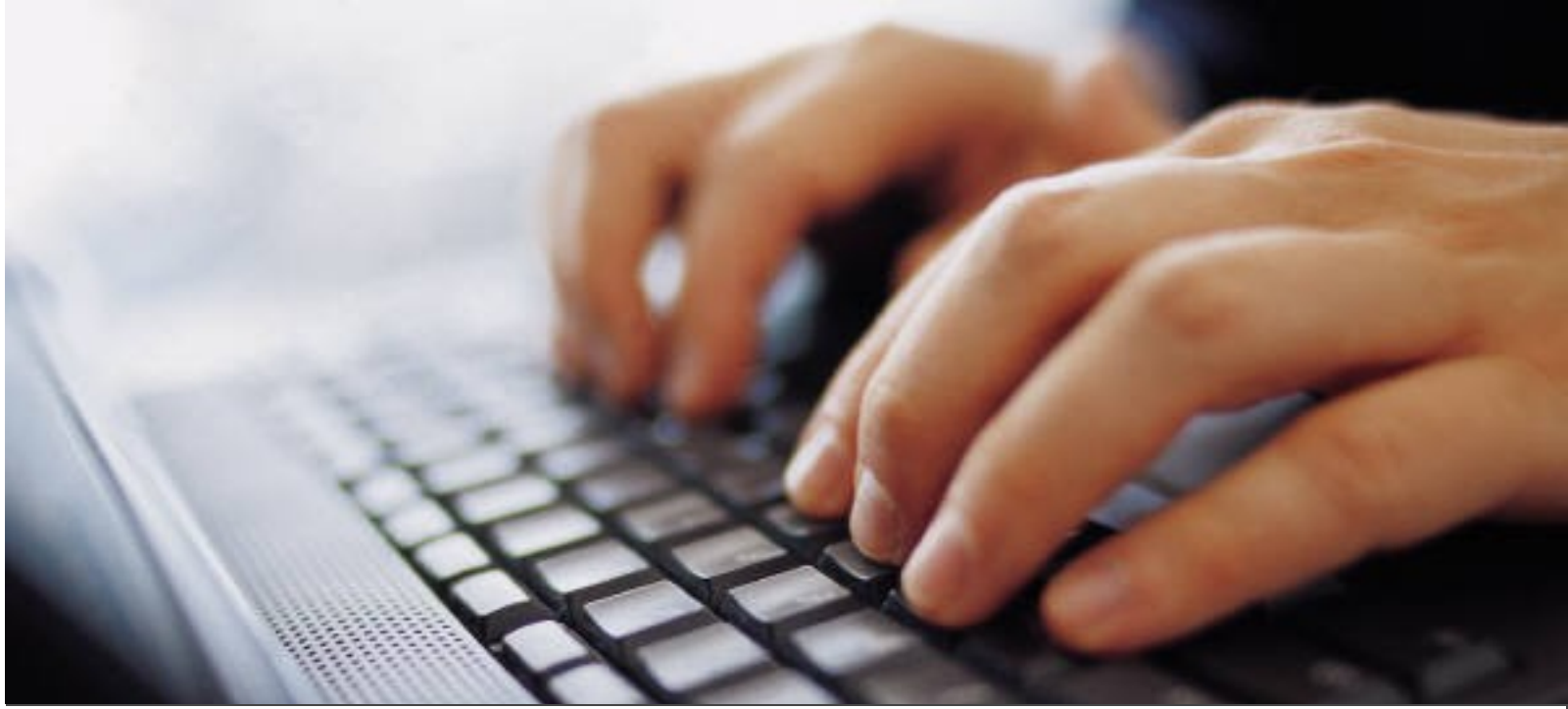
Hadoop / Big Data

Benjamin Renaut <renaut.benjamin@tokidev.fr>

MBDS

2013 - 2014



TP 1

Méthodologie Map/Reduce - programmation Hadoop.

Préparation du TP

- **Installer VirtualBox (https://www.virtualbox.org/).**
- **Importer la machine virtuelle .ova du TP.**
  Il s'agit d'un système GNU/Linux Debian, x86, sur lequel Hadoop a été pré-installé.
- **Démarrer la machine virtuelle.**
- **S'identifier avec le login: `root`, mot de passe: `mbds`.**
- **Démarrer Hadoop avec la commande:**

  ```
  start-all.sh
  ```

  (ignorer les *warnings*)

Préparation du TP

- **Vérifier le bon fonctionnement de Hadoop en exécutant la commande:**

  ```
  hdfs dfsadmin -report
  ```

  La commande vérifie le statut de HDFS. Elle devrait afficher:

  ```
  Datanodes available: 1 (1 total, 0 dead)
  ```

- **Nous allons maintenant compiler le code d'exemple Java du cours (compteur d'occurrence de mots).**
  Objectif: vérifier que l'environnement de compilation soit fonctionnel et que Hadoop soit correctement à même d'exécuter des tâches, et se familiariser avec le processus de compilation.

Préparation du TP

- **Installer le JDK OpenJDK (Java):**

```
apt-get install openjdk-6-jdk
```

- **Exécuter la commande suivante:**

```
wget http://cours.tokidev.fr/bigdata/tp/tp1_package.tgz
```

La commande va télécharger une archive contenant:

- **Le poème ayant servi d'exemple en cours.**
- **Le code d'exemple du cours (classes driver, map et reduce).**
- **Un script permettant la mise en place simplifiée de l'environnement de compilation.**

Préparation du TP

- **Décompresser l'archive:**

```
tar -zxvf tp1_package.tgz
```

- **Exécuter le script classpath:**

```
. classpath
```

Ce script exporte la variable d'environnement CLASSPATH nécessaire à javac pour trouver les librairies Hadoop qui nous permettront de compiler le programme d'exemple.

- **Compiler le programme Hadoop:**

```
javac WCount*java
```

Préparation du TP

- **La compilation a généré trois fichiers .class: un pour chacune de nos classes (driver, map et reduce).**

- **On va désormais packager le programme d'exemple au sein d'un fichier .jar. Créer l'arborescence liée au nom du *package* avec la commande:**

```
mkdir -p org/mbds/hadoop/wordcount
```

Et déplacer les fichiers compilés au sein de cette arborescence:

```
mv *.class org/mbds/hadoop/wordcount
```

- **Générer le .jar:**

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

Préparation du TP

- **En préparation de l'exécution de notre programme Hadoop, nous allons maintenant déplacer le texte du poème sur HDFS. Exécuter la commande:**

  hadoop fs -put poeme.txt /

  et vérifier sa présence avec la commande:

  hadoop fs -ls /

- **Enfin, exécuter notre programme Hadoop avec la commande:**

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

  La commande devrait prendre tout au plus quelques secondes à s'exécuter.

Préparation du TP

Si tout s'est passé correctement, un message « `mapreduce.Job: map 100% reduce 100%` » devrait s'afficher.

- Vérifier la présence des fichiers de résultats dans le répertoire /results avec la commande:

```
hadoop fs -ls /results
```

(un fichier `_SUCCESS` devrait être présent, ainsi qu'un fichier `part-r-00000`).

- Enfin, afficher les résultats finals avec la commande:

```
hadoop fs -cat /results/part-r-00000
```

Le mot « qui » devrait être le plus présent au sein du poème (25 occurrences).

TP - Anagrammes

- **Vous devez maintenant réaliser votre propre programme Hadoop.**

- **<u>OBJECTIF:</u> on dispose d'une liste de mots courants de la langue anglaise (plus de 1000). On souhaite déterminer quels mots sont des anagrammes.**

  On rappelle qu'un mot est un anagramme d'un autre si leurs lettres sont identiques (par exemple, « *lemon* » et « *melon* »).

- **Télécharger le fichier des mots courants avec la commande:**

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

- **<u>Remarque:</u> vous pouvez utiliser Streaming (tel que vu en cours) si vous préférez développer dans un autre langage que Java. L'essentiel est de réaliser le programme Hadoop :-)**

Conseils

- **Pensez bien à appliquer la méthodologie vue en cours.**

  Les questions à se poser: quelle est la clef à utiliser ?
  Quel est le facteur commun entre les anagrammes, c'est à dire la clef qui permettra à Hadoop de les grouper ?

- **Ne pas oublier de déplacer les données d'entrée (le fichier des mots courants) sur HDFS.**

- **Vous pouvez utiliser un IDE plus confortable (Eclipse ou autre) sur vos machines hôtes si vous le préférez pour compiler le .jar, et le transférer sur la machine virtuelle dans un second temps pour tester.**
  Pensez simplement à créer un projet Java « classique » pour générer un .jar, et à ajouter les librairies Hadoop vues en cours au projet (les quatre .jar – télécharger Hadoop pour les obtenir).