Telecom Companies Report
Using Spark and Scala

Table of Contents

Definition

Project Statement

The aim of the project is to find the top 10 customers facing frequent call drops in Roaming (a.k.a. 10CFFCD). For this a CDR (Call Details Record) file is provided. It contains the list of customers for which calls have dropped. The aim of this report is to improve customer experience. 10CFFCD customers can then be called back. In parallel, their roaming providers can be contacted to improve the connectivity issues in specific areas.
For this the report provides the following info:

- The list of the 10CFFCD to call back urgently
- The 10CFFCD log information to supply the roaming providers.

Data Source

https://edureka.wistia.com/medias/799xfn376r/download?media_file_id=73236251

Data Description

The below screen shot shows a snippet of the data available in the CDR file.

VISITOR_LOCN – The caller's location represented as a unique identifier.
CALL_DURATION _SEC – The time in second a call was on before dropping.
PHONE_NO – The caller telephone number.
ERROR_CODE – A unique identifier representing the call drop error.

	A	B	C	D
1	VISITOR_LOCN	CALL_DURATION_SEC	PHONE_NO	ERROR_CODE
2	6022	1	66832145	0x830F09
3	6022	1	75566321	0x869B11
4	6022	1	75566321	0x869B11
5	6022	1	134679876	0x869C08
6	6022	1	229804352	0x756511
7	6022	1	1234567890	0x869C08
8	6022	1	4439088765	0x734F06
9	6022	1	4439088765	0x869C08
10	6022	1	4455123897	0x833B08
11	6022	1	5577993322	0x789F17
12	6022	1	6600443311	0x829F08
13	6022	1	8877123432	0x829554
14	6022	1	8877123432	0x889F04
15	6022	1	8879036752	0x789F17
16	6022	1	8879036752	0x860F16
17	6022	1	9678005533	0x829F44
18	6022	1	9678005533	0x889F04

Source Code Details

The code for this report can be found in the .zip, file aname 'CRDProject.scala'.
The 'sbt' file used for the package building is named 'build.sbt'.

Edureka VM installation

It assumes Edureka VM is installed. For more information, please contact Edureka on http://www.edureka.co/apache-spark-scala-training

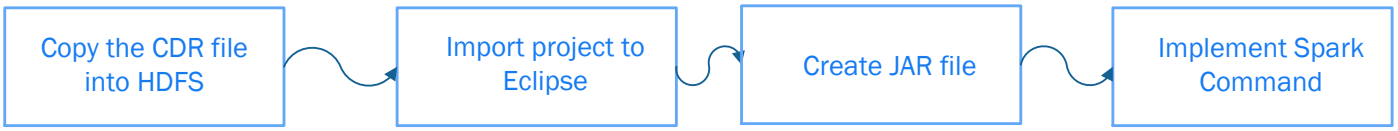
Software and Libraries

- JDK 1.7.0_67
- Eclipse Luna
- Hadoop 2.2.0
- Hive 0.13.1
- Spark 1.5.2
- Scala 2.10.4
- Sbt 0.13. 8

Methodology

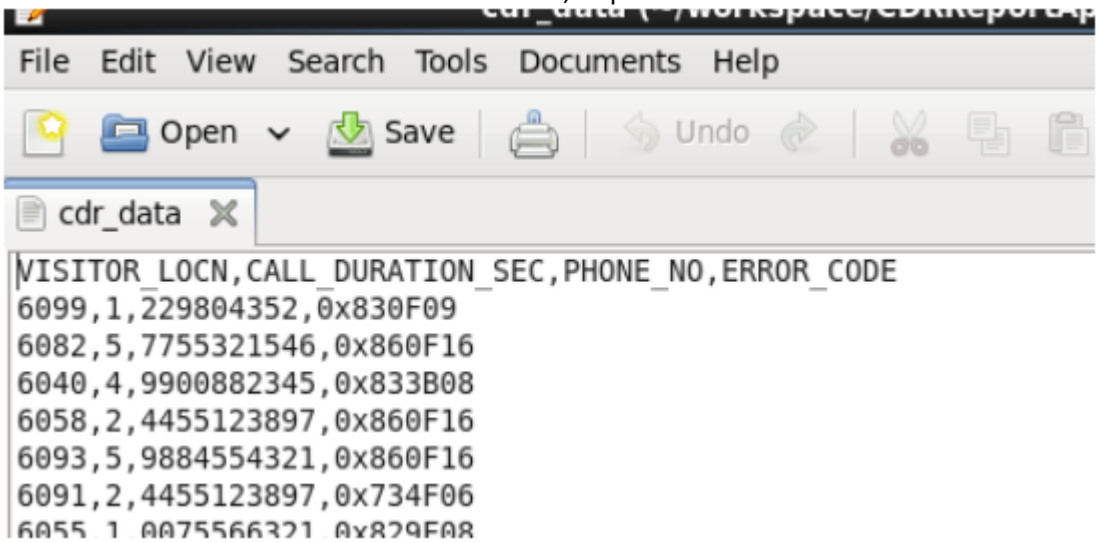
Dataflow Diagram

The below diagram shows the steps involved in running a spark solution to provide an answer to the problem at hand. The first step is to copy the CDR data into the HDFS environment for the Spark context to access the data. The following two steps correspond to the package building/import into the Eclipse environment and the generation of a JAR file. The last step relates to the code implementation that generates the final report output.
The next three sections detail the diagram steps.



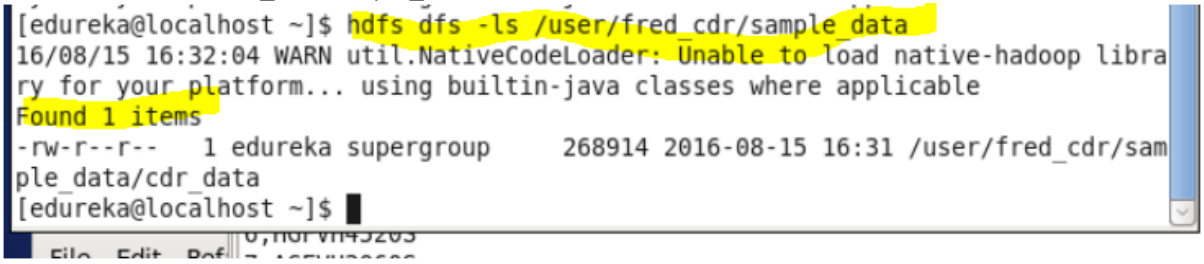
Copy the CDR file into HDFS

1. Create a directory called: '/home/edureka/workspace/CRDRepoertApp/data'
   2.1 Create the file cdr_data
   2.2 Load the CDR schema and data into this file, as per the below screenshot



2. Transfer the CDR file into HDFS, in a command widonw run:
   2.1 hdfs dfs -mkdir /user/fred_cdr
   2.2 hdfs dfs -mkdir /user/fred_ cdr /sample_data
   2.3 hdfs dfs -put "/home/edureka/workspace/CDRReportApp/data/cdr_data" /user/ fred_ cdr /sample_data

3. Check the file exists in HDFS, in a command widonw run:
   hdfs dfs -ls /user/fred_cdr/sample_data



Building and Running the Code

These detailed steps correspond to the second and third steps of the above diagram.

1. Create a directory called: '/home/edureka/workspace/CRDRepoertApp'

2. Create the directory: '/home/edureka/workspace/CDRReportAppApp/scr/main/scala'
   2.1 Create a file 'CRDProject.scala'

3. Create a build.sbt file into '/home/edureka/workspace/CDRReportAppApp/'
   3.1 store the following info (using sudo gedit built.sbt from a terminal window)

```
name := "CDRReportApp Application"

version := "1.0"

scalaVersion  := "2.10.4"

libraryDependencies += "org.apache.spark" %% "spark-core" % "1.6.1"
libraryDependencies += "org.apache.spark" %% "spark-sql" % "1.6.1"
libraryDependencies += "com.databricks" %% "spark-csv" % "1.4.0"

libraryDependencies ++= Seq(
  "org.apache.hadoop" % "hadoop-client" % "2.7.0"
)
```

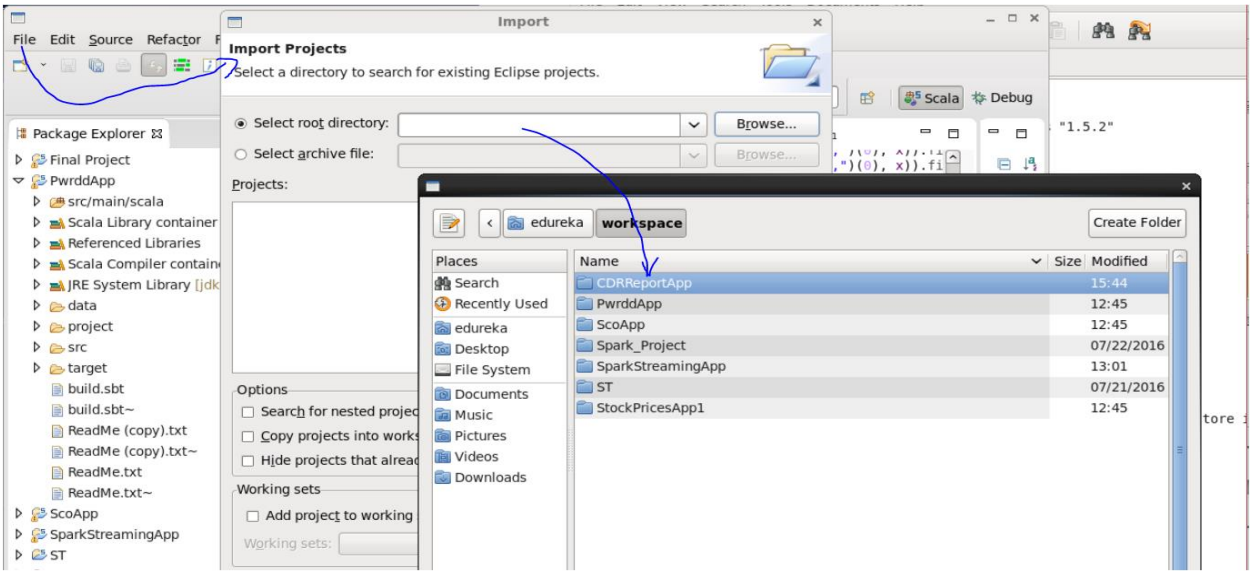
4. Create an Sbt package
   4.1 Open a terminal window
   4.2 Cd to '/home/edureka/workspace/CRDRepoertApp'
   4.3 Type 'sbt package' in the window (as shown below)

```
edureka@localhost:~/workspace/CDRReportApp
File  Edit  View  Search  Terminal  Help
[edureka@localhost ~]$ cd /home/edureka/workspace
[edureka@localhost workspace]$ cd CDRReportApp
[edureka@localhost CDRReportApp]$ sbt package
[info] Loading global plugins from /home/edureka/.sbt/0.13/plugins
[info] Set current project to PwrdApp Application (in build file:/home/edureka/w
orkspace/CDRReportApp/)
[info] Updating {file:/home/edureka/workspace/CDRReportApp/}cdrreportapp...
[info] Resolving com.fasterxml.jackson.module#jackson-module-scala_2.10;2.4.4 ..
[info] Resolving org.apache.directory.server#apacheds-kerberos-codec;2.0.0-M15 .
[info] Resolving org.fusesource.jansi#jansi;1.4 ...
[info] Done updating.
[info] Compiling 1 Scala source to /home/edureka/workspace/CDRReportApp/target/s
cala-2.10/classes...
[info] Packaging /home/edureka/workspace/CDRReportApp/target/scala-2.10/pwrdapp-
application_2.10-1.0.jar ...
[info] Done packaging.
[success] Total time: 18 s, completed Aug 15, 2016 3:31:45 PM
[edureka@localhost CDRReportApp]$
```

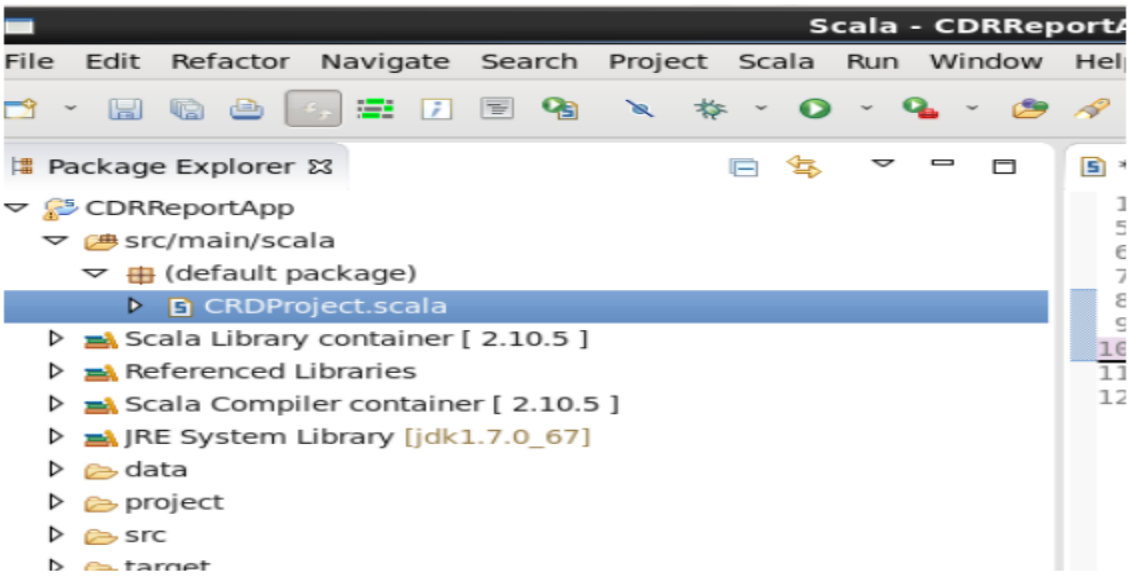
5. Create an Eclipse package
   5.1 Cd to '/home/edureka/workspace/CRDRepoertApp'
   5.2 Type 'sbt eclipse' in the window (as shown below)

```
edureka@localhost:~/workspace/CDRReportApp
File  Edit  View  Search  Terminal  Help
tapp-application_2.10-1.0.jar ...
[info] Done packaging.
[success] Total time: 16 s, completed Aug 15, 2016 3:49:22 PM
[edureka@localhost CDRReportApp]$ eclipse package
:ompilerOracle: exclude org/eclipse/core/internal/dtree/DataTreeNode.forwardDelt
aWith
:ompilerOracle: exclude org/eclipse/jdt/internal/compiler/lookup/ParameterizedMe
thodBinding.<init>
:ompilerOracle: exclude org/eclipse/cdt/internal/core/dom/parser/cpp/semantics/C
PPTemplates.instantiateTemplate
:ompilerOracle: exclude org/eclipse/cdt/internal/core/pdom/dom/cpp/PDOMCPPLinkag
e.addBinding
:ompilerOracle: exclude org/python/pydev/editor/codecompletion/revisited/PythonP
athHelper.isValidSourceFile
:ompilerOracle: exclude org/python/pydev/ui/filetypes/FileTypesPreferencesPage.g
etDottedValidSourceFiles
[edureka@localhost CDRReportApp]$ sbt eclipse
[info] Loading global plugins from /home/edureka/.sbt/0.13/plugins
[info] Set current project to CRDReportApp Application (in build file:/home/edur
eka/workspace/CDRReportApp/)
[info] About to create Eclipse project files for your project(s).
[info] Successfully created Eclipse project files for project(s):
[info] CRDReportApp Application
[edureka@localhost CDRReportApp]$
```

6. Import the project into Eclipse
   6.1 Open eclipse (the Scala IDE)
   6.2 File -> Import -> Existing Projects into Workspace -> Select root directory -> CDRReportAppApp -> Finish
   6.3 Click Finish

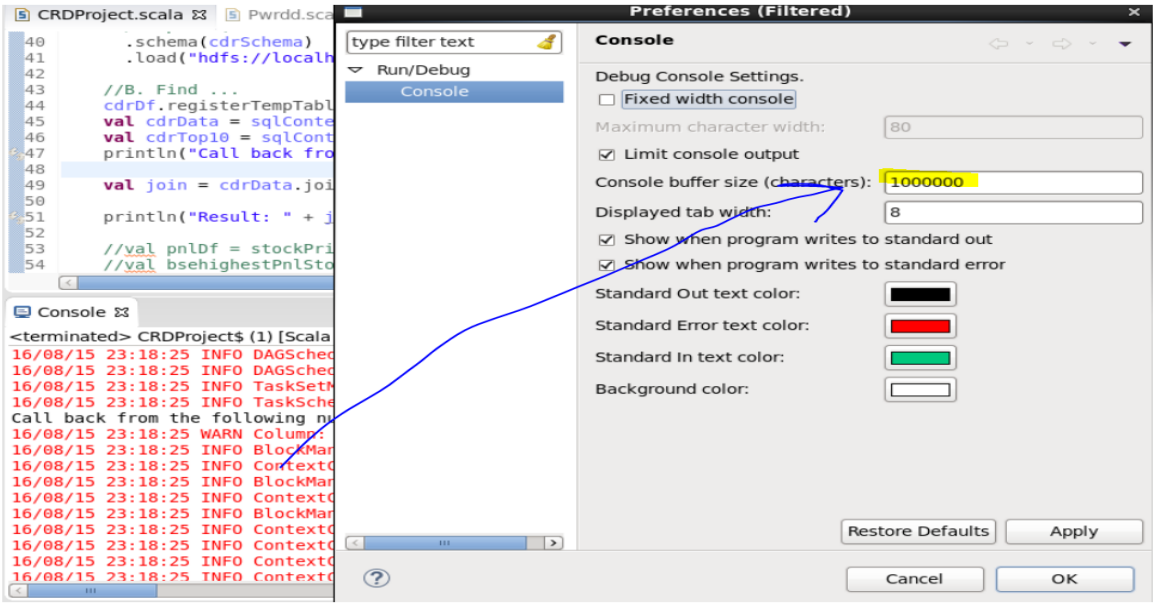


6.4 The Eclipse navigation tree looks like this:

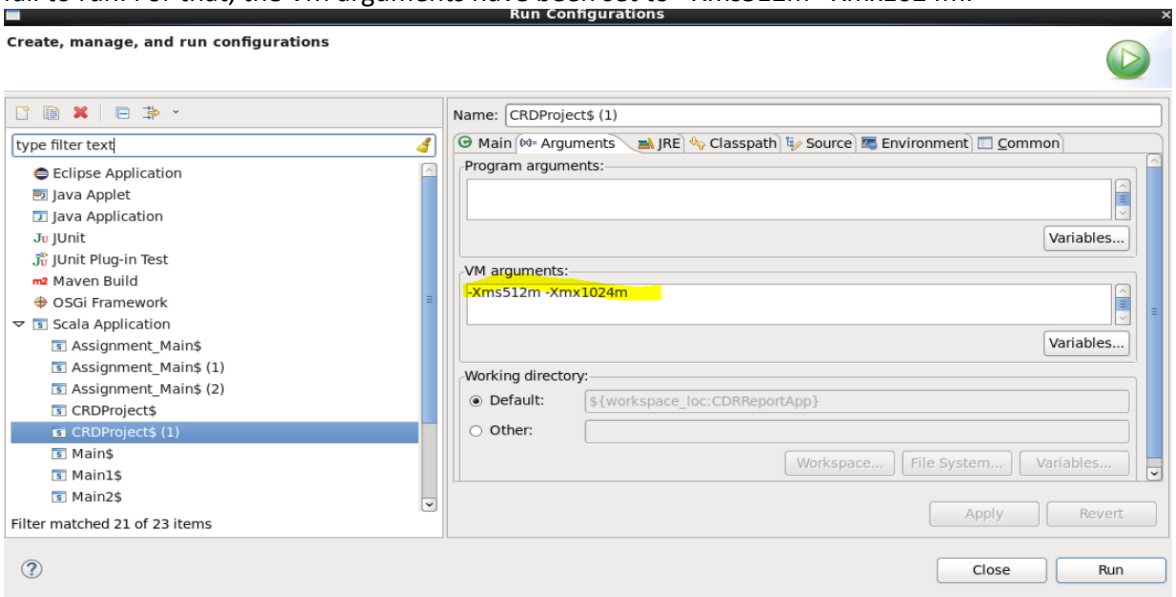


6.4 Setup the Eclipse environment

6.4.1 Ensure the log buffer is large enough. For that set he 'console buffer size' needs to be set to a large enough number, here 1,000,000.



6.4.2 The available process memory should be set to a large enough number, else the process will fail to run. For that, the VM arguments have been set to –Xms512m –Xmx1024m.



Implementation

The below screen shot explains in detail the report implementation.

```
object CRDProject {
  def main(args: Array[String]) {
    //Shared config
    val config = new Configuration();
    val uri = "hdfs://localhost:8020/";
    config.set("fs.defaultFS", uri);
    //Create Spark ctx
    val conf = new SparkConf().setAppName("CDRReportApp Application").setMaster("local[2]");
    val sc = new SparkContext(conf);
    //A. Load the file as table in SparkSQL and register template
    val sqlContext = new org.apache.spark.sql.SQLContext(sc);
    import sqlContext.implicits._; //necessary and after sqlContext is created, else '.toDF' will fail to compile
    //Create the schema, the point is to create a schema is not space in the columns name
    val cdrSchema =  StructType(Array(
                        StructField("VISITOR_LOCN", IntegerType, true),
                        StructField("CALL_DURATION_SEC", IntegerType, true),
                        StructField("PHONE_NO", LongType , true),
                        StructField("ERROR_CODE", StringType, true)));
    //Load the data and add the schema
    val cdrDf = sqlContext.read
      .format("com.databricks.spark.csv")
      .option("header", "true") // Use first line of all files as header
      /*.option("inferSchema", "true") // Automatically infer data types*/
      .schema(cdrSchema)
      .load(uri + "user/fred_cdr/sample_data/cdr_data/");
    //B. Find the 10 most frequent drop out numbers
    cdrDf.registerTempTable("CDR");
    val cdrData = sqlContext.sql("SELECT VISITOR_LOCN,CALL_DURATION_SEC,PHONE_NO,ERROR_CODE FROM CDR")
    val cdrTop10 = sqlContext.sql("SELECT PHONE_NO, COUNT(PHONE_NO) AS COUNT_DROP_OUT FROM CDR GROUP BY PHONE_NO ORDER BY COUNT(PHONE_NO) DESC")
    println("Call back the following number: " + cdrTop10.take(10).toList);
    //C. Find all the relevant information for the roaming provider (for the 10 most frequent drop out numbers)
    val join = cdrData.join(cdrTop10, cdrData("PHONE_NO") === cdrTop10("PHONE_NO"));
    println("Log for the roaming provider (10 most frequent drop outs) : " + join.collectAsList);
  }
```

Define the hdfs/spark shared configurations

Instantiate a spark configuration and start two nodes

Create a spark context

Create the schema for the dataframe

Load the schema and data in a dataframe

Generate the two reports

Report Output

The below screen shots show the reports corresponding to:

- The list of the 10CFFCD to call back urgently

```
Console
<terminated> CRDProject$ (1) [Scala Application] /usr/lib/jvm/jdk1.7.0_67/bin/java (Aug 16, 2016, 1:29:32 PM)
16/08/16 13:29:42 INFO TaskSetManager: Finished task 199.0 in stage 2.0 (TID 202) in 4 ms on localhost (199/200)
16/08/16 13:29:42 INFO Executor: Finished task 196.0 in stage 2.0 (TID 199). 3032 bytes result sent to driver
16/08/16 13:29:42 INFO TaskSetManager: Finished task 196.0 in stage 2.0 (TID 199) in 32 ms on localhost (200/200)
16/08/16 13:29:42 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have all completed, from pool
16/08/16 13:29:42 INFO DAGScheduler: ResultStage 2 (take at CRDProject.scala:48) finished in 1.666 s
16/08/16 13:29:42 INFO DAGScheduler: Job 1 finished: take at CRDProject.scala:48, took 2.775157 s
Call back the following numbers: List([4439088765,595], [8877123432,583], [880983451,579], [9900882345,575], [6600443311,572], [75566321,569], [7755321546,568], [1234567
16/08/16 13:29:42 WARN Column: Constructing trivially true equals predicate, 'PHONE_NO#2L = PHONE_NO#2L'. Perhaps you need to use aliases.
```

The result is a list of ordered arrays by drop out frequency. The first array element corresponds to telephone to call back. The second item represents the call drop frequency.

- The 10CFFCD log information to supply the roaming providers.

```
Console
<terminated> CRDProject$ (1) [Scala Application] /usr/lib/jvm/jdk1.7.0_67/bin/java (Aug 16, 2016, 1:29:32 PM)
16/08/16 13:29:49 INFO Executor: Finished task 198.0 in stage 9.0 (TID 823). 1979 bytes result sent to driver
16/08/16 13:29:49 INFO TaskSetManager: Finished task 198.0 in stage 9.0 (TID 823) in 18 ms on localhost (200/200)
16/08/16 13:29:49 INFO TaskSchedulerImpl: Removed TaskSet 9.0, whose tasks have all completed, from pool
16/08/16 13:29:49 INFO DAGScheduler: ResultStage 9 (collectAsList at CRDProject.scala:52) finished in 2.224 s
16/08/16 13:29:49 INFO DAGScheduler: Job 3 finished: collectAsList at CRDProject.scala:52, took 5.483749 s
Log for the roaming provider (10 most frequent drop outs) : [[6046,5,8877123432,0x860F16,8877123432,583], [6033,4,8877123432,0x789F17,8877123432,583], [6040,2,8877123432,
16/08/16 13:29:50 INFO SparkContext: Invoking stop() from shutdown hook
16/08/16 13:29:50 INFO SparkUI: Stopped Spark web UI at http://10.0.2.15:4040
```

The result is a list of ordered arrays by drop out frequency. Each element represents respectively: i) the call location, ii) the telephone number, iii) the error code, iv) a repeat of the telephone number, v) the number of drop out frequency.