edX Study Lab3

(이삭엔지니어링 인턴 김형근 : hkkim@isaac-eng.com)

Lab 3 – Beyond Hive – Pig and Python

(문서 내용 중 옥색으로 칠한 부분은, 기존의 문서에서 CDH 환경에 맞도록 내용이 변경된 부분이다. CDH 6.0에서 실습하였음.)

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.

Upload the Log Files to Hue

1. Connect to 10.100.3.216:8888
2. Log in Hue (user : hkkim)
3. Set path /user/hkkim
4. Make new directory named **data**
5. In **/user/hkkim/data** directory, make directory **logs**
6. In **/user/hkkim/data/weather** directory, upload text files **heathrow.txt**

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 **Putty**, connect to 10.100.3.216 and enter the following command to view the contents of the **/user/hkkim/data/weather** folder in the HDFS file system.

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

1. 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 '/user/hkkim/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 spacedelimited text file. 3. Close the text editor.
4. In Azure Storage Explorer, in the **Upload** drop-down list, click **Upload Files**. Then upload **scrub_weather.pig** as a block blob to the existing **data** folder.
5. Keep Azure Storage Explorer open – you will use it again in a later procedure.
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 hdfs://iseHA/user/hkkim/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 /user/hkkim/data/scrubbedweather
```

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

```
hdfs dfs -text /user/hkkim/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 Azure Storage Explorer, in the **Upload** drop-down list, click **Upload Files**. Then upload both **convert_temp.py** and **convert_weather.pig** as block blobs to the existing **data** folder.
6. Keep Azure Storage Explorer open – you will use it again in a later procedure.
7. Return to the SSH console for your cluster, and enter the following command to run the Pig script you uploaded to Azure storage:

```
pig hdfs://iseHA/user/hkkim/data/convert_weather.pig
```

8. 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 /user/hkkim/data/convertedweather
```

9. Enter the following command to view the converted weather data: `hdfs dfs -text`

   ```
   /user/hkkim/data/convertedweather/part-m-00000
   ```

10. 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. upload **convert_rain.py**
6. Return to the SSH console for your cluster, and enter the following command in **create_table.hql** to run the Hive Query in Hue

   ```
   CREATE EXTERNAL TABLE weather
   (year INT, month INT, max_temp FLOAT, min_temp FLOAT, frost_days INT,
   rainfall FLOAT, sunshine_hours FLOAT)
   ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
   STORED AS TEXTFILE LOCATION '/user/hkkim/data/convertedweather';
   ```

7. 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;
   ```

8. 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 hdfs://iseHA/user/hkkim/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 edx_hkkim.weather;
   ```

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

    ```
    exit;
    ```

Reference : https://github.com/MicrosoftLearning/Processing-Big-Data-with-Hadoop-in-Azure-HDInsight/blob/master/Labs/Lab%203%20-%20Beyond%20Hive%20-%20Pig%20and%20Python.pdf