Readme

National Center for Biotechnology Information Basic Local Alignment Search Tool (NCBI BLAST) Windows Azure Cloud Service Sample

Lab version: 1.0.0

Last updated: 12/2/2013

**Contents**

[Overview 3](#_Toc364959472)

[Getting Started 5](#_Toc364959473)

[Task 1 - Inspecting the Blast Solution 5](#_Toc364959474)

[Task 2 - Preparing the Blast Solution for Deployment 6](#_Toc364959475)

[Task 4 - Uploading Input Files to Blob Storage 6](#_Toc364959476)

[Task 5 – Configuring Windows Azure environment 8](#_Toc364959477)

[Deployment 10](#_Toc364959478)

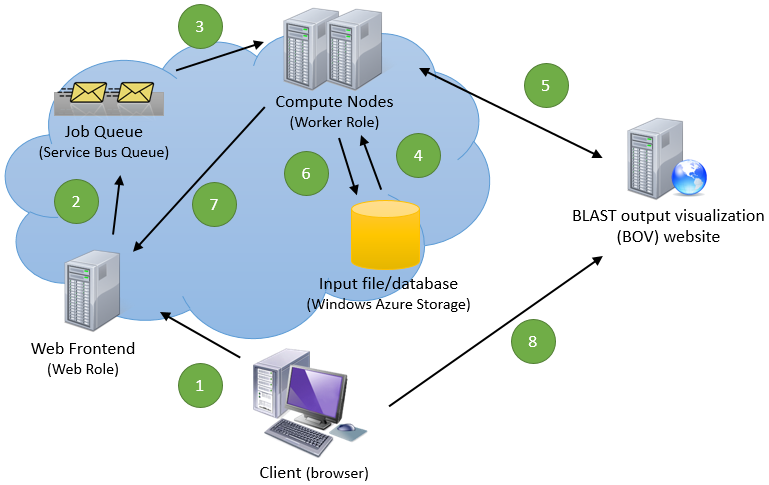
[Testing the service 11](#_Toc364959479)

[Task 1 – Queue a new search task 11](#_Toc364959480)

[Task 2 – Queue a new search task 12](#_Toc364959481)

[Summary 13](#_Toc364959482)

Overview

* 1. The National Center for Biotechnology Information’s Basic Local Alignment Search Tool (NCBI BLAST) HPC Sample demonstrates how to run a [nucleotide match algorithm](http://blast.ncbi.nlm.nih.gov/Blast.cgi) on the human genome using an HPC parametric sweep application.
  2. The parametric sweep application uses a set of input files that contain sequences of nucleotides, comparing them to the human genome database. The application creates output files containing sequence similarities and uploads these files to a BLAST output visualizer (BOV) website.
  3. To run the nucleotide match, the sample uses the blastn utility, which is a part of the [BLAST+](http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download) application.
  4. The architecture of the sample and the steps of its execution are described in Figure 1:
  5. 
  6. Figure 1
  7. Architecture of the BLAST sample
  8. The client application (browser) submits a parametric sweep job to the Web frontend via an AJAX call to the REST API.
  9. The Web frontend pushes the job to a Service Bus queue.
  10. A compute node gets the sweep job from the job queue and executes on it. The compute nodes are competing consumers of the job queue. This pattern allows us to scale out compute nodes as needed and ensures a submitted job is executed at least once.
  11. The parametric sweep application running on a compute node downloads an input file from a Windows Azure blob storage. The input file includes a nucleotide that is to be compared to the human genome database previously downloaded to each Windows Azure compute node (a Worker Role instance).
  12. After completing a sweep index, the BLAST application uploads the resulting matches file to the BLAST output visualization (BOV) website and receives a matching URL for the file’s visualization page.
  13. The output file and the URL are written to Windows Azure storage: the file is uploaded to a blob, and the URL is written in a table.
  14. While the parametric sweep job is running, the compute node notifies the web frontend its progress via a SignalR channel. The web frontend then accesses the Windows Azure table storage to get updated URL list and shows it to the user.
  15. The user can select any of the URLs to see the rendered image for the nucleotide match.
  16. **Note:** To use this sample application, you will need to download the human genome compressed database from the NCBI FTP server, extract the database, and copy it to a Windows Azure blob storage as described later on in the document.

# Key Features

* 1. This sample demonstrates the following features:
  + Uploading a parametric sweep application package to Windows Azure nodes.
  + Uploading and downloading files to Windows Azure blob storage.
  + Writing and reading from Windows Azure table storage.
  + Use Windows Azure Service Bus to provide loose coupling between the web frontend and the compute nodes, allowing the compute nodes to be scaled independently from the web frontend.
  + Use SinglR to push real-time feedbacks from the compute nodes to the web frontend, providing more responsive, engaging user experience.

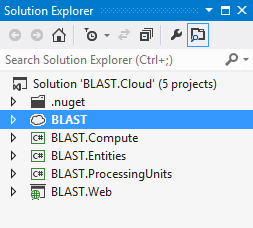
Getting Started

* 1. The following prerequisites are required for running the sample:
  + On Windows Azure
    - An active [Windows Azure subscription](http://www.windowsazure.com/en-us/pricing/purchase-options/)
    - A [Windows Azure storage account](http://www.windowsazure.com/en-us/manage/services/storage/what-is-a-storage-account/)
    - A [Windows Azure Service Bus](http://msdn.microsoft.com/en-us/library/ee732537.aspx) namespace
      1. **Note:** The application runs locally with one compute node. You can deploy the application to Windows Azure and scale out the compute node to multiple instances.
  + On the Development machine:
    - [Microsoft Visual Studio 2012](http://www.microsoft.com/visualstudio/eng/downloads)
    - [Windows Azure SDK and Windows Azure Tools for Microsoft Visual Studio (June 2013)](http://www.windowsazure.com/en-us/develop/net/)

The application runs locally after you configure the connection string to your Windows Azure Service Bus namespace. By default, the application uses Windows Azure emulated storage on the local machine. You should change the storage account to an actual Windows Azure storage account when deploying the application to Windows Azure.

Task 1 - Inspecting the Blast Solution

In this task, you will inspect the **Blast** solution to see the various projects included in this sample.

* 1. Open Microsoft Visual Studio 2012 as administrator.
  2. Open the ***Blast.sln*** solution file located in the ***Blast\Source\Blast*** folder.
  3. Examine the projects tree in the **Solution Explorer** window. The solution tree includes several Windows Azure projects and some class libraries, as shown in Figure 2:
     1. 
     2. Figure 2
     3. The Blast projects tree
  4. The following list describes the purpose of each project in the **Blast** solution:
  + BLAST
    - BLAST is the Windows Azure Cloud Service project that defines a frontend Web Role (BLAST.Web) and a backend Worker Role (BLAST.Compute).
    - A Windows Azure Service Queue connects the Web Role and the Worker Role.
  + BLAST.Compute
    - BLAST.Compute is the backend Worker Role, which can be scaled to multiple instances if needed.
    - The Worker Role does not implement business logics itself. Instead, it hosts Processing Units, which implement actual business logics.
  + BLAST.Entities
    - This Class Library defines business entities.
  + BLAST.ProcessingUnits
    - This Class Library implements business logics.
  + BLAST.Web
    - This this the web frontend.

Task 2 - Preparing the Blast Solution for Deployment

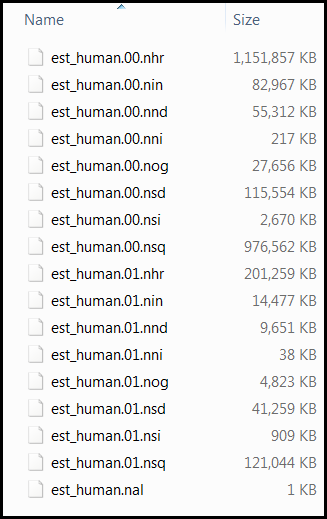
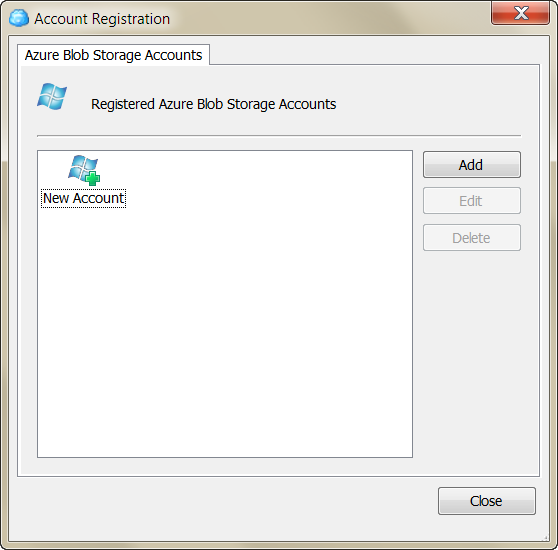
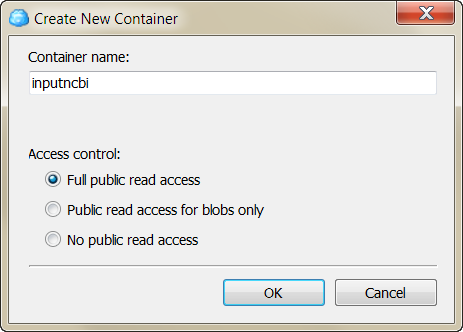
In this task, you will make the necessary adjustments to the projects’ configuration files so you can deploy them to the Windows Azure nodes.

* 1. In the **Solution Explorer** window, double-click on **BAST🡺Roles🡺BLAST.Compute** node to open its properties page.
  2. Go to **Settings** tab.
  3. Modify **Microsoft.ServiceBus.ConnectionString** to the connection string to your Windows Azure Service Bus namespace.
  4. Modify **StorageAccount** to the connection string to your Windows Azure Storage Account.
  5. Modify **HubAddress** to the address where you are planning to deploy the Cloud Service. For example, if your Cloud Service is to be deployed at *http://blast.cloudapp.net/*, then you need to enter the address into this setting. You can also modify this setting after your service has been deployed to Windows Azure.
  6. In the **Solution Explorer** window, double-click on **BAST🡺Roles🡺BLAST.Web** node to open its properties page.
  7. Similarly, update **StorageAccount** and **Microsoft.ServiceBus.ConnectionString**, respectively.

Task 4 - Uploading Input Files to Blob Storage

In this task, you will upload the input files required by the parametric sweep application to a blob in your Windows Azure storage account.

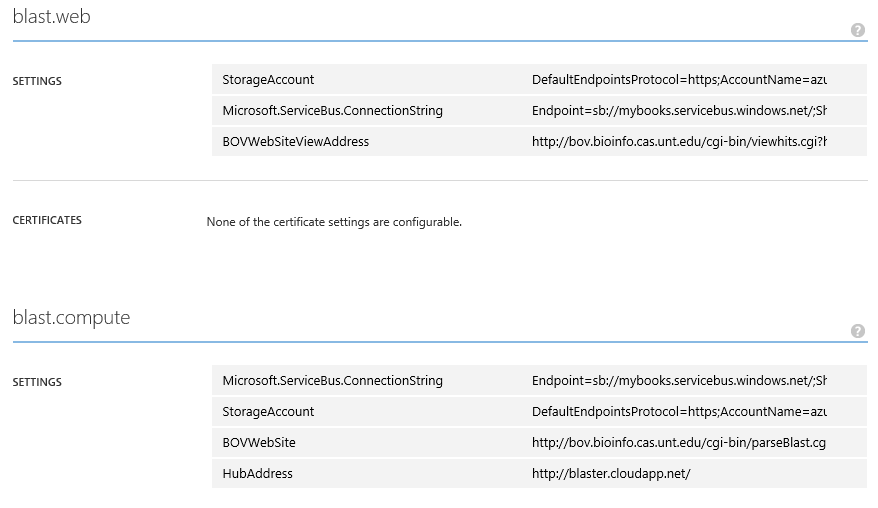
Uploading, downloading, and browsing files in blobs is an easy task if you install one of the blob storage browsing applications, such as [CloudBerry Explorer for Azure Blob Storage](http://cloudberrylab.com/?page=explorer-azure), or the [Azure Storage Explorer](http://azurestorageexplorer.codeplex.com). The following steps are for the CloudBerry Explorer application; you can use the same techniques with Azure Storage Explorer, but the steps may differ.

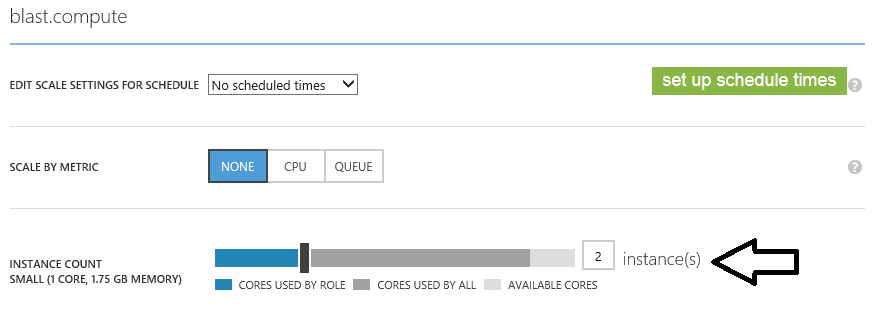
* 1. Download and install CloudBerry Explorer for Azure Blob Storage.
  2. Download and extract the human genome database files from the NCBI FTP server:
     1. Browse to the NCBI FTP server on <ftp://ftp.ncbi.nih.gov/blast/db>
     2. Download the files named **est\_human.00.tar.gz**, and **est\_human.01.tar.gz**.
     3. Extract both .gz files to the same folder. After extracting both files, you should have 17 files, as shown in Figure 3:
     4. 
     5. Figure 3
     6. Content of the extracted NCBI human genome database
  3. Open CloudBerry Explorer for Azure Blob Storage from **Start** | **All Programs** | **CloudBerryLab** | **CloudBerry Explorer for Azure Blob Storage** | **CloudBerry Explorer for Azure Blob Storage**.
  4. Open the **File** menu and select **Azure Blob Storage Accounts**. The **Account Registration** dialog will appear, as shown in Figure 4:
     1. 
     2. Figure 4
     3. The Account Registration dialog
  5. Click the **Add** button, and input the display name of the storage account, the storage account name, and the shared key (the primary access key) of the account.
  6. Click the **Test Connection** button and wait for the approval message. Close the approval message, click OK to add the storage account, and close the **Azure Blob Storage Accounts** dialog.
  7. You should now see your blob storage in the left pane of the application, and your machine (“My Computer”) in the right pane.
  8. Create a new container in the blob by clicking on the **New Container** button in the left pane, as shown in Figure 5:
     1. 
     2. Figure 5
     3. Creating a new blob container
  9. In the **Create New Container** dialog, set the container name to ***inputncbi***, select the **Full public read access** option from the Access control options as shown in Figure 6, and click **OK**.
     1. 
     2. Figure 6
     3. The Create New Container dialog
  10. Locate the newly created container in the list of containers and double-click its name to view its contents (it should be empty for now).
  11. In the right pane, navigate to the **Source\Input** folder that is in the **BLAST** sample folder.
  12. Select all the files (200 files) from the **Input** folder and click the **Copy** button. Click **Yes** in the confirmation message that appears, and then wait for the copy procedure to complete.
  13. Repeat steps 8 and 9 to create another new container, this time naming it ***ncbi***.
  14. In the left pane, click the **Root** folder in the address path to move to the root of the blob.
  15. Locate the ***ncbi***container in the list of containers and double-click its name to see its content (it should be empty for now).
  16. In the right pane, navigate to the folder to which you extracted the database files in step 2 of this task.
  17. Select all the extracted database files (17 files) from the database folder and click the **Copy** button. Click **Yes** in the confirmation message that appears, and then wait for the copy procedure to complete.
      1. **Note:** The size of the database is about 2.5GB, so this operation may take some time, depending on your network bandwidth.
  18. After the upload completes, close the CloudBerry Explorer application.

Task 5 – Configuring Windows Azure environment

* 1. Create a new **JobQueue** Service Queue under your Windows Azure Service Bus namespace.
  2. Create a new **SearchTask** table under your Windows Azure Storage Account.

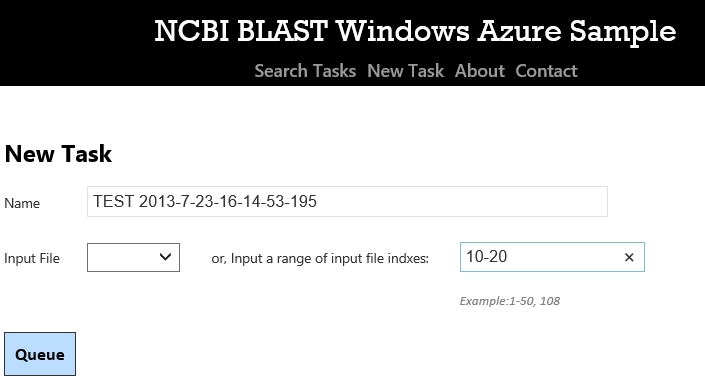
Deployment

* 1. Once your Cloud Service project has been fully configured, in **Solution Explorer**, right click the Cloud Service project (BLAST) and select **Publish…** menu to publish the service to Windows Azure.
  2. After the service is deployed, you can re-configure it on Windows Azure Management Portal. You will need to modify your [**storage account**](http://www.windowsazure.com/en-us/manage/services/storage/how-to-create-a-storage-account/) and [**service bus namespace**](http://www.windowsazure.com/en-us/develop/net/how-to-guides/service-bus-queues/) **credentials.** You also need to add the **BOV web site addresses**, as shown in Figure 7.
  3. 

1. **Figure 7**
2. *Configuring the application on Windows Azure Management Portal*
3. You can also scale out the Worker Role as needed. Because each Worker Role instance only handles one task at a time, you can handle more tasks in parallel if you create multiple instances, as shown in Figure 8:
4. **
5. **Figure 8**
6. *Scaling the Worker Role*

Testing the service

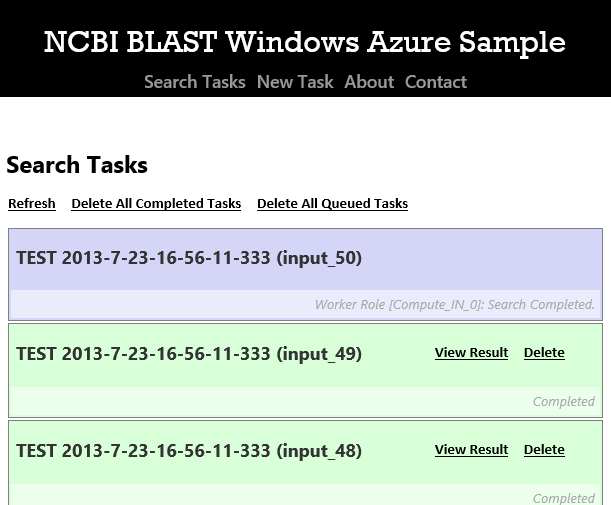
1. Task 1 – Queue a new search task
   1. Open a browser. Navigate to the address of your Cloud Service, which is **http://*[cloud service name]*.cloudapp.net**.
   2. Click the **New Task** link.
   3. On the **New Task** page, pick an input file you want to use, or enter a range of input files. The list of input file reflects the files you have uploaded to ***inputncbi*** container. The system generates a test name based on the server’s current UTC time, but you can modify the name to a friendly name so you can find it more easily.



**Figure 8**

*Enqueue a new task*

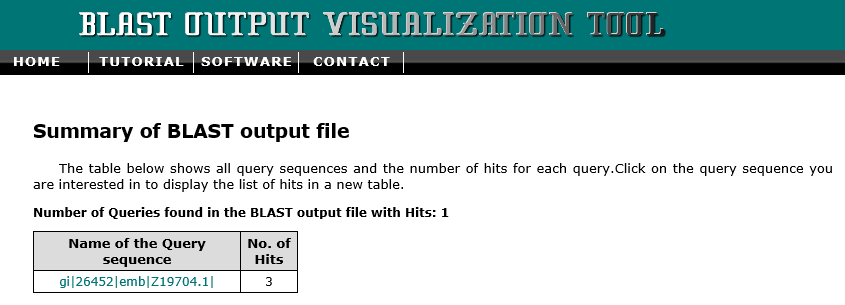
* 1. Once the task is put on the queue, you are redirected to task list page, where you can observe job progress, delete a complete job, retry a failed job, and view task results.



**Figure 9**

*Search task list view*

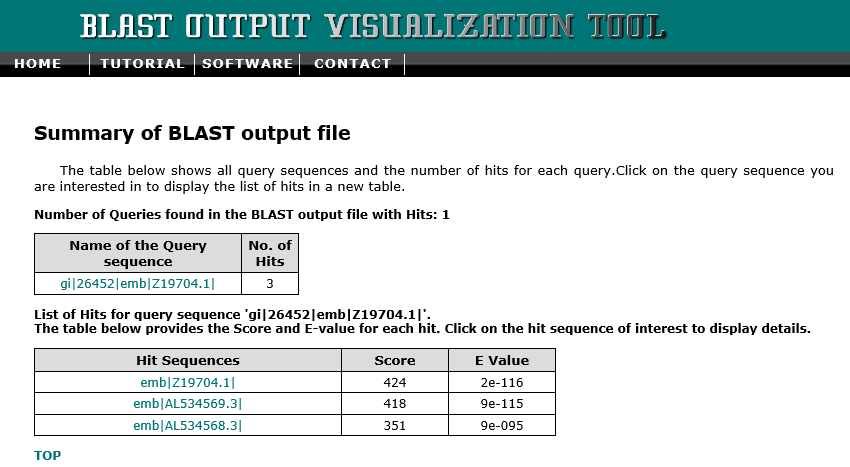
* + 1. **Note:** Because the system needs to download the database files, the very first task after the system has been deployed will take a long time. However, as the system caches database files, jobs after the first will be executed much faster.
  1. After a task completes, you can click on its **View Result** link to view the search result. The link takes you to **bov.bioinfo.cas.unt.edu** (this address is configurable in your Cloud Service settings) to view the result, as shown in Figure 10:



**Figure 10**

*Summary of BLAST output file*

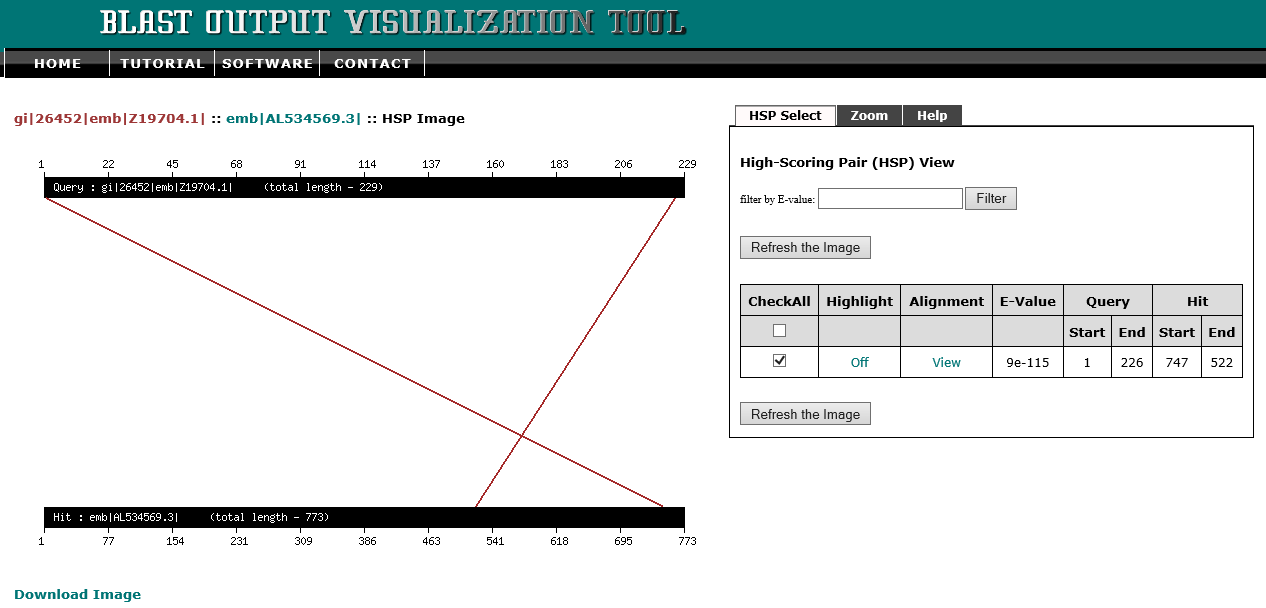
* 1. Click on the link to expand hit sequences, as shown in Figure 11.



**Figure 11**

*List of hit sequences*

* 1. Click on any of hit sequences to view its corresponding HSP image, as shown in Figure 12.



**Figure 12**

*Sample HSP image*

Task 2 – (optional) task management

On search task list view, you can observe task statuses as well as perform simple management tasks:

* 1. Click on the **Delete** link on a task to remove it. Note you can’t delete a job that is being executed.
  2. Click on the **Retry** link to re-submit a task. You can only re-submit a job that has previously failed.
  3. Click on the **View Result** link to view task result.
  4. Click on the **Refresh** link on the top of the page to refresh the page.
  5. Click on the **Delete All Completed Tasks** link to delete all completed tasks.
  6. Click on the **Delete All Queued Tasks** link to delete all queued tasks (that have not been started yet).

Summary

* 1. After running the BLAST sample, you should have learned the following:
  + How to work with Windows Azure blob storage.
  + How to work with Windows Azure table storage.
  + How to deploy Windows Azure web roles to a hosted service.
  + How to package an application for Windows Azure.
  + How to deploy an application to Windows Azure nodes.
  + How to submit a parametric sweep job from the web.
  + How to check a running job’s status.

Copyright 2013 Microsoft Corporation. All rights reserved.   
Except where otherwise noted, these materials are licensed under the terms of the Apache License, Version 2.0. You may use it according to the license as is most appropriate for your project on a case-by-case basis. The terms of this license can be found in  http://www.apache.org/licenses/LICENSE-2.0