EXAMEN HADOOP

(HDFS, MapReduce, HBase, Hive & Pig)

5AL1
Egor BEREZOVSKIY, Mathieu PEQUIN & Guillaume VILLEREZ

HDFS

Utilisation du shell HDFS

1. Créer le dossier /user/cloudera/examen/donnees

```
$ hadoop fs -mkdir -p /user/cloudera/examen/donnees
```

2. Uploader les fichiers users.csv, notations.csv, movies.csv au sein de /user/cloudera/examen/donnees

```
$ cd tp/
$ hadoop fs -put *.csv /user/cloudera/examen/donnees
$ hadoop fs -ls /user/cloudera/examen/donnees
Found 4 items
-rw-r--r--   1 cloudera cloudera        421 2016-04-15 00:51
/user/cloudera/examen/donnees/hbase_ventes.csv
-rw-r--r--   1 cloudera cloudera     236344 2016-04-15 00:51
/user/cloudera/examen/donnees/movies.csv
-rw-r--r--   1 cloudera cloudera    1979173 2016-04-15 00:51
/user/cloudera/examen/donnees/notations.csv
-rw-r--r--   1 cloudera cloudera      22628 2016-04-15 00:51
/user/cloudera/examen/donnees/users.csv
```

3. Afficher l'espace restant sur notre cluster

```
$ hadoop fs –df
Filesystem                                Size       Used    Available
Use%
hdfs://quickstart.cloudera:8020  58613309440  772074714  46234537984
1%
```

4. Afficher le nombre de réplica du fichier notations.csv

```
$ hadoop fs -ls /user/cloudera/examen/donnees/notations.csv
-rw-r--r--   1 cloudera cloudera    1979173 2016-04-15 00:51
/user/cloudera/examen/donnees/notations.csv
```

On peut observer que le fichier est actuellement répliqué une seule fois.

5. Mettre le nombre de réplica à 4 pour le fichier notations.csv

```
$ hadoop fs -setrep 4 /user/cloudera/examen/donnees/notations.scv
$ hadoop fs -ls /user/cloudera/examen/donnees/notations.csv
-rw-r--r--   4 cloudera cloudera    1979173 2016-04-15 00:51
/user/cloudera/examen/donnees/notations.csv
```

On peut observer que la réplication est passée à 4 pour ce fichier.

6. Nous n'avons qu'un seul nœud et par conséquent, un seul DataNode. Quel sera l'impact sur notre Cluster d'avoir un réplica de 4 ?

Tant que notre cluster ne sera composé que d'un seul DataNode, la modification de la réplication n'aura pas d'impact direct. Si jamais de nouveaux DataNode sont par la suite ajoutés, la réplication entrera en jeu jusqu'à que notre fichier soit répliqué 4 fois.

HBase

1. Créer une table « Ventes »
2. Créer une famille « produit » en gardant un historique de 5 versions
3. Créer une famille « client » en gardant un historique de 3 versions

```
$ hbase shell
hbase> create “Ventes”,{NAME => “produit”, VERSIONS => 5}, {NAME =>
“client”, VERSIONS => 3}
```

On crée en même temps nos 2 familles avec un historique de 5 version pour les produits et de 3 pour les clients.

4. Insérer les données contenues dans le fichier « hbase_ventes.csv »

```
$ hadoop fs -put hbase_ventes.csv
/user/cloudera/examen/donnees/hbase_ventes.csv
$ hbase shell
hbase> put “Ventes”,”1”,”produit:id”,”p0001”
hbase> put “Ventes”,”1”,”produit:libelle”,”Clavier microsoft”
hbase> put “Ventes”,”1”,”produit:prix”,”44,9”
hbase> put “Ventes”,”1”,”produit:quantite”,”4”
hbase> put “Ventes”,”1”,”client:nom”,”Dupont”
hbase> put “Ventes”,”1”,”client:prenom”,”JEAN”
hbase> put “Ventes”,”2”,”produit:id”,”p0002”
hbase> put “Ventes”,”2”,”produit:libelle”,”Batterie 90w DELL”
hbase> put “Ventes”,”2”,”produit:prix”,”87,85”
hbase> put “Ventes”,”2”,”produit:quantite”,”10”
hbase> put “Ventes”,”2”,”client:nom”,”Azik”
hbase> put “Ventes”,”2”,”client:prenom”,”Michel”
hbase> put “Ventes”,”2”,”client:prenom”,”Michel”
hbase> scan “Ventes”
1 column=client:nom, timestamp=1460711666621, value=Dupont
1 column=client:prenom, timestamp=1460711695894, value=JEAN
1 column=produit:id, timestamp=1460711405970, value=p0001
1 column=produit:libelle, timestamp=1460711547298, value=Clavier
microsoft
1 column=produit:prix, timestamp=1460711582171, value=44,9
1 column=produit:quantite, timestamp=1460711626119, value=4
2 column=client:nom, timestamp=1460711853373, value=Azik
2 column=client:prenom, timestamp=1460711878855, value=Michel
2 column=produit:id, timestamp=1460711747922, value=p0002
2 column=produit:libelle, timestamp=1460711774030, value=Batterie
90w DELL
2 column=produit:prix, timestamp=1460711804935, value=86,85
2 column=produit:quantite, timestamp=1460711831811, value=10
2 row(s) in 0.0370 seconds
```

5. Modifier la famille « client » afin qu'elle puisse contenir 7 versions

```
hbase> alter "Ventes", { NAME => client, VERSIONS => 7 }
```

6. Mettre à jour le rowid « 1 », la colonne « produit :quantite » saisir « 10 » Remettre à jour ces données en saisissant « 20 »

```
hbase> put "Ventes","1","produit:quantite","10"
hbase> put "Ventes","1","produit:quantite","20"
```

7. Afficher la 1ère version pour la ligne : rowid « 1 », colonne « produit :quantite »

```
hbase> get "Ventes", "1", { COLUMN => "produit:quantite",VERSIONS => 1 }

COLUMN                   CELL
produit:quantite        timestamp=1460712113700, value=4
produit:quantite        timestamp=1460712149700, value=10
produit:quantite        timestamp=1460712193700, value=20
1 row(s) in 0.0080 seconds
```

Hive

Insertion des données par une table de staging

1. Créer une table « movies_tmp » à partir du fichier « movies.csv »

```
CREATE TABLE IF NOT EXISTS movies_tmp (
    movieID INT,
    movieTitle string,
    releaseDate string,
    videoReleaseDate string,
    imdbLink string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' ;
```

2. Insérer les données du fichier « movies.csv » dans la table movies_tmp

```
LOAD DATA INPATH '/user/cloudera/examen/donnees/movies.csv' INTO TABLE
movies_tmp;
```

3. Créer une table « movies » avec comme schéma en chargeant les données de movies_tmp

```
CREATE TABLE IF NOT EXISTS movies (movieID INT, movieTitle string,
releaseDate string, videoReleaseDate string);
```

```
CREATE TABLE IF NOT EXISTS movies (
    movieID INT,
    movieTitle string,
    releaseDate string,
    videoReleaseDate string);

INSERT INTO movies
SELECT movieID, movieTitle,releaseDate,videoReleaseDate FROM movies_tmp;
```

Table externe

1. Créer une table externe « notations »

```
CREATE EXTERNAL TABLE IF NOT EXISTS notations (
    user_id INT,
    item_id INT,
    rating INT,
    timestamp INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
LOCATION '/user/cloudera/examen/donnees/notations';
```

2. Charger les données du fichier « notations.csv » dans la table notations.

```
LOAD DATA INPATH '/user/cloudera/examen/donnees/notations.csv' INTO
TABLE notations;
```

3. Créer une table externe « ventes » faisant référence à la table « ventes » contenue sur HBase. Récupérer l'ensemble de ses familles et colonnes

```
CREATE EXTERNAL TABLE IF NOT EXISTS Ventes(
    id INT,
    libelle string,
    prix FLOAT,
    quantite INT,
    nom string,
    prenom string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES (‘hbase.columns.mapping’ =
‘id:id,libelle:libelle,prix:prix,quantite:quantite,nom:nom,prenom:prenom’)
TBLPROPERTIES ('hbase.table.name' = 'Ventes')
LOCATION '/user/cloudera/examen/donnees/ventes';
```

Partitions

1. Créer la table « users » en partitionnant sur userid

```
SET hive.exec.dynamic.partition=true
SET hive.exec.dynamic.partition.mode=nonstrict
SET hive.exec.max.dynamic.partitions.pernode=1000
SET hive.exec.max.dynamic.partitions=10000

CREATE TABLE users (
   age INT,
   gender STRING,
   occupation STRING,
   zip_code INT)
PARTITIONED BY (user_id INT);
```

2. Charger dynamiquement les données et les partitions au sein de la table users

```
CREATE TABLE users_tmp (
   user_id INT,
   age INT,
   gender STRING,
   occupation STRING,
   zip_code INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '|';

LOAD DATA LOCAL INPATH '/user/cloudera/examen/donnees/users.csv'
OVERWRITE INTO TABLE users_tmp;

FROM user_tmp INSERT OVERWRITE TABLE users PARTITION(user_id) SELECT
user_id, age, gender, occupation, zip_code;
```

Quelques requêtes

1. Donner la moyenne des notes données pour chaque utilisateur (« rating » de la table « notations »)

```
SELECT users.age, AVG(notations.rating) FROM notations JOIN users ON
notations.user_id = users.user_id GROUP BY users.id;
```

2. Donner la moyenne des notes (rating de la table notations) par gender (table users)

```
SELECT users.age, AVG(notations.rating) FROM notations JOIN users ON
notations.user_id = users.user_id GROUP BY users.gender;
```

Pig

Quelques requêtes

1. Donner la moyenne des notes données pour chaque utilisateur (« rating » de la table « notations »)

```
Notes = LOAD '/user/cloudera/examen/donnees/notations.csv' AS (
    user_id:int,
    item_id:int,
    rating:int,
    timestamp:int);

GroupByUser = GROUP Notes BY user_id;
Ratings = FOREACH GroupByUser GENERATE group, AVG(Notes.rating);

DUMP Ratings;
```

2. Donner la moyenne des notes (rating de la table notations) par gender et par zipcode (table users)

```
Notes = LOAD '/user/cloudera/examen/donnees/notations.csv' AS (
    user_id:int,
    item_id:int,
    rating:int,
    timestamp:int);

Users = LOAD '/user/cloudera/examen/donnees/users.csv' USING
PigStorage('|') AS (
    user_id:int,
    age:int,
    gender:chararray,
    occupation:chararray,
    zipcode:int);

NoteByUser = JOIN Notes BY user_id, Users BY user_id;

ByGenderAndZipcode = GROUP NoteByUser BY (gender, zipcode);

Ratings = FOREACH ByGenderAndZipcode GENERATE group.gender,
group.zipcode, AVG(NoteByUser.rating);

DUMP Ratings;
```