Session 18: RDD'S CONTD. & INTRODUCTION TO DATAFRAMES

Assignment 18.1

Student Name: Abarajithan SA

Course: Big Data Hadoop & Spark Training

Start Date: 2017-09-09

End Date: 2017-11-26

Assignment 18.1– Introduction to Dataframes.

Contents

Introduction

In this assignment, we are going to perform some basic Spark RDD operation and see introduction to dataframes.

Problem Statement

1. What is the distribution of the total number of air-travelers per year?
2. What is the total air distance covered by each user per year?
3. Which user has travelled the largest distance till date?
4. What is the most preferred destination for all users?

Dataset – the dataset is available in the below link,

https://drive.google.com/drive/folders/0B_P3pWagdIrrVThBaUdVSUtzbms

Dataset used,

S18_Dataset_Holidays.txt,

```
[acadgild@localhost hadoop]$ cat S18_Dataset_Holidays.txt
1,CHN,IND,airplane,200,1990
2,IND,CHN,airplane,200,1991
3,IND,CHN,airplane,200,1992
4,RUS,IND,airplane,200,1990
5,CHN,RUS,airplane,200,1992
6,AUS,PAK,airplane,200,1991
7,RUS,AUS,airplane,200,1990
8,IND,RUS,airplane,200,1991
9,CHN,RUS,airplane,200,1992
10,AUS,CHN,airplane,200,1993
1,AUS,CHN,airplane,200,1993
2,CHN,IND,airplane,200,1993
3,CHN,IND,airplane,200,1993
4,IND,AUS,airplane,200,1991
5,AUS,IND,airplane,200,1992
6,RUS,CHN,airplane,200,1993
7,CHN,RUS,airplane,200,1990
8,AUS,CHN,airplane,200,1990
9,IND,AUS,airplane,200,1991
10,RUS,CHN,airplane,200,1992
1,PAK,IND,airplane,200,1993
2,IND,RUS,airplane,200,1991
3,CHN,PAK,airplane,200,1991
4,CHN,PAK,airplane,200,1990
5,IND,PAK,airplane,200,1991
6,PAK,RUS,airplane,200,1991
7,CHN,IND,airplane,200,1990
8,RUS,IND,airplane,200,1992
9,RUS,IND,airplane,200,1992
10,CHN,AUS,airplane,200,1990
1,PAK,AUS,airplane,200,1993
```

The above dataset has the data column wise, userID, Destination, arrival, travel mode, travel distance and the year,

We are loading the dataset into the spark context,

val baseRDD = sc.textFile("/home/acadgild/hadoop/S18_Dataset_Holidays.***txt")***

import org.apache.spark.storage.StorageLevel

We are caching the “baseRDD” using the persist(StorageLevel.MEMORY_ONLY)

baseRDD.persist(StorageLevel.MEMORY_ONLY)

```
scala> val baseRDD = sc.textFile("/home/acadgild/hadoop/S18_Dataset_Holidays.txt")
baseRDD: org.apache.spark.rdd.RDD[String] = /home/acadgild/hadoop/S18_Dataset_Holidays.txt MapPartitionsRDD[22] at textFile at <console>:24

scala> import org.apache.spark.storage.StorageLevel
import org.apache.spark.storage.StorageLevel

scala> baseRDD.persist(StorageLevel.MEMORY_ONLY)
res11: baseRDD.type = /home/acadgild/hadoop/S18_Dataset_Holidays.txt MapPartitionsRDD[22] at textFile at <console>:24
```

Task 1 - What is the distribution of the total number of air-travelers per year?

Codes used to achieve the above,

1. ***val baseRDD1 = baseRDD.map(x => (x.split(",")(5).toInt,1))***
2. ***val no_air_travelers = baseRDD1.reduceByKey((x,y)=>(x+y)).foreach(println)***

we are creating a tuple RDD baseRDD1 and mapping the key with numerical value 1.

```
scala> val baseRDD1 = baseRDD.map(x => (x.split(",")(5).toInt,1))
baseRDD1: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[23] at map at <console>:27

scala> baseRDD1.foreach(println)
(1990,1)
(1991,1)
(1992,1)
(1990,1)
(1992,1)
(1991,1)
(1990,1)
(1991,1)
(1992,1)
(1993,1)
(1993,1)
(1993,1)
(1993,1)
(1991,1)
(1992,1)
(1993,1)
(1990,1)
(1990,1)
(1991,1)
(1992,1)
(1993,1)
(1991,1)
(1991,1)
(1990,1)
(1991,1)
(1991,1)
(1990,1)
(1992,1)
(1992,1)
(1990,1)
(1993,1)
(1994,1)
```

We are reducing the number of occurrences using reduceByKey and printing the result. Therefore, Total no of air travelers per year is,

```
scala> val no_air_travelers = baseRDD1.reduceByKey((x,y)=>(x+y)).foreach(println)
(1994,1)
(1992,7)
(1990,8)
(1991,9)
(1993,7)
no_air_travelers: Unit = ()
```

Task 2 - What is the total air distance covered by each user per year?

Codes used to achieve the above,

1. ***val baseRDD2 = baseRDD.map(x => ((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))***
2. ***val distance_user = baseRDD2.reduceByKey((x,y) => (x + y)).foreach(println)***

We are creating a tuple rdd "baseRDD2" and mapping the key and value. Here the userID, year acts as key and the travel distance is value.

```
scala> val baseRDD2 = baseRDD.map(x => ((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))
baseRDD2: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[25] at map at <console>:27

scala> baseRDD2.foreach(println)
((1,1990),200)
((2,1991),200)
((3,1992),200)
((4,1990),200)
((5,1992),200)
((6,1991),200)
((7,1990),200)
((8,1991),200)
((9,1992),200)
((10,1993),200)
((1,1993),200)
((2,1993),200)
((3,1993),200)
((4,1991),200)
((5,1992),200)
((6,1993),200)
((7,1990),200)
((8,1990),200)
((9,1991),200)
((10,1992),200)
((1,1993),200)
((2,1991),200)
((3,1991),200)
((4,1990),200)
((5,1991),200)
((6,1991),200)
((7,1990),200)
((8,1992),200)
((9,1992),200)
((10,1990),200)
((1,1993),200)
((5,1994),200)
```

In the second step, we are reducing the number of occurrences using reduceByKey and printing the result, therefore the total air distance covered by each user per year is show below in the screenshot,

```
scala> val distance_user = baseRDD2.reduceByKey((x,y) => (x + y)).foreach(println)
((3,1992),200)
((3,1993),200)
((5,1991),200)
((6,1991),400)
((10,1993),200)
((5,1992),400)
((8,1991),200)
((8,1990),200)
((1,1993),600)
((5,1994),200)
((2,1993),200)
((2,1991),400)
((4,1990),400)
((10,1992),200)
((3,1991),200)
((1,1990),200)
((10,1990),200)
((6,1993),200)
((9,1992),400)
((8,1992),200)
((7,1990),600)
((9,1991),200)
((4,1991),200)
distance_user: Unit = ()
```

Task 3 - Which user has travelled the largest distance till date?

Codes used below,

1. ***val baseRDD3 = baseRDD.map(x=> (x.split(",")(0),x.split(",")(4).toInt))***
2. ***val largest_dist = baseRDD3.reduceByKey((x,y)=>(x+y)).takeOrdered(1)***

The tuple rdd "baseRDD3" is created to map the key and value from the baseRDD. Here the userID and is key and the travel distance is value,

In the 2nd step, we are reducing the number of occurrences using reduceByKey and using the takeOrdered function to get the result,

largest_dist: Array[(String, Int)] = Array((1,800))

```
scala> val baseRDD3 = baseRDD.map(x=> (x.split(",")(0),x.split(",")(4).toInt))
baseRDD3: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[27] at map at <console>:27

scala> baseRDD3.foreach(println)
(1,200)
(2,200)
(3,200)
(4,200)
(5,200)
(6,200)
(7,200)
(8,200)
(9,200)
(10,200)
(1,200)
(2,200)
(3,200)
(4,200)
(5,200)
(6,200)
(7,200)
(8,200)
(9,200)
(10,200)
(1,200)
(2,200)
(3,200)
(4,200)
(5,200)
(6,200)
(7,200)
(8,200)
(9,200)
(10,200)
(1,200)
(5,200)
```

The required output,

```
scala> val largest_dist = baseRDD3.reduceByKey((x,y)=>(x+y)).takeOrdered(1).foreach(println)
(1,800)
largest_dist: Unit = ()
```

Task 4 – What is the most preferred destination for all users?

Codes used below,

1. ***val baseRDD4 = baseRDD.map(x => (x.split(",")(2),1))***
2. ***val dest = baseRDD4.reduceByKey((x,y)=>(x+y))***
3. ***val dest = baseRDD4.reduceByKey((x,y)=>(x+y)).takeOrdered(1)(Ordering[Int].reverse.on(_._2))***

A tuple rdd created with the destination as key and numerical 1 as value, and we are reducing the number of occurrences using the reduceByKey. Now, the most preferred destination is taken by using the function takeOrdered and ordering the values descending so that we can get the required output.

The output of the each step is shown below,

```
scala> val baseRDD4 = baseRDD.map(x => (x.split(",")(2),1))
baseRDD4: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[35] at map at <console>:27

scala> baseRDD4.foreach(println)
(IND,1)
(CHN,1)
(CHN,1)
(IND,1)
(RUS,1)
(PAK,1)
(AUS,1)
(RUS,1)
(RUS,1)
(CHN,1)
(CHN,1)
(IND,1)
(IND,1)
(AUS,1)
(IND,1)
(CHN,1)
(RUS,1)
(CHN,1)
(AUS,1)
(CHN,1)
(IND,1)
(RUS,1)
(PAK,1)
(PAK,1)
(PAK,1)
(RUS,1)
(IND,1)
(IND,1)
(IND,1)
(AUS,1)
(AUS,1)
(PAK,1)
```

```
scala> val dest = baseRDD4.reduceByKey((x,y)=>(x+y))
dest: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[36] at reduceByKey at <console>:29

scala> dest.foreach(println)
(CHN,7)
(IND,9)
(PAK,5)
(RUS,6)
(AUS,5)
```

The required output,

```
scala> val dest = baseRDD4.reduceByKey((x,y)=>(x+y)).takeOrdered(1)(Ordering[Int].reverse.on(_._2))
dest: Array[(String, Int)] = Array((IND,9))
```