Lab 5: Apache PIG

Pour rappel, Apache Pig est un langage de haut niveau conçu pour le traitement de données massives dans l'écosystème Hadoop. Il permet aux utilisateurs d'écrire des scripts pour effectuer des opérations complexes sur des ensembles de données sans avoir à se plonger dans les détails de MapReduce

L'objectif de ce TP est de :

- Installer Apache pig sur le cluster hadoop avec docker

I. Installation Apache PIG

1. Accéder au master

- *Démarrer et entrer dans le conteneur master pour commencer à l'utiliser*

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

2. Démarrer Installer et configurer Apache PIG

- *Télécharger la télécharger la dernière version d'Apache Pig*

```
wget https://dlcdn.apache.org/pig/pig-0.17.0/pig-0.17.0.tar.gz
```

- *Extraire l'Archive Téléchargée ensuite déplacez le répertoire extrait vers /usr/local/pig*

```
tar -zxvf pig-0.17.0.tar.gz
mv pig-0.17.0 /usr/local/pig
```

- *Supprimez le fichier tar téléchargé pour libérer de l'espace :*

```
rm pig-0.17.0.tar.gz
```

- *Configurer les variables d'environnement pour Pig*

```
vim ~/.bashrc
export PIG_HOME=/usr/local/pig
export PATH=$PATH:$PIG_HOME/bin
```

- *Après avoir enregister le fichier bashrc, appliquer les modifications effectuées*

```
source ~/.bashrc
```

- *Configurer les Paramètres Hadoop*

```
vim $HADOOP_CONF_DIR/hdfs-site.xml
```

```
<property>
      <name>dfs.webhdfs.enabled</name>
    <value>true</value>
</property>
<property>
      <name>dfs.bytes-per-checksum</name>
      <value>512</value>
</property>
```

- *Configurer les Paramètres YARN*

```
vim $HADOOP_CONF_DIR/yarn-site.xml
<property>
      <name>yarn.timeline-service.enabled</name>
      <value>true</value> </property>
<property>
      <name>yarn.timeline-service.hostname</name>
      <value>localhost</value>
</property>
```

A la fin du démarrage, Démarrer le **cluster hadoop**, **timelineserver** et **historyserver**

Timeline Server, composant de yarn qui permet de suivre les job mapreduce issus de la traduction du script pig latin.

History server permet aux utilisateurs d'accéder accéder aux données historiques après qu'un travail Pig soit terminé.

```
./start-hadoop
yarn timelineserver
mapred --daemon start historyserver
```

II. Premier exemple

1. Accéder au grunt shell de Pig

- Pour ouvrir le shell interactive de pig avec une exécution en mode locale.

```
Pig -x local
```

2. Exemple wordcount

- Nous allons tester l'exemple wordcount avec un script pig latin. Pour ce faire, charger le fichier à lire

```
lines = LOAD '/shared_volume/alice.txt';
```

- Parser et nettoyer les données

```
words = FOREACH lines GENERATE FLATTEN(TOKENIZE((chararray)$0)) AS word;
clean_w = FILTER words BY word MATCHES '\\w+';
```

- Réaliser un groupement des mots

```
D = GROUP clean_w BY word;
```

- Générer un résultat sous forme word, count

```
E = FOREACH D GENERATE group, COUNT(Clean_w);
```

- Enregistre le résultat sous forme en local

```
-- Store output data in local output directory
STORE E INTO '/shared_volume/pig_out/WORD_COUNT/';
```

III. employés d'une entreprise

Cet exemple

1. Chargement des données

- On va travailler avec le fichier des employees.txt . Ses colonnes sont séparés par des ‘,’ et contient ID, Nom, Prenom, depno, Région, Salaire et departements.txt avec depno et name
- Copiez ces fichiers sur HDFS

```
hdfs dfs -mkdir input
```

1. *Analyse des employés*

- Quel est le salaire moyen des employés dans chaque département ?
- Combien d'employés travaillent dans chaque département ?
- Lister tous les employés avec leurs départements respectifs.
- Quels sont les employés ayant un salaire supérieur à 60 000 ?
- Quel est le département avec le salaire le plus élevé ?
- Lister tous les départements sans employés.
- Quel est le nombre total d'employés dans l'entreprise ?
- Lister tous les employés de la ville de Paris.
- Quel est le salaire total des employés dans chaque ville ?
- Quels sont les départements qui ont des femmes employées ?

Enregistrer le dernier résultat sur hdfs dans le dossier pigout/employes_femmes

- Vérifier les resultats sur hdfs

IV. *Analyse des films*

Le jeu de données proposé est notre base de films. Les données se trouvent sous format JSON dans deux fichiers séparés

- un exemple de tuple dans films

```
{ "_id": "movie:1", "title": "Vertigo", "year": 1958, "genre":
"drama", "summary": "...", "country": "USA", "director": { "_id": "artist:3"},
 "actors": [{ "_id": "artist:15", "role": "John
             Ferguson" }, { "_id": "artist:16", "role": "Madeleine Elster"}]
 }
```

- un exemple de tuple dans artists.

```
{ "_id": "artist:15", "last_name": "Stewart", "first_name":
"James", "birth_date": "1908" }
```

- Charger les deux fichiers sur hdfs
- Ouvrir le grunt shell de Pig
- charger les deux fichiers selon le format adequat.
- Créez une collection mUSA_annee groupant les films américains par année (code du pays: US).
- Créez une collection mUSA_director groupant les films américains par metteur en scène.
- Créez une collection mUSA_acteurs contenant des triplets (idFilm, idActeur, role). Chaque film apparaît donc dans autant de documents qu’il y a d’acteurs
  Aide: il faut « aplatir » la collection actors imbriquée dans chaque film.
- créez une collection moviesActors associant l’identifiant du film à la description complète de l’acteur.

Aide: utiliser une jointure . Consultez le schéma de mUSA_actors pour connaître le nom des colonnes.

- créez une collection fullMovies associant la description complète du film à la description complète de tous les acteurs.
  Aide: soit une jointure entre moviesActors et movies, puis un regroupement par film, ce qui un contenu correct mais très compliqué , soit un cogroup entre moviesUSA et moviesActors
- Créer une collection `ActeursRealisateurs` donnant pour chaque artiste la liste des films où il/elle a joué (éventuellement vide), et des films qu'il/elle a dirigé. On peut se contenter d'afficher l'identifiant de l'artiste :
  (artist:24,{},{(movie:10,Blade Runner,artist:24,Deckard),
  `(movie:34,Le retour du Jedi,artist:24,Han Solo)})`

- Enregistrer le dernier résultat sur hdfs dans le dossier pigout/ ActeursRealisateurs
- Vérifier les resultats sur hdfs
-

V. *Analyse des vols*

La datasets suivantes comporte 3 fichiers. L'objectif est de répondre aux questions suivantes : Dans ce qui suit, nous proposons une série d'exercices sous forme de requêtes.

- Top 20 des aéroports par volume total de vols

Quels sont les aéroports les plus fréquentés par trafic aérien total.
Pour chaque code d'aéroport, calculez le nombre de vols entrants, sortants et tous les vols.
Variation sur le thème : calculez les chiffres ci-dessus par jour, semaine, mois et au fil des ans.

- Popularité des transporteurs

Certains transporteurs(`carriers`) vont et viennent, d'autres affichent une croissance régulière. Calculez le volume (logarithmique base 10) - le nombre total de vols - chaque année, par transporteur. Les transporteurs sont classés selon leur volume médian (sur une période de 4 ans).

- Proportion de vols retardés

Un vol est retardé si le retard est supérieur à 15 minutes. Calculez la fraction de vols retardés selon différentes granularités temporelles (heure, jour, semaine, mois, année).

- Retards des transporteurs

Y a-t-il une différence dans les retards des transporteurs ? Calculez la proportion de vols retardés par transporteur, classés par transporteur, à différentes granularités temporelles (heure, jour, semaine, mois, année). Là encore, un vol est retardé si le retard est supérieur à 15 minutes.

- Itinéraires les plus fréquentés

Quels itinéraires sont les plus fréquentés ? Une première approche simple consiste à créer un tableau de fréquences pour la paire non ordonnée (i, j) où i et j sont des codes d'aéroport distincts.

La description de la dataset se trouve sur
http://stat-computing.org/dataexpo/2009/the-data.html

```
1       Year   1987-2008
2       Month  1-12
3       DayofMonth     1-31
4       DayOfWeek      1 (Monday) - 7 (Sunday)
5       DepTime        actual departure time (local, hhmm)
6       CRSDepTime     scheduled departure time (local, hhmm)
7       ArrTime        actual arrival time (local, hhmm)
8       CRSArrTime     scheduled arrival time (local, hhmm)
9       UniqueCarrier  unique carrier code
10      FlightNum      flight number
11      TailNum        plane tail number
12      ActualElapsedTime      in minutes
13      CRSElapsedTime in minutes
14      AirTime        in minutes
15      ArrDelay       arrival delay, in minutes
16      DepDelay       departure delay, in minutes
17      Origin origin IATA airport code
18      Dest   destination IATA airport code
19      Distance       in miles
20      TaxiIn taxi in time, in minutes
21      TaxiOut        taxi out time in minutes
22      Cancelled      was the flight cancelled?
23      CancellationCode       reason for cancellation (A = carrier, B = weather, C = NAS, D
= security)
24      Diverted       1 = yes, 0 = no
25      CarrierDelay   in minutes
26      WeatherDelay   in minutes
27      NASDelay       in minutes
28      SecurityDelay  in minutes
29      LateAircraftDelay      in minutes
```

- Sortir de bash de hadoop-master
- Arrêter les trois conteneurs

```
docker stop hadoop-master hadoop-slave1 hadoop-slave2
```