C S 479/579 Big Data

Tutorial 2: Spark

Due: Feb 14, 11:55pm

1. Question 1: Running Spark in Colab Environment

Google Colaboratory (https://colab.research.google.com/), or "Colab" for short is a Google free cloud service. It allows you to write and execute Python in your browser with zero configuration required, free access to GPUs, and easy sharing. In this section, you will learn how to use Spark in Colab environment.

- Use Google Chrome Browser, open this Colab Notebook https://colab.research.google.com/drive/1eYQfi69m67VX8wMu5iO9IPM1FV5cqev4
- Follow the steps on that Colab Notebook. **After finishing, you need to submit the output.zip**.

There are two things to note here:

- The above Colab shows one way to run your Spark application. That is by using Spark Shell. Spark Shell lets you interactively compose and run your application. The other way is that you can write your application using an IDE and then submit your application to Spark. You will learn how to do this in Question 2.
- Spark can refer to a dataset in an external storage system, such as a shared filesystem, HDFS, HBase, or any data source offering a Hadoop InputFormat. The WordCount application in Colab does not use HDFS from Hadoop. See Question 2 for how to tell Spark to refer to a distributed dataset in HDFS.

2. Question 2: Running Spark in the Cloudera QuickStart VM

Follow the steps:

- Open Terminal
- You can skip this step if you want. We can use Spark Shell in Terminal to perform again the steps in the Colab. Use this command to open the Spark Shell:

```
$ pyspark
```

- Put input files to HDFS

```
$ hadoop fs -mkdir /user/cloudera/tutorial2/ /user/cloudera/tutorial2/input

$ hadoop fs -put OnlineRetail.csv /user/cloudera/tutorial2/input

$ hadoop fs -put pg100.txt /user/cloudera/tutorial2/input
```

- Assuming you use an IDE to write the following WordCount program and store it a textfile called `wc.py`:

```
import sys
from pyspark import SparkContext, SparkConf
if __name__ == "__main__":
        # create Spark context with necessary configuration
        sc = SparkContext("local","PySpark WordCount")
        # read data from text file and split each line into words
        words = sc.textFile("hdfs://localhost/user/cloudera/tutorial2/input/pg100.txt")
                .flatMap(lambda line: line.split(" "))
```

```
# count the occurrence of each word
wordCounts = words.map(lambda word: (word, 1)).reduceByKey(lambda a,b:a +b)

# save the counts to output
wordCounts.saveAsTextFile("hdfs://localhost/user/cloudera/tutorial2/output/")
```

The program implements MapReduce using Spark. Look at the bold text where Spark refers to a file in HDFS.

- Submit the program

  ```
  $ spark-submit wc.py
  ```

- View the ouput in HDFS

  ```
  $ hadoop fs -ls tutorial2/output
  ```

- **Save the output to `submit_t2.txt` and submit this file:**

  ```
  $ hadoop fs -cat /user/cloudera/tutorial2/output/part\* > submit_t2.txt
  ```

3. Question 3: Running Spark on CS Servers

You can perform again the steps above on CS servers. It should work for both `spark-shell` and `spark-submit` with some minor adaptations. You can skip this question if you want.