TP n° 1 Hadoop et le système de fichiers distribué

Nom prénom binôme 1

Nom prénom binôme 2 (Pas de trînome sauf groupe impaire).

I. Recherche commandes HDFS

Listez ci dessous les listes de commandes HDFS disponibles (chercher sur google/ou cf mes diapos) et leur fonctionnalité.

Commentez (en comparant aux commandes classiques de chez Ubuntu). Pourquoi n'existes-t-il pas de commande hdfs cd ?

II. Lancement de Hadoop

Nous allons lancer hadoop à l'aide de docker. Pour cela nous allons lancer : trois contenaires représentant respectivement un noeud maître master (Namenode) et deux noeuds workers 1 et 2 (Datanodes).

Faire un schéma ci dessous ou un screenshot figma répresentant : votre PC, la machine virtuelle Docker lançant le noeud master et les deux machines virtuelles docker lançant les noeuds works 1 et 2. Montrer les liens entre les différentes entités. Nommer la VM qui représente le NameNode et les 2 VM qui représentent les DataNode sur votre schéma.

Pour lancer Docker avec les 3 noeuds, faire un projet sur VS Code nommé TP1BigData. Dans le terminal de ce projet lancer les commandes suivantes (commenter ce que fait chacun des commandes) :

docker pull liliasfaxi/hadoop-cluster:latest

docker network create --driver=bridge hadoop

docker run -itd --net=hadoop -p 9870:9870 -p 8088:8088 -p 7077:7077 -p 16010:16010 --name hadoop-master --hostname hadoop-master liliasfaxi/hadoop-cluster:latest

docker run -itd -p 8040:8042 --net=hadoop --name hadoop-worker1 --hostname hadoop-worker1 liliasfaxi/hadoop-cluster:latest

docker run -itd -p 8041:8042 --net=hadoop --name hadoop-worker2 --hostname hadoop-worker2 liliasfaxi/hadoop-cluster:latest

Insérer une capture d'écran de Docker avec les 3 images docker qui run.

Entrer sur le container master à l'aide de la commande :

docker exec -it hadoop-master bash

et lancer hadoop à l'aide de : ./start-hadoop.sh . A chaque fois que vous redemarrer les VM, ils vous faudra relancer cette commande sinon hadoop ne fonctionnera pas.

III. Gestion de fichiers HDFS

Faire un dossier HDFS nommé input à l'aide des commandes de la partie 1.

Vérifier sa création à l'aide des commandes du 1. (Insérer une capture d'écran des deux commandes utilisés).

Télécharger le fichier purchase.txt à l'adresse : https://we.tl/t-uWXO5fIEKW

Extraire le fichier à la racine du dossier TP1BigData. Copier le fichier sur le container master à l'aide d'une commande docker (insérer une capture d'écran de la fonction docker associé). Vérifier que le fichier a bien été copié sur le container master. Puis copier le fichier sur le HDFS à l'aide d'une commande HDFS. Vérifier que le fichier a bien été copié à l'aide d'une autre commande HDFS. Mettre les captures d'écran.

IV. Interfaces web

Aller sur votre navigateur ouvrir les pages : http://localhost:9870 & http://localhost:8088 . Prendre une capture d'écran de chaque page et expliquer ce qu'elles permettent de voir (Container Master ou Container Worker / NameNode ou JobTracker)

Montrer sur une capture d'écran le dossier Input et le fichier purchases.txt crée avant sur une des deux interfaces web.

V. Fonction Map Reduce

MapReduce permet d'executer des programmes de traitement de fichier sur le système hadoop en calculant des clés (map) puis en les triant (reduce). Dans un premier temps sur votre environnement local, créer un fichier countword (en python ou langage de votre choix). Le but est d'avoir les différents morts dans purchases.txt et leur nombre d'occurrences. Faites ce code de traitement et insérer une capture d'écran de celle-ci. Quel est le temps approximatif d'exécution de ce code sur votre PC ?

Nous allons maintenant utiliser la fonction map reduce sur le système hadoop.

Créer maintenant deux codes python map.py et reduce.py en local avec le contenu suivant :

map.py

```
#!/usr/bin/env python
import sys
# Read input from STDIN
for line in sys.stdin:
    # Split the line into words
    words = line.strip().split()
    for word in words:
        # Emit each word with a count of 1
        print(f"{word}\t1")
```

reduce.py

```
#!/usr/bin/env python
import sys
from collections import defaultdict
# Initialize a dictionary to store word counts
word_counts = defaultdict(int)
# Read input from STDIN
for line in sys.stdin:
    word, count = line.strip().split("\t")
    # Accumulate word counts
    word_counts[word] += int(count)

# Emit the final word counts
for word, count in word_counts.items():
    print(f"{word}\t{count}")
```

Il faudra ensuite copier ces deux fichiers sur le container hadoop-master sur docker.

Nous n'avons pas besoin de copier sur le système HDFS. Pour exécuter le mapreduce sur le système HDFS, nous allons utiliser un programme nommé hadoop-streaming (fichier déjà compliqué en exécutable Java car Hadoop est codé en Java).

Pour exécuter le map reduce il suffira de taper la commande suivante sur le container master :

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar \

-files mapper.py,reducer.py -mapper "python3 mapper.py" \

-reducer "python3 reducer.py" \

-input /user/root/input/purchases.txt -output /user/root/output1 (ajuster les fichiers input et output en fonction de votre configuration)

Cependant lors de la première exécution de ce code, il vous montera une erreur. D'où vient l'erreur ?

Nous allons maintenant configurer le fichier mapred-site.xml pour bien exécuter la fonction map reduce. Pour se faire il faudra utiliser vim ou onstaller la bibliothèque nano (commande apt update suivi de apt install nano) puis cd $HADOOP_HOME/usr/local/hadoop et ensuite on fait nano ou vim mapred-site.xml sur le container master puis éditer le fichier de configuration avec :

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>

  <!-- Add the configurations mentioned in the error message -->
  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
  </property>
  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
  </property>
  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
  </property>
</configuration>
```

Exécuter à nouveau mapreduce avec :

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar \
-files mapper.py,reducer.py     -mapper "python3 mapper.py" \
-reducer "python3 reducer.py" \
-input /user/root/input/purchases.txt -output /user/root/output2
```

Recupérer les données mapreduce sur votre PC à l'aide de commandes HDFS & docker (le output de la fonction mapreduce est créée dans le dossier output2). Prendre une capture d'écran de ces résultats puis aller vérifier sur l'interface web (http://localhost:8088) la bonne exécution. Prendre les screenshot des logs files et commenter en comparant la différence de vitesse d'exécution de wordcount en local sur votre PC et via mapreduce sur le HDFS.