DS/CMPSC 410 Sparing 2021

Instructor: Professor John Yen

TA: Rupesh Prajapati and Dongkuan Xu

Lab 7 Part A: Data Partitioning

The goals of this lab are for you to be able to

- Use data partitioning to improve scalability

- Make decisions regarding types of RDD join

- Choose transformations and actions that can benefit, or be benefited by, data partitioning

- Apply the obove to find top k hashtags for tweets after Boston Marathon Bombing.

This lab includes two data sets, each one is a set of tweets related to Boston Marathon Bombing in 2013.

- The first dataset contains Boston Marathon Bombing collected on 4/17/2013.
- The second dataset contains Boston Marathon Bombing collected on 4/25/2013.
- Your mission, should you decide to accept, is to first find hashtags that occur more than 100 times in 4/17/2013 tweets. Then, calculate the total counts of these hashtags (considering both days). You should save these hashtags together with their counts in a text file.
- The Jupyter Notebook below provides more detailed instructions.

Lab 7 Part A: Complete 6 Exercises in this Jupyter Notebook.

Lab 7 Part B: Modify Lab 7A for Data-Partitioning and Spark-submit

1. You choose the number of pre-partitioning the data.
2. Modify your code in Lab 7A to take advantage of data partitioning. Export it as Lab7B.py
3. Modify your code in Lab7B.py to read data files from `/ocean/project/asc200006p/shared/<filenames>` and write your output in your home directory in the project folder (similar to what you did for Lab 5 and 6).
4. Submit Lab 7B to Bridges2 cluster to obtain its run-time information.

Submit the following items for Lab 7

- Completed Jupyter Notebook of Lab 7

- Lab7B.py (used for spark-submit)
- Log file for spark-submit
- Output file generated by Lab7A or Lab7B.

Total Number of Exercises: 6+1

- Exercise 1: 5 points
- Exercise 2: 5 points
- Exercise 3: 5 points
- Exercise 4: 10 points
- Exercise 5: 10 points
- Exercise 6: 5 points
- Part 7B: 30 points ## Total Points: 70 points

Due: midnight, February 8, 2021

The first thing we need to do in each Jupyter Notebook running pyspark is to import pyspark first.

In [43]:
```
import pyspark
```

Once we import pyspark, we need to import an important object called "SparkContext". Every spark program needs a SparkContext object.

This lab will use RDD, not DataFrame, because data partitioning applies only to RDD.

In [44]:
```
from pyspark import SparkConf
from pyspark import SparkContext
```

We then create a Spark Context variable. Once we have a spark context variable, we can execute spark codes.

In [45]:
```
conf = SparkConf()
sc=SparkContext(conf=conf)
sc
```

Out[45]: **SparkContext**

[Spark UI (http://comp-bc-0284.acib.production.int.aci.ics.psu.edu:4040)](http://comp-bc-0284.acib.production.int.aci.ics.psu.edu:4040)

Version
`v3.0.1`
Master
`local[*]`
AppName
`pyspark-shell`

Exercise 7.1 (5 points)

(a) Add your name below.
(b) replace the path below with the path of your home directory.

Answer for Exercise 1

(a) Your Name: Kangdong Yuan

In [46]:
```
text_RDD = sc.textFile("/storage/home/kky5082/ds410/lab7/BostonMarathn4_17_201
3.csv")
text_RDD
```

Out[46]:
```
/storage/home/kky5082/ds410/lab7/BostonMarathn4_17_2013.csv MapPartitionsRDD[1]
at textFile at NativeMethodAccessorImpl.java:0
```

Exercise 7.2 (5 points)

Complete the following code for parsing the text file into tokens. The file contains multiple fields, separated by ",". So, you need to first parse each line into its fields. Then, you can extract tokens from each field.

In [47]:
```
field_RDD = text_RDD.flatMap(lambda line: line.strip().split(","))
field_RDD
```

Out[47]:
```
PythonRDD[2] at RDD at PythonRDD.scala:53
```

```
In [48]: field_RDD.take(10)

Out[48]: ['3.24311E+17',
          'corena_danny',
          '4/17/2013 0:00',
          'RT @JCrossover: Prayers up for Boston..',
          '',
          '3.24311E+17',
          'trainmilehigh',
          '4/17/2013 0:00',
          'New York Yankees Put Boston Rivalry Aside at Tonight?��?��?�s Game http://t.c
         o/nU7rJmSfSQ (via @ABC)',
          '']
```

```
In [49]: token_RDD = field_RDD.flatMap(lambda field: field.split(" "))
```

```
In [50]: token_RDD.take(10)

Out[50]: ['3.24311E+17',
          'corena_danny',
          '4/17/2013',
          '0:00',
          'RT',
          '@JCrossover:',
          'Prayers',
          'up',
          'for',
          'Boston..']
```

Filtering an RDD

The syntax for filter (one type of data trasnformation in spark) is

RDD.filter(lambda parameter : condition)

Notice the syntax is not what is described in p. 38 of the textbook.

The result of filtering the input RDD is the collection of all elements that pass the filter condition (i.e., returns True when the filtering condition is applied to the parameter.

For example, the filtering condition in the pyspark conde below checks whether each element of the input RDD (i.e., token_RDD) starts with the character "#", using Python startswith() method for string.

```
In [51]: hashtag_RDD = token_RDD.filter(lambda token : token.startswith("#"))
         hashtag_RDD

Out[51]: PythonRDD[5] at RDD at PythonRDD.scala:53
```

```
In [52]: hashtag_RDD.take(3)

Out[52]: ['#PrayForBoston', '#Respect?��?�?�', '#PrayForBoston']
```

```
In [53]: hashtag_count_RDD = hashtag_RDD.map(lambda hashtag: (hashtag, 1))
         hashtag_count_RDD

Out[53]: PythonRDD[7] at RDD at PythonRDD.scala:53
```

```
In [54]: hashtag_count_RDD.take(3)

Out[54]: [('#PrayForBoston', 1), ('#Respect?��?�?�', 1), ('#PrayForBoston', 1)]
```

Exercise 7.3 (5 points)

Complete the following code to find out the total number of hashtags that appear in Boston Bombing tweets on 4/17/2021.

Note: You should comment out this line in Lab7B (for spark-submit to cluster), like you comment out all of unnecessary actions in spark-submit mode (e.g., take()).

```
In [55]: hashtag_count_RDD.count()

Out[55]: 494461
```

Note: Consider to change the number of partition below for Lab7B spark-submit mode.

```
In [79]: hashtag_total_RDD = hashtag_count_RDD.reduceByKey(lambda a, b: a + b, 1)
         hashtag_total_RDD

Out[79]: PythonRDD[53] at RDD at PythonRDD.scala:53
```

```
In [101]: total_hashtag_RDD = hashtag_total_RDD.map(lambda x: (x[0], x[1]))
          total_hashtag_RDD.take(2)

Out[101]: [('#PrayForBoston', 20757), ('#Respect?��?�?�', 4)]
```

```
In [102]: top_hashtag_day1_RDD = total_hashtag_RDD.filter(lambda x: x[1] > 100)
In [103]: top_hashtag_day1_RDD.take(10)
```

```
Out[103]: [('#PrayForBoston', 20757),
           ('#Boston', 58467),
           ('#prayforboston', 45593),
           ('#tcot', 5177),
           ('#TeaParty', 117),
           ('#p2', 2206),
           ('#boston', 13908),
           ('#BU', 108),
           ('#staystrong', 223),
           ('#Boston.', 3440)]
```

Exercise 7.4 (10 points)

Complete the code below to

- read the twitter data collected on 4/25/2013
- convert it into fields
- convert it into tokens ### Note: Make sure your variable names for RDDs do not conflict with variable names used earlier for processing the first twitter dataset.

```
In [59]: text3_RDD = sc.textFile("/storage/home/kky5082/ds410/lab7/BostonMarathon4_25_20
         13.csv")
         text3_RDD
```

```
Out[59]: /storage/home/kky5082/ds410/lab7/BostonMarathon4_25_2013.csv MapPartitionsRDD[1
         7] at textFile at NativeMethodAccessorImpl.java:0
```

```
In [60]: field3_RDD = text3_RDD.flatMap(lambda line: line.strip().split(","))
```

```
In [61]: token3_RDD = field3_RDD.flatMap(lambda field: field.split(" "))
         token3_RDD
```

```
Out[61]: PythonRDD[18] at RDD at PythonRDD.scala:53
```

```
In [62]: hashtag3_RDD = token3_RDD.filter(lambda token : token.startswith("#"))
         hashtag3_RDD.persist()
```

```
Out[62]: PythonRDD[19] at RDD at PythonRDD.scala:53
```

```
In [63]: hashtag_count3_RDD = hashtag3_RDD.map(lambda hashtag: (hashtag, 1))
         hashtag_count3_RDD.persist()
```

```
Out[63]: PythonRDD[20] at RDD at PythonRDD.scala:53
```

```
In [64]: hashtag_total3_RDD = hashtag_count3_RDD.reduceByKey(lambda a, b: a + b, 1)
         hashtag_total3_RDD.top(2)
```

```
Out[64]: [('#zzzzzz', 1), ('#zyadtihaiyaar', 1)]
```

```
In [65]: hashtag_total3_RDD.take(2)
```

```
Out[65]: [('#tcot???�#tlot???�#YAL', 1), ('#tcot', 3259)]
```

In [85]:
```
combined_hashtag_RDD = top_hashtag_day1_RDD.leftOuterJoin(hashtag_total3_RDD)
combined_hashtag_RDD.persist()
```

Out[85]:
```
PythonRDD[64] at RDD at PythonRDD.scala:53
```

In [86]:
```
combined_hashtag_RDD.take(4)
```

Out[86]:
```
[('#prayforboston', (45593, 809)),
 ('#p2', (2206, 929)),
 ('#boston', (13908, 3591)),
 ('#BU', (108, 17))]
```

In [91]:
```
def tran_none(x):
    if (x==None) :
        return(0)
    else:
        return(x)
```

In [92]:
```
tran_none(None)
```

Exercise 7.5 (10 points)

Change the following code to use **mapValues**, instead of map. Notice the parameter for the lambda function in mapValues is ONLY the value portion of the key value pairs. For example, the value in the key-value pairs of combined_hashtag_RDD is (`count1` , `count2`) where `count1` is the count of a hashtag (key) from the first twitter dataset, and `count2` is the count of the hashtag from the second twitter dataset.

In [95]:
```
# This statement needs to be MODIFIED for Exercise 7.5
total_hashtag_RDD = combined_hashtag_RDD.map(lambda x: (x[0], x[1][0]+tran_none
(x[1][1])))
```

In [96]:
```
total_hashtag_RDD.take(2)
```

Out[96]:
```
[('#prayforboston', 46402), ('#p2', 3135)]
```

In [113]:
```
top_hashtag_day1day2_RDD = total_hashtag_RDD.filter(lambda x: x[1] > 100)
```

In [114]:
```
top_hashtag_day1day2_RDD.take(2)
```

Out[114]:
```
[('#PrayForBoston', 20757), ('#Boston', 58467)]
```

Exercise 7.6 (5 points)

Modify the path below so that you can save sorted hashtags and their counts in a directory.

In [115]:
```
output_path = "/storage/home/kky5082/ds410/lab7/Lab7_Output"
top_hashtag_day1day2_RDD.saveAsTextFile(output_path)
```

The code above should be replaced by the code here (similar to Lab 5 and 6) for Lab 7B

```
 import os
projectPath=os.environ.get('PROJECT')
output_path = "%s/Lab7Output"%projectPath
top_hashtag_day1day2_RDD.saveAsTextFile(output_path)
```

In [ ]:
```
# top_hashtag_list = total_hashtag_RDD.takeOrdered(100, key= lambda x: -x[1])
```

In [ ]:
```
# print(top_hashtag_list)
```

In [ ]:
```
# The following way to print output file works in the local mode, but not in a
 cluster mode.
# So, this is useful for saving exploratory analytics results for a small/initi
al dataset.
#
#import sys
# with open("/storage/home/juy1/Lab7/Lab7TopHashtags.txt", "w") as f:
#    print(top_hashtag_list, file=f)
```

In [42]:
```
# We need to stop Spark Context, though this is not needed for Spark Session.
sc.stop()
```

In [ ]:
```
```