Sujet de TP
Initiation à Hadoop et MapReduce

Objectifs du TP
Initiation au framework hadoop et au patron MapReduce
Utilisation de docker pour lancer un cluster hadoop de 3 nœuds

Outils et versions
Apache Hadoop : version 3.3.6
Docker version : latest

Hadoop et Docker
Pour déployer le framework Hadoop, nous allons utiliser des contenaires Docker. L'utilisation des contenaires va garantir la consistance entre les environnements de développement et permettra de réduire considérablement la complexité de configuration des machines (dans le cas d'un accès natif) ainsi que la lourdeur d'exécution.

Installation
Nous allons utiliser tout au long de ce TP trois contenaires représentant respectivement un noeud maître (Namenode) et deux noeuds workers (Datanodes).

Vous devez pour cela avoir installé docker sur votre machine, et l'avoir correctement configuré. Ouvrez la ligne de commande, et tapez les instructions suivantes:

1. Téléchargez l'image docker uploadée sur dockerhub:

```
docker pull liliasfaxi/hadoop-cluster:latest
```

2. Créez les trois contenaires à partir de l'image téléchargée. Pour cela:

2.1. Créez un réseau qui permettra de relier les trois contenaires:

```
docker network create --driver=bridge hadoop
```

2.2 Créez et lancez les trois contenaires (les instructions -p permettent de faire un mapping entre les ports de la machine hôte et ceux du contenaire):

```
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
```

3. Entrez dans le contenaire master pour commencer à l'utiliser.

```
docker exec -it hadoop-master bash
```

Le résultat de cette exécution sera le suivant:

```
root@hadoop-master:~#
```

Vous vous retrouverez dans le shell du namenode, et vous pourrez ainsi manipuler le cluster à votre guise. La première chose à faire, une fois dans le contenaire, est de lancer hadoop et yarn. Un script est fourni pour cela, appelé start-hadoop.sh. Lancer ce script.

```
./start-hadoop.sh
```

Le résultat devra rassembler à ce qui suit :

```
root@hadoop-master:~# ./start-hadoop.sh

Starting namenodes on [hadoop-master]
hadoop-master: Warning: Permanently added 'hadoop-master,172.22.0.2' (ECDSA) to the list of known hosts.
hadoop-master: starting namenode, logging to /usr/local/hadoop/logs/hadoop-root-namenode-hadoop-master.out
hadoop-slave1: Warning: Permanently added 'hadoop-slave1,172.22.0.3' (ECDSA) to the list of known hosts.
hadoop-slave2: Warning: Permanently added 'hadoop-slave2,172.22.0.4' (ECDSA) to the list of known hosts.
hadoop-slave2: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-hadoop-slave2.out
hadoop-slave1: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-hadoop-slave1.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of known hosts.
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-root-secondarynamenode-hadoop-master.out


starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn--resourcemanager-hadoop-master.out
hadoop-slave2: Warning: Permanently added 'hadoop-slave2,172.22.0.4' (ECDSA) to the list of known hosts.
hadoop-slave1: Warning: Permanently added 'hadoop-slave1,172.22.0.3' (ECDSA) to the list of known hosts.
hadoop-slave2: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-hadoop-slave2.out
hadoop-slave1: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-hadoop-slave1.out
```

Interfaces web pour Hadoop

Hadoop offre plusieurs interfaces web pour pouvoir observer le comportement de ses différentes composantes. Vous pouvez afficher ces pages en local sur votre machine grâce à l'option -p de la commande docker run. En effet, cette option permet de publier un port du contenaire sur la machine hôte. Les ports exposés sont :

- Le port 9870: qui permet d'afficher les informations du namenode.
- Le port 8088: qui permet d'afficher les informations du resource manager de Yarn et visualiser le comportement des différents jobs.

Une fois votre cluster lancé et prêt à l'emploi, vous pouvez, sur votre navigateur, aller à : http://localhost:50070.

Vous pouvez également visualiser l'avancement et les résultats de vos Jobs (Map Reduce ou autre) en allant à l'adresse: http://localhost:8088

Pensez à bien vérifier que les nœuds workers sont bien actifs et prêts à accueillir des jobs MapReduce.

Premiers pas avec Hadoop

Toutes les commandes interagissant avec le système Hadoop commencent par hadoop fs. Ensuite, les options rajoutées sont très largement inspirées des commandes Unix standard.

Nous présentons dans le tableau suivant les commandes les plus utilisées pour manipuler les fichiers dans HDFS:

Instruction	**Description**
hadoop fs –ls	Afficher le contenu du répertoire racine
hadoop fs –put file.txt	Upload un fichier dans hadoop (à partir du répertoire courant linux)
hadoop fs –get file.txt	Download un fichier à partir de hadoop sur votre disque local
hadoop fs –tail file.txt	Lire les dernières lignes du fichier
hadoop fs –cat file.txt	Affiche tout le contenu du fichier
hadoop fs –mv file.txt newfile.txt	Renommer le fichier
hadoop fs –rm newfile.txt	Supprimer le fichier
hadoop fs –mkdir myinput	Créer un répertoire
hadoop fs –cat file.txt | less	Lire le fichier page par page

Maintenant, créez un répertoire dans HDFS, appelé input.

```
hadoop fs –mkdir -p input
```

Nous allons utiliser le fichier purchases.txt comme entrée pour le traitement MapReduce. Ce fichier se trouve déjà sous le répertoire principal de votre machine master.

Chargez le fichier purchases dans le répertoire input que vous avez créé:

```
hadoop fs –put purchases.txt input
```

Pour afficher le contenu du répertoire input, la commande est:

```
hadoop fs –ls input
```

Pour afficher les dernières lignes du fichier purchases:

```
hadoop fs -tail input/purchases.txt
```

Vous aurez le résultat suivant :

```
root@hadoop-master:~# hadoop fs -tail input/purchases.txt
31      17:59   Norfolk Toys    164.34  MasterCard
2012-12-31      17:59   Chula Vista     Music   380.67  Visa
2012-12-31      17:59   Hialeah Toys    115.21  MasterCard
2012-12-31      17:59   Indianapolis    Men's Clothing  158.28  MasterCard
2012-12-31      17:59   Norfolk Garden  414.09  MasterCard
2012-12-31      17:59   Baltimore       DVDs    467.3   Visa
2012-12-31      17:59   Santa Ana       Video Games     144.73  Visa
2012-12-31      17:59   Gilbert Consumer Electronics    354.66  Discover
2012-12-31      17:59   Memphis Sporting Goods  124.79  Amex
2012-12-31      17:59   Chicago Men's Clothing  386.54  MasterCard
2012-12-31      17:59   Birmingham      CDs     118.04  Cash
2012-12-31      17:59   Las Vegas       Health and Beauty       420.46  Amex
2012-12-31      17:59   Wichita Toys    383.9   Cash
2012-12-31      17:59   Tucson  Pet Supplies    268.39  MasterCard
2012-12-31      17:59   Glendale        Women's Clothing        68.05   Amex
2012-12-31      17:59   Albuquerque     Toys    345.7   MasterCard
2012-12-31      17:59   Rochester       DVDs    399.57  Amex
2012-12-31      17:59   Greensboro      Baby    277.27  Discover
2012-12-31      17:59   Arlington       Women's Clothing        134.95  MasterCard
2012-12-31      17:59   Corpus Christi  DVDs    441.61  Discover
root@hadoop-master:~#
```

MapReduce

Premiers pas : wordcount

Nous allons tester un programme MapReduce grâce à un exemple très simple, le WordCount, l'équivalent du HelloWorld pour les applications de traitement de données. Le Wordcount permet de calculer le nombre de mots dans un fichier donné, en décomposant le calcul en deux étapes:

- L'étape de Mapping, qui permet de découper le texte en mots et de délivrer en sortie un flux textuel, où chaque ligne contient le mot trouvé, suivi de la valeur 1 (pour dire que le mot a été trouvé une fois)
- L'étape de Reducing, qui permet de faire la somme des 1 pour chaque mot, pour trouver le nombre total d'occurrences de ce mot dans le texte.

TESTER MAP REDUCE EN LOCAL

Nous allons ici faire fonctionner l'algorithme map-reduce qui compte les mots d'un fichier texte, en local, i.e. sans exploiter le parallélisme que propose le framework Hadoop. Le programme est constitué de deux scripts Python qui sont appelés successivement selon la méthode décrite ci-dessous.

- Créer un répertoire que vous appellerez *wordcount*.
- Téléchargez depuis internet le livre dracula à l'aide de la commande

```
wget http://www.textfiles.com/etext/FICTION/dracula
```

- Créer deux fichiers *mapper.py* et r*educer.py.*

mapper.py

```
#!/usr/bin/env python3
```

```
import sys

# reading entire line from STDIN (standard input)
for line in sys.stdin:
	line = line.strip()
	# split the line into words
	words = line.split()
	for word in words:
		print ('{}\t{}'.format(word, 1))
```

- Tester le mapper sur une partie du fichier dracula

```
head -50 dracula | python3 mapper.py
```

reducer.py

```
#!/usr/bin/env python3

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# read the entire line from STDIN
for line in sys.stdin:
	# remove leading and trailing whitespace
	line = line.strip()
	# splitting the data on the basis of tab we have provided in mapper.py
	word, count = line.split('\t', 1)
	# convert count (currently a string) to int
	try:
		count = int(count)
	except ValueError:
		# count was not a number, so silently
		# ignore/discard this line
		continue

	if current_word == word:
		current_count += count
	else:
		if current_word:
			# write result to STDOUT
			#print '%s\t%s' % (current_word, current_count)
			print('{}\t{}'.format(current_word, current_count))

		current_count = count
		current_word = word

if current_word == word:
```

```
        #print '%s\t%s' % (current_word, current_count)
        print('{}\t{}'.format(current_word, current_count))
```

- Tester le reducer (sur la totalité du fichier)

```
cat dracula | python3 mapper.py | sort | python3 reducer.py
```

Amélioration du wordcount

Pour stocker le résultat d'exécution du script dans un fichier appelé result.txt, exécuter la commande suivante :

```
cat dracula | python3 mapper.py | sort | python3 reducer.py > results.txt
```

Ouvrer ce fichier avec un éditeur de texte , et regarder les premières lignes. On constate de nombreux problèmes :

- la présence de signes de ponctuation dans les mots.
- les mots commençant par une majuscule sont distingués des mots commençant par une minuscule.

Pour régler ces problèmes, veuiller modifier les scripts précédents :
- pour qu'ils ne distinguent plus les mots qui comportent des majuscules et les mêmes mots qui n'en comportent pas.
- de telle manière que les signes de ponctuation ne soient plus pris en compte.

LANCER MAP REDUCE SUR LE CLUSTER

Préparation des fichiers pour le wordcount

- Dans le terminal du master, créer un dossier wordcount et déplacez-vous dedans.

```
mkdir wordcount
cd wordcount
```

- Télécharger depuis internet le livre de dracula à l’aide de la commande.

```
wget http://www.textfiles.com/etext/FICTION/dracula
```

- Versez ce fichier volumineux sur l'espace HDFS (dans le dossier input précédemment créé)

```
hadoop fs -put dracula input
```

- Vérifiez que le fichier a bien été déposé:

```
hadoop fs -ls input
```

- Supprimer le fichier dracula de votre espace Linux (on n'en a plus besoin!)

```
rm dracula
```

Il faut maintenant rapatrier, sur notre espace Linux, les scripts mapper.py et reducer.py que vous avez manipulés durant la première partie de ce TP. Pour cela, il faut ouvrir un second Terminal (laissez le premier ouvert, il va nous resservir!), et vous déplacer dans le dossier de travail qui contient les scripts mapper.py et reducer.py. La commande suivante permet de copier ces 2 fichiers vers l'espace Linux, dans le dossier wordcount.

```
docker cp mapper.py hadoop-master:/root/wordcount
docker cp reducer.py hadoop-master:/root/wordcount
```

Retenez la syntaxe, car elle vous sera utile plus tard, pour rapatrier les nouveaux scripts Python que vous aurez développés.

Revenez alors vers le premier Terminal (ne fermez pas le second, il sera utile plus tard), et vérifiez avec la commande ls que les 2 fichiers sont bien présents. Il faut maintenant rendre ces 2 scripts exécutables:

```
chmod +x mapper.py
chmod +x reducer.py
```

Exécution sur le cluster

Hadoop map-reduce fonctionne avec le langage Java ; il faut donc utiliser une bibliothèque capable de transformer des instructions Python en instruction Java. C'est le rôle de cette bibliothèque hadoop-streaming-3.3.6.jar (on appelle cela un wrapper).

- Lancer le job Hadoop avec l'instruction suivante :

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar –mapper
mapper.py –reducer reducer.py –file mapper.py –file reducer.py –input input/dracula –output
output
```

- Le Job sera lancé sur le fichier dracula que vous aviez préalablement chargé dans le répertoire input de HDFS. Une fois le Job terminé, un répertoire output sera créé.
- Affichez les dernières lignes du fichier généré output/part-r-00000, avec hadoop fs -tail output/part-r-00000. Le résultat devrait être exactement le même que lors de la première partie du TP.
- Il vous est possible de monitorer vos Jobs Map Reduce, en allant à la page: http://localhost:8088. Vous trouverez votre Job dans la liste des applications.

Remarque : N'oubliez pas! : Entre 2 exécutions, il faut soit utiliser un nouveau nom pour le dossier output soit le supprimer.

Appplication

Objectif : recueillir des informations et calculer des statistiques sur des résultats de ventes stockés dans le fichier purchases.txt.

Pour information, le dataset à traiter comporte 6 champs séparés par des tabulations.

Date	Temps	Magasin	Produit	Coût	Paiement

Écrivez des Jobs Map Reduce permettant, à partir du fichier purchases.txt initial, de

- déterminer le total des ventes par magasin.
- déterminer le total des ventes dans chaque catégorie.
- déterminer le total des ventes dans chaque catégorie en ajoutant un filtrage vertical (pour extraire sauf la catégorie "Costumer Electronics" et "Toys").
- déterminer la somme dépensée dans chaque magasin pour chaque moyen de paiement.
- calculer le total des ventes et son nombre dans tous les magasins confendus.

Veillez à tester votre code en local avant de lancer un job sur le cluster!