Principles of Big Data Management

Project Phase 2

Team-3

Anusha Malineni

Sujitha Puthana

Achyuth Reddy N

Sri Sai Narayana Ramgopal Mangena

1. Introduction

The main aim of this project is analyzing big data collected from twitter about Sports using apache spark for fast analysis of big data. Here we collected twitter data on sports. Later we analyzed the extracted data using spark RDD and Data frames. To write a SQL commands to analyze twitter tweets.

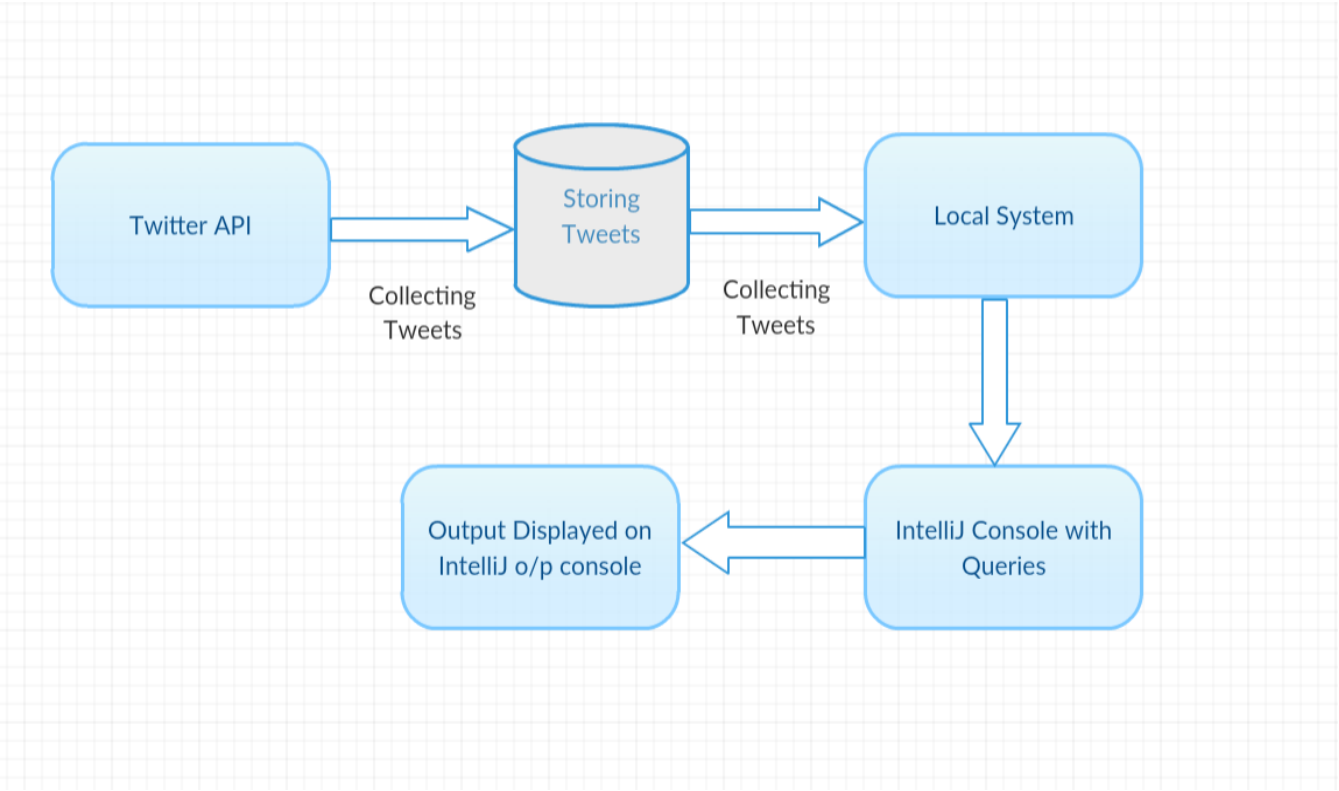
Here we used Spark to processing the twitter data. Because it has many advantages like

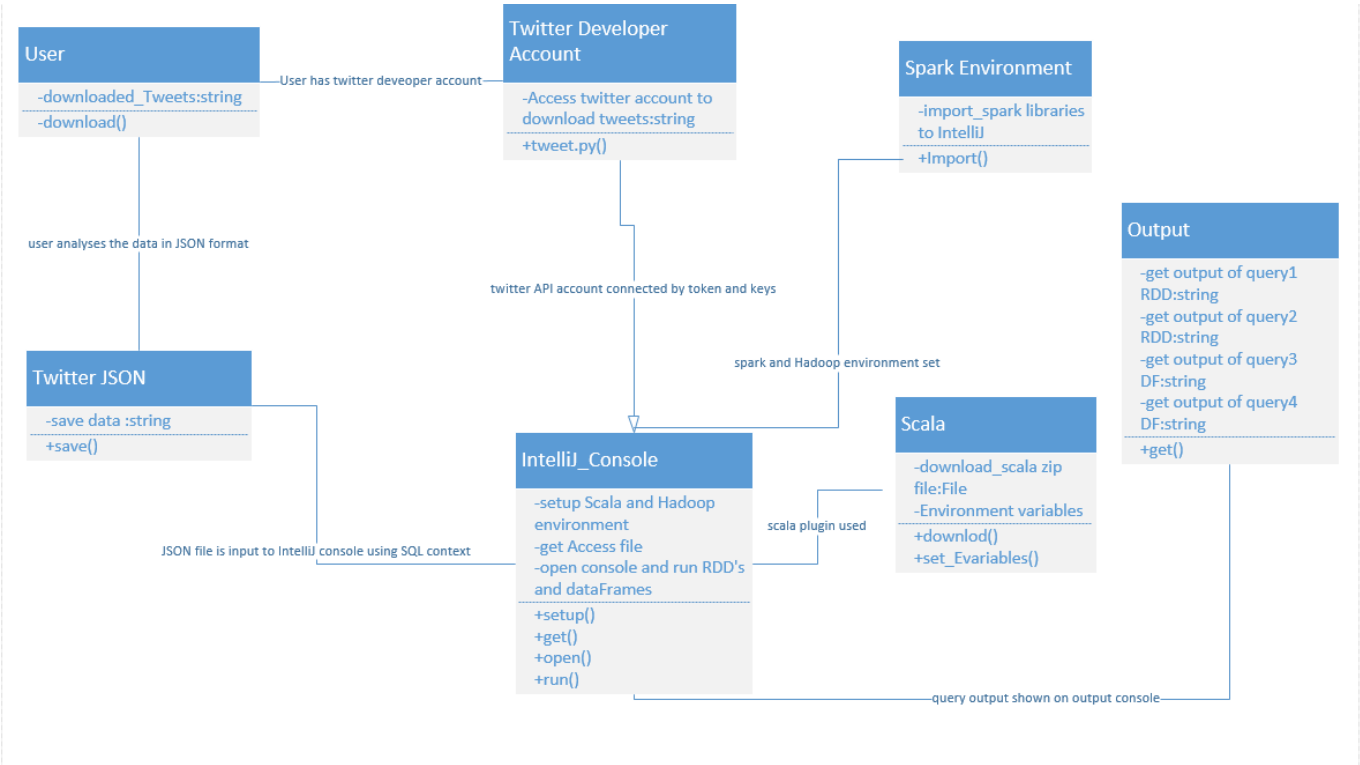
- Speed: Run Programs up to 100x faster than Hadoop Map reduce in memory.
- Ease of Use: Write applications quickly in Java, Scala, Python, R.
- Generality: Combine SQL, streaming, and complex analytics.
- Runs Everywhere: Spark runs on Hadoop, Mesos, standalone, or in the cloud.

2. Software Architecture & Design

Twitter tweets are extracted using the API by running python program using the twitter tokens. The collected tweets are stored in JSON format which enables us to view the twitter data as table format using Notepad++ plug-ins. On the twitter JSON data we run the Spark RDD and Data frames analytical queries from IntelliJ using Scala plug-ins. These queries will give us the output in the IntelliJ output console.

Below diagram shows us the abstract architecture design and UML diagrams.





The Visio file is attached here

UML.vsdx

3. System Requirements:

Software Requirements:

- Apache Spark V 1.6
- Apache Hadoop V 2.6
- JDK 1.8
- Scala V 2.11
- IntelliJ Idea Community V 16

Programming Languages:

- Scala
- Python
- SQL

4.Analytical Queries:

Data Frames & RDD's:

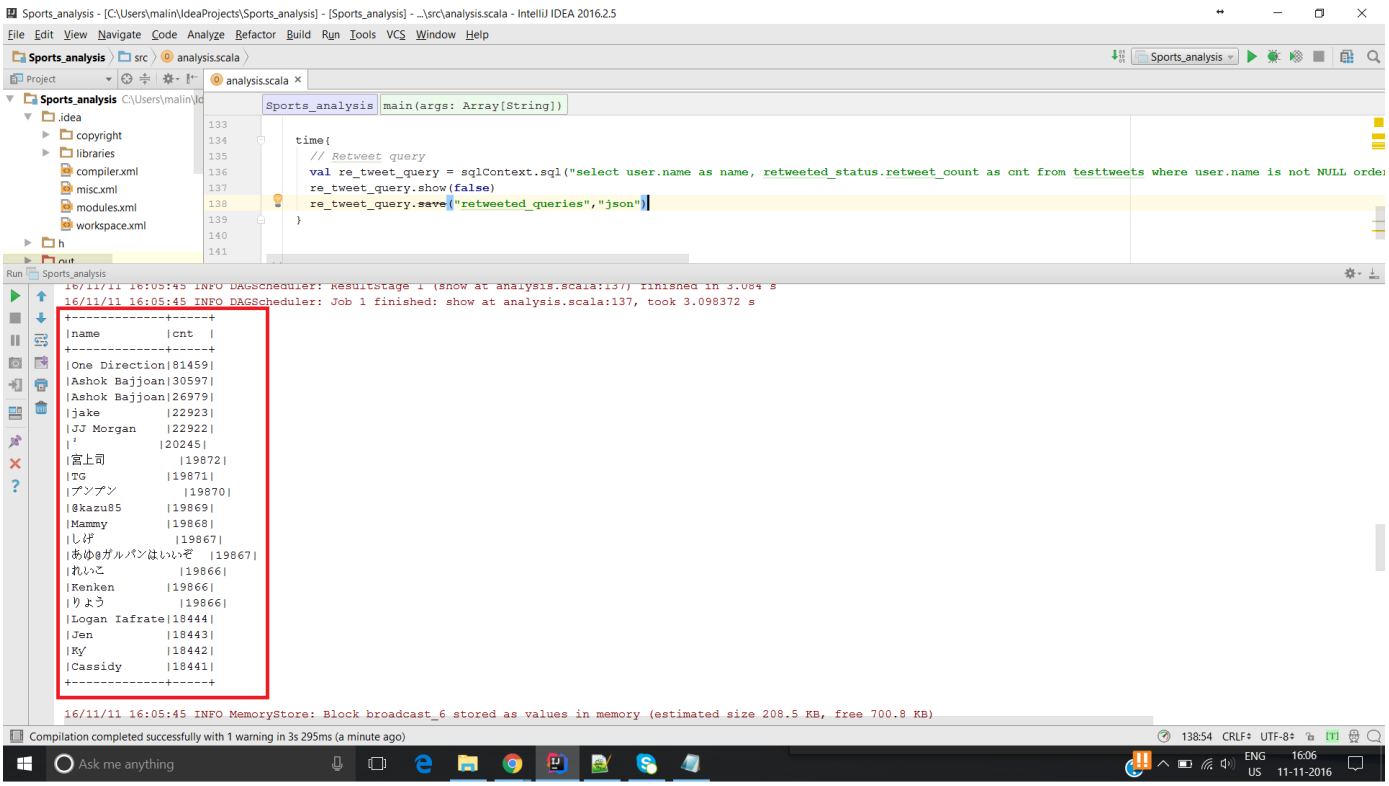
Query1: This query gives us the number of retweet counts of the users, by which we can deduce that most of the people followed certain user's tweet. So, this query gives the result of the most popular person followed based on the retweet count.

```
val textFile = sc.textFile("C:\\Users\\malin\\Downloads\\Sports_tweets.json")

def time[A](f: => A) = {
  val s = System.nanoTime
  val ret = f
  println("Execution time for analysing retweeted count is : " + (System.nanoTime
- s) / 1e9 + " sec")
  ret
}

time{
  // Retweet query
```

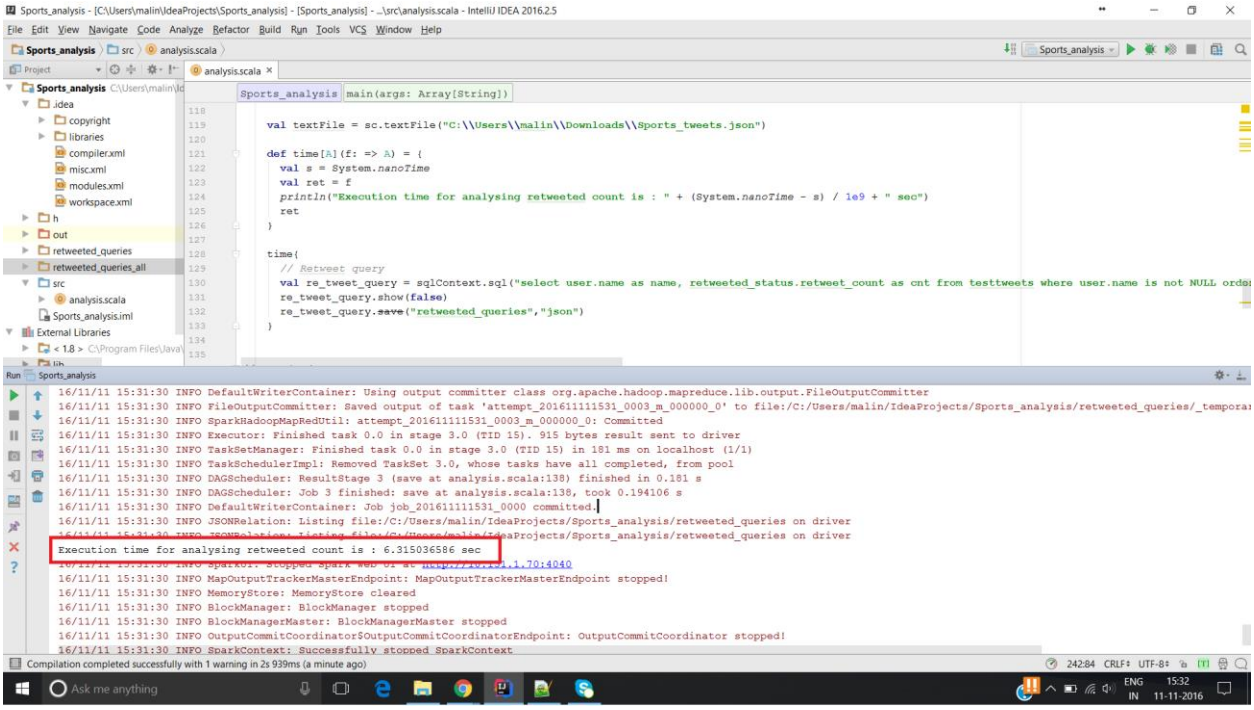
```
  val re_tweet_query = sqlContext.sql("select user.name as name, retweeted_status.retweet_count as cnt from testtweets where user.name is not NULL order by cnt desc limit 20")
  re_tweet_query.show(false)
  re_tweet_query.save("retweeted_queries","json")
}
```

The output of the above query is shown as below and the output file is attached.

re_tweeted_count.json



The run time of the above query is shown in the console of intelliJ idea community.

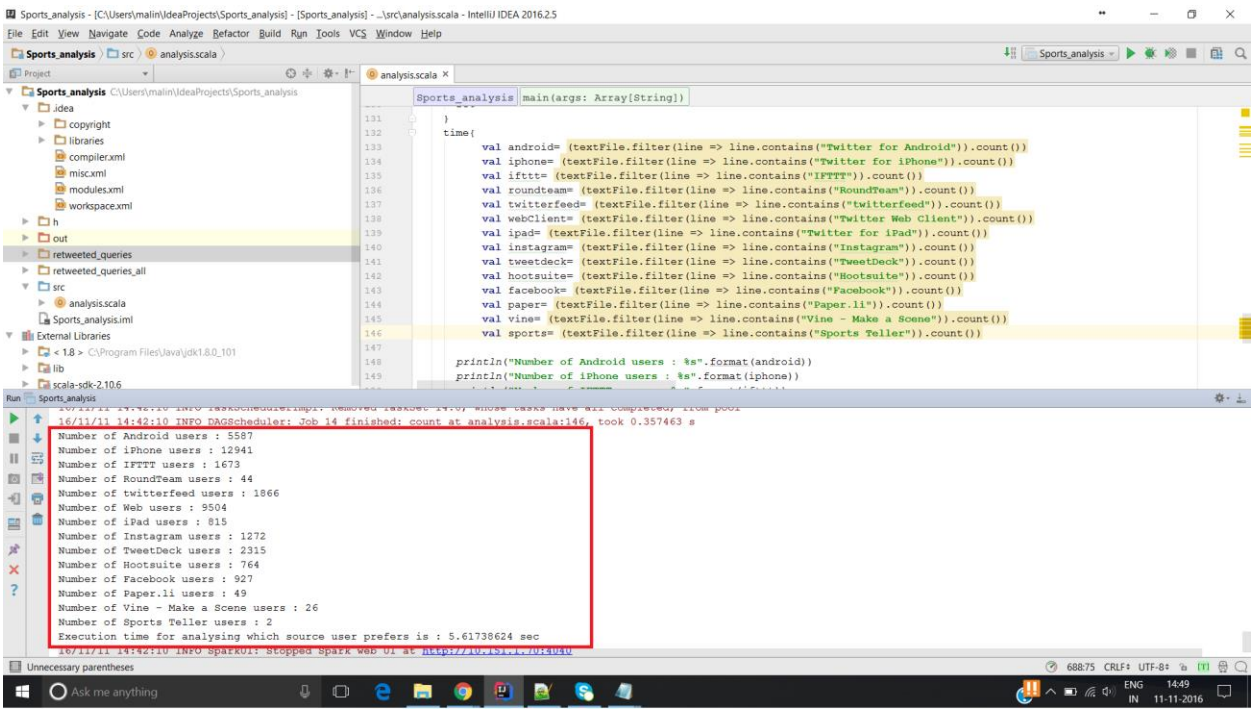


Query2: This RDD query will analyze, from which devices the tweets are tweeted by the users.

```
def time[A](f: => A) = {
  val s = System.nanoTime
  val ret = f
  println("Execution time for analysing which source user prefers is : " +
(System.nanoTime - s) / 1e9 + " sec")
  ret
}
time{
       val android= (textFile.filter(line => line.contains("Twitter for
Android")).count())
      val iphone= (textFile.filter(line => line.contains("Twitter for
iPhone")).count())
      val ifttt= (textFile.filter(line => line.contains("IFTTT")).count())
      val roundteam= (textFile.filter(line => line.contains("RoundTeam")).count())
      val twitterfeed= (textFile.filter(line =>
line.contains("twitterfeed")).count())
      val webClient= (textFile.filter(line => line.contains("Twitter Web
Client")).count())
      val ipad= (textFile.filter(line => line.contains("Twitter for
iPad")).count())
      val instagram= (textFile.filter(line => line.contains("Instagram")).count())
      val tweetdeck= (textFile.filter(line => line.contains("TweetDeck")).count())
      val hootsuite= (textFile.filter(line => line.contains("Hootsuite")).count())
      val facebook= (textFile.filter(line => line.contains("Facebook")).count())
      val paper= (textFile.filter(line => line.contains("Paper.li")).count())
      val vine= (textFile.filter(line => line.contains("Vine - Make a
Scene")).count())
```

```
      val sports= (textFile.filter(line => line.contains("Sports
Teller")).count())

    println(("Number of Android users : %s\n Number of iPhone users : %s \n " +
  "Number of IFTTT users : %s \n Number of RoundTeam users : %s \n " +
  "Number of twitterfeed users : %s \n Number of Web users : %s\n Number of iPad
users : %s \n " +
  "Number of Instagram users : %s \n Number of TweetDeck users : %s \n Number of
Hootsuite users : %s \n " +
  "Number of Facebook users : %s \n Number of Paper.li users : %s \n Number of
Vine - Make a Scene users : %s \n " +
  "Number of Sports Teller users : %s
\n").format(android,iphone,ifttt,roundteam,twitterfeed,webClient,ipad,instagram,tw
eetdeck,hootsuite,facebook,paper,vine,sports))

}
```

The output for the RDD and run time is shown in the below screen shot:

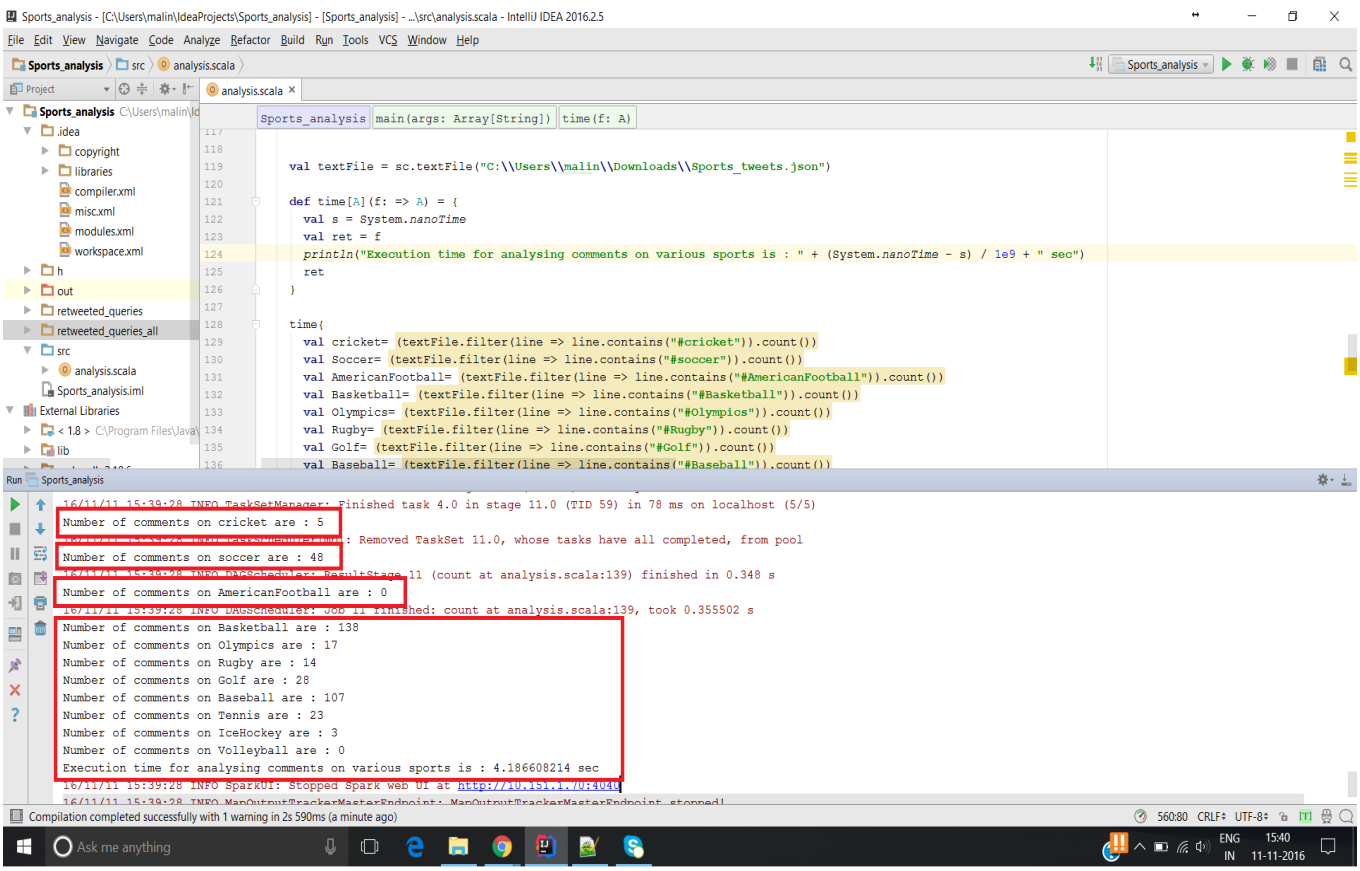


Query 3: This query will give the most popular Hashtag sports topic discussed in the twitter from the source we extracted.

```
def time[A](f: => A) = {
  val s = System.nanoTime
  val ret = f
  println("Execution time for analysing comments on various sports is : " +
(System.nanoTime - s) / 1e9 + " sec")
  ret
}
```

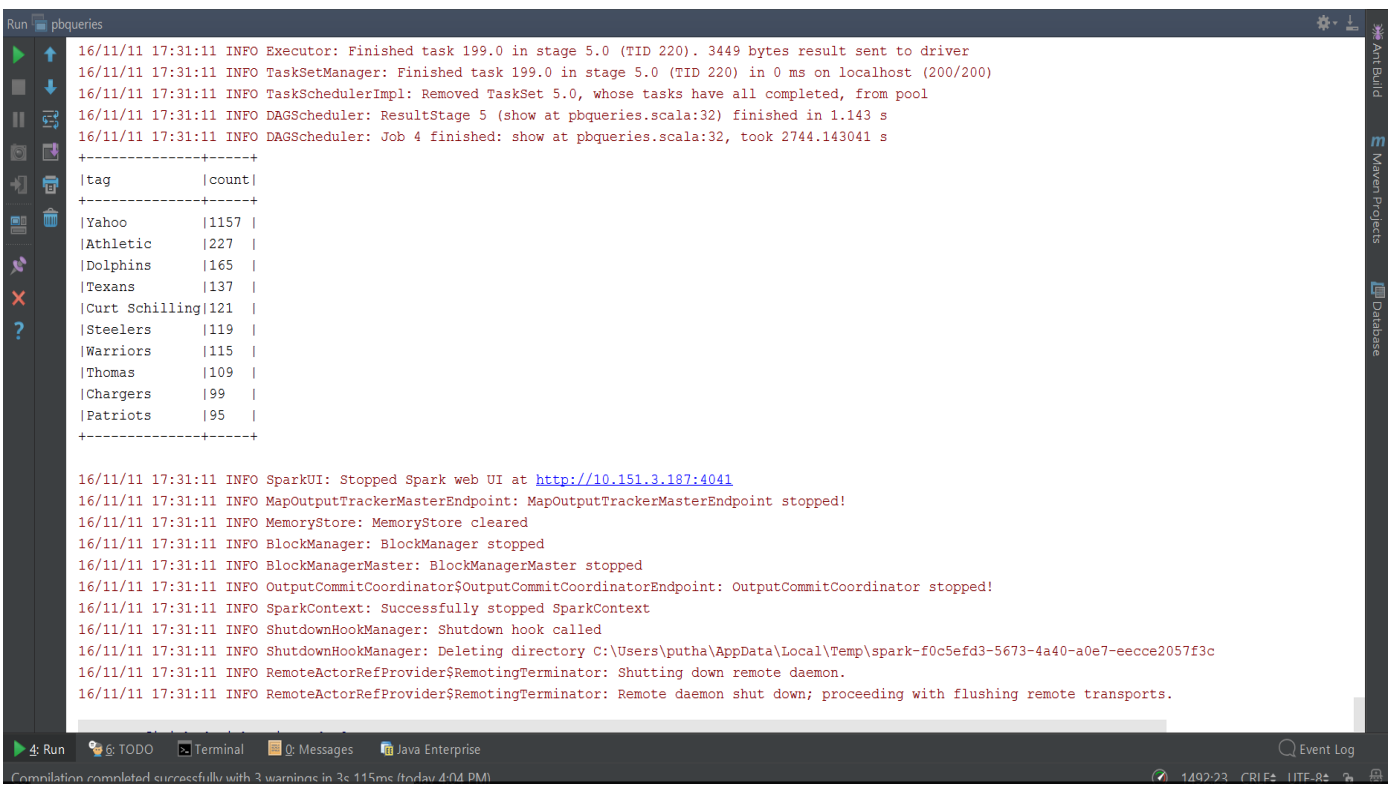
Principles of Big Data Management-Project phase2

```
time{
  val cricket= (textFile.filter(line => line.contains("#cricket")).count())
  val Soccer= (textFile.filter(line => line.contains("#soccer")).count())
  val AmericanFootball= (textFile.filter(line =>
line.contains("#AmericanFootball")).count())
  val Basketball= (textFile.filter(line => line.contains("#Basketball")).count())
  val Olympics= (textFile.filter(line => line.contains("#Olympics")).count())
  val Rugby= (textFile.filter(line => line.contains("#Rugby")).count())
  val Golf= (textFile.filter(line => line.contains("#Golf")).count())
  val Baseball= (textFile.filter(line => line.contains("#Baseball")).count())
  val Tennis= (textFile.filter(line => line.contains("#Tennis")).count())
  val IceHockey= (textFile.filter(line => line.contains("#IceHockey")).count())
  val Volleyball= (textFile.filter(line => line.contains("#Volleyball")).count())

 println(("Number of comments on cricket are : %s \n Number of comments on soccer
are : %s \n " +
  "Number of comments on AmericanFootball are : %s \n Number of comments on
Basketball are : %s \n " +
  "Number of comments on Olympics are : %s \n Number of comments on Rugby are : %s
\n Number of comments on Golf are : %s \n " +
  "Number of comments on Baseball are : %s \n Number of comments on Tennis are :
%s \nNumber of comments on IceHockey are : %s \n " +
  "Number of comments on Volleyball are :
%s").format(cricket,Soccer,AmericanFootball,Basketball,Olympics,Rugby,Golf,Basebal
l,Tennis,IceHockey,Volleyball))}
```



Query 4: This query retrieves the common hashtag topics when compared with hashtag topics given in Blackboard.

```
val anlysis = sqlContext.jsonFile("C:\\Users\\putha\\Desktop\\sujitha\\prin of big data\\project\\blackboard_hashtags.json")
  anlysis.registerTempTable("hashtags")
  anlysis.withColumnRenamed("test","_corrupt_record")

  // Renaming the column name with tag
  val df = anlysis.toDF().withColumnRenamed("_corrupt_record", "tag")
  df.registerTempTable("hash1")

//selecting the hashtags presnt in hash1 and counts
  val t = sqlContext.sql("select hash1.tag,count(testtweets.text) as count from testtweets join hash1 on testtweets.text like concat ('%',hash1.tag,'%') group by hash1.tag order by count desc limit 10")
  t.show(false)
```



Query 5: This query uses twitter API, which takes userid as input and retrieve the first tweet from our extracted data with the tag that contains “soccer”.

```
def main(args: Array[String]) {

    val consumer = new CommonsHttpOAuthConsumer(consumerKey, consumerSecret)
```

```
    consumer.setTokenWithSecret(accessToken, accessSecret)


    val sparkConf = new
SparkConf().setAppName("SparkWordCount").setMaster("local[*]")

    val sc = new SparkContext(sparkConf)

    // Contains SQLContext which is necessary to execute SQL queries
    val sqlContext = new org.apache.spark.sql.SQLContext(sc)

    // Reads json file and stores in a variable
    val tweet =
sqlContext.read.json("C:\\Users\\malin\\Downloads\\Sports_tweets.json")
    tweet.registerTempTable("testtweets")
    val s8 = sqlContext.sql("select user.name as name,user.id from testtweets
where text like '%soccer%'")
    s8.show(false)
    val userid = s8.first().getLong(1)
    println("Selected user id is:"+userid)

    val request = new
HttpGet("https://api.twitter.com/1.1/users/show.json?user_id="+userid)
    consumer.sign(request)
    val client = new DefaultHttpClient()
    val response = client.execute(request)

    println(response.getStatusLine().getStatusCode());
    println(IOUtils.toString(response.getEntity().getContent()))
  }
```

Principles of Big Data Management-Project phase2

Sports_analysis - [C:\Users\malin\IdeaProjects\Sports_analysis] - [Sports_analysis] - ...\src\Api.scala - IntelliJ IDEA 2016.2.5

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Sports_analysis > src > Api.scala

analysis.scala | Api.scala

Project

- Sports_analysis C:\Users\malin\Id
  - .idea
    - copyright
    - libraries
    - compiler.xml
    - misc.xml
    - modules.xml
    - workspace.xml
  - h
  - out
  - retweeted_queries
  - retweeted_queries_all
  - src
    - analysis.scala
      - Sports_analysis
    - Api
  - Sports_analysis.iml
- External Libraries

Api main(args: Array[String])

```
consumer.setTokenWithSecret(accessToken, accessSecret)


val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")

val sc = new SparkContext(sparkConf)

// Contains SQLContext which is necessary to execute SQL queries
val sqlContext = new org.apache.spark.sql.SQLContext(sc)

// Reads json file and stores in a variable
val tweet = sqlContext.read.json("C:\\Users\\malin\\Downloads\\Sports_tweets.json")
tweet.registerTempTable("testtweets")
val s8 = sqlContext.sql("select user.name as name,user.id from testtweets where text like '%soccer%'")
s8.show(false)
val userid = s8.first().getLong(1)
println("Selected user id is:"+userid)
```

Run Api

```
16/11/11 23:01:40 INFO Executor: Finished task 0.0 in stage 2.0 (TID 6). 2717 bytes result sent to driver
16/11/11 23:01:40 INFO TaskSetManager: Finished task 0.0 in stage 2.0 (TID 6) in 16 ms on localhost (1/1)
16/11/11 23:01:40 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have all completed, from pool
16/11/11 23:01:40 INFO DAGScheduler: ResultStage 2 (first at Api.scala:36) finished in 0.016 s
16/11/11 23:01:40 INFO DAGScheduler: Job 2 finished: first at Api.scala:36, took 0.038446 s
Selected user id is:2153103163
200
16/11/11 23:01:41 INFO SparkContext: Invoking stop() from shutdown hook
{"id":2153103163,"id_str":"2153103163","name":"Sports On Notice","screen_name":"SportsOnNotice","location":"Texas, USA","profile_location":{"id":"e0060cda70f5f341","url":"https:\/\/a
16/11/11 23:01:41 INFO SparkUI: Stopped Spark web UI at http://192.168.0.10:4040
16/11/11 23:01:41 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
16/11/11 23:01:41 INFO MemoryStore: MemoryStore cleared
16/11/11 23:01:41 INFO BlockManager: BlockManager stopped
16/11/11 23:01:41 INFO BlockManagerMaster: BlockManagerMaster stopped
16/11/11 23:01:41 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
16/11/11 23:01:41 INFO SparkContext: Successfully stopped SparkContext
16/11/11 23:01:41 INFO ShutdownHookManager: Shutdown hook called
16/11/11 23:01:41 INFO ShutdownHookManager: Deleting directory C:\Users\malin\AppData\Local\Temp\spark-99029f0c-39c7-4edd-adde-ae1b0ad89a5a
```

Compilation completed successfully with 1 warning in 3s 229ms (a minute ago)

183:99 CRLF: UTF-8:

Ask me anything

ENG IN 23:02 11-11-2016