Lab Distributed Big Data Analytics

Worksheet-2: **Getting started with Spark**

Dr. Hajira Jabeen, **Gezim Sejdiu**, **Prof. Dr. Jens Lehmann**

May 2, 2017

In this lab we are going to perform basic RDD and DataFrame operations such as actions and transformations (described on "Spark Fundamentals I"). We are going to cover caching as well in order to speed up any iterative job that we will have during our lab.

IN CLASS

1. Spark basic RDDs & DataFrames operations
   a. After a file (page_links_simple.nt.bz2) have been downloaded, unzipped, and uploaded on HDFS under /yourname folder you may need to create an RDD out of this file. This is created using the spark context and textFile method. Since spark transformations are considered being lazy evaluation, means nothing really happens there. We have to perform this just to tell spark that we want to create an RDD, containing data from the file.
   b. Let's perform some RDD actions on this file by counting the number of triples in the RDD.
   c. Use the filter transformation to return a new RDD with a subset of the triples on the file by checking if the first row contains "#", which on .nt file represent a comment.
   d. Since the data is going to be type of .nt file which inside contains rows of triples in format `<subject> <predicate> <object>` we may need to transform this data into a different format of representation. By using map function we will transform the data into `(Subject, 1)` which we are going to use for counting the subject distributions.
   e. After we have transformed our RDD into (Subject, 1) - which assign every subject a counter (in this case 1) we may need to count how many time the specific subject have been used on our dataset. To do so, we will use reduceByKey
   f. Collect and print subjects and their frequencies.

--Solution--

```
object App {
def main(args: Array[String]) = {
    val input = "src/main/resources/rdf.nt" // args(0)
    println("======================================")
    println("|        Triple reader example       |")
    println("======================================")
    val sparkSession = SparkSession.builder
      .master("local[*]")
                                                  .config("spark.serializer",
"org.apache.spark.serializer.KryoSerializer")
      .appName("WordCount example (" + input + ")")
      .getOrCreate()
    val data = 1 to 100
    val disdata = sparkSession.sparkContext.parallelize(data)

    val triples = sparkSession.sparkContext.textFile(input)
    triples.take(5).foreach(println(_))
    triples.cache()

    val nrOfTriples = triples.count()
    println("Count:  " + nrOfTriples)

    val removeCommentRows = triples.filter(!_.startsWith("#"))
    removeCommentRows.take(5).foreach(println(_))

    val parsedTriples = removeCommentRows.map(parsTriples)
    parsedTriples.take(5).foreach(println(_)
    val subject = parsedTriples.map(f => (f.subject, 1))

    val subjectCOunt = subject.reduceByKey(_ + _)
    println("SubjectCount \n")
    subjectCOunt.take(5).foreach(println(_))

    sparkSession.stop
}

def parsTriples(parsData: String): Triples = {
    val subRAngle = parsData.indexOf('>')
    val predLAngle = parsData.indexOf('<', subRAngle + 1)
    val predRAngle = parsData.indexOf('>', predLAngle + 1)
    var objLAngle = parsData.indexOf('<', predRAngle + 1)
    var objRAngle = parsData.indexOf('>', objLAngle + 1)
    if (objRAngle == -1) {
      objLAngle = parsData.indexOf('\"', objRAngle + 1)
      objRAngle = parsData.indexOf('\"', objLAngle + 1)
    }

    val subject = parsData.substring(1, subRAngle)
    val predicate = parsData.substring(predLAngle + 1, predRAngle)
    val `object` = parsData.substring(objLAngle + 1, objRAngle)

    Triples(subject, predicate, `object`)
  }

case class Triples(subject: String, predicate: String, `object`: String)
}
```

AT HOME

1. Read and explore
   a. Spark Programming Guide
   b. RDD API Examples
   c. DataFrame API Examples
   d. Spark Streaming Programming Guide
   e. Spark Cluster Overview
   f. Spark Configuration, Monitoring and tuning
2. RDF Class Distribution - count the usage of respective classes of a RDF dataset.
   Hint: Class fulfils the rule(`?predicate = rdf:type && ?object.isIRI()`).
   a. Read the nt file into an RDD of triples.
   b. Apply map function for separating triples into (Subject, Predicate, Object)
   c. Apply filter transformation for defining the respective classes.
   d. Count the frequencies of Object and map them into (Object, count).
   e. Return the top 100 classes used in the dataset.
   f. Try to do steps (a - e) within DataFrames.
3. Further readings
   a. Spark: Cluster Computing with Working Sets
   b. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing
   c. Clash of the Titans: MapReduce vs. Spark for Large Scale Data Analytics