High Performance Computing on Microsoft Azure for Scientific and Technical Applications

Many areas of research are compute intensive. Examples include climate modeling, crash simulations, and bioinformatics. As more complex phenomena and greater demands for accuracy increase, so have the demands for cost-effective computational power. Consequently, researchers rely more and more on high performance computing (HPC) that uses clusters of low-cost compute nodes. Further, researchers are turning to the cloud to take advantage of its dynamic allocation of resources, so that they only pay for the resources they actually use. This article is meant to familiarize researchers with the HPC features of Microsoft Azure.

High Performance Computing versus Big Data
HPC is used to solve compute-intensive problems that require a lot of CPU power. Finite element analysis is one example of this type of application. In contrast, a "big data" solution solves data analysis problems by using a cluster of machines to store and analyze an amount of data that can't be handled by a single machine. For example, in computational linguistics, researchers have used Hadoop to map millions of words to geographic locations. Big data solutions tend to rely more on disk IO performance rather than on pure CPU compute power.

A typical HPC node uses CPUs with many cores in order to perform the raw computation. In contrast, a typical big data compute node may have a lot of RAM, and two dozen disks. Of course, clusters can be configured for both IO and CPU power, depending on the application’s requirements.

Microsoft HPC Pack 2012 SP1
Microsoft Azure offers researchers scalable compute resources that can be acquired and released very quickly. If you use Microsoft Azure, there are several ways to manage the compute nodes of your HPC cluster. One way is to use Microsoft HPC Pack 2012 SP1 on a Windows Server, where the server is either local or on a Microsoft Azure VM. (Note that it’s helpful to have some system administration experience in order to configure and manage the HPC Pack.)

The essence of HPC is executing code on many machines at the same time. For HPC Pack 2012 SP1, this means spreading the application’s logic across multiple compute nodes, where each node is a computer that runs Windows. On Microsoft Azure, the equivalent is to run many virtual machines (VMs) simultaneously, all of them working in parallel to solve some problem. 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. HPC Pack 2012 SP1 can run on the premises, as a hybrid solution, or entirely on Microsoft Azure.

Because HPC jobs run their logic simultaneously on multiple machines, HPC Pack 2012 SP1 supports the industry-standard Message Passing Interface (MPI) to let these distributed ranks of code communicate effectively. HPC jobs also commonly exploit multiple cores on each machine they use, and so Windows
HPC Pack 2012 SP1 provides OpenMP, Parallel LINQ (PLINQ), the .NET Framework’s Task Parallel Library, and other software to help developers do this. It also provides debuggers and profilers designed expressly for working with distributed, parallel applications.

To assign the logic of an HPC application to compute nodes, HPC Pack 2012 SP1 provides a scheduler. This scheduler runs on its own dedicated machine called a *head node*.

**What Is a Scheduler**
A scheduler assigns jobs to compute nodes based on their priority, allowing new high-priority jobs to jump to the head of the line. It also lets users specify what kind of resources a job needs, and the scheduler then places the job appropriately. For example, memory-intensive jobs can be assigned to compute nodes in a way that minimizes memory contention. If desirable, the scheduler is also smart enough to avoid letting a few big, high-priority jobs starve all of the others by consuming all of a cluster’s resources.

**Deployment Options**
There several ways to deploy HPC Pack 2012 SP1. One option is a hybrid solution that uses both on premises clusters and Microsoft Azure. Another is to use Microsoft Azure exclusively.

**Hybrid Example**
The following diagram shows an example of HPC Pack 2012 SP1 used in a hybrid configuration.

![Diagram of HPC Pack 2012 SP1 in a hybrid configuration](image)

**Figure 1**
In this example, computation occurs both on the on premises HPC clusters as well as on Microsoft Azure worker nodes. The head node is on the premises, and the scheduler on the head node distributes the application’s logic across the available compute resources. To a user, submitting an HPC job that runs...
partly on the premises and partly in the cloud looks no different than submitting a job that runs entirely in a local cluster.

Benefits of Using a Hybrid Architecture
An organization that has an on-premise installation of HPC may find that it sometimes needs extra compute power. Instead of buying hardware that may be idle most of the time, the organization can use Microsoft Azure for on-demand compute power. The same situation may be true of organizations that are outgrowing their current premises but have no more room to expand. Instead of moving to another data center or adding equipment to one that is already too crowded, they can expand to Microsoft Azure. Using Microsoft Azure also means that the administrative demands of running an HPC cluster, which can be substantial, are no longer an issue.

Microsoft Azure Example
The following diagram shows an example of HPC Pack 2012 SP1 on Microsoft Azure.

![Diagram](image.png)

Figure 2
In this example, everything is located on Microsoft Azure, including the head node, which is installed on a Windows Server 2012 VM. Just as with the hybrid example, a user who submits a Windows HPC job...
sees no difference between running computations on the premises or in the cloud. The scheduler transparently spreads the application’s logic across the Azure worker nodes.

**Benefits of Using a Cloud Architecture**

Putting all of an application’s computing in the cloud can have some clear benefits. The most obvious is that an organization can run HPC jobs without buying and managing its own cluster. Because you only pay for the resources you use, Microsoft Azure is a good choice for organizations whose use of an on premises cluster is too infrequent to justify the expense and management overhead.

Another point to consider is that, compared to a hybrid deployment, running an application entirely in the cloud can make data access easier. Rather than dividing the data between the cloud and on premises installations, or making one part of the application access data remotely, all of the application’s data can be stored in the cloud.

**HPC Application Patterns**

There are several types of HPC applications. One category consists of applications in which logic running on different compute nodes must interact, perhaps to exchange intermediate results, while the application runs. This kind of application commonly relies on some implementation of the Message Passing Interface (MPI), which is a communications protocol for parallel computers.

Another category contains applications where each compute node can work independently of every other compute node. The parallel tasks have no dependencies on each other or any need to communicate with each other. This kind of job is commonly referred to as embarrassingly parallel (or, less pejoratively, delightfully parallel). Two examples of such problems include calculating prime numbers and rendering computer animation, where each frame can be rendered independently of every other frame. In bioinformatics, BLAST searches for multiple queries is another example. To learn how to use Microsoft Azure to create a scalable service for performing BLAST queries, see *Using Microsoft Azure for BLAST* to learn how to perform BLAST queries using Microsoft Azure.

One approach to creating embarrassingly parallel applications is the parametric sweep.

**Parametric Sweep**

Parametric sweep applications run multiple instances of the same program on different sets of input data that are stored in a series of indexed storage items, such as files on a disk or rows in a database table. Each instance of a parametric sweep application runs as a separate task, and many such tasks can execute concurrently, depending on the amount of available cluster resources.

In a parametric sweep application, the compute logic is implemented as an executable. The user passes 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.

When you submit a parametric sweep job to the cluster, you configure the command by using an executable file name or a script file. You can also specify additional properties that define the input and output files, and the sweep index. For more information about how to create a parametric sweep job, see *Define a Parametric Sweep Task*. 
The following figure shows the process of running a parametric sweep job. A user submits the job to the head node. The scheduler determines which worker nodes to use and, with the help of a proxy that runs on each worker node, starts the job’s executable. Each executable gets different parameters passed into it, letting each one work on different data. The job then runs to completion with no further input from the user.

![Diagram showing the process of running a parametric sweep job](image)

**Figure 3**

**Developing HPC Applications**

This section covers the basics of how to develop an HPC application with Windows HPC Server 2012 SP1. When designing or porting an HPC application to Microsoft Azure worker nodes, there are some points to consider that don’t apply to on-premises installations.

- You cannot rely on continuous machine availability for the duration of the job’s execution. Failures and state handling should take this into account.
- You cannot directly access Microsoft Azure nodes using techniques commonly available within an enterprise network cluster, such as using Server Message Block (SMB), or sending a service request directly to a WCF service running in a specific Microsoft Azure node.

**Developing Parametric Sweep Applications**

Developing a parametric sweep application is straightforward. You compile an executable that receives an index parameter, and you use this parameter to access the input data, perform the actual processing, and to output the result to some type of data storage.

When you submit a parametric sweep job, you provide the executable to run, as well as the index range and the step increment. The sweep index parameter is passed directly to the executable as a command-line argument. Any programming language that can run under a worker node can be used to build a parametric sweep application.
Parametric sweeps are ideally suited to the cloud. This is because you can increase the number of compute nodes at will, to suit your immediate needs. Once you no longer need those nodes, you can delete them so that you no longer pay for them.

Migrating UNIX and Linux Applications to Microsoft Azure
Some parametric sweep applications that run on Linux or UNIX systems can be migrated to run under HPC Pack 2012 SP1 and on Microsoft Azure. Eduardo Roloff and colleagues gave a presentation entitled Using Microsoft Azure for High Performance Computing at the Latin American Conference on High Performance Computing (CLCAR) in 2012. In their talk, they discussed how to move UNIX applications to Microsoft Azure. They found that porting HPC applications to Microsoft Azure required modifications to the input and output code but not to the compute code.

Many HPC applications are written in Fortran, but there is no Fortran compiler included with Microsoft Visual Studio. One possible solution is to use Intel's Fortran Composer XE, which integrates with Visual Studio.

Another possibility if you have Linux applications that you would like to port to Microsoft Azure, is to use PToolsWin, from Paratools. This product is a complete development environment that allows you to port high-performance applications from Linux/Unix environments either to Microsoft Azure or to your Windows-based cluster. The environment includes a 64-bit MinGW cross compiler that generates native Windows code, so your application will perform as well as a native Windows application. For an example of how to use PToolsWin, see Cross-Compiling WRF for Microsoft Azure. This page shows how to cross-compile an HPC Linux version of the Weather Research and Forecasting (WRF) model so that it runs on Microsoft Azure.

Two other resources for understanding how to port your applications are The UNIX Custom Application Migration Guide and the UNIX to Windows API Dictionary website.

Microsoft Azure Data Services for HPC Applications
Microsoft Azure offers many ways to save your data. This section gives a brief description of some of them. For complete information, see Introducing Microsoft Azure.

Blobs
Microsoft Azure Blob storage is a service for storing large amounts of unstructured data that can be accessed from anywhere in the world via HTTP or HTTPS. Files can easily be uploaded and downloaded from blob storage. Common uses of Blob storage include serving images or documents directly to a browser and streaming video and audio. It is also well suited for storing and sharing scientific data sets.

Tables
A Microsoft Azure table is a good choice if you need fast access to typed data but don't need to perform complex SQL queries. Tables don't provide relational storage. Instead, they use a NoSQL approach called a key/value store. The basic data item stored in a table is called an entity. An entity is a collection of properties that are name value pairs. Each entity has 3 fixed properties called PartitionKey, RowKey and Timestamp. You can store up to 252 additional properties in an entity.
Service Bus
Microsoft Azure Service Bus allows applications to exchange data, no matter where they are located. All the applications may run on Microsoft Azure, some might run on another cloud platform, or they may be located within a data center. The Service Bus provides a variety of queuing services. The simplest is one-to-one, which means a sender places messages on the queue and a recipient retrieves them. Along with one-to-one queues, the Service Bus also provides a publish-and-subscribe mechanism, for one-to-many communications. Service Bus also allows direct communication through its relay service, which provides a secure way to interact through firewalls.

SQL Database
SQL Database provides relational storage. Formerly called SQL Azure, SQL Database provides all of the key features of a relational database management system, such as atomic transactions, concurrent data access by multiple users with data integrity, and ANSI SQL queries.

SQL Database is a managed service. A managed service means that you don't have to perform any of the administrative grunt work, such as managing the hardware infrastructure and keeping the database and operating system software up to date. All of this is done for you. SQL Database also provides a federation option that distributes data across multiple servers (this practice is also known as sharding). This is useful for applications that work with large amounts of data or need to spread data access requests across multiple servers for better performance.

Setting Up and Deploying HPC
To deploy HPC Pack entirely to Microsoft Azure requires that you set up a Windows 2012 IaaS VM for the head node. To get started, see Windows HPC Pack 2012 Tutorial (1/5): Installing Server 2012 on an IaaS VM. This tutorial shows you how to perform the following steps:

1. Create a Windows 2012 IaaS VM.
2. Download the HPC Pack and unzip.
3. Create a domain controller and join the head node to it.
4. Log on as a domain user and get ready to install HPC Pack 2012.

For more information about installing HPC Pack, see Microsoft HPC Pack 2012: Getting Started.

Deploying the Application to Microsoft Azure
If your application files are on the premises then you must deploy them to Microsoft Azure. The following diagram illustrates the process.
High Performance Computing on Microsoft Azure for Scientific and Technical Applications

Figure 4 shows that there are three steps to follow.

1. Use the `hpcpack create` command to package the application files.
2. Use the `hpcpack upload` command to upload the packages to Microsoft Azure storage.
3. Use the `clusrun hpcsync` command on the headnode to copy or synchronize the packages from Microsoft Azure storage to each of the Microsoft Azure compute nodes; `clusrun` is a utility that runs the specified command on multiple nodes.

You can learn more about the `hpcpack` commands on the TechNet `hpcpack` page. You can learn more about the `hpcsync` command on the TechNet `hpcsync` page.

A Parametric Sweep Example

At a high-level, here are the steps to follow.

1. Deploy the application files to Microsoft Azure.
2. Upload the frames that you want to render to Microsoft Azure blob storage.
3. Create a parametric sweep task.
4. Run a batch file that copies a frame from the blob to local storage, renders it, copies the image to the blob, and deletes the original image that is in local storage.
5. Examine blob storage to make sure that it contains the TIFF frames.
Deploying the Application
Here are the steps for deploying the parametric sweep application to the Microsoft Azure nodes.

**Note:** For the simplicity of this exercise, we assume all of these commands will be run from the HPC cluster headnode.

1. Open a command prompt as administrator. You can also use the Visual Studio command prompt from Microsoft Visual Studio Tools menu.

2. Navigate to the `ImageRendering` labs folder, and use the `hpcpack create` command to create the rendering package for uploading. In the command, specify the package name and the content directory of the package.

3. Upload the deployment package to the Microsoft Azure package storage. Change the value of the `nodetemplate` parameter to the name of your Microsoft Azure node template.

   ```
   hpcpack upload Source\aqsis.zip /nodetemplate:"Default AzureNode Template" /relativePath:aqsis
   ```

4. If you already have Microsoft Azure nodes started in the HPC cluster, you need to copy or sync the new software package to them. To sync the Microsoft Azure nodes with the new packages stored in the application packages blob, run the following command on the hpc cluster headnode.

   ```
   clusrun /nodegroup:AzureNodes hpcsync
   ```

5. You can check the file `deploy.cmd`, which is in the deployment folder, and update any related configuration information.

   ```
   cluscfg setenvs "CCP_PACKAGE_ROOT=c:\apps"
   clusrun /nodegroup:AzureNodes xcopy \\hpc-cluster\apps\aqsis\*.c:\apps\aqsis\ /YE
   hpcpack upload ..\aqsis.zip /nodetemplate:"Default AzureNode Template" /relativePath:aqsis
   clusrun /nodegroup:AzureNodes hpcsync
   ```

Uploading the Frames
There are many ways to upload data to blob storage. A simple way to upload, download, and browse files in blob containers is to install a blob storage browsing application, such as the CloudBerry Explorer for Azure Blob Storage, or the Azure Storage Explorer. For example, by using the Cloudberry application, you can create a blob storage container for the frames and then copy them to Microsoft Azure Blob Storage by using the application’s GUI. The following screenshot shows the input files uploaded to a blob container by using the Cloudberry application.
Creating a Parametric Sweep Task
To create a parametric sweep task, you first connect to the HPC job scheduler and create a new job. The following C# shows an example of how to do this.

```csharp
Scheduler scheduler = new Scheduler();
scheduler.Connect(headnode);

// Define job settings
ISchedulerJob job = scheduler.CreateJob();
job.Name = "Aqsis on Azure";
job.MinimumNumberOfCores = 1;
job.MaximumNumberOfCores = 5;
job.UnitType = JobUnitType.Core;
// Let the scheduler calculate the required resources for the job
job.AutoScaleMax = true;
job.NodeGroups.Add(targetNodes);
```

The next step is to add a parametric task to the new job and submit it to the scheduler. A parametric sweep job requires a start value, an increment value, and an end value. The first iteration of the task uses the start value. The value is then incremented by the specified increment value for each successive
iteration until the end value is reached. The following C# code shows an example of how to add a parametric sweep job. Note that Aqsis is the name of the 3D rendering program.

```csharp
// Create a parametric sweep task
ISchedulerTask task = job.CreateTask();
task.Type = TaskType.ParametricSweep;
task.StartValue = 0;
task.EndValue = endValue;
task.IncrementValue = 1;
// Run the aqsis command to render the images
// The (*) wildcard is used as a placeholder for the current index value
Console.WriteLine("Running job");
job.AddTask(task);
scheduler.SubmitJob(job, username: null, password: null);

Running the Batch File
Each compute node runs a batch file named run.cmd when it runs the parametric sweep job. The batch file runs the AzureBlobCopy utility to download the current file from the input blob, then decompresses it and runs the Aqsis 3D rendering application. Here is the batch file.

REM Use the input parameter as a frame index.
set frame=%1

REM Setup the executable, input, and output folders.
set root=%CCP_PACKAGE_ROOT%\Aqsis
set inputdir=%CCP_WORKDIR%\%CCP_JOBID%\%CCP_TASKID%\input
set outputdir=%CCP_WORKDIR%\%CCP_JOBID%\%CCP_TASKID%\output
if not exist %inputdir% mkdir %inputdir%
if not exist %outputdir% mkdir %outputdir%

REM Pull input data from blob storage.
%root%\bin\AzureBlobCopy.exe -Action Download -BlobContainer input -LocalDir %inputdir% -FileName %frame%.zip

REM Unzip the input file, run the executable, and create output data.
%root%\bin\rar.exe e -y %inputdir%\%frame%.zip %outputdir%
%root%\bin\aqsis.exe -
shaders:"%root%\displacement:%root%\shaders\imager:%root%\shaders\light:%root%\shaders\surface:%root%\shaders\volume" -displays="%root%\bin" %outputdir%\%frame%.rib

REM Upload the output files to blob storage.
%root%\bin\AzureBlobCopy.exe -Action Upload -BlobContainer output -LocalDir %outputdir% -FileName %frame%.tif

REM remove local files
del /Q %inputdir%\%frame%.zip
```
del /Q %outputdir%\%frame%.rib
del /Q %outputdir%\%frame%.tif

**Using AzCopy**
An alternative to the **AzureBlobCopy** utility is to use the **AzCopy** command line utility, which has many useful features. For example, you can copy between blob storage and a local disk or between storage accounts. Copying between storage accounts can occur in the background. The copy operation continues until it completes, even after the **AzCopy** command exits. You can download the **AzCopy** utility from [GitHub](https://github.com).

**Examining Blob Storage**
When the job completes, you can examine the resulting TIFF files that are in blob storage. The following screenshot shows an example that uses the Cloudberry application.

![Figure 6](source.png)

**Summary of Websites**


For a general introduction to Microsoft Azure, see [Introducing Microsoft Azure](http://azure.microsoft.com/en-us/documentation/articles/fundamentals-introduction-to-azure/).


For third-party tools to help you convert Linux HPC applications to Azure, see PToolsWin, from Paratools. For an example of how to use PToolsWin, see Cross-Compiling WRF for Microsoft Azure on the Paratools website.


For information about Intel's Fortran compiler, which is compatible with Visual Studio, see Intel Composer XE Suites (http://software.intel.com/en-us/intel-composer-xe).


You can learn more about the hpcpack commands on the TechNet hpcpack page. (http://technet.microsoft.com/en-us/library/gg481764.aspx)

You can learn more about the hpcsync command on the TechNet hpcsync page. (http://technet.microsoft.com/en-us/library/gg481752(v=wss.10).aspx)

The AzCopy command line utility offers many useful features for handling blobs. You can download it from GitHub (https://github.com/downloads/WindowsAzure/azure-sdk-downloads/AzCopy.zip). For an overview of AzCopy, see the Microsoft Azure Storage Team Blog. (http://blogs.msdn.com/b/windowsazurestorage/archive/2012/12/03/azcopy-uploading-downloading-files-for-windows-azure-blobs.aspx)

Two possible tools you might consider to manage Microsoft Azure Blob Storage are Cloudberry Explorer for Microsoft Azure (http://www.cloudberrylab.com/free-microsoft-azure-explorer.aspx) and Azure Storage Explorer (http://azurestorageexplorer.codeplex.com/).

© 2013 Microsoft Corporation. All rights reserved. Except where otherwise noted, content on this site is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.