increase in the usage of multi-tenant applications on the Azure platform. For example, for a multi-tenant CRM software that is built on Azure, once every tenant is created, storage and database resources must be allocated. When an upgrade version is required, the administrator can conveniently implement the application update. All these factors need to be taken account for a multi-tenant application.

Traditional single-tenant application migration

When migrating traditional single-tenant Web applications to the cloud, the software vendor may not want to change the existing program architecture. Some software vendors would continue to use the existing architecture, and deploy a set of procedures for every tenant, as shown below.
The advantage in this approach is that there are separate databases for different tenants, simplifying the design of the data model extension, as well as meeting the unique needs of different tenants; when there is a failure, recovering data would be relatively simple. In addition, user data isolation level would be the highest, providing best security.

Meanwhile, it also raises some issues: when the instance of database installation is increased, maintenance and acquisition costs rise. The difference between this solution and the deployment of a more traditional single-tenant set of data is whether the software is unifying deployed on the cloud platform. When dealing with tenants requiring high levels of data isolation like banks or hospitals, we can choose this model, which improves tenant security, though with a higher price. However, if the allowable budget is much lower, and the tenants require a lower level of data isolation, this solution is usually unacceptable to most operators.

From an application development and maintenance point of view, the administrator needs to maintain various versions; even one change will require an update on all applications. Since the computing resource is not shared, there will be enormous wastage of computing resources in the implementation procedure. With regard to deployment management, this is a huge challenge to the administrator, since every single application needs to be managed individually.

Very seldom, this framework is used for applications hosting large numbers of tenants. For these scenarios, if there is a further increase in tenants, the system needs to implement a different solution as shown below.

A typical multi-tenant application

Figure 2 shows a typical multi-tenant application, in which all tenants have access to the same applications, sharing data source and application. Each tenant hosts its own data, isolated from other tenants.

Advantages are:

- Solving upgrade and maintenance issue, keeping version changes and updates at a minimum. Since the program is a shared version for all tenants, upgrade error is therefore reduced.
- Building on a shared architecture, the program maximizes the use of computing resources.
- Easy deployment management.

However, this situation has also raised the following issues:

- Interaction of resources. During a failure, data recovery can be more difficult, because the recovery database involves data of other tenants.
- Lower security.
Multi-tenant application on Azure

Storage resources allocation

As the number of tenant increases, applications and databases need to have good expansion. Windows Azure allows administrators to call upon PowerShell to expand storage and computing resources. A Windows Azure storage account can contain an unlimited number of Containers, as long as their total size is under 100TB. Figure 3 illustrates a multi-tenant storage architecture, with every tenant assigned a Container. Each tenant has access to operating authority over its own Container.
Computing resources extension

Let’s look at the application computing resources provided by Windows Azure Web and Worker roles, and how they should be used in a multitenant solution to host Web services, Web sites, and other general processing.

The clearest use of Windows Azure Web and Worker roles in a multi-tenant application is to provide additional (or to reduce!) compute resources on demand in the service of the tenants, with the notion that a single role instance serves multiple tenants.

Worker Roles are used to provide background processing. In the fully multi-tenant scenario, the approach you take is to have some Worker roles to loop processes work items agnostic of which tenant created it -- hence you are sharing the compute resources across all tenants.

Database solution for multi-tenant

For the database, there are three options:

- Single-tenant single database: one database for each tenant
- Multi-tenant single database: multiple tenants share a database
- Multi-tenant multi-database: the system consists of multiple databases; a database can contain data from multiple tenants
The comparative advantages and disadvantages are as follows:

<table>
<thead>
<tr>
<th></th>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-tenant single</td>
<td>• High security</td>
<td>• Excessive number of databases</td>
</tr>
<tr>
<td>database</td>
<td>• Database not too large</td>
<td>• Upgrade complexity</td>
</tr>
<tr>
<td></td>
<td>• Database failure affects only a single tenant</td>
<td>• Database waste of resources</td>
</tr>
<tr>
<td></td>
<td>• Least amount of code changes</td>
<td></td>
</tr>
<tr>
<td>Multi-tenant single</td>
<td>• Less number of databases required</td>
<td>• Performance affected when the database gets too big</td>
</tr>
<tr>
<td>database</td>
<td>• Easy upgrade and maintenance</td>
<td>• Poor security</td>
</tr>
<tr>
<td></td>
<td>• Database resources rarely wasted</td>
<td>• Database failure affects the entire system</td>
</tr>
<tr>
<td>Multi-tenant multi</td>
<td>• Moderate number of databases</td>
<td>• Code implementation difficult</td>
</tr>
<tr>
<td>database</td>
<td>• More effective use of database resources</td>
<td>• Database segmentation and merger operations</td>
</tr>
<tr>
<td></td>
<td>• Certain security and isolation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Flexible distribution</td>
<td></td>
</tr>
</tbody>
</table>

Table 1

Scale SQL Database

Federations in SQL Database are used to achieve greater scalability and performance. One or more tables within a database are split by row and portioned across multiple databases (Federation members). This type of horizontal partitioning is often referred to as “sharding.” The primary scenarios in which this is useful are those in which it is necessary to achieve scale, performance, or to manage capacity. When designing a federation, one of the most important design decisions is what value to federate on. Ideally you want to select a key that allows you to federate data from multiple, related tables so related rows are stored together. For example, in the case of a multi-tenant application you might select the tenant_id. The rows within each federated table that specify the same tenant_id value would be stored in the same federation atomic unit.
In the SQL Azure Federation, the Federation Key must be the table’s primary key or clustered index. Related tables for all tenants to have a Tenancy ID field and this field are added to the primary key, which is the premise of using SQL Azure Federation. Since the only requirement of the Federation Key is the same as that of the primary key, it will not undermine the existing database schema.

Multi-tenant application metrics

With regard to multi-tenant applications on Azure, since each tenant uses different resources, the backend system needs to have a metric for every tenant. Under normal circumstances, we can consider the following aspects:

- Storage service metrics, including storage size and storage transactions for each tenant. For example, to enable metrics for the Blob, Table, and Queue services.
- Database metrics: the size of the SQL Database.
- Measure of computing and network usage.

Symbio Azure team

For different multi-tenant applications, a need exists to analyze user scenarios, since there is no definite answer on which solution to deploy.

The Symbio Azure team provides one to two weeks of in-depth analysis for migration of traditional applications to the cloud. We provide complete solution, including migration steps and all documentation. Simultaneously, for multi-tenant applications, Symbio provides best practices for SQL Azure Federation. With a Symbio solution, clients can implement migration to the cloud platform independently, or engage Symbio’s development services. Symbio also provides continuous development services for the Windows Azure platform for multi-tenant applications and mobile applications, large-scale Web applications and enterprise applications integration.

For more information contact marketing@symbio.com or complete a Request for Service at www.symbio.com/contactus