

TP Spark streaming programming

Daniel Hagimont
daniel.hagimont@irit.fr

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

1. Installation

The same as in the previous lab.

2. Development

For vscode, add (from the previous lab) the following jar in your project :

\$SPARK_HOME/jars/spark-streaming_2.11-2.4.3.jar

\$SPARK_HOME/jars/apache-log4j-extras-1.2.17.jar

3. Execution

The same as in the previous lab.

4. Test the WordCount application

```
import java.util.Arrays;
import org.apache.log4j.Level;
import org.apache.log4j.Logger;
import org.apache.spark.SparkConf;
import org.apache.spark.streaming.Durations;
import org.apache.spark.streaming.api.java.JavaDStream;
import org.apache.spark.streaming.api.java.JavaPairDStream;
import org.apache.spark.streaming.api.java.JavaReceiverInputDStream;
import org.apache.spark.streaming.api.java.JavaStreamingContext;
import scala.Tuple2;

public class WordCountStreaming {

    public static void main(String args[]) {

        Logger.getLogger("org").setLevel(Level.OFF);
        Logger.getLogger("akka").setLevel(Level.OFF);

        // Create the context with a 1 second batch size
        SparkConf sparkConf = new SparkConf().setAppName("WordCountStreaming");
        JavaStreamingContext ssc = new JavaStreamingContext(sparkConf, Durations.seconds(1));

        // Create a JavaReceiverInputDStream on target ip:port and count the
        // words in input stream of \n delimited text (eg. generated by 'nc')
        JavaReceiverInputDStream<String> lines = ssc.socketTextStream("localhost", 9999);
        JavaDStream<String> words = lines.flatMap(x -> Arrays.asList(x.split(" ")).iterator());
        JavaPairDStream<String, Integer> wordCounts = words.mapToPair(s -> new Tuple2<>(s, 1))
            .reduceByKey((i1, i2) -> i1 + i2);

        wordCounts.print();

        ssc.start();
        try {
            ssc.awaitTermination();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

You can test this application with :

- in a first terminal, run "nc -lk 9999" – you will type words in this terminal
- in a second terminal, launch the Spark Streaming applications
 - you need at least 2 threads for receiving data and processing data
 - spark-submit --class WordCountStreaming --master local[2] wc.jar

5. Different versions of WordCount

Modify the application to compute the word count every 10 seconds over the last 10 seconds. It's just one line, using the `reduceByKeyAndWindow()` transformation.

Modify the application to compute the cumulated word count (from the beginning of the application). It's just few lines, using the `updateStateByKey()` transformation.

6. A monitoring application

You are given a `CPUProbe` application which accepts TCP a connection and sends on this TCP connection the monitored CPU load of 10 nodes.

You have to implement a Spark Streaming application which computes, every 5 seconds over the last 10 seconds, the average CPU load of each node.

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