Hupi-Streaming-Localisation-Torus

HUPI (http://www.hupi.fr/)

Co-funded by the Erasmus+ Programme of the European Union

TOWARD AN OPEN RESOURCES USING SERVICES

Géolocalisation en streaming - Geolocation streaming

Sujet - Subject

Nous allons récupérer en streaming les données de l'usine ecoles (http://ecoles.hupi.io/ (http://ecoles.hupi.io/)). Nous récupérerons les données et afficheront votre position géographique (country).

> *Here, we are going to collect in streaming the data of the ecoles platform (http://ecoles.hupi.io/ (http://ecoles.hupi.io/)). We will collect the data and display your location (country).*

Importation des libraries - Library import

```
:dp org.apache.spark %% spark-streaming-kafka % _
```

```
// Modules pour la liaison entre Kafka et Spark / Packages for linking Kafka et Spark
import org.apache.spark.streaming._
import org.apache.spark.streaming.kafka.KafkaUtils
```

SparkContext

```
// Si il n'y a pas de SparkContext, il faut le créer / Create a SparkContext if there is not :
//import org.apache.spark.SparkContext # Si il n'existe pas déjà / if it doesn't exist
//val sc = SparkContext(appName="StreamingKafka")
```

Data for geolocalisation

```
// We need 2 document to determine the location with the IP adress.

val user = "onrewind"
val geo_blocks = sc.textFile(s"hdfs://R610-1.pro.hupi.loc/user/$user/data/source/geolitecity_blocks")
val geo_location = sc.textFile(s"hdfs://R610-1.pro.hupi.loc/user/$user/data/source/geolitecity_location")

// location = [ idloc, pays, longitude, latitude ]
val location = geo_location.map(_.split(","))
                     .map(line => (line(0), List(line(1), line(5), line(6)))).persist()
// blocks = [ idloc, Ipmin, Ipmax ]
val blocks = geo_blocks.map(_.split(","))
                    .map(line => (line(2).replace("\"",""), line(0).replaceAll("\"","").toLong, line(1).replaceAll
location.count()
blocks.count()
```

```
// Conversion of ID adress
def ipToDec( ipString:String) : Long = {
    var sum:Long = 0
    var ipAddressInArray: Array[java.lang.String] = ipString.split("\\.")
    var i:Int = 0
    for( i <- 0 to ipAddressInArray.length - 1) {
      var   power:Int = 3 - i;
      var   ip:Long = ipAddressInArray(i).toLong
      sum += ip * Math.pow(256, power).toLong;
    }
    return sum
}
```

Liaison avec Kafka - Link with Kafka

```
val intervalle = 20 // Fenêtre de x secondes / x seconds window

val ssc =  new StreamingContext(sc, Seconds(intervalle))

val zkQuorum = "hupi.node1.pro.hupi.loc:2181"

val topic = Map("hupi_hupilytics" -> 1)

val identifiant = "Torus" // A modifier pour chaque personne / to modify for each user

val streamdata = KafkaUtils.createStream(ssc, zkQuorum, identifiant, topic)
```

```
// fonction d'affichage / Print function
def get_output(rdd: RDD[String]) = {
  println("Your IP adress : ")
  val li = rdd.collect()
  for(x <- li){
    println("- "+ x)
  }
  println("")
  println("")
}
```

```
// Map for visualization
val geo = widgets.GeoPointsChart(Seq((43.447868, -1.554658, "init")))
geo
```

```
// IP address

// We display all IP adress
val actions = streamdata.map(_._2).map(rdd => (rdd.substring(rdd.indexOf("\"ip\"")+6,rdd.indexOf("action")-3) ) )
actions.foreachRDD(l=> get_output(l))

// Then we calculate the location (latitude, longitude)
val geolocalisation = actions.map( l => blocks.filter( w=> ( w._2<=ipToDec(l) && w._3>=ipToDec(l) ) ).map(l=> l._1)
                             .map( l => location.filter(w=> (w._1==l) ).map(w=> (w._2(1).toDouble,w._2(2).toDouble)
                             .map( l => (l._1,l._2,"init"))
//geolocalisation.foreachRDD(l=> get_output(l.map(w=>w.toString) ) ) // for printing latitude and longitude
geolocalisation.foreachRDD(l=> geo.applyOn(l.collect())) // print on the above map
```

Lancement du programme - Program's launching

```
ssc.start()
val duration = 3*intervalle*1000.toLong  // 1 minute
ssc.awaitTerminationOrTimeout(duration)
```

```
ssc.stop(stopSparkContext=false,stopGracefully=true) // Stop streaming
```

```
ssc.getState()
```

Build: | **buildTime**-*Tue Mar 15 07:45:59 UTC 2016* | **formattedShaVersion**-*0.6.3-61efca56c2a45613ec609f350dc4911b166b3f28-SNAPSHOT* | **sbtVersion**-*0.13.8* | **scalaVersion**-*2.11.6* | **sparkNotebookVersion**-*0.6.3* | **hadoopVersion**-*2.6.0-cdh5.5.2* | **jets3tVersion**-*0.7.1* | **jlineDef**-*(jline,2.12)* | **sparkVersion**-*1.5.0* | **withHive**-*true* | **withParquet**-*true* |.