DO A PRACTICAL EXPERIMENT ON BIG DATA

CLASS-VIII

Instructor: Palash Gupta

Re-Cap

- Accessing HDFS using Python
- Loading HDFS Files in Spark
- Running Basic Transformation and Action in Spark
- Understanding Spark Submit Details

What are we cover today?

- Revisit Transformation and Actions
- Understand a real problem
- Design & Implement the Solution
- Understand the Final Assignment
- Q&A and Ending Speech

Revisit Transformation and Action

We will be using following simple file content to analyze multiple transformations and actions scenarios.

```
[hadoop@bdrenfdludcf01 ~]$ cat /home/hadoop/a.txt
This is our second file what we are going to put into our HDFS
This will be just starting
Python file operation is interesting and we are going to start soon
```

Revisit Transformation and Action Contd.

Word Count

```
>>> counts = sc.textFile("/home/hadoop/a.txt").flatMap(lambda line: line.split(" ")).map(lambda word: (word, 1)).reduceByKey(lambda a, b: a + b)
>>> counts.collect()
[('soon', 1), ('start', 1), ('HDFS', 1), ('just', 1), ('our', 2), ('Python', 1), ('going', 2), ('operation', 1), ('is', 2), ('file', 2), ('will', 1), ('what', 1), ('are', 2), ('to', 2), ('be', 1), ('into', 1), ('interesting', 1), ('we', 2), ('and', 1), ('This', 2), ('second', 1), ('put', 1), ('starting', 1)]
```

FlatMap Function

```
>>> counts = sc.textFile("/home/hadoop/a.txt").flatMap(lambda line: line.split(" "))
>>> counts.collect()
['This', 'is', 'our', 'second', 'file', 'what', 'we', 'are', 'going', 'to', 'put', 'into', 'our', 'HDFS', 'This', 'will', 'be', 'just', 'starting', 'Python', 'file', 'operation', 'is', 'interesting', 'and', 'we', 'are', 'going', 'to', 'start', 'soon']
```

Revisit Transformation and Action Contd.

Map Function

```
>>> counts = sc.textFile("/home/hadoop/a.txt").flatMap(lambda line: line.split(" ")).map(lambda word: (word, 1))
>>> counts.collect()
[('This', 1), ('is', 1), ('our', 1), ('second', 1), ('file', 1), ('what', 1), ('we', 1), ('are', 1), ('going', 1), ('to', 1), ('put', 1), ('into', 1), ('our', 1), ('HDFS', 1), ('This', 1), ('will', 1), ('be', 1), ('just', 1), ('starting', 1), ('Python', 1), ('file', 1), ('operation', 1), ('is', 1), ('interesting', 1), ('and', 1), ('we', 1), ('are', 1), ('going', 1), ('to', 1), ('start', 1), ('soon', 1)]
```

ReduceByKey

```
>>> counts = sc.textFile("/home/hadoop/a.txt").flatMap(lambda line: line.split(" ")).map(lambda word: (word, 1)).reduceByKey(lambda a, b: a + b)
>>> counts.collect()
[('soon', 1), ('start', 1), ('HDFS', 1), ('just', 1), ('our', 2), ('Python', 1), ('going', 2), ('operation', 1), ('is', 2), ('file', 2), ('will', 1), ('what', 1), ('are', 2), ('to', 2), ('be', 1), ('into', 1), ('interesting', 1), ('we', 2), ('and', 1), ('This', 2), ('second', 1), ('put', 1), ('starting', 1)]
```

Revisit Transformation and Action Contd.

Let us simulate some different problem statements:

Number of words per first letter wise

```
>>> counts = sc.textFile("/home/hadoop/a.txt").flatMap(lambda line: line.split(" ")).map(lambda word: (word[0], word)).groupByKey().mapValues(lambda word: len(word))
>>> counts.collect()
[('o', 3), ('t', 2), ('T', 2), ('P', 1), ('j', 1), ('g', 2), ('H', 1), ('w', 4), ('i', 4), ('s', 4), ('p', 1), ('b', 1), ('f', 2), ('a', 3)]
```

What happened if we do reduceByKey

```
>>> counts = sc.textFile("/home/hadoop/a.txt").flatMap(lambda line: line.split(" ")).map(lambda word: (word[0], word)).reduceByKey(lambda a, b: a + b)
>>> counts.collect()
[('o', 'ourouroperation'), ('t', 'toto'), ('T', 'ThisThis'), ('P', 'Python'), ('j', 'just'), ('g', 'goinggoing'), ('H', 'HDFS'), ('w', 'whatwewillwe'), ('i', 'isintoisinteresting'), ('s', 'startsoonsecondstarting'), ('p', 'put'), ('b', 'be'), ('f', 'filefile'), ('a', 'andareare')]
```

Revisit Transformation and Action Contd.

Map Function

```
>>> counts=sc.textFile("/home/hadoop/a.txt").flatMap(lambda line: line.split(" ")).map(lambda word: (word[0], word))
>>> counts.collect()
[('T', 'This'), ('i', 'is'), ('o', 'our'), ('s', 'second'), ('f', 'file'), ('w', 'what'), ('w', 'we'), ('a', 'are'), ('g', 'going'), ('t', 'to'), ('p', 'put'), ('i', 'into'), ('o', 'our'), ('H', 'HDFS'), ('T', 'This'), ('w', 'will'), ('b', 'be'), ('j', 'just'), ('s', 'starting'), ('P', 'Python'), ('f', 'file'), ('o', 'operation'), ('i', 'is'), ('i', 'interesting'), ('a', 'and'), ('w', 'we'), ('a', 'are'), ('g', 'going'), ('t', 'to'), ('s', 'start'), ('s', 'soon')]
```

MapValues Function

```
>>> counts = sc.textFile("/home/hadoop/a.txt").flatMap(lambda line: line.split(" ")).map(lambda word: (word[0], word)).groupByKey().mapValues(lambda word: len(word))
>>> counts.collect()
[('o', 3), ('t', 2), ('T', 2), ('P', 1), ('j', 1), ('g', 2), ('H', 1), ('w', 4), ('i', 4), ('s', 4), ('p', 1), ('b', 1), ('f', 2), ('a', 3)]
```

Revisit Transformation and Action Contd.

Number of unique words per first letter wise

```
>>> counts = sc.textFile("/home/hadoop/a.txt").flatMap(lambda line: line.split(" ")).map(lambda word: (word[0], word)).groupByKey().mapValues(lambda word: len(set(word)))
>>> counts.collect()
[('o', 2), ('t', 1), ('T', 1), ('P', 1), ('j', 1), ('g', 1), ('H', 1), ('w', 3), ('i', 3), ('s', 4), ('p', 1), ('b', 1), ('f', 1), ('a', 2)]
```

Is it efficient method? NO

Number of unique words per first letter wise – Efficient Method

```
>>> counts = sc.textFile("/home/hadoop/a.txt").repartition(6).flatMap(lambda line: line.split(" ")).distinct().map(lambda word: (word[0], word)).groupByKey().mapValues(lambda word: len(word))
>>> counts.collect()
[('o', 2), ('g', 1), ('i', 3), ('H', 1), ('b', 1), ('t', 1), ('w', 3), ('T', 1), ('P', 1), ('j', 1), ('p', 1), ('f', 1), ('s', 4), ('a', 2)]
```

Understanding a Real Life Problem

Problem Description:

Lincoln Labs set up an environment to acquire nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment but peppered it with multiple attacks.

All the raw responses are kept in file named **corrected**

Detail schema is given in **Columns.xlsx**

The following analytics shall be done:

1. Select tcp network interactions with more than 1 second duration and no transfer from destination
   Note: The output column can be "duration | dst_bytes"

1. Find protocol type wise network interaction count
   Note: The output column can be "protocol_type | count"

1. Count how many interactions last more than 1 second, with no data transfer from destination, grouped by protocol type. Output can be "protocol_type | count"

1. Count them by label and protocol type. Output can be "label | protocol_type | count"

Note that: If label is not normal then it is considered as attack

Schema:

Microsoft Excel Worksheet

Real Data Set

Check File URL

Instruction to be followed:

1. Raw Capture file shall be loaded in HDFS /RAW/ directory daily from external program

1. Process all the outputs in Apache Spark in standalone cluster mode

1. Store all the results in HDFS /RAW/ directory in CSV format

1. Schedule the spark job to run every day at 03:00PM

Design & Implement the Solution

Designing & Implementing:

- Inspect the real data source – Identify payload type
- Identify required data type and unit
- Identify required field
- Understand input and output format and location
- Confirm the way Spark has to run if specified
- Plan the packages you would need to use
- Plan for the resources you might need as per your cluster configuration
- Plan the efficient data flow by applying required repartition, sequence of transformation & action and keeping only required data set in actual execution. Better to use pyspark in this stage of analysis.
- Do Development
- Perform Testing either using pyspark or local mode
- Prepare Shell script or other method to be used in scheduler (if specified)
- Configure Scheduler and ensure required dependencies are running (HDFS, Spark etc.)

Understanding Final Assignment

Problem Description:

We all know that Covid-19 pandemic is going on world-wide and almost every corner of the world is impacted globally. You are given with a Covid-19 dataset collected from the internet to do certain analysis.

All the raw responses are kept in file named
Covid_Analysis_DataSet.csv

Detail schema is self-explanatory.

The following analytics shall be done:

1. Month, Year and countryterritoryCode wise, please derive Infection Rate and Death Rate.

   Note: The output format shall be shown as

   Month;Year;CountryCode;InfectionRate;DeathRate
   September;2020; AFG;3;10
   October;2020; AFG;5;12

   Infection Rate = (cases / TestPerformed) * 100%
   Death Rate = (deaths / cases) * 100%

Instruction to be followed:

1. Consider Raw Capture file shall be loaded in HDFS /**ASSIGNMENT** directory hourly from external program. So, you will get the mentioned file in the designed location periodically.

1. Process all the outputs in Apache Spark in Standalone Cluster Mode

1. Store all the results in HDFS /**OUTPUT** in mentioned CSV format (; separated file)

1. Schedule the spark job to run in every hour at 15 minutes

Real Data Set

Check File URL

Thank You to All

Connect with linkedin

https://www.linkedin.com/in/enggpalashgupta/

E-mail

spline_palash@yahoo.com

Yet any Query

http://forum.fountainit.com.bd/

QUESTION & ANSWER

THANKS FOR ATTENDING THE CLASS & YOUR CO-OPERATION

References

- https://spark.apache.org/docs/2.3.0/sql-programming-guide.html
- https://spark.apache.org/docs/latest/configuration.html
- https://sparkbyexamples.com/
- https://spark.apache.org/docs/2.1.0/api/python/pyspark.html
- https://www.nodalpoint.com/spark-data-frames-from-csv-files-handling-headers-column-types/
- https://stackoverflow.com/questions/36608559/how-to-assign-and-use-column-headers-in-spark
- https://jaceklaskowski.gitbooks.io/