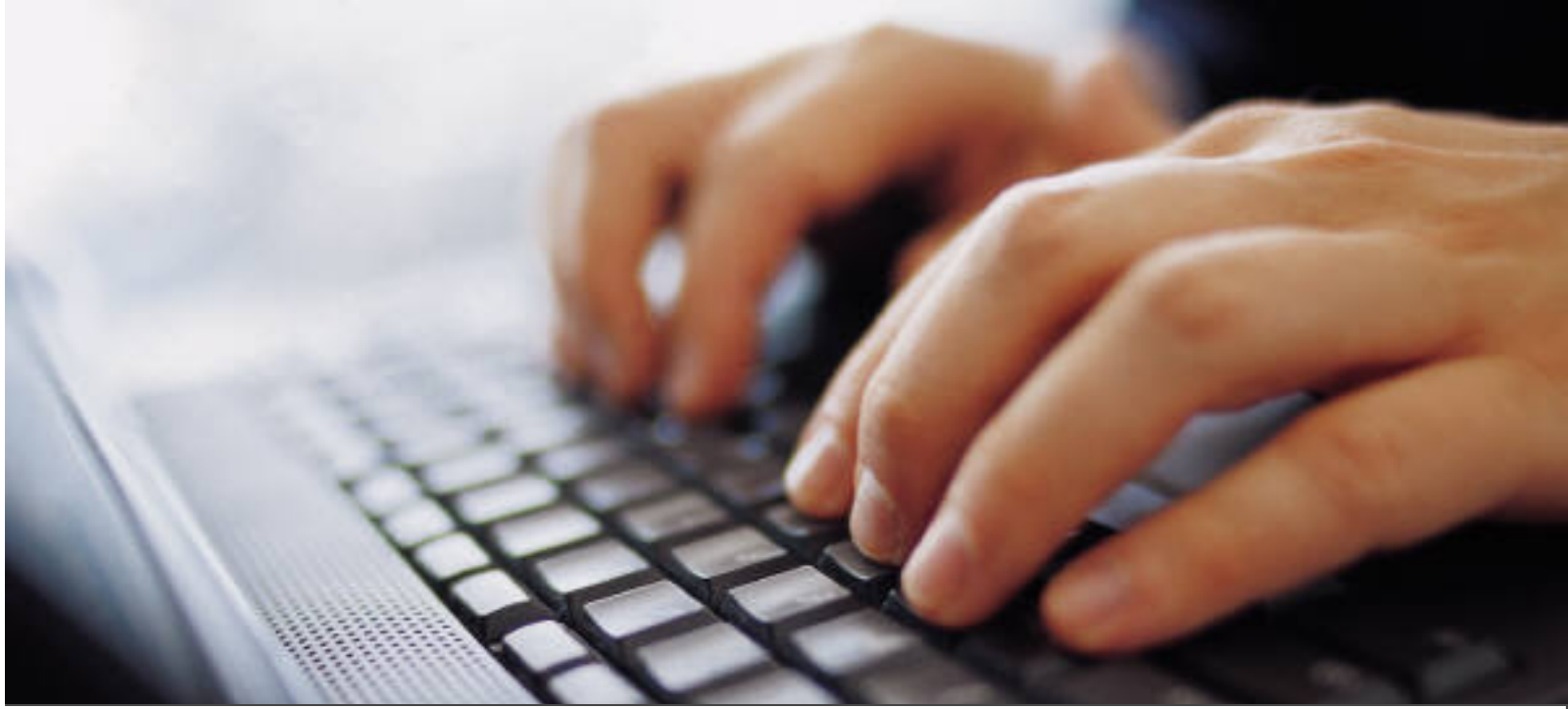
Hadoop / Big Data

Benjamin Renaut <renaut.benjamin@tokidev.fr>

MBDS

2019 - 2020



TP 6

Apache Spark

Setup

- **For this exercise, yet again the same virtual machine will be used.**
- **You can start by launching the virtual machine. It is not necessary (at this step at least) to start Hadoop (neither Yarn nor HDFS).**
- **Spark has already been installed in the VM.**

Part 1 – Spark shell

- **Your first step, in order to familiarize yourself with the Spark shell, will be to execute the example shown during the course (the anagrams detection).**

- **Start by launching the Spark shell in local mode, simulating a two-nodes cluster:**

  pyspark --master "local[2]"

- **Then, attempt to load and check out the file of common English words used previously (make sure it is available in the local filesystem):**

```
words=sc.textFile('common_words_en_subset.txt')
words.collect()
```

  Observe the results; make sure the data has been successfully loaded.

Part 1 – Spark shell

- **Then, run the map:**

```
tuples=words.map(lambda x: (''.join(sorted(list(x))), x))
tuples.collect()
```

Observe the results.

- **Then, run the groupByKey (the rough equivalent to Hadoop's « shuffle »):**

```
grouped=tuples.groupByKey().mapValues(lambda x: list(x))
grouped.collect()
```

Observe the results.

Part 1 – Spark shell

- **Then, run filter:**

```
filtered=grouped.filter(lambda x: len(x[1])>1)
filtered.collect()
```

Observe the results.

- **Afterwhich you can run the reduceByKey:**

```
res=filtered.reduceByKey(lambda a,b: "%s, %s" % (a, b))
res.collect()
```

Observe the results.

Part 1 – Spark shell

- **Finally, also apply the reduce to the unfiltered RDD and save both datasets on disk:**

```
res2=grouped.reduceByKey(lambda a,b: "%s, %s" % (a, b))
res.saveAsTextFile('file:///home/mbds/res-words-filtered')
res2.saveAsTextFile('file:///home/mbds/res-words-unfiltered')
```

  Make sure you indeed managed to export the results in the two directories « res-words-filtered » and « res-words-unfiltered ».

- **You can also of course try out various other operations described during the course to familiarize yourself with them.**

Part 2 – Spark development

- **You must now implement a first Python Spark program, to solve the « common friends » issue outlined during the first Hadoop course.**

- **Your input data:**

  `http://cours.tokidev.fr/bigdata/tps/tp5_friends.txt`

  You can download said data to the VM using:

  `wget http://cours.tokidev.fr/bigdata/tps/tp5_friends.txt`

- **Remember that each line describes a social network user followed by his list of friends; your objective is to pinpoint all common friends in the social network (for all possible couples of distinct users).**

Part 2 – Spark development

- **You then must implement a Spark version of the graph breadth-first-search algorithm described – and implemented for Hadoop – in previous course material. Remember that you will need to loop and implement a stopping condition once all nodes are flagged as having been processed.**

- **The input data can be downloaded to the VM using:**

```
wget http://cours.tokidev.fr/bigdata/tps/tp5_graph.txt
```

- **Test your program first in local mode (make sure to simulate at least two nodes); then in Hadoop mode (you will then need to start Hadoop first).**

- **It is advised to start with a first, one-step graph run instead of the whole looped breadth-first-search logic directly for ease of debug.**

Part 2 – Spark development

- **Finally, you must develop the same sales analysis software that was written during the first practical exercise session.**

- **Remember that your program must be able to perform four different analysis types; and allow the user to pick which using a command line argument. As when running an Hadoop program, here too you can specify additional command line argument; using a synopsis similar to:**
  `spark-submit --master "local[2]" FICHIER.py arg1 arg2…`
  … you can then retrieve those command line arguments from Python using the `sys.argv` global variable (after doing `import sys`).

- **The input data is identical to the one previously used:**

  http://cours.tokidev.fr/bigdata/tps/sales_world_10k.csv

Part 2 – Spark development

- **For the development tasks, you can either write your code on your host machine and then either copy the file or copy-paste in the VM; or alternatively write your code in the VM directly.**

- **You can also attempt to install Spark on your host machine if you so wish (which may prove challenging on Windows systems).**

- **On the virtual machine, you can test your programs by doing:**
  `spark-submit --master "local[2]" FICHIER.py`

- **If you programs work, it is advised to then attempt to run them in « true » Hadoop mode (after starting Hadoop) using, for example:**

```
spark-submit --master "yarn" FICHIER.py
```