TP2- INITIATION A HADOOP ET MAP-REDUCE

I- Objectifs du TP :

1) Identifiez comment générer rapidement de la valeur en utilisant le **pipeline Hadoop**.
2) **Utilisation de la machine virtuelle** : Cloudera fournit une machine virtuelle où Hadoop, ainsi qu'un grand nombre d'outils de son écosystème, sont préinstallés.
3) Tirez parti des avantages en tirant parti des images prédéfinies de l'écosystème Hadoop par des sociétés telles que Cloudera et Hortonworks

Remarque : Pour notre TP, nous utilisons le langage Python pour développer les Mappers et les Reducers. Les traitements intermédiaires (comme le tri par exemple) sont effectués automatiquement par Hadoop.

II- Préambule :

Ce TP est inspiré de la formation "Intro to Hadoop and Map Reduce" fait par Cloudera2 et publié sur Udacity. Ils fournissent une machine virtuelle où Hadoop, ainsi qu'un grand nombre d'outils de son écosystème, sont préinstallés.

Une fois la VM lancée, vous êtes automatiquement identifié comme étant un utilisateur cloudera avec le **username cloudera** et le **mot de passe cloudera**. Ce même mot de passe est utilisé pour l'accès root à MySQL, l'accès à Hue et à Cloudera Manager.

.

III- Instructions de création de répertoires et de téléchargement de données:

Dans le tableau suivant, nous résumons les commandes les plus utilisées dans Hadoop:

hadoop fs –ls	Afficher le contenu du répertoire racine

hadoop fs –put file.txt	Upload un fichier dans hadoop (à partir du repertoire courant)
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 –cat file.txt | less	Lire le fichier page par page
hadoop fs –mkdir myinput	Créer un répertoire

Exemple de paramètres supplémentaires de la commande cd :
cd / : Permet de se retrouver à la racine du disque.

cd ~ : ou cd Accéder directement au répertoire de l'utilisateur.

cd .. : Remonter dans le répertoire parent à partir de là où vous êtes.

cd - : Permet de revenir au répertoire précédent.

[référence : https://juliend.github.io/linux-cheatsheet/]

Manipulations et exécution de programmes map-reduce sous hadoop

 # Version Hadoop :
hadoop version

 # Affichage des fichiers sous un répertoire
Hadoop fs -ls TP1/
hadoop fs -ls -R TP1/

 # Déplacer un répertoire
hadoop fs -mv <src> <dest>

 # Copier un fichier de HDFS dans HDFS
hadoop fs -cp <src> <dest>
Exemple : <src> TP1/data/purchases.txt
<dest> TP1/data/

 # Télécharger un fichier
wget ….

Attribuer la permission à un fichier
chmod +x code/reducer.py

Application 1 : Wordcount

Créer les deux codes python suivants :

Programme Mapper

```
#!/usr/bin/env python2
import sys
for line in sys.stdin:
   line = line.strip()
   words = line.split()
   for word in words:
      print '%s\t%s' % (word, 1)
```

Programme Reducer

```
#!/usr/bin/env python2
from operator import itemgetter
import sys
current_word = None
current_count = 0
word = None
for line in sys.stdin:
   line = line.strip()
   word, count = line.split('\t', 1)
   try:
      count = int(count)
   except ValueError:
      continue
   if current_word == word:
      current_count += count
   else:
      if current_word:
         print '%s\t%s' % (current_word, current_count)
      current_count = count
      current_word = word
if current_word == word:
   print '%s\t%s' % (current_word, current_count)
```

 # Tester votre programme mapper
echo "le groupe CD de troisième année de cette année est formidable !!!" |
code/mapper2.py

Application sur des données réelles

Afficher le dernier kilo-octet du fichier sur stdout.
hadoop fs -tail hadoop/purchases.txt

Pour afficher le contenu de votre fichier texte purchase.txt
qui est présent dans votre répertoire hadoop.
hadoop fs -cat hadoop/purchases.txt

 # Tester vos codes en local :
head -50 Downloads/purchases.txt |code/mapper2.py |sort |code/reducer.py

 # Exécuter map-reduce sous hadoop :
hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-streaming-2.6.0-mr1-cdh5.13.0.jar -mapper code/mapper2.py -reducer code/reducer.py -file code/mapper2.py -file code/reducer.py -input myinput/purchases.txt -output joboutput

Remarque : nombre de map = nombre de block (en fonction du volume du fichier)
Généralement le nombre de reducer est au voisinage de 10% du nombre de mapper

 # Afficher le fichier de sortie :
hadoop fs -tail output/part-r-00000

 # consulter la liste des tâches map-reduce
mapred job -list all

 # consulter le statut du job
mapred job -status job_1670364086924_0001

#😉 Forcer l'arrêt une tâche

mapred job -status job_1670364086924_0002

Remarque : vous pouvez consulter ces informations via la page web.

--- all applications

Ecrire vos propres Mappers et Reducers pour :

1) Donner le chiffre d'affaire par magasin
2) Donner la liste des ventes par catégorie de produits.

- Quelle est la valeur des ventes pour la catégorie Toys?
- Et pour la catégorie Consumer Electronics?

3) Quel est le nombre total des ventes et la valeur totale des ventes de tous magasins confondus?

Tester les solutions :
head -50 Downloads/purchases.txt | code/mapper3.py
head -50 Downloads/purchases.txt | code/mapper3.py | sort |code/reducer3.py

Pour aller loin :
Utiliser un cluster AWS-EMR (Elastic Map Reduce) de Amazon pour exécuter un Job Map Reduce de votre choix sur un vrai cluster distribué. [https://aws.amazon.com/fr/emr/]

Autres commandes :

- Signaler la quantité d'espace utilisé et disponible sur le système de fichiers actuellement monté

  hadoop fs -df hdfs:/

- Liste toutes les commandes shell du système de fichiers hadoop

  hadoop fs

- Commande pour que le nœud de nom quitte le mode sans échec

  hadoop fs –expunge

- La commande *'-get'* est équivalente à la commande *'-copyToLocal'*

Travail demandé :

1. Démarrer votre machine virtuelle et lancer un terminal
2. En utilisant certaines commandes présentées dans le tableau ci-dessus

    - Créer un répertoire TP
    - Créer deux sous-répertoires (sous TP) code et data dans lesquels on sauvegardera respectivement les codes de nos mappers et reducers, et les données sources et résultat.

3. Télécharger le fichier purchases.txt à partir du lien suivant : **https://drive.google.com/file/d/1Ar12GZISlf-JMQSj1FQsy0_XhSqMH4F9/view?usp=sharing**

Description des données : L'ensemble de données contient les données relatives aux données de vente de chaque magasin Walmart au cours d'une année donnée. Cet ensemble de données contient des variables catégorielles et des variables numériques. Il existe 6 attributs et plus de 10 000 000 instances.

4. Déplacez-vous sous le répertoire ~/TP/data, et y importer le fichier purchases.txt.

IV- Utilisez les concepts de base, HDFS et MapReduce de Big Data Hadoop Platform pour résoudre un problème de traitement de gros volumes de données :

Il est maintenant temps de se familiariser avec Hadoop Map Reduce.

Généralement, le code Map Reduce est écrit en java. Mais avec une fonctionnalité appelée Hadoop Streaming, vous pouvez écrire vos mappeurs et

réducteurs dans n'importe quelle langue. Nous utiliserons python pour écrire le travail de réduction de la carte.

Application : Analyse des données de vente pour la chaîne de magasins à l'aide de Cloudera Platform.

- Nous allons utiliser un programme mapper et le reducer pour calculer les ventes totales de chaque magasin aux États-Unis.
- Analyser et déterminer les revenus générés dans différents magasins d'un pays / d'une région.

Outils :

- Utilisez les concepts de base, HDFS et MapReduce de Big Data Hadoop Platform pour résoudre le problème.
- Traitement de gros volumes de données en parallèle en divisant le travail en un ensemble de tâches indépendantes.
- Utilisez le fichier de processus MapReduce pour le diviser en morceaux et le traiter en parallèle.

Travail demandé :

1. Créer un répertoire appelé myinput
2. Copier le fichier purchases.txt dans le répertoire myinput
3. Afficher les dernières lignes du fichier

Solution :

- Créer un répertoire dans HDFS, appelé myinput. Pour cela, taper:

hadoop fs –mkdir myinput

- Pour copier le fichier purchases.txt dans HDFS sous le répertoire myinput, taper la commande:

hadoop fs –put purchases.txt myinput/

- Pour afficher le contenu du répertoire myinput, la commande est:

hadoop fs –ls myinput

4. Le dossier du projet contient les fichiers **mapper.py** et **reducer.py**.

- **Afficher le fichier mapper.py**

Description : le mapper lit le morceau du fichier et divise le fichier à l'aide du délimiteur de tabulation. Ensuite, il imprime uniquement l'emplacement du magasin et le coût de l'article comme sortie.

Tester ce mapper en local sur les 50 premières lignes du fichier purchases.txt en tapant l'instruction suivante, directement à partir de votre répertoire code:

head -50 ../data/purchases.txt | ./mapper.py

- **Afficher le code réducer**

Description : le reducer.py prend l'entrée triée et continue de vérifier si la nouvelle clé est égale à la clé précédente. Lorsque le changement se produit, il imprime le nom du magasin et le total des ventes du magasin.

Tester ce Reducer sur le disque local, en utilisant cette instruction.

head -50 ../data/purchases.txt |./mapper.py |sort |./reducer.py

Lancer un Job sur Hadoop Lancer un job sur Hadoop implique qu'on fera appel au mapper puis au reducer sur une entrée volumineuse, et qu'on obtiendra à la

fin un résultat, directement sur HDFS. Pour faire cela, l'instruction à exécuter sur notre machine virtuelle est du type :

Une fois que vous avez écrit le mapper et le reducer, vous pouvez démarrer le travail de réduction de carte à l'aide de la commande:

hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-streaming-2.6.0-mr1-cdh5.13.0.jar -mapper code/mapper2.py -reducer code/reducer.py -file code/mapper2.py -file code/reducer.py -input myinput/purchases.txt -output joboutput

Remarque 1 : Nous utilisons Hadoop Streaming qui permet de créer et lancer des jobs MapReduce avec tout type d'exécutable ou script en tant que mapper et reducer. Ici nos scripts sont en Python, mais les mappers et reducers pourraient être des classes Java, des utilitaires unix, des scripts R, etc. La manière standard est d'écrire des programmes MapReduce en Java via l'API Java MapReduce.

Remarque 2 : Le répertoire d'entrée doit contenir un seul fichier. Le répertoire de sortie ne doit pas exister avant l'exécution de l'instruction.

Remarque 3 : Après avoir exécuté cela, le calcul prend un certain temps. Vous pouvez observer le fonctionnement des mappeurs et des réducteurs dans le terminal de sortie. Le terminal affichera une variété d'informations comme le pourcentage d'achèvement du travail.

- **Sortie :** La sortie de notre programme est enregistrée dans le fichier joboutput / part-00000

  Vous pouvez afficher la sortie dans le même terminal en exécutant la commande :

  hadoop fs -ls joboutput

Au lieu de visualiser l'ensemble de la sortie, vous pouvez afficher une plus petite partie de notre sortie en utilisant la commande :

hadoop fs -cat joboutput / part-00000 | Moins

Une fois votre tâche terminée, vous pouvez maintenant supprimer ce fichier de sortie de travail de notre système HDFS. La raison en est que nous ne pouvons pas générer différentes charges de travail de sortie avec le même nom de fichier de sortie. Ceci peut être réalisé en exécutant la commande suivante
hadoop fs -rm -r -f joboutput

Pendant ce temps, vous pouvez exporter les données dans un fichier texte et l'enregistrer dans votre répertoire si vous le souhaitez pour une utilisation future.

http://localhost:50030/jobtracker.jsp.