David Chappell

October 2012

WINDOWS AZURE DATA MANAGEMENT

CHOOSING THE RIGHT TECHNOLOGY

Sponsored by Microsoft Corporation

Contents

Windows Azure provides several different ways to store and manage data. This diversity lets users of the platform address a variety of different problems. Yet diversity implies choice: Which data management option is right for a particular situation?

This short overview is intended to help you answer that question. After a quick summary of the data management technologies in Windows Azure, we'll walk through several different scenarios, describing which of these technologies is most appropriate in each case. The goal is to make it easier to choose the right options for the problem you face.

Windows Azure Data Management: A Summary

Windows Azure has four main options for data management[1]:

- *SQL Database*, a managed service for relational data. This option fits in the category known as *Platform as a Service (PaaS)*, which means that Microsoft takes care of the mechanics—you don't need to manage the database software or the hardware it runs on.
- *SQL Server in a VM*, with SQL Server running in a virtual machine created using Windows Azure Virtual Machines. This approach lets you use all of SQL Server, including features that aren't provided by SQL Database, but it also requires you to take on more of the management work for the database server and the VM it runs in.
- *Blob Storage*, which stores collections of unstructured bytes. Blobs can handle large chunks of unstructured data, such as videos, and they can also be used to store smaller data items.
- *Table Storage*, providing a NoSQL key/value store. Tables provide fast, simple access to large amounts of loosely structured data. Don't be confused, however: Despite its name, Table Storage doesn't offer relational tables.

All four options let you store and access data, but they differ in a number of ways. Figure 1 summarizes the differences that most often drive the decision about which data management technology to use.

[1] For an introduction to these technologies, see *Windows Azure Data Management and Business Analytics*, David Chappell.

	Storage Type	*Access Mechanisms*	*Transaction Support*	*Maximum Size*	*Price (US Dollars)*
SQL Database	Relational tables	*Protocol*: TDS *Interfaces:* ADO.NET, Entity Framework, etc.	Across all tables in a database	150 GB per database, much larger with federation	$9.99 to $0.999 /GB per month
SQL Server in a VM	Relational tables	*Protocol*: TDS *Interfaces*: ADO.NET, Entity Framework, etc.	Across all tables in a database	Several TBs per database, depending on VM size	$2.02 to $0.125 per hour, plus blob storage for data disks
Blob Storage	Binary objects	*Protocol*: REST *Interfaces*: Windows Azure Storage Client, Windows file I/O	None	200 GB per block blob, 1 TB per page blob	$0.125 to $0.037/GB per month, $0.01/ 100,000 operations
Table Storage	Key/value store	*Protocol*: OData *Interfaces*: Windows Azure Storage Client	Across entities in the same partition in a table	100 TB per table	$0.125 to $0.037/GB per month, $0.01/ 100,000 operations

Figure 1: Different Windows Azure data management options have different characteristics.

Despite their differences, all of these data management options provide built-in replication. For example, all data stored in SQL Database, Blob Storage, and Table Storage is replicated synchronously on three different machines within the same Windows Azure datacenter. Data in Blob Storage and Table Storage can also be replicated asynchronously to another datacenter in the same region (North America, Europe, or Asia). This geo-replication provides protection against an entire datacenter becoming inaccessible, since a recent copy of the data remains available in another nearby location. And because databases used with SQL Server running in a VM are actually stored in blobs, this data management option also benefits from built-in replication.

It's also worth noting that third parties provide other data management options running on Windows Azure. For example, ClearDB provides a managed MySQL service, while Cloudant provides a managed version of its NoSQL service. It's also possible to run other relational and NoSQL database management systems in the Windows and Linux VMs that Windows Azure provides.

Choosing Data Management Technologies: Examples

Choosing the right Windows Azure data management option requires understanding how your application will work with data in the cloud. What follows walks through some common scenarios, evaluating the Windows Azure data management options for each one.

Creating an Enterprise or Departmental Application for a Single Organization

Suppose your organization chooses to build its next enterprise or departmental application on Windows Azure. You might do this to save money—maybe you don't want to buy and maintain more hardware—or to avoid complex procurement processes or to share data more effectively via the cloud. Whatever the reason, you need to decide which Windows Azure data management options your application should use.

If you were building this application on Windows Server, your first choice for storage would likely be SQL Server. The relational model is familiar, which means that your developers and administrators already understand it. Relational data can also be accessed by many reporting, analysis, and integration tools. These benefits still apply to relational data in the cloud, and so the first data management option you should consider for this new Windows Azure application is SQL Database. Most of the existing tools that work with SQL Server data will also work with SQL Database data, and writing code that uses this data in the cloud is much like writing code to access on-premises SQL Server data.

There are some situations where you might not want to use SQL Database, however, or at least, not only SQL Database. If your new application needs to work with lots of binary data, such as displaying videos, it makes sense to store this in blobs—they're much cheaper—perhaps putting pointers to these blobs in SQL Database. If the data your application will use is very large, this can also raise concerns. SQL Database's limit of 150 gigabytes per database isn't stingy, but some applications might need more.

One option for working with data that exceeds this limit is to use SQL Database's federation option. Doing this lets your database be significantly larger, but it also changes how the application works with that data in some ways. A federated database contains some number of federation members, each holding its own data, and an application needs to determine which member a query should target.

Another choice for working with large amounts of data is to use Windows Azure Table Storage. A table can be much, much bigger than a non-federated database in SQL Database, and tables are also significantly cheaper. Yet using tables requires your developers and administrators to understand a new data model, one with many fewer capabilities than SQL Database offers. Choosing tables also means giving up the majority of data reporting, analysis, and integration tools, since most of these are designed to be used with relational data.

Still, enterprise and departmental applications primarily target an organization's own employees. Given this limited number of users, a database size limit of 150 gigabytes may well be enough. If it is, sticking with SQL Database and the relational world you know is likely to be the best approach.

One more option worth considering is SQL Server running in a Windows Azure VM. This entails more work than using SQL Database, especially if you need high availability, as well as more ongoing management. It does let you use the full capabilities of SQL Server, however, and it also allows creating larger databases without using federation. In some cases, these benefits might be worth the extra work this option brings.

Creating a SaaS Application for Many Organizations

Using relational storage is probably the best choice for a typical business application used by a single enterprise. But what if you're a software vendor creating a Software as a Service (SaaS) application that will be used by many enterprises? Which cloud storage option is best in this situation?

The place to start is still relational storage. Its familiarity and broad tool support bring real advantages. With SQL Database, for example, a small SaaS application might be able to use just one database to store data for all of its customers. Because SaaS applications are typically multi-tenant, however, which means that they support multiple customer organizations with a single application instance, it's common to create a separate database for each of these customers. This keeps data separated, which can help sooth customer security fears, and it also lets each customer store up to 150 gigabytes. If this isn't enough, an application can use SQL Database's federation option to

spread the application's data across multiple databases. And as with enterprise applications, using SQL Server in a VM can sometimes be a better alternative, such as when a SaaS application needs the full functionality of the product or when you need a larger relational database without using federation.

Yet Windows Azure Table Storage can also be attractive for a SaaS application. Its low cost improves the SaaS vendor's margins, and its scalability allows storing huge amounts of customer data. Tables are also very fast to access, which helps the application's performance. If the vendor's customers expect traditional reports, however, especially if they wish to run these reports themselves against their data, tables are likely to be problematic. Similarly, if the way the SaaS application accesses data is complicated, the simple key/value approach implemented by Table Storage might not be sufficient. Because an application can only access table data by specifying the key for an entity, there's no way to query against property values, use a secondary index, or perform a join. And given the limitations of transactions in Table Storage—they're only possible with entities stored in the same partition—using them for business applications can create challenges for developers.

Table Storage also brings one more potential drawback for SaaS applications. Especially for smaller software vendors, it's common to have some customers demand an on-premises version of the application—they don't (yet) want SaaS. If the application uses relational storage on Windows Azure, moving both code and data to Windows Server and SQL Server can be straightforward. If the SaaS application uses Table Storage, however, this gets harder, since there's no on-premises version of tables. When using the same code and data in the cloud and on-premises is important, Table Storage can present a roadblock.

These realities suggest that if relational storage can provide sufficient scale and if its costs are manageable, it's probably the best choice for a SaaS application. Either SQL Server in a VM or SQL Database can be the right choice, depending on the required database size and the exact functionality you need. One more issue to be aware of is that because SQL Database is a shared service, its performance depends on the load created by all of its users at any given time. This means, for example, that how long a stored procedure takes to execute can vary from one day to the next. A SaaS application using SQL Database must be written to allow for this performance variability.

One final point: An application certainly isn't obligated to use just one of the platform's storage options. For example, a SaaS application might use SQL Database for customer data, Table Storage for its own configuration information, and Blob Storage to hold uploaded video. Understanding the strengths and weaknesses of each option is essential to choosing the right technology.

Creating a Scalable Consumer Application

Suppose you'd like to create a new consumer application in the cloud, such as a social networking app or a multi-player game. Your goal is to be wildly successful, attracting millions of users, and so the application must be very scalable. Which Windows Azure storage option should you pick?

Relational storage, whether with SQL Database or SQL Server in a VM, is less attractive here than it is for the more business-oriented scenarios we've looked at so far. In this scenario, scalability and low cost are likely to matter more than developer familiarity and the ability to use standard reporting tools. Given this, Table Storage should probably be the first option you consider. Going this route means that your application must be designed around the simple kinds of data access this key/value store allows, but if this is possible, tables offer fast data access and great scalability at a low price.

Depending on the kind of data your application works with, using Blob Storage might also make sense. And relational storage can still play a role, such as storing smaller amounts of data that you want to analyze using traditional business intelligence (BI) tools. In large consumer applications today, however, it's increasingly common to use an inexpensive, highly scalable NoSQL technology for transactional data, such as Table Storage, then use the open-source technology Hadoop for analyzing that data. Because it provides parallel processing of unstructured data, Hadoop can be a good choice for performing analysis on large amounts of non-relational data. Especially when that data is already in the cloud, as it is with Windows Azure tables, running Hadoop in the cloud to analyze that data can be an attractive choice. Microsoft provides a Hadoop service as part of Windows Azure, and so this approach is a viable option for Windows Azure applications.

Creating a Big Data Application

By definition, a big data application works with lots of data. Cloud platforms can be an attractive choice in this situation, since the application can dynamically request the resources it needs to process this data. With Windows Azure, for instance, an application can create many VMs to process data in parallel, then shut them down when they're no longer needed. This can be significantly cheaper than buying and maintaining an on-premises server cluster.

As with other kinds of software, however, the creators of big data applications face a choice: What's the best storage option? The answer depends on what the application does with that data. Here are two examples:

- An application that renders frames for digital effects in a movie might store the raw data in Windows Azure Blob Storage. It can then create a separate VM for each frame to be rendered and move the data needed for that frame into the VM's local storage[2]. This puts the data very close to the application—it's in the same VM—which is important for this kind of data-intensive work. VM local storage isn't persistent, however. If the VM goes down, the work will be lost, and so the results will need to be written back to a blob. Still, using the VM's own storage during the rendering process will likely make the application run significantly faster, since it allows local access to the data rather than the network requests required to access blobs.

- As mentioned earlier, Microsoft provides a Hadoop service on Windows Azure for working with big data. Called *HDInsight*, a Windows Azure application that uses it processes data in parallel with multiple instances, each working on its own chunk of the data. A Hadoop application might read data directly from blobs or copy it into a VM's local storage for faster access.

One more concern with running big data applications in the cloud is getting the data into the cloud in the first place. If the data is already there—maybe it's created by a consumer SaaS application using Table Storage—this isn't a problem. If the big data is on premises, however, moving it into Blob Storage can take a while. In cases like this, it's important to plan for the time and expense that moving the data will entail.

Creating a Data Streaming Application

An application that does data streaming, such as an online movie service, needs to store lots of binary data. It also needs to deliver this data effectively to its users, perhaps even to users around the world. Windows Azure can address both of these requirements.

---

[2] A VM's local storage is sometimes called *instance storage*.

To store the data, a data streaming application can use Blob Storage. Blobs can be big, and they're inexpensive to use. To deliver the data in a blob, an application can use the Windows Azure Content Delivery Network (CDN). By caching data closer to its users, the CDN speeds up delivery of that data when it's accessed by multiple customers. The Windows Azure CDN also supports smooth streaming to help provide a better video experience for diverse clients, including Windows 8, iOS, Android, and Windows Phone.

Moving a Windows Server Application to Windows Azure

It sometimes makes sense to move code and data from the on-premises Windows Server world to Windows Azure. Suppose an enterprise elects to move an existing application into the cloud to save money or free up datacenter space or for some other reason. Or imagine that a software vendor decides to create a SaaS version of an existing application, reusing as much of the application's code as possible. In situations like these, what's the best choice for data management?

The answer depends on what the application already uses. If the goal is to move to the cloud with as little change as possible, then using the Windows Azure options that most closely mirror the on-premises world is the obvious choice. For example, if the application uses SQL Server, the simplest option will likely be to use SQL Server running in a Windows Azure VM. This lets the application use whatever aspects of SQL Server it needs, since the same product is now running in the cloud. Alternatively, it might make sense to modify the application a bit to use SQL Database. Like SQL Server, SQL Database can be accessed using Entity Framework, ADO.NET, JDBC, PHP, and other options, which allows moving the application without extensive changes to its data access code. Similarly, an application that relies on the Windows file system can use the NTFS file system provided by Windows Azure drives to move code with as little change as possible.

Creating an On-Premises Application that Uses Cloud Data

Not all applications that use cloud data need to run in the cloud. It can sometimes be useful to store data in Windows Azure, then access that data from software running somewhere else. Here are a few examples:

- An application running on mobile phones can rely on a large database stored in the cloud. For instance, a mapping application could require a database that's too large to store on the phone, as might a mobile application that does language translation.
- A business might store data in the cloud to help share the information across many users, such as its divisions or a dealer network. Think about a company that provides a large parts catalog, for example, that's updated frequently. Storing this data in the cloud makes it easier to share and to update.
- The creators of a new departmental application might choose to store the application's data in the cloud rather than buy, install, and administer an on-premises database management system. Taking this approach can be faster and cheaper in quite a few situations.
- Users that wish to access the same data from many devices, such as digital music files, might use applications that access a single copy of that data stored in the cloud.

This kind of application is sometimes referred to as a *hybrid*: It spans the cloud and on-premises worlds. Which storage option is best for these scenarios depends on the kind of data the application is using. Large amounts of geographic data might be kept in Blob Storage, for example, while business data, such as a parts catalog or data

used by a departmental application, might use SQL Database. As always, the right choice varies with how the data is structured and how it will be used.

Storing Backup Data in the Cloud

One more popular way to use Windows Azure storage is for backups. Cloud storage is cheap, it's offsite in secure data centers, and it's replicated, all attractive attributes for data backup.

The most common approach for doing this is to use Blob Storage. Because blobs can be automatically copied to another datacenter in the same region, they provide built-in geo-replication. If desired, users are free to encrypt backups as well before copying them into the cloud. Another option is to use StorSimple, an on-premises hardware appliance that automatically moves data in and out of Windows Azure Blob Storage based on how that data is accessed. Especially for organizations new to the public cloud, using blobs for backup data can be a good way to get started.

Final Thoughts

Applications use many different kinds of data, and they work with it in different ways. Because of this, Windows Azure provides several options for data management. Using these options effectively requires understanding each one and the scenarios for which it makes the most sense.

Each of the platform's data management options—SQL Database, SQL Server in a VM, Blob Storage, and Table Storage—has strengths and weaknesses. Once you understand what your needs are, you can decide which tradeoffs to make. Is cost most important? Does massive scalability matter, or is the ability to do complex processing on the data more important? Do you need to use standard business intelligence tools on the data? Will you be doing analysis with big data? Whatever your requirements, the goal of Windows Azure is to provide a data management technology that's an effective solution for your problem.

About the Author

David Chappell is Principal of Chappell & Associates (www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps people around the world understand, use, and make better decisions about new technologies.