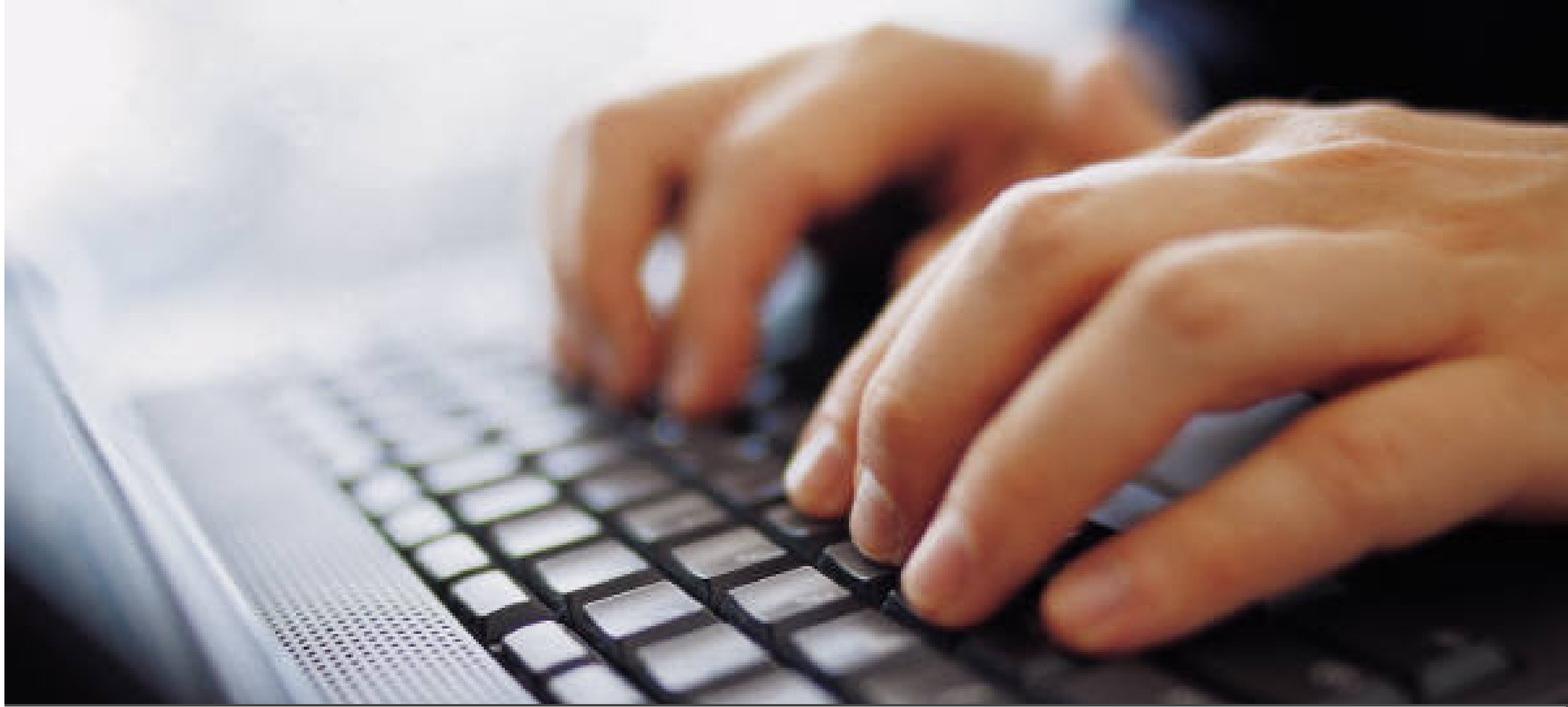
Hadoop / Big Data

Benjamin Renaut <renaut.benjamin@tokidev.fr>

MBDS

2019 – 2020



TP 1

Map/Reduce methodology – Hadoop development

Setup

- Install VirtualBox (https://www.virtualbox.org/).

- Import the exercice session virtual machine .ova file.
  It is a GNU/Linux Virtual Machine, running Debian x64.

- Start the virtual machine.

- Login with the following credentials. Login: `mbds`, Password: `password`. Read the explicative doc describing how to connect using PuTTY/SSH for convenience.

- Start Hadoop with the following command:

```
start-hadoop
```

  (ignore *warnings* if any)

Setup

- **Check Hadoop is indeed working properly by doing:**

  ```
  hdfs dfsadmin -report
  ```

  The command checks the HDFS status. It should print, among other things:

  ```
  Live datanodes (1)
  ```

  … with various information on the Data Node following.

- **We will now compile the course example (the word counter).**

  The objective is to make sure your development/compilation environment is functional for the next steps, and to familiarize yourself with the process.

Setup

- To execute your Hadoop programs, you will need to generate a jar file, and then copy it to the virtual machine to execute it.

- For the development itself and to generate the jar, you have several options:

  - Use IntelliJ, loading Hadoop dependencies using Gradle. A project archive is provided in this case.
  - Use IntelliJ, loading Hadoop dependencies using Maven. A project archive is provided in this case as well.
  - Use Eclipse, loading Hadoop dependencies using Maven. A project archive is provided in this case as well.
  - Use another IDE, loading Hadoop dependencies manually or using Gradle or Maven.
  - Compile the code and build the jar yourself manually directly on the virtual machine.

Setup

- If you with to use IntelliJ with Gradle, download and import the following project on your machine: http://cours.tokidev.fr/bigdata/tps/hadoop_intellij_project_gradle.zip

- If you with to use IntelliJ with Maven, download and import the following project on your machine: http://cours.tokidev.fr/bigdata/tps/hadoop_intellij_project_maven.zip

- If you with to use Eclipse with Maven, download and import the following project on your machine: http://cours.tokidev.fr/bigdata/tps/hadoop_eclipse_project_maven.zip

- If you picked one of those three options, you can skip the following setup slides.

Setup

- If you wish to use another IDE, you can download the Java source code directly here:
  http://cours.tokidev.fr/bigdata/tps/wordcount.zip

- And in order to load the Hadoop dependencies, you can either:

  - Use the following Gradle configuration:
    http://cours.tokidev.fr/bigdata/tps/build.gradle.txt

  - Use the following Maven configuration:
    http://cours.tokidev.fr/bigdata/tps/pom.xml.txt

  - Load the jar dependencies manually in your project from the following archive:
    http://cours.tokidev.fr/bigdata/tps/hadoop_3.1.3_deps.zip

Setup

- **Finally, if you wish to directly compile and build the jar files in the virtual machine, you will need to copy your code there or directly edit your code inside the VM.**

- **Once the code is on the VM, you can then build the class files and then the jar file, bearing in mind the path hierarchy must match your classpath.**

- **As an example, to compile and build the jar file manually for this first example (the word counter), you would do in the VM:**

  - **Downloading the code**
    ```
    wget http://cours.tokidev.fr/bigdata/tps/wordcount.zip
    ```
  - **Deflating it**
    ```
    unzip wordcount.zip && rm -rf wordcount.zip
    ```

Setup

- Compiling it

```
javac WCount.java WCountMap.java WcountReduce.java
```

- Constructing the classpath hierarchy and moving the class files into it:

```
mkdir -p org/mbds/hadoop/wordcount
mv WCount*.class org/mbds/hadoop/wordcount/
```

- And, finally, generate the jar and clean up:

```
jar -cvf mbds_wcount.jar -C . org
rm -rf org
```

Setup

- No matter which setup you picked, you should now be able to compile and build the jar file for the example and move it to the VM.

- The code itself is that of the example shown in the course: the word counter.

- Start now by building the jar and copying it to the virtual machine; refer to the explicative doc for file copying using SSH if needed.

Setup

- Once your jar file has been moved to the VM, you should now download the example poem text file, and then move it to the HDFS filesystem in order for your program to be able to read it.

- The next steps assume your jar file is named « wordcount.jar » ; adjust the commands are required.

- Start by downloading the poem file in the virtual machine using the following command:

```
wget http://cours.tokidev.fr/bigdata/tps/poeme.txt
```

- Then move this file to the HDFS filesystem:

```
hadoop fs -put poeme.txt /
```

Setup

- **Then check it is indeed available on HDFS:**
```
hadoop fs -ls /
```
  (the « poeme.txt » file must appear)

- **Then, execute your program:**

```
hadoop jar wordcount.jar org.mbds.hadoop.tp.WCount /poeme.txt /results
```

  (adjust the filename and classpath as needed; also, note that we use /results to store our final results on HDFS, and that the program will fail if that directory already exists)

- **We can then check our results on HDFS, for example by doing:**
```
hadoop fs -ls /results
hadoop fs -cat /results/*
```

TP – Anagrams

- You must now write your own Hadoop program.

- <u>OBJECTIVE:</u> you are provided with a list of common english words. We wish to pinpoint which words are anagrams of one another.

  As a reminded, two words are anagrams if their letters are the same but in different orders (such as « melon » and « lemon », for example).

- Download the common words file using the command:

```
wget http://cours.tokidev.fr/bigdata/tps/common_words_en_subset.txt
```

- <u>Remark:</u> you can use Streaming (as seen during the course) if you feel more comfortable using another langage instead of Java. The main purpose here is to implement an efficient M/R algorithm to solve the problem.

Remarks

- **Apply the map/reduce methodology as described during the course.**

  As yourself: what should my key be ? What is the common factors between anagrams, and how could that help ?

- **Do not forget that you will need to copy the input data file to HDFS for processing.**

TP – Sales analysis

- You must now write another map/reduce Hadoop program.
- We will work now on a sales record; you can download the file here:

  http://cours.tokidev.fr/bigdata/tps/sales_world_10k.csv
- It is a CSV file. Please check its format; it has many columns. Also note it has a header ligne, which may be problematic when loading the file from your Hadoop implementation.
- We are interested in specifically five columns: the sales region (« Region »), the sales country (« Country »), the type of item bought (« Item Type »), the sales channel (« Sales channel » : online or offline), and finally the total sale profit (« Total profit »).

TP – Sales analysis

- **Your program should be able to perform various analysis tasks.**

- **It is <u>highly advised</u> to allow your program to adapt its behaviour / pick an analysis task depending on a command line argument.**

- **Alternatively and if you don’t see how to perform this, you can implement a separate software for each of the required tasks.**

- **<u>Remark</u>: the Configuration object that we create in the Driver class can also be obtained later on from the Map and Reduce classes; and it also allows passing values between those classes, which may prove useful for this program. You are advised to look for the way to obtain the Configuration instance from the context object in the map and reduce classes, as well as looking up the getter and setter methods for the Configuration class.**

TP – Sales analysis

- Your software should be able to perform the following tasks:
  - Obtain the total profit for any given world region.
  - Obtain the total profit for any given country.
  - Obtain the total profit for any given item type.
  - For each item type, provide:
    - How many sales were performed online.
    - How many sales were performed offline.

    … and for each of those quantities, how much the combined total profit for those sales was.