Processing Big Data with Hadoop in Azure HDInsight

Lab 2B – Advanced Hive Techniques

Overview

In this lab, you will use Tez to improve the performance of Hive queries, create partitioned Hive tables, create indexes on Hive tables, and create a Hive view.

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 computer with the following software installed:
    - Microsoft Azure PowerShell
- 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. Specifically, you must have signed up for an Azure subscription, installed and configured Azure PowerShell, and imported the publisher settings for your Azure subscription into PowerShell.

When working with cloud services, transient network errors can occasionally cause scripts to fail. If a script fails, and you believe that you have entered all of the required variables correctly; wait a few minutes and run the script again.

Using Tez

Using Tez can significantly enhance the performance of Hive queries, but it is not enabled by default in HDInsight 3.2 clusters. In this exercise you will observe the effect of enabling Tez for HiveQL queries.

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. In a web browser, navigate to http://azure.microsoft.com. Then click **Portal**, and if prompted, sign in using the Microsoft account that is associated with your Azure subscription.
2. In the Microsoft Azure portal, view the **HDInsight** page and verify that there are no existing HDInsight clusters in your subscription.

3. Click **NEW** (at the bottom of the page) and then click **CUSTOM CREATE**. Then use the New HDInsight Cluster wizard to create a new cluster with the following settings. Click the arrows to navigate through all of the wizard pages:
   - **Cluster Name**: *Enter a unique name (and make a note of it!)*
   - **Cluster Type**: Hadoop
   - **Operating System**: Windows Server 20012 R2 Datacenter
   - **HDInsight Version**: 3.2 (HDP 2.2, Hadoop 2.6)
   - **Data Nodes**: 2
   - **Region**: *Select any available region*
   - **Head Node Size**: A3 (4 cores, 7 GB memory)
   - **Data Node Size**: A3 (4 cores, 7 GB memory)
   - **HTTP User Name**: *Enter a user name of your choice (and make a note of it!)*
   - **HTTP Password**: *Enter and confirm a strong password (and make a note of it!)*
   - **Enable the remote desktop for cluster:** Selected
   - **RDP User Name:** *Enter another user name of your choice (and make a note of it!)*
   - **RDP Password:** *Enter and confirm a strong password (and make a note of it!)*
   - **Expires on**: Select tomorrow's date
   - **Enter the Hive/Oozie Metastore**: Unselected
   - **Storage Account**: Create New Storage
   - **Account Name**: *Enter a unique name for your storage account (and make a note of it!)*
   - **Default Container**: *Enter a unique name for your container (and make a note of it!)*
   - **Additional Storage Accounts**: 0
   - **Additional scripts to customize the cluster**: *None*
4. Wait for the cluster to be provisioned and the status to change to **Running** (this can take a while.)

Upload Files and Create Hive Tables

1. In the C:\HDILabs\Lab2B folder, rename **Upload Hive Data.txt** to **Upload Hive Data.ps1** (you may need to modify the *View* options for the folder to see file extensions). Then right-click **Upload Hive Data.ps1** and click **Edit** to open the script in the Windows PowerShell interactive script environment (ISE).
2. Change the values assigned to the **$clusterName**, **$storageAccountName**, and **$containerName** variables to match the configuration of your HDInsight cluster.
3. Review the rest of the code in the script, and note that it performs the following actions:
   a. Uploads the **CreateHiveTables.txt** file to **/data** in your Azure storage account, and then starts a Hive job to run the HiveQL script that the file contains. This script creates a Hive table named **rawlog** on the **/data/logs** folder.
   b. Uploads the files in the local **iislogs_gz** subfolder to **/data/logs** in your Azure storage container (overwriting any existing files of the same name). These files are compressed IIS web server log files.
4. Save the PowerShell script, and on the toolbar, click **Run Script**. Then wait several minutes for the script to finish and close Windows PowerShell ISE.

Note: If you installed Azure PowerShell after August 14th 2015, you may see the following error. You can ignore this.

```
Get-AzureHDInsightJobOutput : Could not load file or assembly
'Microsoft.WindowsAzure.Storage, Version=3.0.3.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35' or one of its dependencies. The system cannot find
the file specified.
```

Query a Hive Table

1. In a web browser, navigate to http://azure.microsoft.com. Then click **Portal**, and if prompted, sign in using the Microsoft account that is associated with your Azure subscription.
2. In the Azure portal, on the **HDInsight** page, select your HDInsight cluster and click **Query Console**. Then log into the query console using the HTTP user name and password for your cluster.
3. In the HDInsight query console, view the **Hive Editor** page.
4. In the **Query Name** box, type **Query Log Data (MR)**. Then replace the default **Select** statement with the following code (you can copy and paste this from **Query Log Data.txt** in the C:\HDILabs\Lab02B folder):

   ```
   SELECT log_date, COUNT(*), SUM(sc_bytes), SUM(cs_bytes)
   FROM rawlog
   WHERE SUBSTR(log_date, 1, 1) <> '#'
   GROUP BY log_date
   ORDER BY log_date;
   ```

5. Click **Submit** and wait for the **Query Log Data (MR)** job to be listed in the **Job Session** table.
6. While the job is still running, change the value in the **Query Name** box to **Query Log Data (Tez)**. Then modify the query by as follows:

   ```
   set hive.execution.engine=tez;

   SELECT log_date, COUNT(*), SUM(sc_bytes), SUM(cs_bytes)
   FROM rawlog
   WHERE SUBSTR(log_date, 1, 1) <> '#'
   GROUP BY log_date
   ORDER BY log_date;
   ```

7. Click **Submit** and wait for the **Query Log Data (Tez)** job to be listed in the **Job Session** table.
8. When the status of each job changes to **Completed**, click the job name in the **Query Name** column of the **Job Session** table. This opens a new tab containing the output generated by the job in which you can view the **Job Log**, noting the time taken to execute the job. Then close the **Job Status** tab for the job.

Partitioning Data

Hive provides three ways to separate out the storage of data in a Hive table into multiple subfolders and files. This can improve performance when queries commonly filter on specific columns, or when the job can be effectively split across multiple reducer nodes.

Create a Partitioned Table

1. In the HDInsight query console, view the **Hive Editor** page.
2. In the **Query Name** box, type **Partition Log Data**. Then replace the current query code with the following code (you can copy and paste this from **Partition.txt** in the C:\HDILabs\Lab02B folder):

   ```
   set hive.execution.engine=tez;

   CREATE EXTERNAL TABLE partitionedlog
   (log_day int,
    log_time STRING,
    c_ip STRING,
    cs_username STRING,
    s_ip STRING,
    s_port STRING,
    cs_method STRING,
   ```

```
 cs_uri_stem STRING,
 cs_uri_query STRING,
 sc_status STRING,
 sc_bytes INT,
 cs_bytes INT,
 time_taken INT,
 cs_user_agent STRING,
 cs_referrer STRING)
PARTITIONED BY (log_year int, log_month int)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
STORED AS TEXTFILE LOCATION '/data/partitionedlog';

SET hive.exec.dynamic.partition.mode=nonstrict;
SET hive.exec.dynamic.partition = true;
INSERT INTO TABLE partitionedlog PARTITION(log_year, log_month)
SELECT DAY(log_date),
       log_time,
       c_ip,
       cs_username,
       s_ip,
       s_port,
       cs_method,
       cs_uri_stem,
       cs_uri_query,
       sc_status,
       sc_bytes,
       cs_bytes,
       time_taken,
       cs_user_agent,
       cs_referrer,
       YEAR(log_date),
       MONTH(log_date)
FROM rawlog
WHERE SUBSTR(log_date, 1, 1) <> '#';
```

3. Review the code, noting that it creates a table named **partitionedlog** that is partitioned on columns named **log_year** and **log_month**, and then loads the log data from the **rawlog** table into the partitions of the **partitionedlog** table.
4. Click **Submit**, and wait for the query to complete.
5. When the query as completed, in the HDInsight query console, view the **File Browser** page, and navigate to the ***storage_account/container*****/data/partitionedlog** folder (where *storageaccount* is you Azure storage account and *container* is the blob container used by your HDInsight cluster). Note that this folder contains a folder for each year in the log data (there is only one year; 2008).
6. Click the **log_year=2008** folder and note that the data is further partitioned into a folder for each month. Then view the contents of the **log_month=1** folder and note that each month partition contains one or more data files for the log entries in that month. By partitioning the data in this way, queries such as the following one will benefit from having to search a smaller volume of data to retrieve the results:

```
SELECT log_day, log_time, c_ip
FROM partitionedlog
WHERE log_year=2008 AND log_month=6;
```

Create a Skewed Table

1. In the HDInsight query console, view the **Hive Editor** page.

2. In the **Query Name** box, type **Skew Log Data**. Then replace the current query code with the following code (you can copy and paste this from **Skew.txt** in the C:\HDILabs\Lab02B folder):

```
set hive.execution.engine=tez;

CREATE EXTERNAL TABLE skewedlog
(log_date DATE,
 log_time STRING,
 c_ip STRING,
 cs_username STRING,
 s_ip STRING,
 s_port STRING,
 cs_method STRING,
 cs_uri_stem STRING,
 cs_uri_query STRING,
 sc_status STRING,
 sc_bytes INT,
 cs_bytes INT,
 time_taken INT,
 cs_user_agent STRING,
 cs_referrer STRING)
SKEWED BY (cs_uri_stem) ON ('/default.aspx') STORED AS DIRECTORIES
ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
STORED AS TEXTFILE LOCATION '/data/skewedlog';

INSERT INTO TABLE skewedlog
SELECT log_date,
       log_time,
       c_ip,
       cs_username,
       s_ip,
       s_port,
       cs_method,
       cs_uri_stem,
       cs_uri_query,
       sc_status,
       sc_bytes,
       cs_bytes,
       time_taken,
       cs_user_agent,
       cs_referrer
FROM rawlog
WHERE SUBSTR(log_date, 1, 1) <> '#';
```

3. Review the code, noting that it creates a table named **skewedlog** that skews the data into directories based on a **cs_uri_stem** value of **'/default.aspx'**, and then loads the log data from the **rawlog** table into the partitions of the **skewedlog** table.
4. Click **Submit**, and wait for the query to complete.
7. When the query as completed, in the HDInsight query console, view the **File Browser** page, and navigate to the ***storage_account*/*container*/data/skewedlog** folder. Note that this folder contains a folder for rows with a **cs_uri_stem** value of **'/default.aspx'** and a folder for all other rows. Skewing the data in this way can improve queries when a frequently filtered column contains a high percentage of rows with the same value. For example, if approximately 50% of web requests are for the **/default.aspx** page, skewing the data halves the volume of data that must be searched by the following query:

```
SELECT log_date, log_time, c_ip
FROM skewedlog
```

```
WHERE cs_uri_stem = '/default.aspx';
```

Create a Clustered Table

1. In the HDInsight query console, view the **Hive Editor** page.
2. In the **Query Name** box, type **Cluster Log Data**. Then replace the current query code with the following code (you can copy and paste this from **Cluster.txt** in the C:\HDILabs\Lab02B folder):

   ```
   set hive.execution.engine=tez;

   CREATE EXTERNAL TABLE clusteredlog
   (log_date DATE,
    log_time STRING,
    c_ip STRING,
    cs_username STRING,
    s_ip STRING,
    s_port STRING,
    cs_method STRING,
    cs_uri_stem STRING,
    cs_uri_query STRING,
    sc_status STRING,
    sc_bytes INT,
    cs_bytes INT,
    time_taken INT,
    cs_user_agent STRING,
    cs_referrer STRING)
   CLUSTERED BY (c_ip) INTO 10 BUCKETS
   ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
   STORED AS TEXTFILE LOCATION '/data/clusteredlog';

   INSERT INTO TABLE clusteredlog
   SELECT log_date,
          log_time,
          c_ip,
          cs_username,
          s_ip,
          s_port,
          cs_method,
          cs_uri_stem,
          cs_uri_query,
          sc_status,
          sc_bytes,
          cs_bytes,
          time_taken,
          cs_user_agent,
          cs_referrer
   FROM rawlog
   WHERE SUBSTR(log_date, 1, 1) <> '#';
   ```

3. Review the code, noting that it creates a table named **clusteredlog** that clusters the data into 10 buckets based on the **cs_ip** column value, and then loads the log data from the **rawlog** table into the partitions of the **clusteredlog** table.
4. Click **Submit**, and wait for the query to complete.
8. When the query as completed, in the HDInsight query console, view the **File Browser** page, and navigate to the ***storage_account/container/*****data/clusteredlog** folder. Note that this folder contains ten files for the data in the table. Clustering the data in this way can improve queries that use JOIN operations based on the clustered keys, for example, the following query joins the **clusteredlog** table to a hypothetical **sales** table:

```
set hive.optimize.bucketmapjoin=true;
SELECT /*+ MAPJOIN(s,c) */ s.user_id, c.log_date
FROM sales AS s JOIN clusteredlog AS c
ON s.client_ip = c.c_ip;
```

Note: The **set** command instructs Hive to optimize map join operations using bucketed (clustered) data. The **/*+ MAPJOIN(s,c) */** hint instructs Hive to perform joins during the Map phase of the MapReduce job generated by the query.

Creating an Index

Hive supports the creation of indexes to help optimize query performance.

Create a Table

1. In the HDInsight query console, view the **Hive Editor** page.
2. In the **Query Name** box, type **Create Table**. Then replace the current query code with the following code (you can copy and paste this from **Indexed Table.txt** in the C:\HDILabs\Lab02B folder):

   ```
   set hive.execution.engine=tez;

   CREATE EXTERNAL TABLE indexedlog
   (log_year int,
    log_month int,
    log_day int,
    log_time STRING,
    c_ip STRING,
    cs_username STRING,
    s_ip STRING,
    s_port STRING,
    cs_method STRING,
    cs_uri_stem STRING,
    cs_uri_query STRING,
    sc_status STRING,
    sc_bytes INT,
    cs_bytes INT,
    time_taken INT,
    cs_user_agent STRING,
    cs_referrer STRING)
   ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
   STORED AS TEXTFILE LOCATION '/data/indexedlog';

   INSERT INTO TABLE indexedlog
   SELECT YEAR(log_date),
          MONTH(log_date),
          DAY(log_date),
          log_time,
          c_ip,
          cs_username,
          s_ip,
          s_port,
          cs_method,
          cs_uri_stem,
          cs_uri_query,
          sc_status,
          sc_bytes,
          cs_bytes,
   ```

```
        time_taken,
        cs_user_agent,
        cs_referrer
FROM rawlog
WHERE SUBSTR(log_date, 1, 1) <> '#';
```

3. Review the code, noting that it creates a table named **indexedlog**, and then loads the log data from the **rawlog** table into the partitions of the **indexedlog** table. Despite its name, the **indexedlog** table has no indexes defined on it.
4. Click **Submit**, and wait for the query to complete.
5. When the query as completed, in the HDInsight query console, view the **File Browser** page, and navigate to the ***storage_account/container/*data/indexedlog** folder. Note that this folder contains the data file for the table.

Create an Index

1. In the HDInsight query console, view the **Hive Editor** page.
2. In the **Query Name** box, type **Create Index**. Then replace the current query code with the following code (you can copy and paste this from **Index.txt** in the C:\HDILabs\Lab02B folder):

```
set hive.execution.engine=tez;

CREATE INDEX idx_month ON TABLE indexedlog(log_month)
AS 'COMPACT'
WITH DEFERRED REBUILD;
```

3. Review the code, noting that it creates a compact index on the **indexedlog** table.
4. Click **Submit**, and wait for the query to complete.
5. When the query as completed, in the HDInsight query console, view the **File Browser** page, and navigate to the ***storage_account/container/*hive/warehouse** folder. Note that this folder is the default folder for Hive, and contains a folder for the **hivesampletable** table. In addition, a folder named **default_indexedlog_idx_month_** has been created for the index.
6. View the contents of the **default_indexedlog_idx_month_** folder, and note that it is empty.

Rebuild an Index

1. In the HDInsight query console, view the **Hive Editor** page.
2. In the **Query Name** box, type **Rebuild Index**. Then replace the current query code with the following code (you can copy and paste this from **Rebuild Index.txt** in the C:\HDILabs\Lab02B folder):

```
set hive.execution.engine=tez;

ALTER INDEX idx_month ON indexedlog REBUILD;
```

3. Review the code, noting that it rebuilds the **idx_month** index on the **indexedlog** table.
4. Click **Submit**, and wait for the query to complete.
5. When the query as completed, in the HDInsight query console, view the **File Browser** page, and navigate to the ***storage_account/container/*hive/warehouse/default_indexedlog_idx_month_** folder. Note that this folder now contains index files. These files contain the addresses of the storage blocks in the table data files that contain the indexed value. This enables Hive to quickly locate the rows requested by queries that filter on indexed values that are stored next to one another. For example, the following query filters on the indexed **log_month** column:

```
SELECT sum(cs_bytes)
FROM indexedlog
WHERE log_month=6;
```

Using a View

Hive supports the creation of views to abstract complex queries.

Create a View

1. In the HDInsight query console, view the **Hive Editor** page.
2. In the **Query Name** box, type **Create View**. Then replace the current query code with the following code (you can copy and paste this from **View.txt** in the C:\HDILabs\Lab02B folder):

```
set hive.execution.engine=tez;

CREATE VIEW vDailySummary
AS
SELECT log_date,
       COUNT(*) AS requests,
       SUM(cs_bytes) AS inbound_bytes,
       SUM(sc_bytes) AS outbound_bytes
FROM rawlog
WHERE SUBSTR(log_date, 1, 1) <> '#'
GROUP BY log_date;
```

3. Review the code, noting that it creates a view named **vDailySummary** that retrieves data from the **rawlog** table.
4. Click **Submit**, and wait for the query to complete.

Query a View

1. In the HDInsight query console, view the **Hive Editor** page.
2. In the **Query Name** box, type **Query View**. Then replace the current query code with the following code (you can copy and paste this from **Query View.txt** in the C:\HDILabs\Lab02B folder):

```
set hive.execution.engine=tez;

SELECT * FROM vDailySummary;
```

3. Review the code, noting that it retrieves all rows and columns from the **vDailySummary** view.
4. Click **Submit**, and wait for the query to complete.
5. When the job completes, click its name in the **Query Name** column of the **Job Session** table to open a new tab containing the output generated by the job, and view the output returned by the query.
6. Close the web browser.

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 Azure portal, click the **HDInsight** tab.
2. Select the row containing your HDInsight cluster, and then at the bottom of the page, click **Delete**. When prompted to confirm the deletion, click **Yes**.

3. Wait for your cluster to be deleted, and then click the **Storage** tab, and if necessary refresh the browser to view the storage account that was created with your cluster.
4. Select the row containing the storage account, and then at the bottom of the page, click **Delete**. When prompted to confirm the deletion, enter the storage account name and click **OK**.
5. Close the browser.