TP 1 : Hadoop, Map/Reduce

Le Tp consiste à exécuter un job Map reduce en python (en JAVA, si vous le souhaiter) dans 3 environnements différents :

- **Mode local (standalone)** : dans ce mode, tout s'exécute au sein d'une seule JVM, en local. C'est le mode recommandé en phase de développement.
- **Mode local pseudo-distribué (pseudo-distribué)** : dans ce mode, le fonctionnement en mode cluster est simulé par le lancement des tâches dans différentes JVM exécutées localement.
- **Mode distribué (fully-distributed)** : c'est le mode d'exécution réel d'Hadoop. Il permet de faire fonctionner le système de fichiers distribué et les tâches sur un ensemble de machines.

I) Installation de cluster Hadoop

1. Installer un cluster Hadoop dans un environnement avec 1 Master et 4 slaves sur VMs ou Docker.

- 2 Go de RAM pour le Master
- 512 Go de RAM pour chaque Slave

Lien utile :

- https://medium.com/@jootorres_11979/how-to-set-up-a-hadoop-3-2-1-multi-node-cluster-on-ubuntu-18-04-2-nodes-567ca44a3b12
- https://www.edureka.co/blog/setting-up-a-multi-node-cluster-in-hadoop-2-x/
- https://dev.to/awwsmm/installing-and-running-hadoop-and-spark-on-ubuntu-18-393h

Ces liens sont à titre indicatif

2. Assurez-vous d'avoir le dossier hadoop (Le path étant signifié dans les commandes exécuter plus haut) et que hadoop marche en exécutant la commande **hadoop –version**
3. Assurez-vous d'accéder à la UI du gestionnaire de ressources **YARN**

II) Map/Reduce (Word count)

1) Télécharger le répertoire **TP 1 Hadoop** ainsi que les scripts mapper et reducer

2) Télécharger les fichiers .txt (et les copier dans hadoop/wc/input) qui serviront de corpus pour les jobs Word Count MapReduce

3) S'assurer de donner les droits d'exécutions sur les fichiers mapper.py et reducer.py depuis répertoire de hadoop

4) Assurer vous que le mapper donne le résultat escompté en exécutant en ligne de commande depuis le dossier hadoop la commande suivante :
cat input/5000.txt | ./mapper0.py

Que signifie le symbole ''|''

5) Assurer vous que le mapper donne le résultat escompté en exécutant depuis le dossier hadoop la commande suivante : **cat input/5000.txt | ./mapper.py**

Que signifie le symbole du pipe ''|''

6) Exécuter maintenant : **cat wc/input/5000.txt | ./mapper.py | sort**
Qu'observez-vous ?

7) Exécutez maintenant la commande complète
cat wc/input/5000.txt | ./mapper.py | sort | ./reducer0.py

Et vérifiez que le programme produit le résultat attendu. Si ça fonctionne, félicitations! Vous venez d'exécuter votre premier programme Map/Reduce en Utilisant propre implémentation de Map/Reduce (en ligne de commande)

Maintenant nous allons exécuter le Job sur dossier de fichier input contenant **5000.txt** et **20417-8.txt** avec une exécution en streaming

8) La chaine d'exécutions du job MapReduce

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
-input /user/hadoop/wc/input \
-output /user/hadoop/wc/output \
-file /home/hadoop/mapper.py \
-mapper /home/hadoop/mapper.py \
```

```
-file /home/hadoop/reducer.py \
-reducer /home/hadoop/reducer.py
```

Que signifie chaque ligne de code ?

9) Faire un benchmark en faisant varier le nombre (nbrtask) de reduce tasks suivant la chaine d'exécution ci-dessous

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
-input /user/hadoop/wc/input \
-output /user/hadoop/wc/output \
-file /home/hadoop/mapper.py \
-mapper /home/hadoop/mapper.py \
-file /home/hadoop/reducer.py \
-reducer /home/hadoop/reducer.py \
-jobconf mapred.reduce.tasks= nbrtask
```

Quel commentaire faites-vous ?

Que pouvez-vous conclure ?

NB : Après traitement de MapReduce les résultats seront inscrits dans le output

10) Modifier le word count pour compter cette fois ci le nombre de mots commençant par une lettre donner

Format de Output :

a 12

b 92

c 233

C 99

III) Map/Reduce (Transcription de requete SQL en Map/Reduce)

Soit deux tables Customer.txt et Order.txt avec les schéma ci-dessous

• Customer(id, startDate, nom)

• Order(#id, total)

Transcrire les requêtes ci-dessous en map/reduce

Requete 1

```
-- Nombre d'achat pour chaque mois --
    SELECT MONTH(c.start_date), COUNT(*)
    FROM customer c
    GROUP BY MONTH(c.start_date)
```

Requete 2

```
-- Ensemble des enregistrements cliens tri ées par nom --

    SELECT *
    FROM customer c
    ORDER BY c.name;
```

Requete 3

```
    -- Ensemble des commandes avec le nom du client (jointure) --
    SELECT c.name, o.total
    FROM customer c, order o
    WHERE c.id = order.id;
```

Requete 4

```
    -- Montant moyen des commandes par client --
    SELECT c.name, AVG(o.total)
    FROM customer c, order o
    WHERE c.id = o.id
    GROUP BY c.name;
```

Les fichiers Customer.txt et Order.txt se trouve dans le répertoire du projet.

TP 2 : Spark

I) Installation de SPARK

1. Installer un cluster Hadoop dans un environnement avec 1 Master et 4 slaves sur VMs ou Docker.

2 Go de RAM pour le Master

512 Go de RAM pour chaque Slave

Lien utile :

https://www.linkedin.com/pulse/how-setup-install-apache-spark-311-cluster-ubuntu-shrivastava/

https://medium.com/@madtopcoder/install-a-spark-cluster-on-virtualbox-fad075449521

https://events.prace-ri.eu/event/1316/attachments/1761/3452/Installing%20Environment-2022.pdf

Ces liens sont à titre indicatif

2. Assurez-vous d'avoir spark qui marche en éxéxutant les commandes suivantes :

-spark-submit

-pyspark

-spark-shell

3. Assurez-vous d'accéder à la UI de spark

II) Word Count avec Spark

- Exécuter le word count en spark
- Modifier le word count pour compter cette fois ci le nombre de mots commençant par une lettre donner

Augmenter le nombre d'exécuteurs, nombre de mémoire et de CPU par exécuteurs, également le nombre de Drivers, nombre de mémoire et de CPU par Drivers

Que pouvez-vous conclure ?

III) Utilisation de RDD et Spark Dataframe ou Spark SQL

Soit deux tables Customer.txt et Order.txt avec les schéma ci-dessous

• Customer(id, startDate, nom)

• Order(#id, total)

Transcrire les requêtes ci-dessous en map/reduce

Requete 1

```
-- Nombre d'achat pour chaque mois --
    SELECT MONTH(c.start_date), COUNT(*)
    FROM customer c
    GROUP BY MONTH(c.start_date)
```

Requete 2

```
-- Ensemble des enregistrements cliens tri ées par nom --

    SELECT *
    FROM customer c
    ORDER BY c.name;
```

Requete 3

```
    -- Ensemble des commandes avec le nom du client (jointure) --
    SELECT c.name, o.total
    FROM customer c, order o
    WHERE c.id = order.id;
```

Requete 4

```
    -- Montant moyen des commandes par client --
    SELECT c.name, AVG(o.total)
    FROM customer c, order o
    WHERE c.id = o.id
    GROUP BY c.name;
```

Les fichiers Customer.txt et Order.txt se trouve dans le répertoire du projet.

1. Résoudre le problème ci-dessous avec spark RDD
2. Avec Spark Dataframe
3. Avec Spark SQL

Ce TP est à faire en par groupe de 3 étudiants.

NB : Un rapport de TP de (15-20 pages maximum) est à rendre au plus tard le samedi 04 Février à 00h.