TP Spark programming

Daniel Hagimont
daniel.hagimont@irit.fr

The goal of this labwork is to program an application with Spark.

1. Installation

- FYI, I made it on my laptop with jdk1.8.0_202 and spark-2.4.3-bin-hadoop2.7
- pre-requisite
    - you should have Java installed and the JAVA_HOME variable defined
    - you should have hadoop installed as in the previous labwork
- install Spark
    - untar the spark-2.4.3-bin-hadoop2.7.tgz archive
    - define environment variables

```
export SPARK_HOME=<path>/spark-2.4.3-bin-hadoop2.7
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

2. Development

- you can use vscode to develop applications
    - create a Java project
    - Add jars to your project

```
$SPARK_HOME/jars/spark-core_2.11-2.4.3.jar
$SPARK_HOME/jars/scala-library-2.11.12.jar
$SPARK_HOME/jars/hadoop-common-2.7.3.jar
```

- you must package your application in a jar
    - assuming that you have a package **foo** in your project **spark**

```
jar cf wc.jar -C ~/eclipse-workspace/hadoop/bin foo
```

3. Execution

- your application (source code) should refer to local files (local to the file system)
- spark-submit --class <classname> --master local <jarfile>

4. Test the WordCount application

Here is the code of the WordCount application in Spark :

```
public class WordCount {

        public static void main(String[] args) {
          String inputFile = "filesample.txt";
          String outputFile = "result";

          SparkConf conf = new SparkConf().setAppName("WordCount");
          JavaSparkContext sc = new JavaSparkContext(conf);

          long t1 = System.currentTimeMillis();

          JavaRDD<String> data =
                        sc.textFile(inputFile).flatMap(s -> Arrays.asList(s.split(" ")).iterator());

          JavaPairRDD<String, Integer> counts =
                        data.mapToPair(w -> new Tuple2<String, Integer>(w,1)).
                        reduceByKey((c1,c2) -> c1 + c2);

          counts.saveAsTextFile(outputFile);

          long t2 = System.currentTimeMillis();

          System.out.println("======================");
          System.out.println("time in ms :"+(t2-t1));
          System.out.println("======================");

        }
}
```

5. Treatment of meteorology data

Implement the same application as in the previous labwork (Hadoop), but with Spark.

NB : you can browse Spark API : https://spark.apache.org/docs/latest/api/java/index.html