TP2 Premiers pas avec MapReduce

ITI4.2 - INSA Rouen

December 12, 2022

Les objectifs de ce TP sont:

- ☐ Création d'une image Pomdman disposant Hadoop et génération d'un conteneur lié.
- ☐ WordCount: compilation du programme et comprendre comment fonctionner un job MapReduce.
- ☐ Petit tweak: WordLength
- ☐ Anagrams: Implémenter votre propre programme MapReduce.

1. Setup image Podman

Veuillez consulter le sujet *TP1.2: Dockerfile et première image Podman*. Pour ce TP concrètement, les étapes sont un peu plus simples (idem veut dire que vous pouvez reprendre exactement le contenu qui se trouve dans le sujet 1.2):

- 1. Image de base: idem
- 2. Répertoire de travail: idem
- 3. Installation des paquets/services/bibliothèques essentiels: remplacer (vim nodejs npm) par **nano** (parce qu'on en aura besoin).
- 4. Installation des logiciels: Installez **seulement** Hadoop.
- 5. Variables d'environnement: Définissez seulement les variables qui servent pour tourner Hadoop:
  - JAVA_HOME
  - HADOOP_HOME
  - HADOOP_CONF_DIR
  - LD_LIBRARY_PATH
  - PATH=$PATH
- 6. Ajout des dossiers utils pour le TP: Indiquez dans le Dockerfile, l'ajout du dossier *input* dans */home/bgd/input* et *wordcount* dans *home/bgd/java/wordcount*.
  Hint: Grâce à l'instruction `ADD`, avec la syntaxe suivante:

```
ADD dossier_source dossier_destination
```

Vous pouvez ensuite créer une image Podman:

`podman image build -t tp2:latest .` (**Attention**: le point est **très** important)

Et lancer un conteneur lié à cette image:

`podman run -itd --name tp2etu --hostname tp2etu tp2:latest`

(cf Readme pour comprendre mieux)

Si tout se passe bien, vous allez vous allez prompter dans le conteneur directement:

`root@tp2etu:/home/bgd#`

Avec `ls`, vérifiez que les dossiers `input` et `java` sont bien présents.

Attention: Une fois le conteneur est lancé, il est conseillé de le laisser ouvert jusqu'à la fin du TP (c'est à dire garder le terminal ouvert et de ne pas faire `exit` par curiosité).

2. WordCount

(Message pour Maxime) Cf la partie 2.WordCount du sujet que t'as fait (je le push également sur le Git). Normalement toutes les commandes fonctionnent pareil **sauf** au niveau de l'exécution des programmes Java, au lieu de:

```
javac WordcountMain.java WordcountMap.java WordcountReduce.java
```

ce serait:

```
javac *.java -cp $(hadoop classpath)
```

3. WordLength

- Copiez les fichiers WordCount, renommer en WordLength (en faisant attention à changer tous les "wordcount" en "wordlength" dans tous les fichiers).

- Modifiez le programme pour calculer, à partir d'un fichier texte, le nombre de mot suivant leurs longueurs:
  - Short words: moins que 3 lettres.
  - Average words: moins que 6 lettres.
  - Long words: les restes.

- Exécutez le programme en prennant en entrée le fichier *book.txt*(**Attention**: supprimez le dossier `output` avant l'exécution). Pour vérifier que votre programme fonctionne, la sortie de `cat output/*` serait:

```
root@tp2etu:/home/bgd# cat ouput/*
Average words: 138300
Long words: 73471
Short words: 57414
```

4. Anagrams

(Message pour Maxime) Cf la partie 3.Anagrams du sujet que t'as fait (je le push également sur le Git). Normalement toutes les commandes fonctionnent pareil