Infrastructure Big Data

Examen

Durée: 1h, documents autorisés

Understanding questions (6 points)

Instructions: In the MCQ, several answers may be possible on some questions.

Question 1

We consider the execution of the wordcount application on a Spark cluster composed of several servers. The execution should run faster than a centralized sequential version of wordcount :

- [ ] for all input files
- [ ] for input files with a minimal size
- [ ] for input files up to a maximal size

Question 2

In a Spark/HDFS cluster on top of Linux, what are the strong requirements (meanning it cannot run without it)?

- [ ] ssh server installed on each node
- [ ] same type of processor on each node
- [ ] same number of processor on each node
- [ ] Java installed on each node

Question 3

In a distributed Spark/HDFS deployment :

- [ ] the Master daemon is running on the master node
- [ ] the NameNode daemon is running on all slave nodes
- [ ] the NameNode daemon is running on the master node
- [ ] the Worker daemon is running on all slave nodes
- [ ] the DataNode daemon is running on the master node

Question 4

In Spark, the `reduceByKey()` method:

- [ ] results into a dataset where each key is unique
- [ ] results into a dataset where the number of different keys is reduced

- ☐ gathers all the pairs on the master node
- ☐ distributes all the pairs over the slave nodes

Question 5

In Spark, the `reduceByKey (func)` method :

- ☐ can be applied to any JavaRDD
- ☐ can only be applied to a JavaPairRDD
- ☐ func is a function which aggregates 2 pairs into one pair
- ☐ func is a function which aggregates 2 values into one value

Question 6

In Spark, the `mapValues (func)` method:

- ☐ can be applied to any JavaRDD
- ☐ can only be applied to a JavaPairRDD
- ☐ func is a function which transforms a value into another value
- ☐ func is a function which aggregates 2 values into one value

Problem (14 points)

We consider a large file including meteorological data. Each line in the file provides an observation:

```
day month year temperature city
```

For instance : `12 07 1995 30 Paris` is an observation that on the 12th of july 1995, the temperaiure was 30 degrees in Paris.

You can extract the fields from a line L with `L.splitO[0]`, `L.splitO[1]`, etc.
Your algorithms start with the following `RDD` (initialized with a file available in HDFS) :

```
JavaRDD<String> data = sc.textFile(inputFile);
```

Question 1

We want to compute for each city the average temperature. What's wrong in the following algorithm ? justify (1 point)

```java
JavaPairRDD<String, Integer> average = data.mapToPair(
    s -> new Tuple2<String, Integer> (s.split()[4],
                                      Integer.parseInt(s.split()[3])))
    .reduceByKey((t1,t2)->(t1 + t2)/2);
```

Question 2

Propose a correct solution to the previous question (1 point)

Question 3

We want to compute for each year the number of cities where temperature is measured (2 points).

Result should be a `JavaPairRDD<year, count>`

Question 4

We want to extract the list of cites where 2022 was the warmest year (considering the maximal temperature) (3 points).

Result should be a `JavaRDD<city>`