

# TP Hadoop programming

Daniel Hagimont  
daniel.hagimont@irit.fr

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

## 1. Installation

- FYI, I made it on my laptop with jdk1.8.0\_202 and hadoop-2.7.1. Hadoop only compiles with java-8 and only runs with java-8 and java-11.

- pre-requisite

- you should have Java installed and the JAVA\_HOME variable defined
- you should have your ssh keys configured to allow ssh to localhost

- install Hadoop

- untar the hadoop-2.7.1.tar.gz archive
- define environment variables

```
export HADOOP_HOME=<path>/hadoop-2.7.1
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
```

- to execute Hadoop in local mode, you have to configure the following files

- <hadoop-home>/etc/hadoop/core-site.xml:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

- <hadoop-home>/etc/hadoop/hdfs-site.xml:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

- <hadoop-home>/etc/hadoop/hadoop-env.sh:  
hardcode the JAVA\_HOME variable

- you can use vscode to develop applications, create a Java project

- you have to add the following jars in your java projet
  - <hadoop-home>/share/hadoop/common/hadoop-common-2.7.1.jar
  - <hadoop-home>/share/hadoop/mapreduce/hadoop-mapreduce-client-common-2.7.1.jar
  - <hadoop-home>/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.7.1.jar

## 2. Test the wordCount application

Here is the code of the WordCount application :

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

    public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>{
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        public void map(Object key, Text value, Context context )
            throws IOException, InterruptedException {
            String tokens[] = value.toString().split(" ");
            for (String tok : tokens) {
                word.set(tok);
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values, Context context )
            throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        long t1 = System.currentTimeMillis();
        int res = job.waitForCompletion(true) ? 0 : 1;
        long t2 = System.currentTimeMillis();
        System.out.println("time in ms =" + (t2 - t1));
        System.exit(res);
    }
}
```

Here is how you can test it, assuming that you have a package **foo** in your project **hadoop**.

```
hdfs namenode -format
start-dfs.sh
jps
hdfs dfs -mkdir /input
hdfs dfs -put filesample.txt /input
jar cf wc.jar -C <path-to-vscode-project>/bin foo    (replace foo by "." if no package)
hadoop jar wc.jar foo.WordCount /input /output
hdfs dfs -cat /output/*
stop-dfs.sh
```

You can test it with any text file.

### 3. Treatment of meteorology data

You are given a file containing meteorological data about cities all over the planet. You have to implement two map-reduce jobs that :

- compute the maximum temperature for each year
- compute the number of months which had a temperature higher than a given value (a parameter).
- the format of a line in the file is : day : month : year : temperature : city