Processing Big Data with Hadoop in Azure HDInsight

Lab 3 – Beyond Hive: Pig and Custom UDFs

Overview

While Hive is the most common technology used to process big data in Hadoop, you can also process data using Pig and by creating custom user-defined functions for use in both Pig and Hive.

In this lab, you will use Pig to process data. You will run Pig Latin statements and create Pig Latin scripts that cleanse, shape, and summarize data. You will then create custom UDFs using Python, and use them from both Pig and Hive.

What You'll Need

To complete the labs, you will need the following:

- A web browser
- A Microsoft account
- A Microsoft Azure subscription
- A Microsoft Windows, Linux, or Apple Mac OS X computer on which the Azure CLI has been installed.
- The lab files for this course

Note: To set up the required environment for the lab, follow the instructions in the **Setup** document for this course.

Processing Data with Pig

Pig is designed to provide a high-level processing engine over MapReduce that can be used to process structured, semi-structured, and unstructured data by executing a sequence of commands to transform the data. Pig commands are specified in a language called Pig Latin, which is designed to be intuitive and easy to use, while at the same time providing powerful functionality to make complex transformations to data.

Provision an Azure Storage Account and HDInsight Cluster

Note: If you already have an HDInsight cluster and associated storage account, you can skip this task.

1. Use the steps provided in Lab 1 to provision a Hadoop HDInsight cluster on Linux and an associated storage account.
2. Make a note of the details of your HDInsight cluster configuration.

View and Upload the Source Data

The source data you need to process consist of monthly weather observation data for Heathrow airport in London from 1948 to 2015. You must upload this to Azure storage for processing with HDInsight. The instructions here assume you will use the Azure CLI to do this, but you can use any Azure Storage tool you prefer.

1. Use a text editor to view the **heathrow.txt** file in the **HDILabs\Lab03** folder where you extracted the lab files for this course. Note that the file contains monthly weather observation data for Heathrow airport in London from 1948 to 2015. The first few lines of the file contain unstructured text, which is followed by a multiple rows of tab-delimited values.

   Note: The file contains public sector information licensed under the Open Government License.

2. Close the text file without saving any changes.
3. Open a command line console, and enter the following command to switch the Azure CLI to *resource manager* mode.

   ```
   azure config mode arm
   ```

   Note: If a *command not found* error is displayed, ensure that you have followed the instructions in the setup guide to install the Azure CLI.

4. Enter the following command to log into your Azure subscription:

   ```
   azure login
   ```

5. Follow the instructions that are displayed to browse to the Azure device login site and enter the authentication code provided. Then sign into your Azure subscription using your Microsoft account.
6. Enter the following command to view your Azure resources:

   ```
   azure resource list
   ```

7. Verify that your HDInsight cluster and the related storage account are both listed. Note that the information provided includes the resource group name as well as the individual resource names.
8. Enter the following command on a single line to determine the connection string for your Azure storage account, specifying the storage account and resource group names you noted earlier:

   ```
   azure storage account connectionstring show storage_account -g
   resource_group
   ```

9. Note the connection string, copying it to the clipboard if your command line tool supports it.
10. If you are working on a Windows client computer, enter the following command to set a system variable for the connection string:

    ```
    SET AZURE_STORAGE_CONNECTION_STRING=your_connection_string
    ```

    If you are using a Linux or Mac OS X client computer, enter the following command to set a system variable for the connection string (enter the connection string in quotation marks):

```
export AZURE_STORAGE_CONNECTION_STRING="your_connection_string"
```

11. Enter the following command to upload the **heathrow.txt** file to a blob named **data/weather/heathrow.txt** in the container used by your HDInsight cluster; replacing *data_file* with the local path to heathrow.txt (for example *c:\HDILabs\Lab03\heathrow.txt* or *HDILabs/Lab03/heathrow.txt*) and *container* with the name of the default storage container used by your cluster:

```
azure storage blob upload data_file container data/weather/heathrow.txt
```

12. Wait for the file to be uploaded. Keep the command line console open, you will return to it later in the lab.

Connect to the Cluster

To process the weather data using Pig, will need to open an SSH console that is connected to your cluster:

If you are using a Windows client computer:

1. In the Microsoft Azure portal, on the **HDInsight Cluster** blade for your HDInsight cluster, click **Secure Shell**, and then in the **Secure Shell** blade, under **Windows users**, copy the **Host name** (which should be ***your_cluster_name*-ssh.azurehdinsight.net**) to the clipboard.
2. Open PuTTY, and in the **Session** page, paste the host name into the **Host Name** box. Then under **Connection type**, select **SSH** and click **Open**. If a security warning that the host certificate cannot be verified is displayed, click **Yes** to continue.
3. When prompted, enter the SSH username and password you specified when provisioning the cluster (not the cluster login).

If you are using a Mac OS X or Linux client computer:

1. In the Microsoft Azure portal, on the **HDInsight Cluster** blade for your HDInsight cluster, click **Secure Shell**, and then in the **Secure Shell** blade, under **Linux, Unix, and OS X users**, note the command used to connect to the head node.
2. Open a new terminal session, and enter the appropriate command to connect, specifying your SSH user name (not the cluster login) and cluster name as necessary:

```
ssh your_ssh_user_name@your_cluster_name-ssh.azurehdinsight.net
```

3. If you are prompted to connect even though the certificate can't be verified, enter **yes**.
4. When prompted, enter the password for the SSH username.

Note: If you have previously connected to a cluster with the same name, the certificate for the older cluster will still be stored and a connection may be denied because the new certificate does not match the stored certificate. You can delete the old certificate by using the **ssh-keygen** command, specifying the path of your certificate file (**f**) and the host record to be removed (**R**) - for example:

```
ssh-keygen -f "/home/usr/.ssh/known_hosts" -R clstr-ssh.azurehdinsight.net
```

View the Uploaded Data File

Now that you have a remote command line for your cluster, you can verify that the weather observation data has been uploaded to the shared cluster storage.

1. In the SSH console window, enter the following command to view the contents of the **/data/weather** folder in the HDFS file system.

   ```
   hdfs dfs -ls /data/weather
   ```

2. Verify that the folder contains the heathrow.txt file you uploaded.

Use the Grunt Shell to Run Pig Commands

1. In the console containing an SSH session to your cluster, enter the following command to start the Grunt shell:

   ```
   pig
   ```

2. In the Grunt shell, enter the following Pig Latin statement to load the files in the /data/weather folder into a relation called **Source**. The data is loaded into a tab-delimited schema with seven columns. Lines with no tab characters will result in a row (or *tuple*) containing a single value in the first column and NULL values in the remaining columns.

   ```
   Source = LOAD '/data/weather' USING PigStorage('\t') AS
   (year:chararray, month:chararray, maxtemp:float, mintemp:float,
   frost:int, rainfall:float, sunshine:float);
   ```

3. Ignore any warning messages, and enter the following Pig Latin statement to filter the data so that only tuples with a value in the **maxtemp** and **mintemp** columns are included.

   ```
   Data = FILTER Source BY maxtemp IS NOT NULL AND mintemp IS NOT NULL;
   ```

4. Enter the following Pig Latin statement to further filter the data to remove the header row.

   ```
   Readings = FILTER Data BY year != 'yyyy';
   ```

5. Enter the following Pig Latin statement to group the data by year.

   ```
   YearGroups = GROUP Readings BY year;
   ```

6. Enter the following Pig Latin statement to calculate average **maxtemp** and average **mintemp** for each year.

   ```
   AggTemps = FOREACH YearGroups GENERATE group AS year,
   AVG(Readings.maxtemp) AS avghigh, AVG(Readings.mintemp) AS avglow;
   ```

7. Enter the following Pig Latin statement to sort the temperature data by year.

   ```
   SortedResults = ORDER AggTemps BY year;
   ```

8. Enter the following Pig Latin statement to display the results in the console. This runs a MapReduce job to perform all of the transformations you have entered so far.

   ```
   DUMP SortedResults;
   ```

9. Wait for the job to complete, and then view the results, which include the average monthly maximum and minimum temperatures for each year. Then enter the following command to exit Pig.

   ```
   quit;
   ```

10. Keep the SSH console open, you will return to it in the next procedure.

Run a Pig Latin Script

1. Use a text editor to open the **scrub_weather.pig** file in the **HDILabs\Lab03** folder.
2. Review the Pig Latin code in this file, and note that the script performs the following tasks:
    a. Loads the tab-delimited data in the **/data/weather** folder into a schema that includes the columns **year**, **month**, **maxtemp**, **mintemp**, **frostdays**, **rainfall**, and **sunshinehours**.
    b. Filters the data to remove text notes and header rows.
    c. Replaces any "---" values (which indicate missing data) in the **sunshinehours** column with an empty string.
    d. Splits the data into a relation in which **sunshinehours** contains a "'**#**" character denoting a sensor reading (which makes the data dirty) and a relation in which **sunshinehours** is already clean.
    e. Cleans the rows in which **sunshinehours** contains a "**#**" by using the SUBSTRING and INDEXOF functions.
    f. Re-combines the cleaned rows with the rows that were already clean.
    g. Sorts the data by year and month.
    h. Stores the cleaned and sorted data in the **/data/scrubbedweather** folder as a space-delimited text file.
3. Close the text editor.
4. In the command line console for your local computer, in which you previous used the Azure CLI to upload a text file to Azure storage, enter the following command to upload the Pig script file, replacing *pig_script* with the local path to the **scrub_weather.pig** file, and *container* with the name of the default storage container used by your cluster:

    ```
    azure storage blob upload pig_script container data/scrub_weather.pig
    ```

5. Keep the command line window open, you will return to it later in this lab.
6. Return to the SSH console for your cluster, and enter the following command to run the pig script you uploaded to Azure storage:

    ```
    pig wasb:///data/scrub_weather.pig
    ```

7. Wait for the script to finish, then enter the following command to verify that the script has written the results to the output folder:

    ```
    hdfs dfs -ls /data/scrubbedweather
    ```

8. Enter the following command to view the cleaned weather data:

    ```
    hdfs dfs -text /data/scrubbedweather/part-r-00000
    ```

9. Keep the SSH console for your cluster open, you will return to it later in the lab.

Using Python User-Defined Functions

In many scenarios, Hive and Pig provide all the functionality you need to transform and query your data. However, in some cases you might need to implement custom data processing logic that would be complex or impossible to achieve in Hive or Pig alone. Rather than resort to writing your own custom MapReduce components to achieve this; you can create a UDF that can be called from Hive or Pig. You can create UDFs in many languages, including Java and Microsoft C#; but increasingly Python is becoming the programming language of choice for Big Data processing.

Use a Python UDF from Pig

Pig provides native support for Jython – a Java implementation of Python. This enables you to write Pig Latin code that calls Python UDFs directly.

1. Use a text editor to open the **convert_temp.py** file in the local **HDILabs\Lab03** folder. Then review the Python code this file contains, and note that the code defines an output schema that includes a Pig *bag* structure named **f_readings**. The bag contains fields named **year**, **month**, **maxtemp**, **mintemp**, **frostdays**, **rainfall**, and **sunshinehours**. The code then defines a function named **fahrenheit** with an input parameter named **c_reading**. This function:
    a. Splits the input parameter into **year**, **month**, **maxtemp**, **mintemp**, **frostdays**, **rainfall**, and **sunshinehours** variables.
    b. Creates a variable named **maxtemp_f**, which is calculated as **maxtemp** multiplied by nine divided by five and added to 32 (the equation to convert Celsius to Fahrenheit).
    c. Similarly, creates a variable named **mintemp_f** with a value of **mintemp** converted to Fahrenheit.
    d. Returns the **year**, **month**, **maxtemp_f**, **mintemp_f**, **frostdays**, **rainfall**, and **sunshinehours** variables.
2. When you have finished viewing the code, close the text editor without saving any changes.
3. Use the text editor to open the **convert_weather.pig** file in the local **HDILabs\Lab03** folder. Then review the Pig Latin code this file contains, and note that it performs the following tasks:
    a. Registers the **convert_temp.py** Python file as a Jython UDF.
    b. Loads the **scrubbedweather** source data you generated in the previous exercise into a relation named **Source**, with a single character array value for each line of text.
    c. Creates a relation named **ConvertedReadings** that uses the **fahrenheit** function in the **convert_temp.py** file to generate each row.
    d. Stores the **ConvertedReadings** relation in the **/data/convertedweather** folder.
4. When you have finished viewing the code, close the text editor without saving any changes.
5. In the local command window in which you previous used the Azure CLI to upload files to Azure storage, enter the following command to upload the Python code file, replacing *python_file* with the local path to the **convert_temp.py** file, and *container* with the name of the default storage container used by your cluster:

```
azure storage blob upload python_file container data/convert_temp.py
```

6. Enter the following command to upload the Pig script file, replacing *pig_script* with the local path to the **convert_weather.pig** file, and *container* with the name of the default storage container used by your cluster:

```
azure storage blob upload pig_script container data/convert_weather.pig
```

7. Keep the command window open, you will return to it later in this lab.
8. Return to the SSH console for your cluster, and enter the following command to run the Pig script you uploaded to Azure storage:

```
pig wasb:///data/convert_weather.pig
```

9. Wait for the script to finish, then enter the following command to verify that the script has written the results to a file named **part-m-00000** in the output folder:

```
hdfs dfs -ls /data/convertedweather
```

10. Enter the following command to view the converted weather data:

```
hdfs dfs -text /data/convertedweather/part-m-00000
```

11. Keep the SSH console for your cluster open, you will return to it later in the lab.

Use a Python UDF from Hive

Hive supports Python UDFs through a *streaming* technique, in which data is passed between Hive and Python through the **stdin** and **stdout** interfaces. You can write a Python script to read each line of text and process the data, and then use the print command to return the results to Hive.

1. Use a text editor to open the **create_table.hql** file in the local **HDILabs\Lab03** folder. Then review the HiveQL code this file contains, and note that the code creates a table named weather based on the **/data/convertedweather** folder generated by the **convert_weather.pig** Pig script in the previous procedure. The table includes a column named **rainfall**, in which the amount of rainfall is measured in millimeters.
2. When you have finished viewing the code, close the text editor without saving any changes.
3. Use the text editor to open the **convert_rain.py** file in the local **HDILabs\Lab03** folder. Then review the Python code this file contains, and note that it performs the following tasks:
    a. Reads each line of text input from the standard input port.
    b. Removes the new-line character in each line.
    c. Splits the remaining text into fields based on a tab-delimiter.
    d. Calculates a **rainfall_inches** field by converting the value in the **rainfall_mm** field to inches.
    e. Writes the fields to the standard output buffer, replacing **rainfall_mm** with **rainfall_inches**.
4. When you have finished viewing the code, close the text editor without saving any changes.
5. In the command window in which you previous used the Azure CLI to upload files to Azure storage, enter the following command to upload the Python code file, replacing *python_file* with the local path to the **convert_rain.py** file, and *container* with the name of the default storage container used by your cluster:

    ```
    azure storage blob upload python_file container data/convert_rain.py
    ```

6. Enter the following command to upload the HiveQL script file, replacing *hive_script* with the local path to the **create_table.hql** file, and *container* with the name of the default storage container used by your cluster:

    ```
    azure storage blob upload hive_script container data/create_table.hql
    ```

7. Return to the SSH console for your cluster, and enter the following command to run the Hive script you uploaded to Azure storage:

    ```
    hive -i wasb:///data/create_table.hql
    ```

8. Wait for the script to open the Hive command line interface, and then enter the following query to verify that the weather table has been created on the output previously generated by Pig:

    ```
    SELECT * FROM weather;
    ```

9. Enter the following code to import the Python code you uploaded and use the UDF it defines to query the **weather** table (you can copy and paste this code from **query_hive_table.txt** in the **HDILabs\Lab03** folder):

    ```
    add file wasb:///data/convert_rain.py;

    SELECT TRANSFORM (year, month, max_temp, min_temp, frost_days,
    rainfall, sunshine_hours)
      USING 'python convert_rain.py' AS
      (year INT, month INT, max_temp FLOAT, min_temp FLOAT, frost_days INT,
    rainfall FLOAT, sunshine_hours FLOAT)
    ```

```
FROM weather;
```

10. View the results, noting that **rainfall** (the second-last column) contains values in inches instead of millimeters.
11. Enter the following command to exit Hive:

    ```
    exit;
    ```

12. Close all command windows.

Cleaning Up

Now that you have finished this lab, you can delete the HDInsight cluster and storage account.

Note: If you are proceeding straight to the next lab, omit this task and use the same cluster in the next lab. Otherwise, follow the steps below to delete your cluster and storage account.

Delete the HDInsight Cluster

If you no longer need the HDInsight cluster used in this lab, you should delete it to avoid incurring unnecessary costs (or using credits in a free trial subscription).

1. In the Microsoft Azure portal, click **Resource Groups**.
2. On the **Resource groups** blade, click the resource group that contains your HDInsight cluster, and then on the **Resource group** blade, click **Delete**. On the confirmation blade, type the name of your resource group, and click **Delete**.
3. Wait for your resource group to be deleted, and then click **All Resources**, and verify that the cluster, and the storage account that was created with your cluster, have both been removed.
4. Close the browser.