David Chappell

June 2011

WINDOWS HPC SERVER AND WINDOWS AZURE

HIGH-PERFORMANCE COMPUTING IN THE CLOUD

Sponsored by Microsoft Corporation

Copyright © 2011 Chappell & Associates

Contents

High-Performance Computing and the Cloud

The essence of high-performance computing (HPC) is processing power. Whether implemented using traditional supercomputers or more modern compute clusters, HPC requires having lots of available computing resources.

Windows HPC Server 2008 R2 lets organizations provide these resources. By letting those machines be managed as a cluster, then providing tools for deploying and running HPC applications on that cluster, the product makes Windows a foundation for high-performance computing. Many organizations today use Windows HPC Server clusters running in their own data centers to solve a range of hard problems.

But with the rise of cloud computing, the world of HPC is changing. Why not take advantage of the massive data centers now available in the cloud? For example, Microsoft's Windows Azure provides on-demand access to lots of virtual machines (VMs) and acres of cheap storage, letting you pay only for the resources you use.

The potential benefits for HPC are obvious. Rather than relying solely on your own on-premises computing resources, using the cloud gives you access to more compute power when you need it. For example, suppose your on-premises systems are sufficient for most—but not all—of your organization's workloads. Or suppose one of your HPC applications sometimes needs more processing power. Rather than buy more machines, you can instead rely on a cloud data center to provide extra computing resources on demand. Depending on the kinds of applications your organization runs, it might even be feasible to eventually move more and more of your HPC work into the cloud. This allows tilting HPC costs away from capital expense and toward operating expense, something that's attractive to many organizations.

Yet relying solely on the cloud for HPC probably won't work for most organizations. The challenges include the following:

- Like other applications, HPC applications (often called *jobs*) sometimes work on sensitive data. Legal, regulatory, or other limitations might make it impossible to store or process that data in the cloud.
- A significant number of HPC jobs rely on applications provided by independent software vendors (ISVs). Because many of these ISVs have yet to make their applications available in the cloud, a significant number of HPC jobs can't use cloud platforms today.
- Jobs that need lots of computing power often rely on large amounts of data. Moving all of that data to a cloud data center can be problematic.

While the rise of cloud computing will surely have a big impact on the HPC world, the reality is that on-premises HPC computing isn't going away. Instead, providing a way to combine the two approaches—on-premises and cloud—makes sense. This is exactly what Microsoft has done with Windows HPC Server and Windows Azure, supporting all three possible combinations: applications that run entirely on-premises, applications that run entirely in the cloud, and applications that are spread across both.

This paper describes how Windows HPC Server R2 Service Pack (SP) 2 and Windows Azure can work together. After a brief tutorial on each one, it walks through the options for combining them. The goal is to provide an architectural overview of what this technology is and why HPC customers should care.

Technology Basics

Understanding how Windows HPC Server and Windows Azure can be combined for high-performance computing requires understanding the basics of both technologies. This section walks through the fundamentals of each one, starting with Windows Azure.

Windows Azure

Windows Azure is a public cloud platform, providing Internet-accessible computers running in Microsoft data centers. Its customers run applications and store data on these machines, paying Microsoft directly for the computing and storage resources they use. Figure 1 shows the main Windows Azure components.

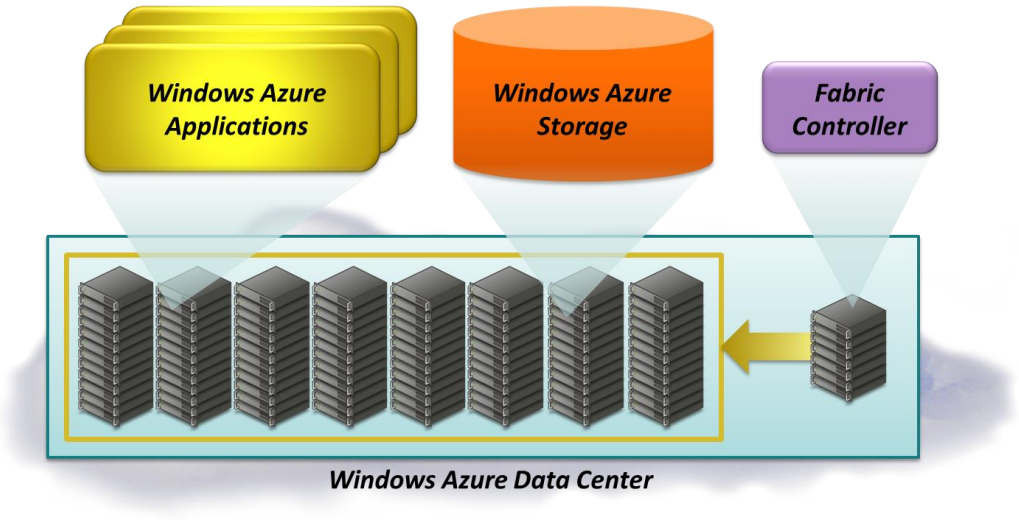

Figure 1: A Windows Azure data center runs applications and stores data, all managed by the Windows Azure fabric controller.

As the figure shows, every Windows Azure application runs on computers in a Windows Azure data center. Those applications (or even applications running on-premises) can use Windows Azure storage, which provides two main options:

- *Blobs*, offering unstructured storage for binary data.
- *Tables*, providing structured storage for large data sets. (Don't be confused—despite their name, these tables don't provide relational storage.)

All of this is managed by the *fabric controller*, an application running on its own dedicated set of machines. Among other things, the fabric controller handles any updates required to the computers in this Windows Azure data center, such as operating system patches.

Whatever its purpose, a Windows Azure application is always implemented as one or more *roles*. Three role types are supported today:

- *Web roles*, which are typically used for code that accepts HTTP requests via Internet Information Services (IIS).
- *Worker roles*, providing a more general option that can be used for various kinds of processing.
- VM roles, which allow a Windows Azure customer to upload a VM image as a Windows Server 2008 R2 VHD, then have Windows Azure execute that image.

Whatever roles an application uses, Windows Azure requires it to run at least two instances of each one. These instances are typically interchangeable—the failure of any one won't take down the application—and each instance runs in its own VM. Depending on the load it needs to handle, an application might run two, ten, or fifty instances of a role. Figure 2 illustrates this idea.

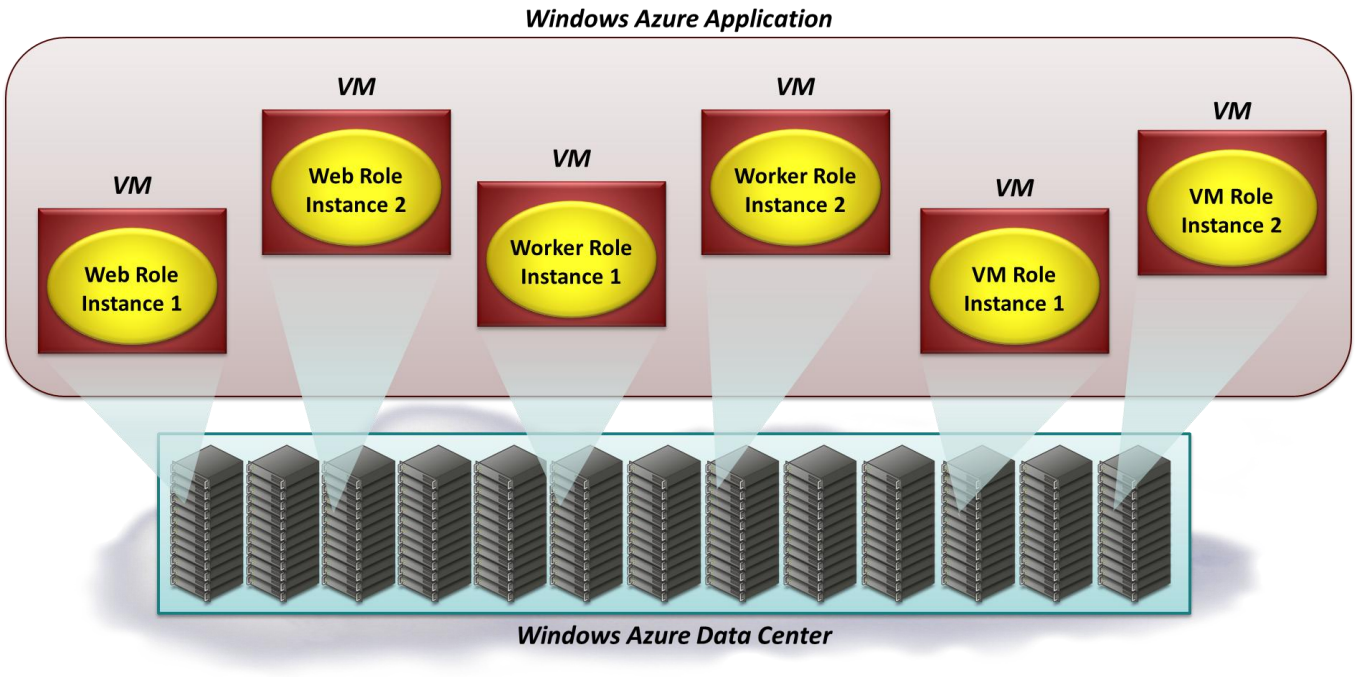

Figure 2: A Windows Azure application consists of some combination of Web role instances, Worker role instances, and VM role instances, with each instance running in its own virtual machine.

In this example, the Windows Azure application is implemented using one Web role, one Worker role, and one VM role. It's currently running two instances of each role, but changing this is straightforward. A developer or administrator can tell Windows Azure to increase or decrease the number of instances, for example, or the application can do this itself.

As the figure shows, the Windows Azure fabric controller assigns each instance to a different computer. This helps make applications more fault-tolerant, since a single hardware failure won't take down the entire application. The fabric controller also monitors each running application. If an instance fails, the fabric controller starts a new instance of the same role within a specified time defined by the platform's service level agreement (SLA). And because each instance has one or more dedicated processer cores, an application's performance is predictable—it doesn't share those cores with other applications.

Windows Azure is based on Windows, and so the applications it runs can be created using the .NET Framework. Those applications can also be built using raw C++, however, or other technologies such as PHP and Java. And while Visual Studio is today's most commonly used tool for creating Windows Azure applications, it's not the only choice—developers are free to use whatever tools they like.

However it's built, a Windows Azure application can access data stored in various places. One common choice is to use the blobs and tables provided by Windows Azure storage. Another option is to use SQL Azure, Microsoft's cloud-based service for relational data. An application is also free to access on-premises data within an organization—there's no requirement that a cloud application use only data stored in the cloud.

Making good decisions about using Windows Azure requires a basic understanding how it's priced. Here's a summary of the main things to know:

- For compute, customers are charged per instance (i.e., per VM) per hour. The price ranges from $0.05 to $0.96 an hour based on the instance's size, with options ranging from one to eight cores. Note that this charge isn't usage-based; an instance that sits unused for an hour will incur the same charge as an instance that spends an hour doing intensive calculations.
- For storage, data kept in Windows Azure blobs and tables costs $0.15 per gigabyte per month. Customers also pay $0.01 per 10,000 operations on this data, such as reads and writes.
- For bandwidth, customers pay $0.10 per gigabyte in and $0.15 per gigabyte out. These costs are incurred only for data moved into and out of a Windows Azure data center—there's no bandwidth charge for Windows Azure applications that access blobs or tables (or SQL Azure data) in the same data center.

Cloud platforms offer a new approach to deploying, using, and paying for computing resources. With Windows Azure, Microsoft provides a way for Windows-oriented developers to embrace this new style of computing.

Windows HPC Server

The essence of high-performance computing is dividing an application into separate components, then running those components simultaneously on multiple machines. With Windows HPC Server, this means spreading the application's logic across multiple *compute nodes*, each a physical computer or VM running Windows. The application might run the same logic on every compute node, with each instance processing different data, or it might run different logic on different nodes.

To assign the logic of a Windows HPC application to compute nodes, Windows HPC Server provides a *job scheduler*. This scheduler runs on its own dedicated machine called a *head node*. Figure 3 shows how this looks.

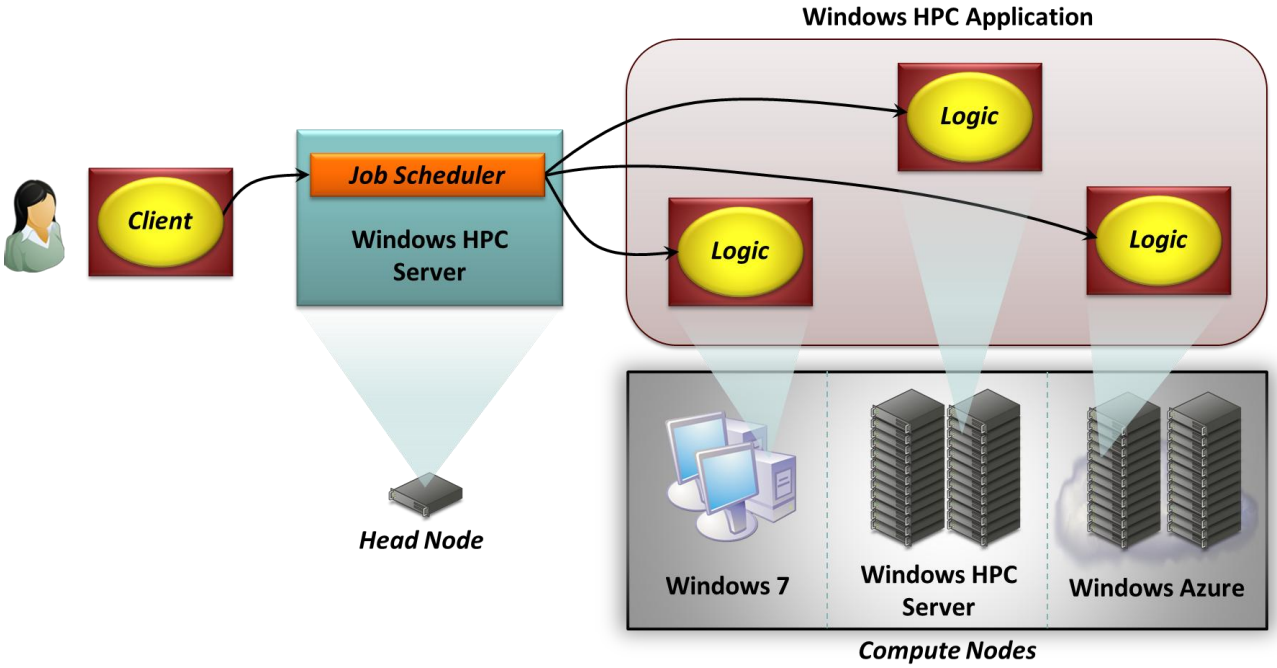

Figure 3: A job scheduler running on a head node distributes the logic of a Windows HPC application across a group of compute nodes.

As the figure shows, a user submits a Windows HPC job from a workstation to the head node. The job scheduler parcels the job's logic across the compute nodes that this application has access to.

Traditionally, a computing cluster contained only on-premises servers. With Windows HPC Server today, however, a cluster can include any combination of three options:

- Desktop workstations running Windows 7.
- On-premises servers running either Windows HPC Server 2008 R2 or its predecessor, Windows HPC Server 2008. These can be physical or virtual servers.
- Windows Azure instances, including both Worker roles and VM roles. (Web roles aren't typically used with Windows HPC Server.)

Assigning work to compute nodes isn't a simple task, and so the job scheduler is a sophisticated piece of software. Among other things, it runs applications based on their priority, allowing new high-priority jobs to jump to the head of the line. It also lets the user submitting a job specify what kind of resources the job needs, then place the job appropriately. The scheduler is also smart enough to avoid letting a few big, high-priority jobs starve all of the others by hogging all of a cluster's resources.

Using this infrastructure, Windows HPC Server can run several kinds of applications. The options today include the following:

- Applications where components running on different computers in the cluster interact with one another while the application is running. Because this interaction typically happens via a technology called the Message Passing Interface (MPI), these are commonly known as *MPI applications*. Many scientific and technical problems can be addressed with MPI applications, including those that require simulating physical processes such as car crashes and nuclear reactions.

- Applications where components running on different computers in the cluster *do not* interact with one another while the application is running. Because applications like these are so easy to run in parallel, they're referred to as *embarrassingly parallel applications*. This kind of application is especially common in financial services, addressing problems such as evaluating the risk of an investment portfolio.

- *Excel applications* that offload a workbook's calculations to a cluster, speeding up work that would take much longer to complete on a single machine. Excel workbooks are used for a wide range of problems today, many of which perform time-consuming calculations that can benefit from running on a cluster.

Much of what Windows HPC Server offers is focused on creating and running applications. Yet managing a cluster and the applications that run on it is also a non-trivial task. For example, an administrator must create the cluster, then determine things such as which compute nodes are available for each job. An application might have a specific set assigned to it, or it might be able to draw compute nodes from a generally available pool. To make life easier for cluster administrators, Windows HPC Server includes *HPC Cluster Manager*. Figure 4 shows an example of how this tool looks.

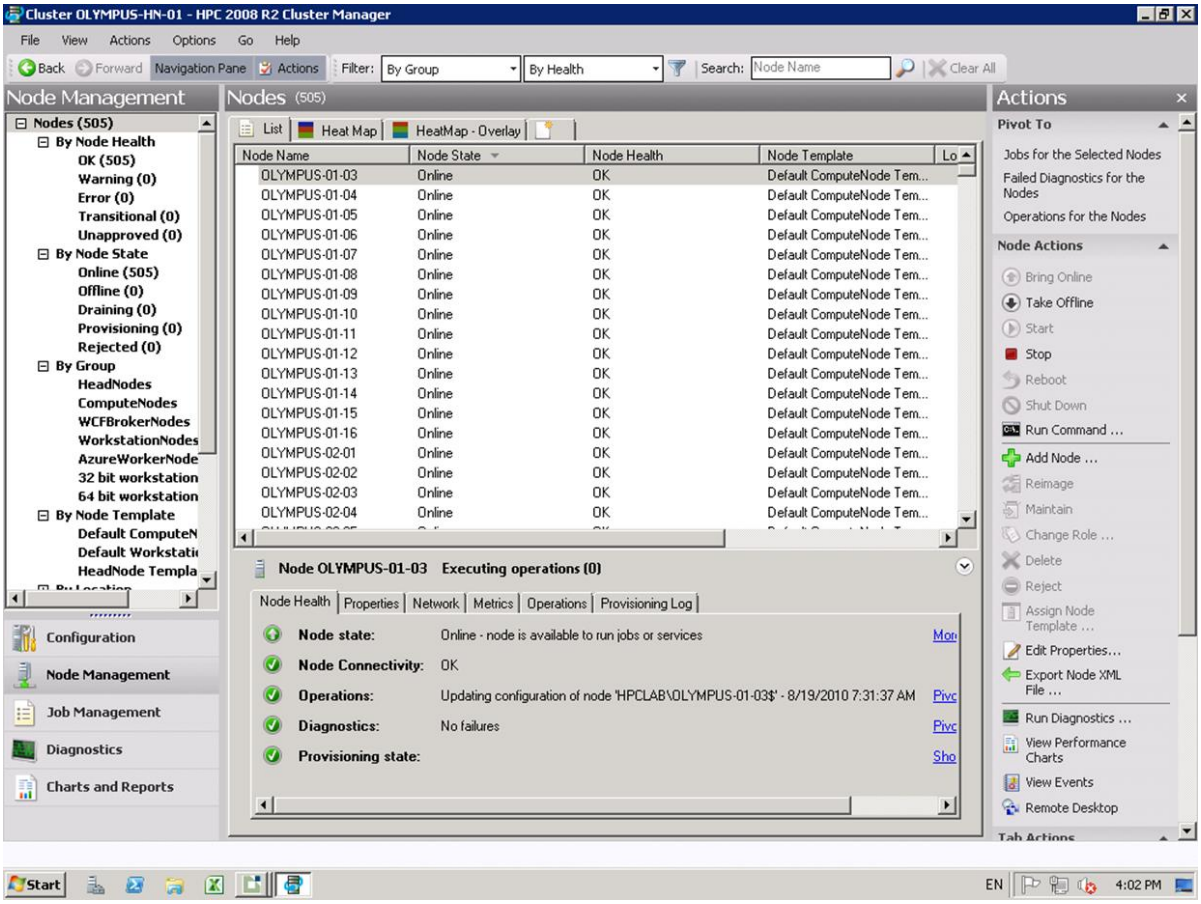

Figure 4: HPC Cluster Manager lets a cluster administrator add nodes to a cluster, monitor and manage jobs running on the cluster, and much more.

As the figure suggests, an administrator uses this tool to accomplish a variety of tasks: managing a cluster's nodes, managing jobs running on that cluster, running diagnostic tests on the cluster, creating charts and reports about the cluster, and more. HPC Cluster Manager also provides graphical views, including a heat map showing the utilization of each node in a cluster. And because a cluster can contain any combination of desktop workstations, on-premises servers, and Windows Azure instances, HPC Cluster Manager lets a cluster administrator work with all three in a consistent way.

High-performance computing isn't an especially simple area. Building and running HPC applications requires specialized software, as does managing the clusters on which they run. Windows HPC Server aims at providing a cohesive solution for all of these problems.

Using Windows HPC Server with Windows Azure

HPC depends on processing power. Windows Azure provides a vast amount of processing power on demand. It's obvious: The two are a natural combination.

But for all of the reasons described earlier, the rise of the cloud doesn't signal the end of on-premises HPC computing. Instead, the on-premises and cloud worlds can work together, as the marriage of Windows HPC Server and Windows Azure demonstrates. This combination supports two approaches today:

- Keeping the head node and some compute nodes on premises, with extra compute nodes in the cloud.
- Keeping only the head node on premises, with all compute nodes in the cloud.

This section looks at both options.

Using On-Premises Computing and Cloud Computing Together

From an architectural perspective, running a Windows HPC application that uses both on-premises compute nodes and Windows Azure instances is straightforward. Figure 5 shows how this looks.

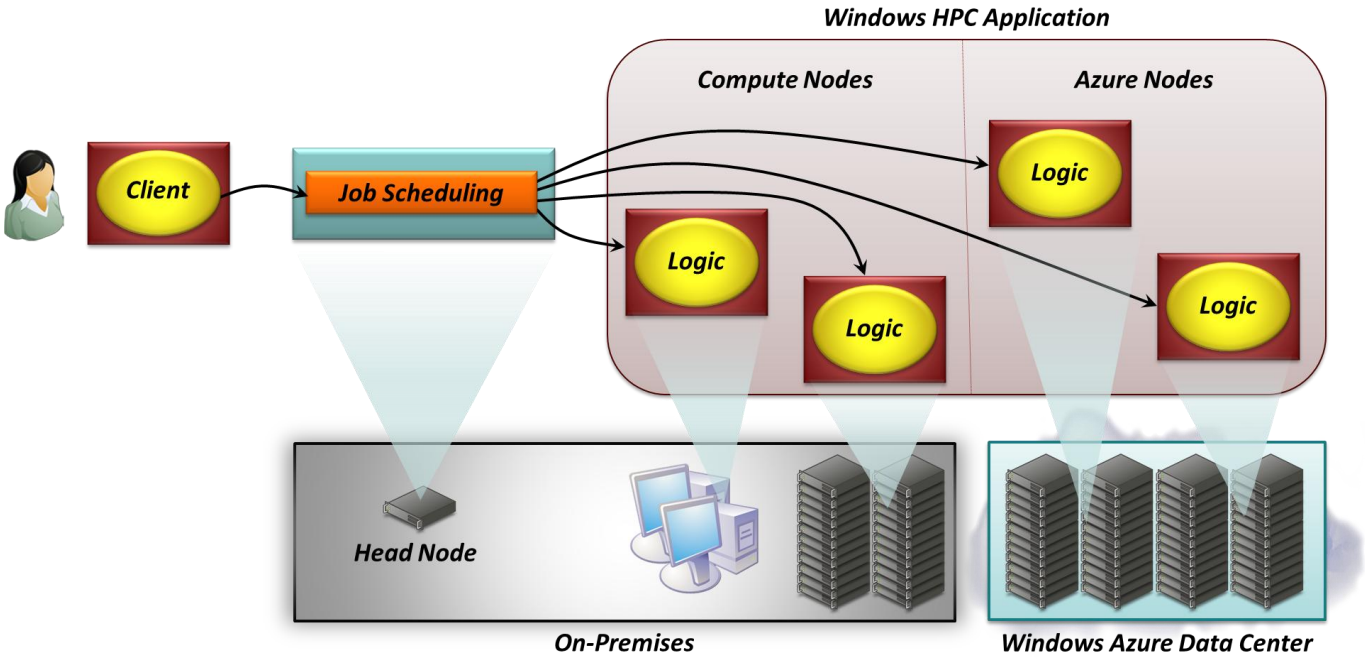

Figure 5: A Windows HPC application can use both on-premises compute nodes and Azure nodes.

The head node remains on premises, and a user submits the HPC job in the usual way. The scheduler on the head node then parcels out the application's logic across the available compute resources. In this case, those resources include both on-premises compute nodes and Azure nodes, which are Worker role and/or VM role instances running in a Windows Azure data center.

To a user, submitting a Windows HPC job that runs partly on-premises and partly in the cloud looks just as it always does—there's no difference from submitting one that runs entirely on-premises. For the Windows HPC administrator, however, some extra work is required to make this possible. While the admin still uses HPC Cluster Manager and other familiar tools, she must set up a Windows Azure account and configure the Azure nodes. The

administrator also specifies which jobs are allowed to use which compute resources, whether those resources are compute nodes in the on-premises cluster or Azure nodes in the cloud.

Azure nodes aren't created on demand when a job is submitted. Instead, the administrator explicitly allocates these VMs, then shuts them down when they're no longer needed. For example, an administrator might create 25 Azure nodes at 9 am each day, then take them down at 6 pm. This can be done manually through HPC Cluster Manager or programmatically via a (potentially automated) script. Using Azure nodes costs money—each one is a Windows Azure Worker role or VM role instance—and so Windows HPC Server gives an administrator control over when those instances are created and how long they run.

Although it's not shown in the diagram, this application almost certainly reads data in and writes data out. That data might be stored on premises, with the local compute nodes having direct access while the Azure nodes access it across the Internet. The application's data might also be stored in the cloud using, say, blobs in Windows Azure storage, or it might be divided between on-premises storage and cloud storage. Windows HPC Server doesn't mandate where data is stored or how it's accessed—it's up to the creator of each application to make this decision.

What kind of applications would benefit from combining on-premises compute nodes and cloud-based Azure nodes? When would this approach be useful? Here are some examples:

- An application that sometimes needs extra compute power could use Windows Azure to provide this on demand. Rather than investing in more machines on-premises, an organization can use the cloud to get the resources it needs at a lower cost.
- An organization that periodically needs extra HPC compute nodes might use Windows Azure to provide them. Think of a financial services firm, for example, that runs a group of CPU-intensive jobs once a week. Rather than expand its existing on-premises cluster, then watch those new machines sit idle most of the time, this firm could instead allocate Azure nodes only when needed,.
- An organization might be running out of room for its on-premises cluster. Rather than expand by adding more physical machines to a crowded data center, they could instead choose to expand into the cloud by using Azure nodes. As mentioned earlier, this also avoids the capital expense of buying new hardware, replacing it with the operating expense of using resources in the cloud. And because the Windows Azure fabric controller automates routine tasks like patching Windows, the administrative costs of an HPC environment can also be reduced.

Whatever the reason, developers creating HPC applications that run across Windows HPC Server and Windows Azure see a familiar world. They can still use Visual Studio (or other Windows development tools) and rely on the product's standard profiling and debugging support. The big change is that their application can now have access to significantly more computing power.

Using Cloud Computing Exclusively

If putting some compute nodes in Windows Azure makes sense—and it often does—why not put all of them in the cloud? Windows HPC Server also supports this option, as Figure 6 shows.

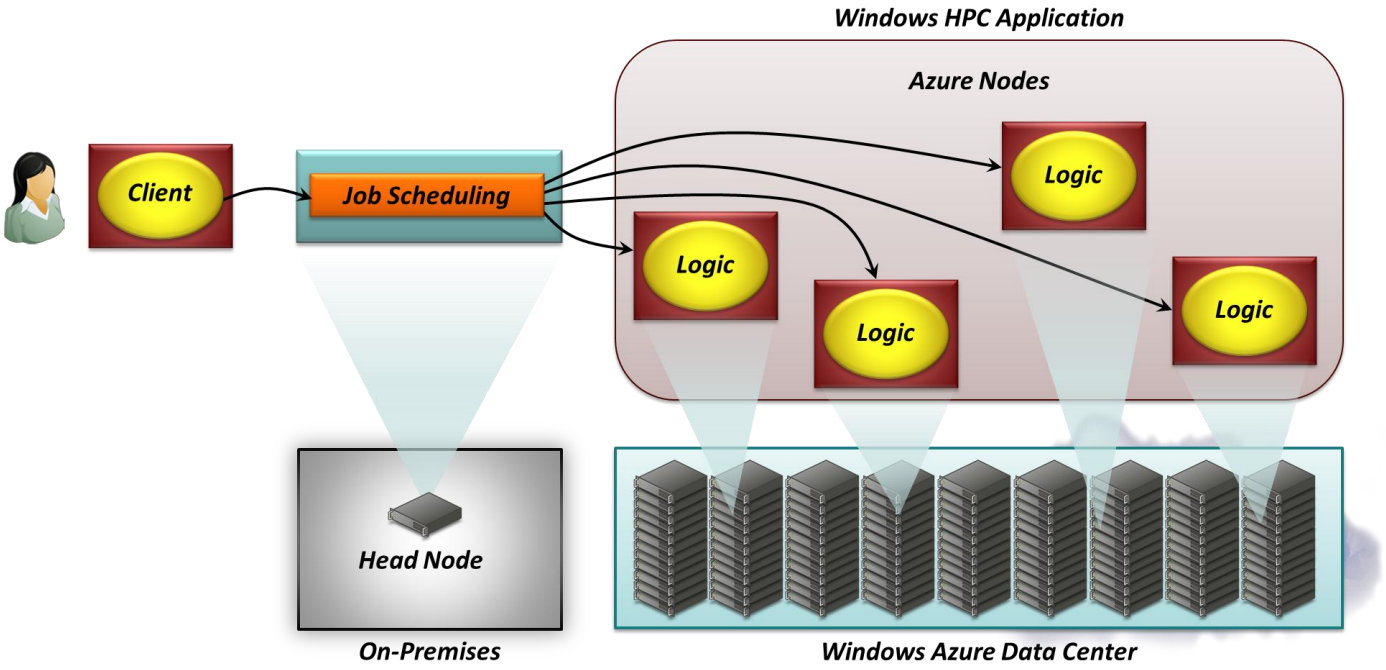

Figure 6: A Windows HPC application can rely entirely on Azure nodes.

In this case, only the head node remains on premises. All of the processing is done in Azure nodes running in a Windows Azure datacenter. Once again, the user submitting a Windows HPC job sees no difference—the scheduler transparently spreads the application's logic across the Azure nodes.

Putting all of an application's computing in the cloud can have some clear benefits. They include these:

- An organization can run HPC jobs without buying and managing its own cluster. Maybe the utilization of an on-premises cluster wouldn't be high enough to justify the expense, or perhaps a firm needs HPC capabilities only once a month. Whatever the situation, having just the head node on premises with all computing done on Azure nodes can make economic sense.
- The cost of entry to HPC can be significantly lower than with an on-premises cluster. Rather than spending a chunk of money up front on hardware, an organization can instead pay only for the HPC resources it needs when it uses them.
- Compared to dividing an HPC application between on-premises resources and Windows Azure, running the application entirely in the cloud can make data access easier. Rather than splitting the data between the two worlds or making one part of the application access data remotely, all of the application's data can be stored in the cloud.
- For some organizations, the SLA provided by Windows Azure might provide more reliability than their own data center. And since the Windows Azure fabric controller automatically handles updates to system software, the instability these can cause is minimized.

Whether you choose to split an HPC application between on-premises resources and the cloud or to run that application entirely in the cloud, the potential benefits of combining HPC and cloud computing are evident. While it's not right for every situation, expect to see an increasing amount of HPC workloads taking advantage of what the cloud has to offer.

Scenarios

To understand how Windows HPC Server and Windows Azure can work together, it's useful to look at each of the three kinds of applications this combination supports today: embarrassingly parallel jobs, Excel jobs, and MPI jobs[1]. What follows takes a brief look at each one.

Embarrassingly Parallel Applications

In an embarrassingly parallel job, the logic running on each compute node can work in isolation—there's no need for the components to interact while the job runs. Windows HPC Server supports two approaches to creating this kind of application: *service-oriented architecture (SOA)* applications and *parametric sweep* applications.

As the name suggests, the compute logic in a SOA application is implemented as a service. With Windows HPC Server, each service is implemented using Windows Communication Foundation (WCF), allowing the service to expose its operations via SOAP or REST or something else. This approach lets users interact with the job while it's running.

In a parametric sweep application, the compute logic is implemented as an ordinary executable. Rather than exposing services that let a user interact with the job while it's running, the submitter instead can pass different initial parameters to the executable on each compute node when the job starts running. This lets each executable apply the same logic to different data. And while parametric sweep applications can't interact with users, they also don't require using WCF. This can make them easier for scientists and other non-professional developers to create.

To see how this kind of solution works with Windows HPC Server and Windows Azure, suppose you need to write a risk analysis application using a Monte Carlo simulation. This application is likely to be embarrassingly parallel—it can easily be broken up into chunks, each of which works independently. Each chunk runs on its own compute node, and each one can read and write data from wherever the developer chooses, such as a shared file system.

With this kind of simulation, running on more nodes typically leads to better results. Suppose that most of the time, running the application using only on-premises compute nodes is sufficient. In some cases, however, greater accuracy might be required, something that using more compute nodes can provide. Rather than investing in new computers that won't be fully used, the organization running this application might instead choose to rely on Windows Azure to do the extra work. Figure 7 shows how this looks.

[1] Windows HPC Server 2008 R2 also supports LINQ to HPC applications for addressing data-intensive problems. In its current beta release, LINQ to HPC applications must run entirely on-premises—they can't use Windows Azure—although Microsoft has announced plans to add cloud support.

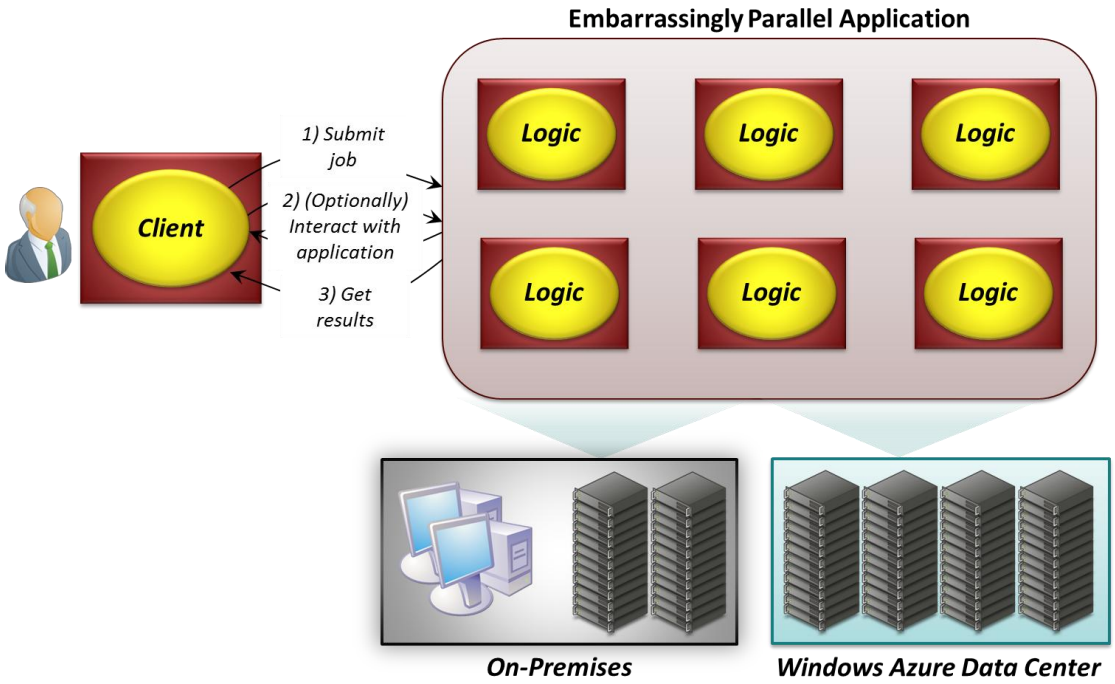

Figure 7: An embarrassingly parallel application might have its work divided between an on-premises data center and Windows Azure.

As the figure shows, the user submits the job as always (step 1). (Although it's not shown here, this relies on the head node as described earlier.) Once the job is executing, the user can interact with it if it's a SOA application (step 2). In any case, the job will eventually return results to the user (step 3).

The Azure nodes used here might be either Worker role instances or VM role instances. If it's a SOA job, its creator probably chose Worker roles, since they're simpler to use and they support WCF. Some existing applications can be challenging to run in Worker roles, however. If this is a parametric sweep job, for instance, it might rely on existing Windows software that can't easily be installed in a Worker role. In this case, the application's creator can instead generate a Windows Server 2008 R2 VHD that contains the application's software, then upload this VHD into Worker roles. Windows HPC Server makes things simpler by providing a graphical tool to do this.

Figure 7 shows an embarrassingly parallel application spread across both on-premises compute nodes and Azure nodes. It's also possible, of course, to run the application entirely in the cloud or entirely on-premises. It's up to the user to decide which option makes the most sense.

Excel Applications

Excel can use a Windows HPC cluster in several different ways. The options are:

- Using Excel as the client for a SOA job. With Visual Studio Tools for Office, a developer can write a custom SOA application that gets executed from an Excel workbook.

- Running Excel user-defined functions (UDFs) on a cluster. UDFs allow creating custom code that can be called from spreadsheet cells just as if they were built-in Excel functions. The code that implements a UDF can run on a Windows HPC cluster.
- Running Excel workbooks on a cluster. An organization can install Excel on multiple compute nodes in a cluster, then have each copy run the same workbook simultaneously with different data.

With any of these options, the cluster can be entirely on-premises, entirely in the cloud, or divided between the two.

For example, suppose a financial analyst creates an Excel workbook that models the behavior of a group of transactions. If the number of transactions is usually small, the analyst might normally run the workbook on his own desktop workstation. In some cases, however, he might need to work with a much larger number of transactions. To do this effectively, he can execute the workbook on Windows Azure instead. Figure 8 shows how this looks.

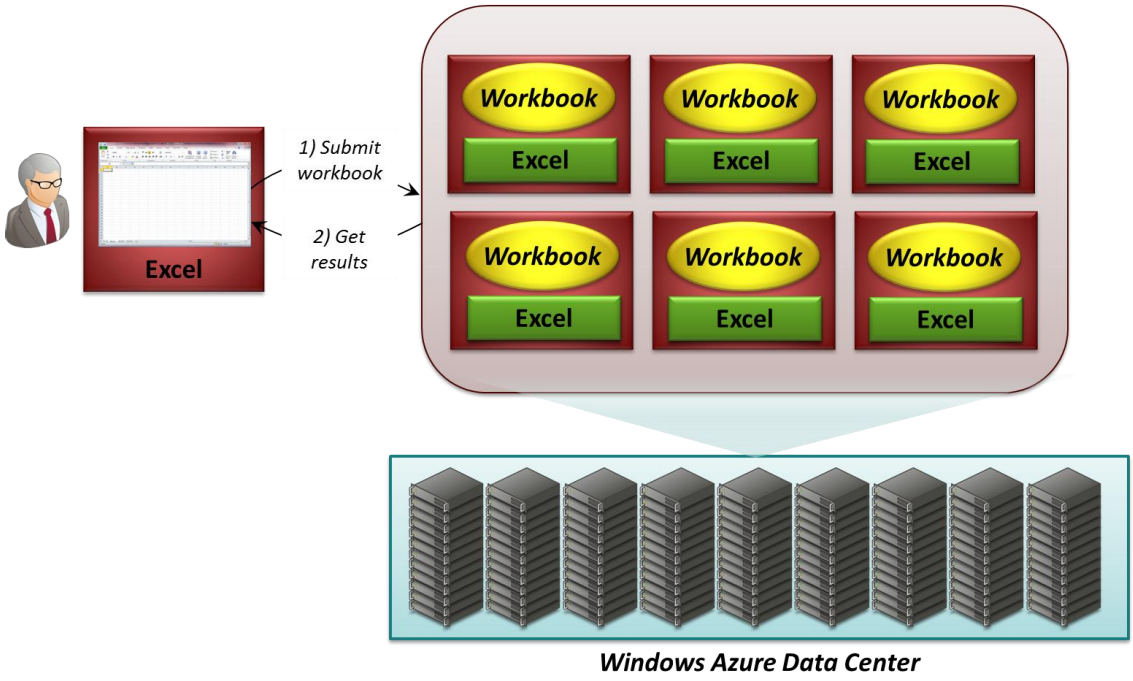

Figure 8: An Excel application can perform workbook calculations on Windows Azure.

In this scenario, the user sends the workbook to Windows Azure as usual (step 1). Because Excel is installed on multiple Azure nodes, the workbook's calculations can run simultaneously on each one, with every copy processing different data. When the job completes, the result is returned to the user (step 2).

Running Excel on Windows Azure in this way requires using VM roles—Excel can't be installed in a Worker role. Before this job is run, someone would need to create a Windows Server 2008 R2 VHD that included Excel, then upload this image to Windows Azure. Most likely, an administrator would handle this, letting the financial analyst remain unaware of these technical details.

MPI Applications

With both embarrassingly parallel applications and Excel applications, each chunk of logic works independently—the chunks don't need to interact while the job runs. With MPI applications, however, the chunks communicate to exchange intermediate results or for other reasons. Yet these jobs can still run on Windows Azure, as Figure 9 shows.

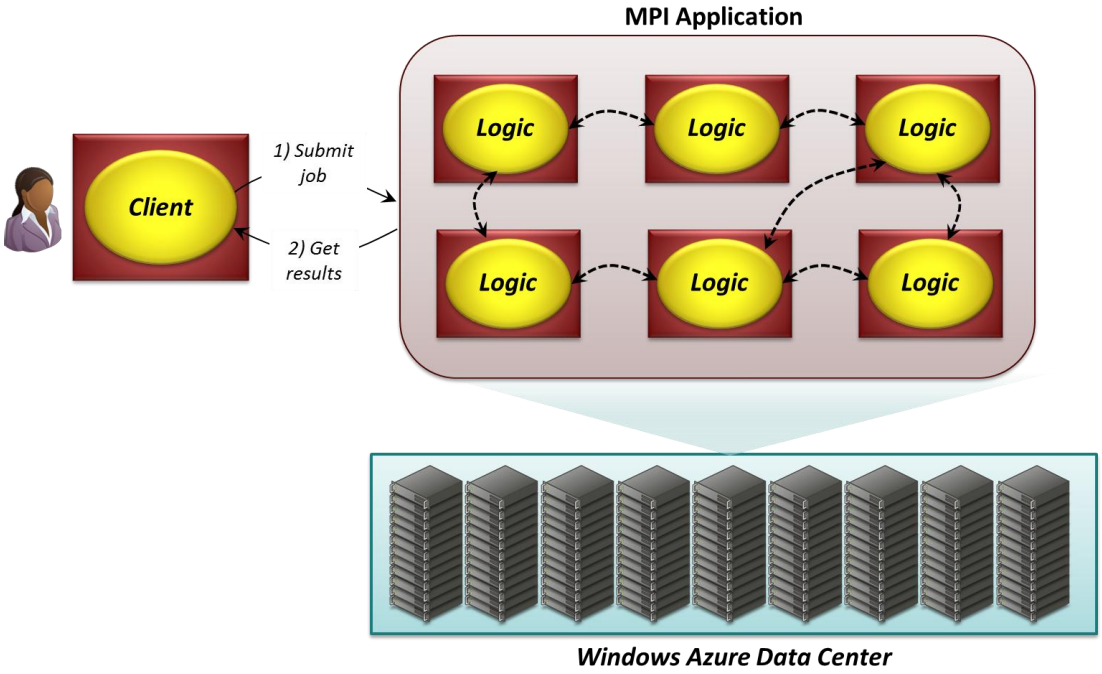

Figure 9: An MPI application can run entirely on Windows Azure.

MPI applications are typically batch jobs. As shown here, the user submits the job (step 1), then gets the job's result returned to her (step 2). As in the earlier cases, an MPI application can run entirely on Windows Azure, as in this example, entirely on-premises, or be divided between the two.

An MPI application can use either Worker roles or VM roles. VM roles require slightly more administration, but they can also make it easier to run some existing software, as described earlier. If an organization wishes to run a purchased MPI application on Windows Azure, for example, using VM roles might make this simpler, minimizing the need for Windows Azure-specific modifications by the application's creator.

It's worth pointing out that because the components in an MPI job must interact while the job runs, on-premises clusters that support this kind of application commonly connect their servers with high-bandwidth networks such as InfiniBand. Windows Azure datacenters don't provide these high-speed connections between machines, and so MPI jobs that depend on very fast communication might not perform well in the cloud. Similarly, dividing an MPI job's processing between on-premises systems and Windows Azure can be problematic for the same reason: The communication between these two worlds might not be fast enough. Still, a significant fraction of MPI jobs can run quite well in the cloud, and it's a good option in some situations.

Conclusion

High-performance computing is entering a new era. The enormous scale and low cost of cloud computing resources is sure to change how and where HPC applications run. Ignoring this change isn't an option.

Yet the rise of the cloud doesn't imply the death of today's clusters. Data issues, networking limitations, and other concerns mean that some HPC jobs remain best suited for on-premises computing. This reality is why Windows HPC Server supports all three possibilities:

- Running applications entirely in on-premises systems.
- Running applications entirely in the cloud.
- Running applications across both on-premises systems and the cloud.

HPC applications can help solve some of the world's most important problems. Cloud computing is surely among the most important technology trends today. It shouldn't be surprising that the two are a natural combination.

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.