David Chappell
& Associates

WINDOWS AZURE AND ISVS

A GUIDE FOR DECISION MAKERS

DAVID CHAPPELL

JULY 2009

SPONSORED BY MICROSOFT CORPORATION

CONTENTS

ISVS AND CLOUD COMPUTING

Why should an independent software vendor (ISV) care about cloud computing? The answer is simple: Using the cloud has the potential to increase an ISV's revenues and/or decrease its costs. Running code and storing data on computers in large Internet-accessible data centers owned by somebody else can offer compelling advantages. Anyone responsible for charting the course of an ISV ought to be thinking seriously about how cloud computing will affect their business.

One option for an ISV looking to benefit from the cloud is to exploit Windows Azure. Designed to support highly scalable and reliable applications, Windows Azure is Microsoft's flagship cloud platform technology. This paper takes a look at how and why ISVs might use Windows Azure. The goal is to make clear why cloud computing is important for ISVs and to illustrate how they might use this new platform.

Before diving into the topic, it's worth summarizing up front some of the main ideas. Here are the key points to understand:

- A primary goal of Windows Azure is to be a platform on which ISVs can create Software as a Service (SaaS) applications. Customers are increasingly interested in having a SaaS option for the software they buy. To satisfy this demand and to keep pace with the competition, many ISVs will choose to offer a SaaS version of their current or future products. Creating a SaaS application requires building a highly scalable, highly reliable cloud-based service that can be used simultaneously by many customer organizations. Building your own foundation for this makes no more sense than would writing your own operating system for an on-premises application. Just as Windows provides a foundation for traditional on-premises applications, Windows Azure can provide a foundation for SaaS applications.

- Cloud computing needn't be an all-or-nothing proposition, and so SaaS applications aren't the only choice. An existing on-premises application can be enhanced with cloud-based functionality, such as running some code or storing a subset of data on Windows Azure. This incremental approach to using a cloud platform can save money and improve a current application's functionality. It can also provide a low-risk way to gain experience with this new kind of technology.

- Cloud platforms aren't useful only for firms that create end-user applications. If you're an ISV that provides infrastructure add-ons or developer aids for the on-premises Windows environment, it's likely that you can also find value-added products to create for Windows Azure. As more computing moves into the cloud, finding these new offerings is likely to be an important way to maintain your revenue stream.

- A cloud platform such as Windows Azure is different from traditional hosting. From a technical perspective, Windows Azure provides simpler administration, as well as services designed to create scalable and reliable applications. The business differences include minimal up-front commitment and easier ways to increase and decrease the computing resources your application uses. These differences mean that Windows Azure can potentially provide better technology and lower costs for ISV applications.

Initially, Windows Azure is likely to be used to support today's applications in the cloud. It's worth pointing out, however, that cloud platforms offer services we haven't seen before, such as access to large numbers of cheap CPUs and massively scalable data storage. Along with support for the world we already

know, we should expect to see creative ISVs finding ways to do wholly new things with this new kind of platform.

A BRIEF OVERVIEW OF WINDOWS AZURE

Making good decisions about how to use Windows Azure requires a basic understanding of the platform. This section provides an overview of the technology and its associated business model.

TECHNOLOGY

Windows Azure has three main parts: a Compute service that runs applications, a Storage service that stores data, and a Fabric that supports the Compute and Storage services. Figure 1 shows this breakdown.

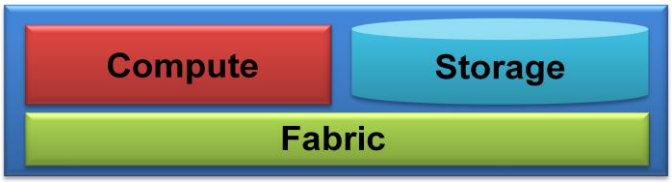

Figure 1: Windows Azure has three main parts: The Compute service, the Storage service, and the Fabric upon which both depend.

To use the Compute service, a developer creates a Windows application. This application might be written using C# and the .NET Framework, using C++ and the Win32 APIs, or in some other way. However it's built, the application must be implemented as *Web roles*, *Worker roles*, or both. Figure 2 illustrates this idea.

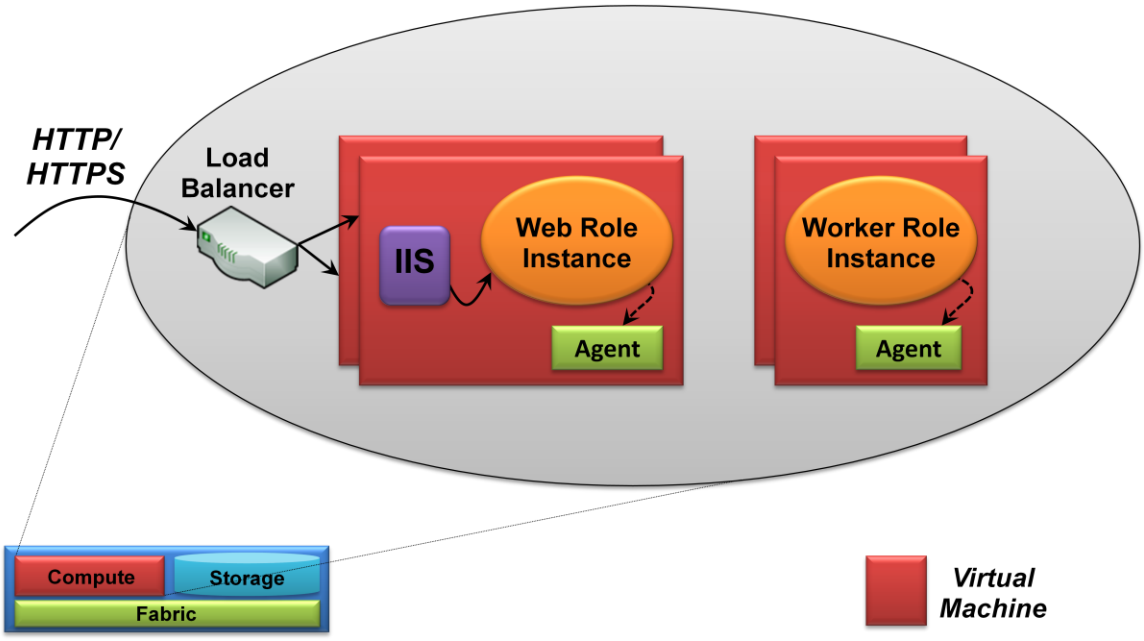

Figure 2: Applications built on the Windows Azure Compute service can consist of Web role instances, Worker role instances, or both.

As the name suggests, a Web role instance accepts Web requests. It can be created using ASP.NET or another technology that works with Internet Information Services (IIS). Whatever technology is used, Windows Azure provides built-in hardware load balancing across all Web role instances in a particular application.

For functions that aren't intended to respond directly to Web requests, a Windows Azure application can also contain Worker role instances. A Worker role instance is just a Windows application with a main(), and it can run indefinitely. Among other things, this model allows creating scalable applications where Web role instances accept requests, then pass them to Worker role instances to be processed. And while both Web role instances and Worker role instances are ordinary Windows applications, a few things, such as logging, require direct access to Windows Azure. To allow this, applications can call directly into a Windows Azure agent, as Figure 2 shows.

Notice that each instance—Web role or Worker role—runs inside its own virtual machine (VM). This provides isolation, letting Windows Azure applications run with full trust, and it also allows a clear view into application performance, since there's a defined mapping between VMs and processor cores. But a developer doesn't explicitly create VMs. Instead, she uploads an application to Windows Azure, together with an XML configuration file that specifies how many Web role instances and Worker role instances should run. Once this is done, Windows Azure creates the required number of VMs, then monitors their execution. If an instance fails, Windows Azure will start a new one, making sure that the specified number of Web role and Worker role instances are always running. (This work is done by the *Fabric Controller*, software that's in charge of all machines in a particular instance of the Fabric.) To increase or decrease the number of running instances, the owner of an application can change the value for either instance type in the application's configuration. Windows Azure automatically creates or shuts down VMs to match this new setting.

Given that Windows Azure applications are essentially Windows apps, it shouldn't be surprising that developers can create them with Visual Studio. This tool provides templates for creating cloud applications as Web roles, Worker roles, or both. Windows Azure also provides a *Development Fabric*, which is a facsimile of Windows Azure that runs on a local machine. Developers can use this to create their code and do initial testing, then upload the app to Windows Azure when it's ready.

Applications usually need persistent storage, and so Windows Azure provides its own cloud-based mechanisms for storing and retrieving data. The platform offers three storage options, all accessed via standard HTTP GETs, PUTs, and DELETEs. Figure 3 illustrates this.

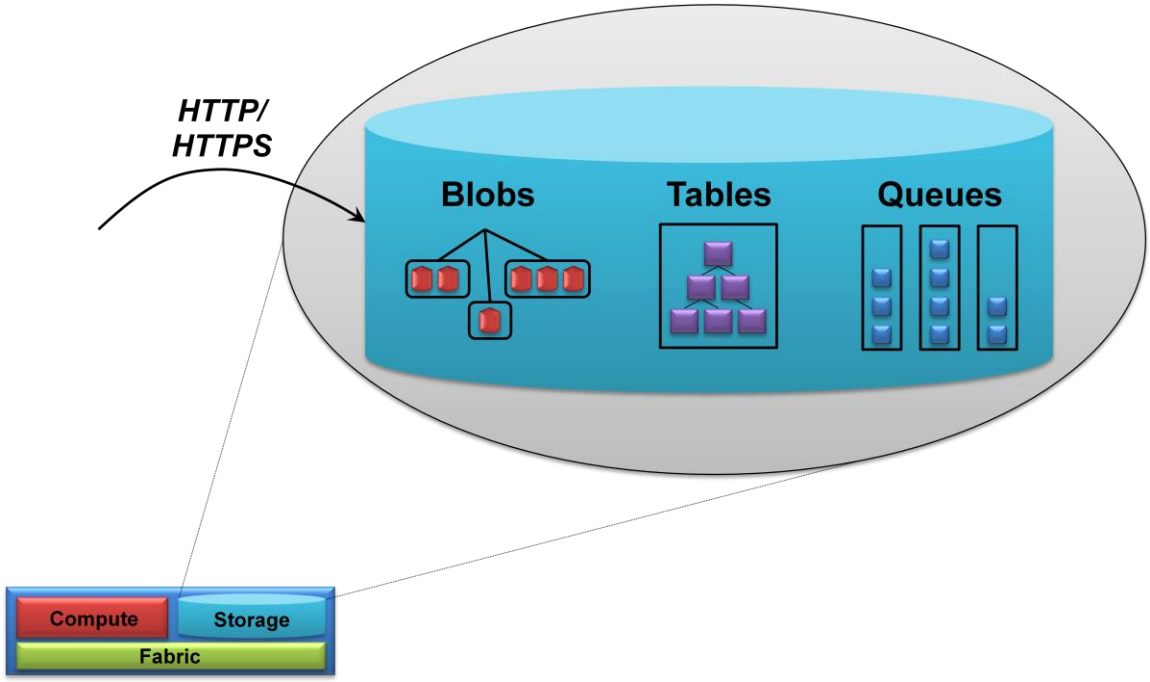

Figure 3: The Windows Azure Storage service can be accessed by Windows Azure applications or by applications running elsewhere.

The three kinds of Windows Azure storage are:

- Blobs: allow storing large binary objects, such as videos and images.
- Tables: provide highly scalable entity-based storage (not relational tables).
- Queues: allow sending and receiving messages, such as between an application's Web role instances and Worker role instances.

It's important to point out that all three of these can also be accessed by applications that aren't running on the Windows Azure Compute service. For example, an on-premises or hosted application might choose to store large video files as Windows Azure blobs.

The Windows Azure platform also includes SQL Azure Database (previously known as SQL Data Services). SQL Azure Database offers standard relational storage based on SQL Server, complete with stored procedures and more. While a single database in SQL Azure Database can't hold as much information as a single Windows Azure Storage table, these databases offer a familiar storage model accessible through ADO.NET and other widely used data access mechanisms.

For a more detailed look at Windows Azure, see Introducing Windows Azure. For a survey of the entire Windows Azure platform (formerly known as the Azure Services Platform), see Introducing the Azure Services Platform: An Early Look at Windows Azure, .NET Services, SQL Services, and Live Services.

BUSINESS MODEL

A primary attraction of a cloud platform, and one of its biggest differences from traditional hosting, is consumption-based pricing. With traditional hosting or an in-house data center, the owner of an application typically pays for a specific set of machines for a fixed amount of time. For applications with variable load, this fixed set must be large enough to handle the highest peak, which means that capacity goes unused at non-peak times.

Cloud platforms aren't like this. With Windows Azure, for example, the owner of an application pays for the compute and storage resources he uses. When the application's load is light, he might request, say, three Web role instances and two Worker role instances. When the load is heavy, he might up his request to six Web role instances and four Worker role instances. In both cases, the application's owner pays only for the resources his application is using.

The Windows Azure pricing scheme is relatively simple, with three main variables: compute time, data storage and access, and bandwidth transferred in and out. For compute time, an application's owner is charged a fixed amount for each hour a VM is running. For data, there's a fixed charge per gigabyte per month, along with a charge for operations against stored data, such as GETs and PUTs. For bandwidth, an application's owner pays a fixed amount per gigabyte of data transferred in or out of a Windows Azure datacenter. (There are no bandwidth fees for accessing data within a datacenter, however.) The charges are:

- Compute = $0.12/hour for each running VM
- Storage = $0.15/GB stored per month, $0.01 / 10,000 operations against stored data
- Bandwidth = $0.10/GB transferred in, $0.15/GB transferred out

Windows Azure applications can also use SQL Azure Database for storage. Once again, the charging is usage-based, with two main variables: how much data is stored (along with the number of query hours allowed, i.e., how much CPU time can be used by SQL queries on this data), and bandwidth transferred in and out. The charges are:

- Web Edition: $9.99/month for a relational database with up to 1 GB of storage and a maximum of 10 query hours
- Business Edition: $99.99/month for a relational database with up to 10 GB of storage and a maximum of 100 query hours
- Bandwidth = $0.10/GB transferred in, $0.15/GB transferred out

Although it's not available at the platform's initial launch, Microsoft says that it will eventually offer commitment-based pricing discounts. Customers willing to commit to specific minimum usage levels will see their charges reduced. Commitments help Microsoft plan more efficiently, and so the discounts reflect these lower costs.

As with its other offerings, Microsoft also has a partner program for Windows Azure. One part of this is an Azure QuickStart, which provides a central place for ISVs and other partners to begin exploring the Windows Azure platform and to connect with Microsoft and other customers using this platform. The

company also has a Development Accelerator offer, which gives ISVs willing to make a six month commitment pricing discounts for developer use of the Windows Azure platform. Forums, phone support, and other traditional services are available as well, reflecting Microsoft's understanding of the fundamental role that ISVs play in making Windows Azure successful.

USING WINDOWS AZURE: SOME OPTIONS FOR ISVS

Working out whether a cloud platform can improve your business requires thinking through how you might use one. Accordingly, this section looks at some of the primary ways in which ISVs can use Windows Azure.

USING CLOUD STORAGE FROM YOUR ON-PREMISES APPLICATION

Perhaps the simplest way an ISV application might use Windows Azure is to store data. As mentioned earlier, Windows Azure Storage can be accessed from on-premises applications as well as from Windows Azure applications. For example, an application that currently does back-ups to an on-premises storage system might instead choose to use Windows Azure Storage blobs. This can improve backup reliability, since like everything else in Windows Azure Storage, blobs are replicated at least three times. It might also lower costs, given the economies of scale provided by Microsoft's very large data centers. Or think of an application that provides large amounts of data to its users: videos, audio files, or something else. Rather than storing this data locally, the application might again choose to use Windows Azure Storage blobs for higher availability, lower costs, or both.

And Windows Azure Storage isn't the only option. Using SQL Azure Database, an application can also store relational data in the cloud. For example, an application that needs to share a set of relational tables across multiple instances running in different locations might benefit from having that data accessible in one place. There are some limitations—a single database in SQL Azure Database can't exceed ten gigabytes, for example—so this service isn't exactly like running SQL Server in the cloud. Still, there are some potential advantages. Much of the work required to administer a database goes away with SQL Azure Database, since Microsoft does it for you. The availability of your data also goes up, since like Windows Azure Storage, SQL Azure Database stores three copies of all data to protect against hardware failures. The cost of this storage might even be lower than on-premises storage, especially for smaller organizations.

Using Microsoft's cloud platform to store your data requires trusting that platform. The best way to build trust is to start small, then if warranted, expand from that base. An ISV that's considering using Windows Azure or SQL Azure Database might well find that using its storage services from an on-premises application lets them gain experience with this new approach before making a bigger commitment. It can be a good way to dip your toe into the water of cloud computing.

COMBINING CLOUD COMPUTING WITH YOUR ON-PREMISES APPLICATION

If putting an application's data in the cloud can make sense, so can using the cloud to run some of your app's code. For example, think of an ISV with an application that could benefit from creating an on-line marketplace among all of the companies that use this solution. Building this functionality on a cloud platform such as Windows Azure should be significantly faster and cheaper than building it entirely from scratch. Or suppose an on-premises application could sometimes benefit from more processor cores to

run CPU-intensive loads. This application could create a number of Windows Azure Worker role instances to do this work, then shut them down when they're no longer needed. As always, the customer would be charged only for the resources they use, i.e., the hours these Worker role VMs were actually running.

There are plenty of situations in which moving an existing application entirely into the cloud isn't practical. Porting millions of lines of code to a cloud platform might be too risky, too expensive, or just not worthwhile. In cases like this, adding new functionality that runs in the cloud can make more sense. Because Windows Azure provides a pre-built platform for running cloud applications, it can make creating this kind of code easier.

CREATING A SAAS VERSION OF YOUR APPLICATION

Being an ISV has long meant installing software directly on your customers' machines. With SaaS applications, this is no longer true. Unlike conventional packaged software, a SaaS application runs in an Internet-accessible data center, and it's typically delivered to customers via the Web.

SaaS applications can offer real benefits for customers, including the following:

- Less risk: Unlike conventional packaged software, SaaS applications needn't require a large up-front investment. Instead, customers can typically try the application for no charge before purchasing it, letting them have more confidence that the application will provide business value.
- A more attractive price structure: SaaS applications commonly provide usage-based pricing, such as a charge per user per month. This lets customers start small, then add users as needed. It also allows replacing a capital expense—buying software—with an operating expense, which can be a draw for some organizations.
- Faster and cheaper deployment: Rather than installing software on local machines, the users of a SaaS application typically access the app via an ordinary Web browser.
- Easier upgrades: Instead of upgrading its own copies of a purchased software package, the customer of a SaaS application can rely on the SaaS provider to upgrade the application centrally.

The advantages of SaaS are certainly appealing for some customers and some kinds of applications. This approach also has drawbacks, however, and so it's not right for every application. The biggest challenges that SaaS customers face include:

- Trust: Can a customer really trust the provider of a SaaS application? Will the application always be available? Can the application provider be trusted with sensitive corporate data? Trust in the provider is the single biggest barrier that most customers face with SaaS applications. It's worth pointing out that this trust must also extend to the cloud platform a SaaS application is built on.
- Regulatory and compliance issues: Many businesses are required by governments to conform to a variety of standards, such as Sarbanes-Oxley, Basel II, and others. If a SaaS application (and the cloud platform it's built on) can't meet these obligations, the business can't use it.
- Customization: Traditional on-premises software packages can usually be customized in various ways. SaaS applications, by contrast, are typically *multi-tenant*, which means that a single copy of the

software is shared by all users. While customization is still possible, it's often more limited than with traditional packaged software.

- Integration with on-premises applications: A SaaS application must provide some way to integrate its code and data with on-premises application. This includes identity integration, since customers typically want single sign-on.

- Management: Most on-premises tools for managing and monitoring applications don't work well with SaaS applications today. While this will likely change, the situation today is off-putting for some customers.

Just as SaaS applications have pros and cons for organizations that buy software, they also offer advantages and disadvantages to companies that sell software. The benefits of SaaS applications for ISVs include:

- The potential for more sales: Since customers face less risk and a smaller up-front financial commitment, making initial sales can be faster and cheaper. SaaS applications can also be attractive to new categories of customers, such as smaller organizations, because they require less in-house IT expertise.

- Easier customer upgrades: Rather than convincing each customer to replace an on-premises package, an ISV can upgrade all users of its SaaS application at once. This can significantly reduce an ISV's support costs, since there's no longer a requirement to support many old versions of an app.

Nothing is free, however, and so the move to a SaaS world also presents some drawbacks for ISVs. Some of the most visible are the following:

- A different sales and revenue model: Money trickles in rather than coming in up-front license fees. At least initially, this may mean lower margins.

- Less customer lock-in: Without the sunk costs of a purchased and perhaps extensively customized on-premises application, customers might find it easier to switch to a competitor.

- Less services revenue from customization: ISVs that derive a significant share of their income from customization services might find this revenue reduced for SaaS applications.

- Technical challenges: Creating a SaaS application requires a different technical skill set from the one most ISVs already have. Rather than creating familiar on-premises software, your development organization must now build a highly scalable and highly reliable shared application. Also, because Windows Azure doesn't always behave identically to the Windows Server environment, some changes to existing code might be required. For example, while Windows Azure load balances requests across an application's Web role instances, it doesn't support session affinity. This allows much better scalability, but it might not be the way your application is currently designed.

The truth is evident: SaaS applications have a significant role to play, but they won't obliterate on-premises software. Every ISV that provides a packaged application must look at its offerings and make a decision: What makes sense to provide as SaaS? Especially if your competitors are going down this path, creating a SaaS version of a current on-premises application might be the right option.

As mentioned earlier, a primary goal of Windows Azure is to support SaaS applications. Many aspects of its design reflect this goal. For example, a SaaS application must be more scalable than an on-premises application because it will support multiple customers simultaneously. The Web role/Worker role division in Windows Azure is intended to help create massively scalable applications by dividing work between a Web front end and Worker back end. Similarly, Windows Azure Storage tables are expressly designed to hold very large amounts of data, much more than can be managed by a single relational database management system.

A SaaS application must also be more reliable than an on-premises application, since a failure affects all customers, not just one. By monitoring every running instance of an application, the Windows Azure Fabric can help achieve this. If an instance or a VM or even an entire machine fails, the Fabric Controller will restart another one to take its place. The Fabric Controller also allows upgrading a running application without shutting it down, an important service for a SaaS app that must be continuously available.

One more important attribute for an effective SaaS application is elasticity, i.e., the ability to handle peaks in demand. With a conventional application, a data center must be able to support the maximum load this application will ever see. This is certainly possible, but it's expensive and wasteful—most of the data center's capacity will probably lie unused most of the time. With Windows Azure, however, addressing this problem is simpler. Recall that the owner of an application can change the number of running instances on the fly, relying on the Fabric Controller to create or shut down VMs as needed. Since Windows Azure customers are charged only for the resources they use, this lets them pay for a large number of computing resources only when those resources are really needed. When the load diminishes, the application can shrink back to its normal size.

SaaS applications really do have different requirements from traditional on-premises applications. These differences are a big part of why Windows Azure is designed the way it is. An ISV that creates a SaaS application is free to build its own platform, and originally some did—there was no alternative. Yet with the rise of cloud platforms, ISVs creating SaaS applications can now focus on their business logic instead of infrastructure. Going forward, it's likely that a majority of SaaS applications will be created on a cloud platform such as Windows Azure.

PROVIDING SUPPORTING SERVICES FOR CLOUD PLATFORMS

Many ISVs today provide infrastructure and management services for on-premises environments. Many others provide software that makes life easier for developers. While some of these solutions aren't relevant in the cloud, others are. And new opportunities exist as well, places where ISVs can make money by adding value to Windows Azure.

For example, Windows Azure provides access to performance data about running applications. A tool that aggregated this data, then presented it through an effective user interface could help Windows Azure customers manage their cloud applications. Windows Azure also provides APIs that let an application change the number of running Web role instances and Worker role instances on the fly rather than relying on a person to do this. The platform does not, however, provide software that monitors the application's load, then uses these APIs to adjust the number of running instances accordingly. An ISV might fill this hole, providing code that lets developers easily add this behavior to their applications.

The advent of cloud platforms is a big change for ISVs. With any big change, it often makes sense to do things a piece at a time, such as using Windows Azure's cloud storage or offloading a part of your app to the cloud. If this works well—and makes financial sense—you can then move on to bigger steps when they're justified, such as creating a full SaaS version of your application.

COMPARING WINDOWS AZURE WITH ALTERNATIVES

Windows Azure doesn't exist in a vacuum—there are other approaches. This section compares Windows Azure with two of its most obvious alternatives: traditional hosting and cloud platforms that offer VMs on demand.

TRADITIONAL HOSTING

The first stop for most people looking for an outsourced place to run their applications is a hosting provider. In traditional hosting, a customer requests a fixed set of resources and commits to pay for those resources for a defined period of time. For example, an ISV wishing to run a SaaS application might contract with a hoster to provider six Windows servers for a year, paying a pre-defined amount for this service.

Hosting has plenty of advantages. Using a hoster is frequently cheaper than running an in-house data center, especially for smaller organizations. It also lets the customer avoid the complexity of running its own data center while still having total control over the machines it's using. While the advent of cloud platforms will likely cut into the traditional hosting business, this model isn't going away—it still makes sense in plenty of situations.

Yet at least for some applications, a cloud platform such as Windows Azure is a better choice. The advantages include:

- The ability to quickly increase the number of servers in use: While a hoster might take days to make a new machine available, a Windows Azure application can get a new VM up and running in minutes.
- The ability to quickly decrease the number of servers in use: Hosters commonly require a commitment to a fixed set of servers that are provisioned just for you. With Windows Azure, an application can reduce the number of VMs it's using—and thus the cost of running this application—by decreasing the number of Web role and/or Worker role instances. There's no up-front commitment to a minimum number of servers.
- The ability to provide services explicitly designed for highly scalable, highly available applications: Hosters generally provide standard Windows systems, leaving it up to their customers to do whatever else is necessary to run their applications successfully. As described earlier, a cloud platform such as Windows Azure can be explicitly designed to support applications with very high scalability and availability requirements.
- Less administrative overhead: Unlike Windows Azure, hosters commonly give customers full administrative access to their machines. The trade-off is that more administrative work is required, everything from patching operating systems to managing database management systems. With Windows Azure (and SQL Azure Database), most of this work is done for you, saving time and money.

VMS ON DEMAND

A number of vendors, including Amazon, Mosso, GoGrid, and others, offer virtual machines on demand. Unlike traditional hosters, these vendors typically provide usage-based charging with no required commitment and rapidly available VMs. In other words, they provide cloud platforms.

Windows Azure is also a cloud platform, but even though it uses (and charges via) VMs, it differs in important ways from platforms that offer VMs on demand. With a purely VM-based platform, the situation is in some ways much like hosting: You have complete control, including administrative access to your VMs, but you also bear full responsibility for configuring and managing those VMs and the software they contain. With Windows Azure, you supply only a Windows application, along with instructions about how many instances to run. The platform itself takes care of everything else, including updating system software when required.

Another important difference is in how relational data is handled. With typical VM-based platforms, you can run a relational database in a VM, just as you'd run the same database on premises or at a hoster. This certainly works, but it requires installing, maintaining, and administering this database yourself. Ensuring reliability can also be challenging, since typical shared-disk clusters often aren't possible. In the Azure world, an application can instead use SQL Azure Database. As described earlier, this technology provides a Microsoft-managed relational store that writes all data multiple times for reliability. Once again, you lose the ability to have total control but gain simplicity and built-in reliability.

CONCLUSIONS

Like all new platforms, Windows Azure will succeed only if ISVs choose to build applications on it. Microsoft clearly understands this, and so making their new cloud platform attractive to this audience is a priority. The core attractions are these:

- Because Windows Azure lets ISVs run applications and store data in a very large data center while paying only for the resources used, it can provide appealing economics.
- By providing a ready-made platform designed to support scalable and reliable cloud applications, Windows Azure reduces the time and money required to create and run SaaS applications and other cloud-based code.

Cloud computing looks like the next great wave in our industry. Just as ISVs have had to adapt to the changes brought by PCs, mobile devices, and other new platforms, they now need to decide how to exploit cloud platforms. And just as Windows played a significant part in those earlier shifts, Windows Azure is poised to take an important role in this new world. If you're responsible for charting your firm's path, understanding and evaluating the Azure environment makes good sense.

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.