Lab 4.1: Data Formats

Overview

In this lab, we will read/write data in several formats

Builds on

None

Run time

15-20 minutes

Data for this lab

We supply several data files that you'll use in this lab and later labs. All data files are in the ***spark-labs/data*** folder. You'll need to include this in the path when loading - e.g. *spark-labs/data/people.json*

- ***people.json***: Small simple data file with name/gender/age info.
- ***twinkle/*.data***: e.g. *twinkle/100M.data*. Simple text files of varying sizes based on "Twinkle twinkle" nursery rhyme.
- ***github.json***: Archive containing github activity for a single day
- ***wiki-pageviews.txt***: Wikimedia page view data for one day

We will be loading these files into Spark, and save them in varying formats. In later labs, we'll examing the DataFrames that are created, and use them for transformations.

Load Text and JSON Data

Tasks

- Go to your Zeppelin shell
- In the shell, use the `SparkSession` and `DataFrameReader` to load data from the following two files
    - *people.json* as JSON data
    - *wiki-pageviews.txt* as text data Store
    - the resulting data in a var, e.g.

```
// Scala
> val folksDF = // Load people.json here
> val viewsDF = // Load wiki-pagesviews.txt here
```

```
# Python
> folksDF = // Load people.json here
> viewsDF = // Load wiki-pagesviews.txt here
```

- Look at the Web UI - Jobs pages after each load.
    - You'll see that the JSON load happens immediately, but there will be no job for the text load.
    - This is because Spark will scan the JSON file to infer the schema - more on this soon.
- Call count on `viewsDF` , as shown below, to get the count of items in it.
    - Look at the Web UI - Jobs page again (refresh it).
    - You'll see there is a job showing now - it needs to actually execute once an action is invoked.
    - You can click on the job to get the job detail - it will show how much data was read.
- ```
  > viewsDF.count()
  ```
  View some of the data in each resulting dataframe.
    - Don't view all the data - there may be many megabytes.
    - Just look at a few lines using the following code to limit the amount of data seen..

```
> folksDF.limit(5).show()
> viewsDF.limit(5).show()
```

Read More Complex Data

Tasks

- In the Spark Shell, load data from *github.json*.
    - As before, store the result in a var (e.g. `githubDF`) for ease of use. Look
    - at some of the data there.
```

- Just look at a few rows (use `limit()`) - there is a lot here also.

Write Data

Tasks

- Write out the data in `folksDF` in parquet format (to *spark- labs/output/folks.parquet*), then view it on the file system.
  - It will be in a separate folder under the *output* folder.
  - With the data files below that folder.
  - Use the Hadpoop commands below to view the output files.
  - Note that we've trunacted the file names of the part- files (e.g. 87ac---) to make it easier to view

```
// Write the data ...
// View it on file system
> %sh
> hdfs dfs -ls spark-labs/output
> hdfs dfs -ls spark-labs/output/folks.parquet

Found 2 items
-rw-r--r--   1 zeppelin hdfs         44 2017-09-28 17:00 spark-
labs/output/ReadMe.txt
drwxr-xr-x   - zeppelin hdfs          0 2017-09-28 17:23 spark-
labs/output/folks.parquet
Found 2 items
-rw-r--r--   1 zeppelin hdfs          0 2017-09-28 17:23 spark-
labs/output/folks.parquet/_SUCCESS
-rw-r--r--   1 zeppelin hdfs        773 2017-09-28 17:23 spark-
labs/output/folks.parquet/part-00000-87ac---.snappy.parquet
```

Note: Remember that writing from Spark creates a **folder** with the name you provide for output (e.g. *folks.parquet*). Inside the folder there will be a file for each partition where data resides (named something similar to *part-00000-87acbfb2-99c3-4d05- bbeb-c5ed295fa8f9.snappy.parquet*). snappy is a compresssion format, that is used by default when saving in parquet format. Saving as csv behaves similarly (but with no snappy compression).

Optional Tasks

- Write out the data in `folksDF` in CSV format (to *spark-labs/output/folks.csv*), as

described below:

- Call `coalesce(1)` on the dataframe before writing.
  - This will gather all the data on one partition, saving the data in a single file.
  - Otherwise, there will be one file per partition.
- Include headers in the output, by calling `option("header", true)` on the `DataFrameWriter` before the call to write it out in csv format.

- View your output, using Hadoop commands as shown below
  - Note that your filename will be different - use the filenames as they appear on your filesystem (seen as the results of `hdfs dfs -ls`)

```
// Write the data ...
// View it on file system
>  %sh
> hdfs dfs -ls spark-labs/output
> hdfs dfs -ls spark-labs/output/folks.csv

Found 3 items
-rw-r--r--   1 zeppelin hdfs         44 2017-09-28 17:00 spark-
labs/output/ReadMe.txt
drwxr-xr-x   - zeppelin hdfs          0 2017-09-28 17:23 spark-
labs/output/folks.csv
drwxr-xr-x   - zeppelin hdfs          0 2017-09-28 17:23 spark-
labs/output/folks.parquet
Found 2 items
-rw-r--r--   1 zeppelin hdfs          0 2017-09-28 17:23 spark-
labs/output/folks.csv/_SUCCESS
-rw-r--r--   1 zeppelin hdfs         55 2017-09-28 17:23 spark-
labs/output/folks.csv/part-00000-23be---.csv
```

```
// View one of the parts by "cat"ing the file
> %sh
> hdfs dfs -cat spark-labs/output/folks.csv/part-00000-23bef8ca-ef78-
4719-b7bd-51639c1b5bf8.csv

age,gender,name
35,M,John
40,F,Jane
20,M,Mike
52,F,Sue
```

More Optional Tasks

- Write out the data in `githubDF` in parquet format (to *spark- labs/output/github.parquet*)
  - Does it work? On our system we got an `OutOfMemoryError` for the Java heap.
    - Using the spark shell in local mode with one thread
  - Why? Because it's a big file, and processing it all on one partition just uses too much memory.
  - It should work on your system - list the contents of the *github.parguet* folder that was created.
  - How many parts do you see?

```
> %sh
> hdfs dfs -ls spark-labs/output/github.parquet
>
Found 3 items
-rw-r--r--   1 zeppelin hdfs          0 2017-10-03 16:05 spark-
labs/output/github.parquet/_SUCCESS
-rw-r--r--   1 zeppelin hdfs    5232585 2017-10-03 16:05 spark-
labs/output/github.parquet/part-00000-b885----.snappy.parquet
-rw-r--r--   1 zeppelin hdfs    5785423 2017-10-03 16:05 spark-
labs/output/github.parquet/part-00001-b885----.snappy.parquet
```

Summary

It's fairly easy to load and write data in varying formats. However, with complex data, the format can be complex. We'll be working more with this data, and will show ways of simplifying the portions we work with.