Hadoop – TextFile

Loading data into HDFS

$cat > football.txt

India's football team captain Sunil Chhetri recently took to Twitter pleading fans to support his team in a four-nation tournament held in Mumbai. Virat Kohli, the captain of national cricket team, doesn't have to issue such appeals as he mostly plays in front of a full house. But football is gaining a following in India, more so after the launch of the professional Indian Super League. Over the next month, cricket will take a backseat as India joins the world in watching the FIFA World Cup.

Storing football.txt into HDFS (input-data floder)

$hadoop fs -mkdir input-data

$hadoop fs -put football.txt input-data

Starting Spark shell

$spark-shell

1. Create RDD for above text file with 3 partitions.

scala>val rdd = sc.textFile("input-data/football.txt",3)

2. display the contents of file using RDD.

scala>rdd.take(3) Or rdd.collect.foreach(println)

```
scala> val rdd = sc.textFile("input-data/football.txt",3)
rdd: org.apache.spark.rdd.RDD[String] = input-data/football.txt MapPartitionsRDD[1] at textFile at <console>:24

scala> rdd.collect.foreach(println)
India's football team captain Sunil Chhetri recently took to Twitter pleading fans to support his team in a four-nation tournament
held in Mumbai. Virat Kohli, the captain of
national cricket team, doesn't have to issue such appeals as he mostly plays in front of a full house. But football is gaining a fo
llowing in India, more so after the launch of the
professional Indian Super League. Over the next month, cricket will take a backseat as India joins the world in watching the FIFA W
orld Cup.
```

3. create individual RDD for first 3 lines.

```
scala> val L1 = sc.parallelize(List(allLines(0)))
L1: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[2] at parallelize at <console>:26

scala> L1.collect
res4: Array[String] = Array(India's football team captain Sunil Chhetri recently took to Twitter pleading fans to support his team
in a four-nation tournament held in Mumbai. Virat Kohli, the captain of)

scala> val L2 = sc.parallelize(List(allLines(1)))
L2: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[3] at parallelize at <console>:26

scala> L2.collect
res5: Array[String] = Array(national cricket team, doesn't have to issue such appeals as he mostly plays in front of a full house.
But football is gaining a following in India, more so after the launch of the)

scala> val L3 = sc.parallelize(List(allLines(2)))
L3: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[4] at parallelize at <console>:26

scala> L3.collect
res6: Array[String] = Array(professional Indian Super League. Over the next month, cricket will take a backseat as India joins the
world in watching the FIFA World Cup.)
```

4. Give a wordcount for each token in the file and save it to another file.
scala>val wordsCount = rdd.flatMap(x => x.split(" ")).map(x => (x,1)).reduceByKey((x,y) => x+y).saveAsTextFile("/outputs")

5. Differentiate between functionality of map and flatmap.

S.No	Map	FlatMap
1	A map is a transformation operation in Apache Spark. It applies to each element ofRDD and it returns the result as new RDD. In the Map, operation developer can define his own custom business logic. The same logic will be applied to all the elements of RDD.	A flatMap is a transformation operation. It applies to each element of RDD and it returns the result as new RDD. It is similar to Map, but FlatMap allows returning 0, 1 or more elements from map function.
2	Spark Map function takes one element as an input process it according to custom code and returns one element at a time. Map transforms an RDD of length N into another RDD of length N. The input and output RDDs will typically have the same number of records.	A FlatMap function takes one element as an input process it according to custom code and returns 0 or more element at a time. flatMap() transforms an RDD of length N into another RDD of length M.
3	Spark map function expresses a one-to-one transformation.	Spark flatMap function expresses a one-to-many transformation.

6. Display data in each partition individually and load them into separate files.

```
nitishpandey@hadoop-m:~$ hadoop fs -ls /outputs
Found 4 items
-rw-r--r--   2 nitishpandey hadoop          0 2019-05-04 05:47 /outputs/_SUCCESS
-rw-r--r--   2 nitishpandey hadoop        220 2019-05-04 05:47 /outputs/part-00000
-rw-r--r--   2 nitishpandey hadoop        265 2019-05-04 05:47 /outputs/part-00001
-rw-r--r--   2 nitishpandey hadoop        189 2019-05-04 05:47 /outputs/part-00002
nitishpandey@hadoop-m:~$ hadoop fs -ls /outputs/part-00000
-rw-r--r--   2 nitishpandey hadoop        220 2019-05-04 05:47 /outputs/part-00000
nitishpandey@hadoop-m:~$ hadoop fs -cat /outputs/part-00000
(football,2)
(his,1)
(Chhetri,1)
(Cup.,1)
(next,1)
(world,1)
(Indian,1)
(full,1)
(have,1)
(cricket,2)
(such,1)
(four-nation,1)
(Kohli,,1)
(national,1)
(more,1)
(of,3)
(pleading,1)
(team,2)
(gaining,1)
(the,6)
(India,,1)
nitishpandey@hadoop-m:~$
```

```
nitishpandey@hadoop-m:~$ hadoop fs -cat /outputs/part-00001
(captain,2)
(Sunil,1)
(issue,1)
(plays,1)
(is,1)
(Over,1)
(front,1)
(a,4)
(mostly,1)
(took,1)
(tournament,1)
(he,1)
(following,1)
(World,1)
(watching,1)
(professional,1)
(take,1)
(League.,1)
(support,1)
(fans,1)
(so,1)
(launch,1)
(appeals,1)
(joins,1)
(recently,1)
nitishpandey@hadoop-m:~$
```

```
nitishpandey@hadoop-m:~$ hadoop fs -cat /outputs/part-00002
(after,1)
(India's,1)
(But,1)
(Virat,1)
(doesn't,1)
(month,,1)
(backseat,1)
(Twitter,1)
(will,1)
(to,3)
(house.,1)
(as,2)
(team,,1)
(in,5)
(FIFA,1)
(held,1)
(Mumbai.,1)
(Super,1)
(India,1)
```