

TP5: Apache PIG - Solutions

1 Installation Apache PIG

```
# Accéder au master
docker exec -it hadoop-master bash

# Telecharger et installer Pig
wget https://dlcdn.apache.org/pig/pig-0.17.0/pig-0.17.0.tar.gz
tar -zxf pig-0.17.0.tar.gz
mv pig-0.17.0 /usr/local/pig
rm pig-0.17.0.tar.gz

# Variables d'environnement
vim ~/.bashrc
# Ajouter:
export PIG_HOME=/usr/local/pig
export PATH=$PATH:$PIG_HOME/bin

source ~/.bashrc

# Configurer HDFS (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 YARN (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>

# Demarrer services
./start-hadoop
yarn timelineserver
mapred --daemon start historyserver
```

2 WordCount

```
pig -x local
```

```

lines = LOAD '/shared_volume/alice.txt';
words = FOREACH lines GENERATE FLATTEN(TOKENIZE((chararray)$0)) AS word;
clean_w = FILTER words BY word MATCHES '\w+';
D = GROUP clean_w BY word;
E = FOREACH D GENERATE group, COUNT(clean_w);
STORE E INTO '/shared_volume/pig_out/WORD_COUNT/';

```

3 Employes

```

hdfs dfs -mkdir input
hdfs dfs -put employees.txt input/
hdfs dfs -put departements.txt input/
pig

```

```

-- Charger donnees
employees = LOAD 'input/employees.txt' USING PigStorage(',') AS (id:int, nom:chararray, prenom:chararray, depno:int, region:chararray, salaire:double);
departements = LOAD 'input/departements.txt' USING PigStorage(',') AS (depno:int, name:chararray);

-- 1. Salaire moyen par departement
grp_dep = GROUP employees BY depno;
sal_moy = FOREACH grp_dep GENERATE group AS depno, AVG(employees.salaire) AS salaire_moyen;

-- 2. Nombre employes par departement
nb_emp = FOREACH grp_dep GENERATE group AS depno, COUNT(employees) AS nb_employes;

-- 3. Employes avec leurs departements
emp_dep = JOIN employees BY depno, departements BY depno;
liste_emp = FOREACH emp_dep GENERATE employees::id, employees::nom, employees::prenom, departements::name;

-- 4. Employes salaire > 60000
sal_sup = FILTER employees BY salaire > 60000;

-- 5. Departement avec salaire le plus eleve
max_sal = FOREACH grp_dep GENERATE group AS depno, MAX(employees.salaire) AS max_salaire;
ordre_max = ORDER max_sal BY max_salaire DESC;
top_dep = LIMIT ordre_max 1;

-- 6. Departements sans employes
dep_emp = JOIN departements BY depno LEFT, employees BY depno;
dep_vides = FILTER dep_emp BY employees::id IS NULL;
liste_vides = FOREACH dep_vides GENERATE departements::depno, departements::name;

-- 7. Nombre total employes
all_emp = GROUP employees ALL;
total = FOREACH all_emp GENERATE COUNT(employees) AS total_employes;

-- 8. Employes de Paris
paris_emp = FILTER employees BY region == 'Paris';

-- 9. Salaire total par ville

```

```

grp_ville = GROUP employees BY region;
sal_ville = FOREACH grp_ville GENERATE group AS ville,
    SUM(employees.salaire) AS salaire_total;

-- 10. Departements avec femmes (suppose colonne sexe existe)
-- Adaptation: filtre sur prenom feminins ou ajout colonne sexe
femmes = FILTER employees BY /* condition femme */;
grp_fem = GROUP femmes BY depno;
dep_femmes = FOREACH grp_fem GENERATE group AS depno;
STORE dep_femmes INTO 'pigout/employees_femmes';

```

```

hdfs dfs -ls pigout/employees_femmes
hdfs dfs -cat pigout/employees_femmes/part-*

```

4 Analyse Films

```

hdfs dfs -put films.json input/
hdfs dfs -put artists.json input/
pig

```

```

-- Charger JSON
films = LOAD 'input/films.json' USING JsonLoader(
    '_id:chararray', 'title:chararray', 'year:int', 'genre:chararray',
    'country:chararray', 'director:{'_id:chararray}', 
    'actors:{t:{_id:chararray, role:chararray}}');

artists = LOAD 'input/artists.json' USING JsonLoader(
    '_id:chararray', 'last_name:chararray', 'first_name:chararray',
    'birth_date:chararray');

-- 1. Films USA par annee
filmsUSA = FILTER films BY country == 'US';
mUSA_annee = GROUP filmsUSA BY year;

-- 2. Films USA par realisateur
mUSA_director = GROUP filmsUSA BY director._id;

-- 3. Films USA avec acteurs (triplets)
filmsUSA_flat = FOREACH filmsUSA GENERATE _id AS idFilm,
    FLATTEN(actors) AS (idActeur:chararray, role:chararray);
mUSA_acteurs = FOREACH filmsUSA_flat GENERATE idFilm,
    idActeur, role;

-- 4. Films avec description complete acteurs
moviesActors = JOIN mUSA_acteurs BY idActeur, artists BY _id;
moviesActors = FOREACH moviesActors GENERATE
    mUSA_acteurs::idFilm, artists::_id, artists::last_name,
    artists::first_name, mUSA_acteurs::role;

-- 5. Description complete films et acteurs
fullMovies_join = JOIN moviesActors BY idFilm, filmsUSA BY _id;
fullMovies_grp = GROUP fullMovies_join BY filmsUSA::_id;
fullMovies = FOREACH fullMovies_grp GENERATE group, fullMovies_join;

-- Alternative avec COGROUP
moviesUSA = FOREACH filmsUSA GENERATE _id, title, year, genre;
fullMovies2 = COGROUP moviesUSA BY _id, moviesActors BY idFilm;

```

```
-- 6. Acteurs et Realisateurs
acteurs_films = FOREACH mUSA_acteurs GENERATE idActeur AS artistId,
    idFilm, role;
real_films = FOREACH filmsUSA GENERATE director._id AS artistId,
    _id AS idFilm, title;

acteurs_grp = GROUP acteurs_films BY artistId;
acteurs_list = FOREACH acteurs_grp GENERATE group AS artistId,
    acteurs_films AS films_joues;

real_grp = GROUP real_films BY artistId;
real_list = FOREACH real_grp GENERATE group AS artistId,
    real_films AS films_diriges;

ActeursRealisateurs = COGROUP acteurs_list BY artistId,
    real_list BY artistId;
ActeursRealisateurs = FOREACH ActeursRealisateurs GENERATE
    group AS artistId, acteurs_list.films_joues,
    real_list.films_diriges;

STORE ActeursRealisateurs INTO 'pigout/ActeursRealisateurs';
```

```
hdfs dfs -ls pigout/ActeursRealisateurs
```

5 Analyse Vols

```
hdfs dfs -put flights*.csv input/
pig
```

```
flights = LOAD 'input/flights*.csv' USING PigStorage(',')
AS (Year:int, Month:int, DayofMonth:int, DayOfWeek:int,
    DepTime:int, CRSDepTime:int, ArrTime:int, CRSArrTime:int,
    UniqueCarrier:chararray, FlightNum:int, TailNum:chararray,
    ActualElapsedTime:int, CRSElapsedTime:int, AirTime:int,
    ArrDelay:int, DepDelay:int, Origin:chararray, Dest:chararray,
    Distance:int, TaxiIn:int, TaxiOut:int, Cancelled:int,
    CancellationCode:chararray, Diverted:int, CarrierDelay:int,
    WeatherDelay:int, NASDelay:int, SecurityDelay:int,
    LateAircraftDelay:int);

-- 1. Top 20 aeroports par volume
dep = FOREACH flights GENERATE Origin AS airport;
arr = FOREACH flights GENERATE Dest AS airport;
all_flights = UNION dep, arr;
grp_airport = GROUP all_flights BY airport;
vol_airport = FOREACH grp_airport GENERATE group AS airport,
    COUNT(all_flights) AS total_vols;
top20 = ORDER vol_airport BY total_vols DESC;
top20 = LIMIT top20 20;

-- Variation: par jour
dep_date = FOREACH flights GENERATE Origin AS airport, Year,
    Month, DayofMonth;
arr_date = FOREACH flights GENERATE Dest AS airport, Year,
    Month, DayofMonth;
all_date = UNION dep_date, arr_date;
grp_date = GROUP all_date BY (airport, Year, Month, DayofMonth);
```

```

vol_date = FOREACH grp_date GENERATE FLATTEN(group),
    COUNT(all_date) AS vols;

-- 2. Popularite transporteurs
grp_carrier = GROUP flights BY (UniqueCarrier, Year);
vol_carrier = FOREACH grp_carrier GENERATE
    FLATTEN(group) AS (carrier, year),
    LOG10((double)COUNT(flights)) AS log_volume;

median_carrier = GROUP vol_carrier BY carrier;
median_val = FOREACH median_carrier GENERATE group AS carrier,
    /* calcul median */ vol_carrier;
sorted_carriers = ORDER median_val BY /* median */ DESC;

-- 3. Proportion vols retardes
delayed = FOREACH flights GENERATE
    (ArrDelay > 15 ? 1 : 0) AS is_delayed, Year, Month, DayofMonth;
grp_delay = GROUP delayed BY (Year, Month, DayofMonth);
prop_delay = FOREACH grp_delay GENERATE FLATTEN(group),
    (double)SUM(delayed.is_delayed) / COUNT(delayed) AS proportion;

-- 4. Retards par transporteur
delayed_carrier = FOREACH flights GENERATE UniqueCarrier,
    (ArrDelay > 15 ? 1 : 0) AS is_delayed, Year, Month;
grp_carr_delay = GROUP delayed_carrier BY (UniqueCarrier, Year, Month);
prop_carr = FOREACH grp_carr_delay GENERATE FLATTEN(group),
    (double)SUM(delayed_carrier.is_delayed) / COUNT(delayed_carrier)
    AS proportion_delayed;
sorted_prop = ORDER prop_carr BY proportion_delayed DESC;

-- 5. Itineraires les plus frequentes
routes = FOREACH flights GENERATE
    (Origin < Dest ? CONCAT(Origin, ' - ', Dest) :
    CONCAT(Dest, ' - ', Origin)) AS route;
grp_routes = GROUP routes BY route;
freq_routes = FOREACH grp_routes GENERATE group AS route,
    COUNT(routes) AS frequence;
top_routes = ORDER freq_routes BY frequence DESC;
top_routes = LIMIT top_routes 20;

```

```

# Arreter
docker stop hadoop-master hadoop-slave1 hadoop-slave2

```