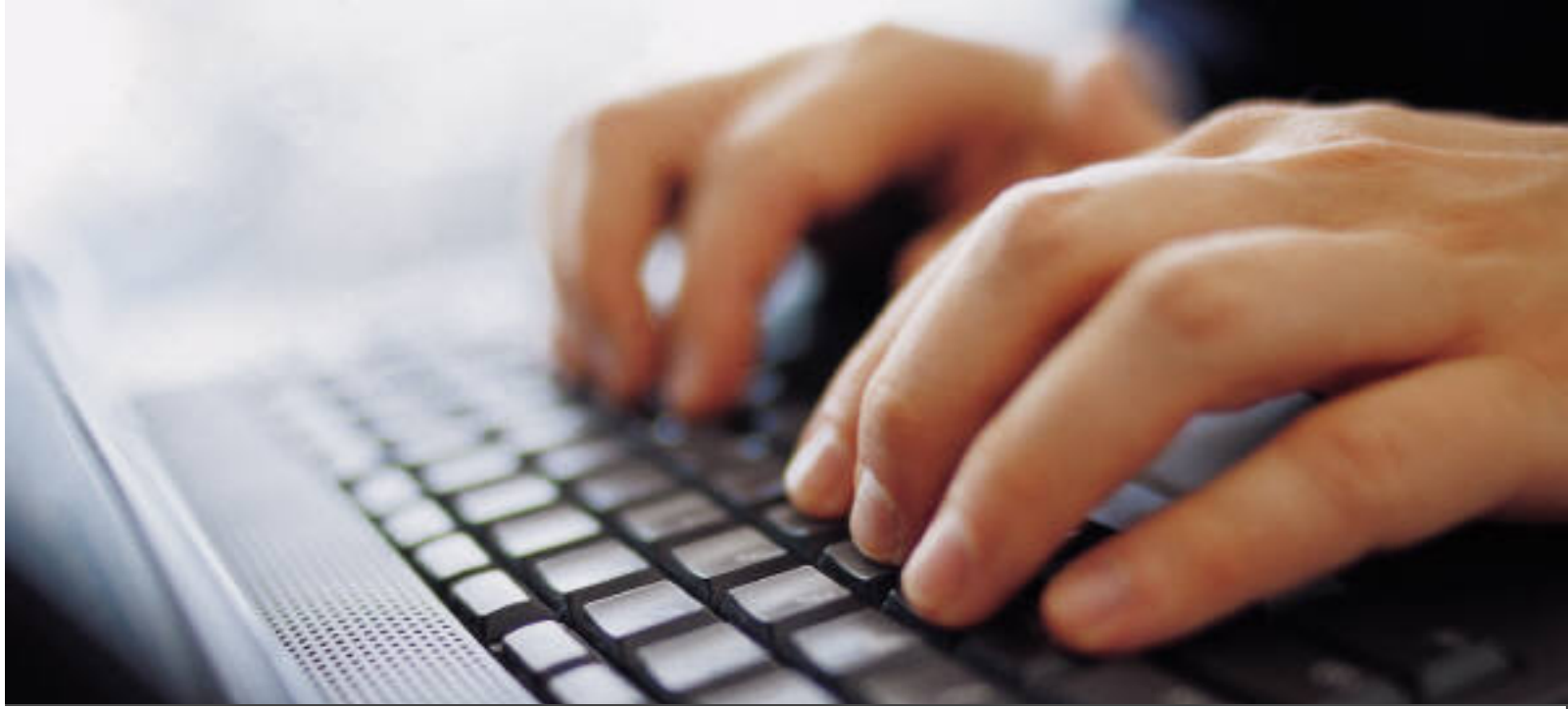
Hadoop / Big Data

Benjamin Renaut <renaut.benjamin@tokidev.fr>

MBDS

2019 - 2020



TP 6

Apache Spark

Préparation du TP

- **Pour ce TP, on va utiliser la même machine virtuelle que pour les TPs précédents.**

- **Commencez par démarrer la machine virtuelle. Il n'est pas nécessaire de lancer Hadoop / HDFS.**

- **Spark a d’ores et déjà été installé dans la machine virtuelle.**

Partie 1 – Le shell

- **Votre premier objectif, pour vous familiariser avec Spark, est d'exécuter un des exemples du cours: le détecteur d'anagrammes.**

- **Commencez par lancer le *shell* Spark python en mode local, en émulant 2 nœuds *worker*:**

  pyspark --master "local[2]"

- **Puis, tentez de charger le fichier des mots communs (assurez-vous de l'avoir bien à disposition dans le répertoire courant, pas sur HDFS):**

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

  Observez le résultat; vérifiez que les données soient correctement chargées.

Partie 1 – Le shell

- **Puis, lancez le map:**

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

Observez le résultat.

- **Lancez ensuite le group by (l'équivalent du « shuffle » Hadoop):**

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

Observez le résultat.

Partie 1 – Le shell

- **Ensuite, lancez le filtre:**

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

Observez le résultat.

- **Lancez ensuite le reduce:**

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

Observez le résultat.

Partie 1 – Le shell

- **Enfin, effectuez le reduce également sur le RDD non-filtré et sauvegardez les données sur le disque:**

```
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')
```

  Vérifiez que vous ayez bien les résultats dans les deux répertoires « res-words-filtered » et « res-words-unfiltered » dans le home de l'utilisateur mbds.

- **N'hésitez pas à essayer d'autres opérations vues en cours sur les données et à observer ce qui se passe.**

Partie 2 – Développement

- **Vous devez développer un premier programme Spark, en python, capable de résoudre le problème du réseau social et des « amis en commun » vu en cours précédemment.**

- **Vos données d'entrée:**

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

  Vous pouvez les télécharger sur la VM avec:

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

- **Pour mémoire, chaque ligne indique un utilisateur suivi de la liste de ses amis; vous devez appliquer l'algorithme map/reduce vu en cours pour obtenir, pour chaque couple d'utilisateurs, leur liste d'amis en commun.**

Partie 2 – Développement

- **Vous devez également développer une version Spark/Python du parcours de graphe vu précédemment en cours. Votre script doit lui-même vérifier, *via* l'analyse des RDDs produits, si le parcours est complètement terminé (tous les nœuds à noir), et continuer jusqu'à ce que ça soit le cas.**

- **Les données d'entrées sont téléchargeables avec:**

```
wget http://cours.tokidev.fr/bigdata/tps/tp6_graph.txt
```

- **Testez votre programme d'abord en mode local, puis en mode Hadoop/Yarn.**

- **Un conseil: commencez par développer et tester une étape du parcours avant d'implémenter la logique de boucle jusqu'à ce que la condition soit respectée.**

Partie 2 – Développement

- **Enfin, vous devez pour finir développer le même programme d'analyse de ventes que celui qui a été demandé dans le TP1.**

- **Pour mémoire, celui-ci doit proposer 4 modes d'analyse différents; et le mode d'analyse doit être choisi au moment du lancement par l'utilisateur. Comme pour Hadoop, il est possible de spécifier des arguments additionels; par exemple:**
  `spark-submit --master "local[2]" FICHIER.py arg1 arg2…`
  … et vous pouvez récupérer ces arguments en Python par le biais de la variable `sys.argv` (aprés avoir `import sys`).

- **Les données d'entrée sont identiques à celles du TP1:**

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

Partie 2 – Développement

- **Pour les tâches de développement, vous pouvez rédiger votre code Python sur la machine hôte et le copier/coller ou le déplacer *via* SSH/WinSCP sur la machine virtuelle.**

- **Vous pouvez également tenter d'installer Spark sur votre machine hôte (peu fiable sous Windows).**

- **Sur la VM, vous pouvez tester votre programme en faisant:**
  `spark-submit --master "local[2]" FICHIER.py`

- **S'il fonctionne, tentez ensuite de démarrer Hadoop (`start-hadoop`) et d'exécuter votre programme *via* Hadoop/Yarn (pensez à modifier vos emplacements fichiers):**

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