Apache Hive

Datawarehouse sur HDFS

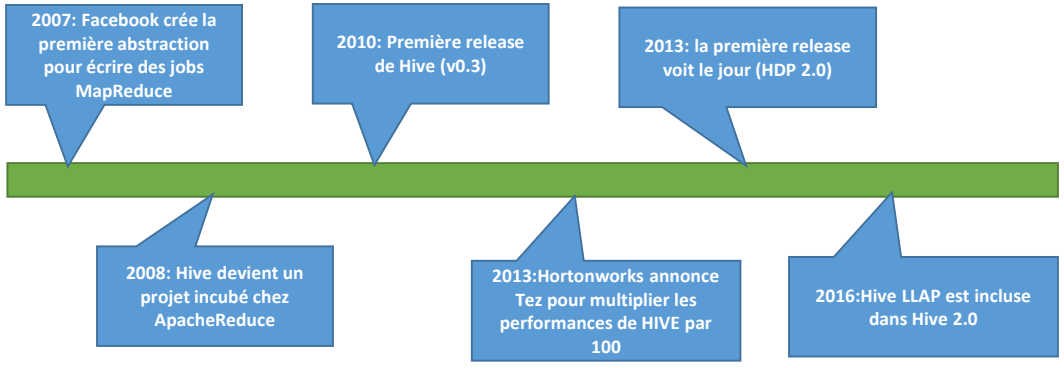
Agenda

- Introduction
- Terminologie
- Types de données
- HiveQL
- Tables internes vs Tables externes
- Partitionnement des tables
- Bucketing des tables
- Architecture
- Formats de fichiers supportés
- Les transactions sur Hive

C’est quoi Apache Hive

- Hive est un outil de datawarehousing open source construit au dessus de Hadoop
- Il permet d’analyser de larges volumes de données stockées sur HDFS
- Initialement développé pour des utilisateurs ayant un background SQL
- Hive offre un langage (HQL) similaire à SQL pour l’intérogation de données stockées sur HDFS
- Les requêtes HQL sont transformées en Job MapReduce(Tez) qui sont ensuite lancés sur YARN
- Hive permet de cacher la complexité de l’API MapReduce/HDFS pour un utilisateur final:
  - Pas besoin de connaitre Java pour utiliser Hive
  - Pas besoin de connaitre HDFS pour utiliser Hive
  - Suffit de connaitre SQL

Apache Hive: un peu d'histoire



Terminologie

- Database:
  - Catalogue de tables
  - Espace de nommage servant à éviter les conflits de nommage pour les tables, vues, partitions, colonnes, ….
- Table:
  - Unité de stockage de données ayant le même schéma
- Partition: diviser une table en fonction suivant les valeurs d'une ou plusieurs colonnes
  - Chaque table peut avoir une ou plusieurs clés de partitionnement
  - Chaque valeur de la clé de partitionnement définit une partition
  - Elle sert à optimiser les requêtes
  - Les données de chaque partition sont stockées dans un repertoire dédié
  - Exemple:
    - Partitionner les données de vente par date (il est à la charge de l'utilisateur de s'assurer que les données d'une partition correspondent bien à la colonne)
- Bucket/Cluster:
  - Chaque partition est stockée dans un bucket en fonction de la valeur du hash d'une certaine colonne
  - Les buckets peuvent être utilisés pour échantillonner efficacement les données d'une partition

Types de données

Primitifs

- TINYINT (1 byte signed integer)
- SMALLINT (2 byte signed integer)
- INT (4 byte signed integer)
- BIGINT (8 byte signed integer)
- BOOLEAN
- FLOAT
- DOUBLE
- STRING
- BINARY

Complexes

- Tableau:
  - ARRAY<data_type>
- Maps:
  - MAP<primitive_type, data_type>
- Structures:
  - STRUCT<col_name : data_type, ...>
- Union:s
  - UNIONTYPE<data_type, data_type, ...>
- TIMESTAMP
- DATE
- INTERVAL

```
CREATE TABLE employees (
  name         STRING,
  salary       FLOAT,
  subordinates ARRAY<STRING>,
  deductions   MAP<STRING, FLOAT>,
  address      STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>);
```

HiveQL, SQL pour Hive

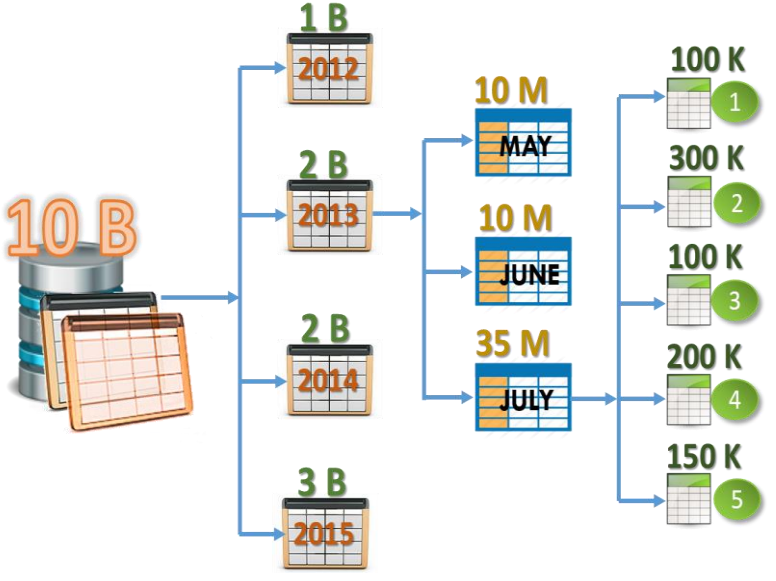
- HiveQL supporte
  - *DDL: create, alter, drop*
  - *DML: load, insert, select*
  - *Fonction utilisateurs (UDF): un mécanisme pour étendre les fonctions offertes par Hive (e.g. une fonction qui transforme un string en majuscule)*
  - *Appel à des programmes externes MapReduce*

Table interne vs table externe

- Les tables internes:
  - Ce sont des tables entièrement gérées par Hive
  - Leurs données sont stockées par défaut dans un répertoire HDFS dédié au warehouse Hive
    - ***/apps/hive/warehouse***
  - La suppression d'une table entraine la suppression des données et des métadonnées
- Tables externes:
  - Elles sont mappées sur un répertoire stocké sur HDFS
  - La suppression d'une table n'entraine que la suppression des méta-données du méta-store

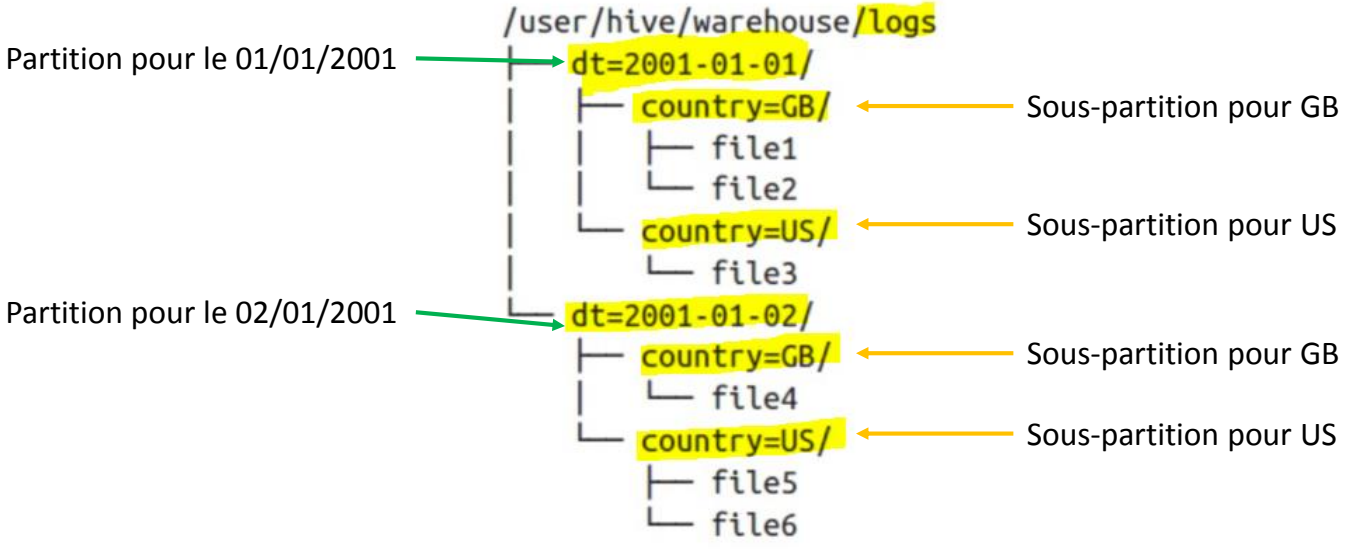
Hive: partitionnement des tables (1)

- Les tables sur Hive peuvent être organisées en partitions
- Le partitionnement permet de diviser une table en parties reliées en se basant sur les valeurs d'une ou plusieurs clés de partitionnement
  - Date, cite, département, ….
- Le partitionnement permet de solliciter uniquement les parties pouvant répondre à une requête



Hive: partitionnement des tables (2)

- Le partitionnement est physique:
  - les partitions sont stockées dans une arborescence de sous-repertoires dédiées



Hive: partitionnement des tables (3)

Création de tables

```
Hive> CREATE TABLE table_name (column1 data_type, column2 data_type) PARTITIONED BY (partition1 data_type, partition2 data_type,….);
```

Renommer une partition

```
Hive> ALTER TABLE table_name PARTITION (partition1=xxx,partition2=yyy) RENAME TO PARTITION (partition1=XXX,partition2=YYY);
```

Ajouter une partition

```
hive> ALTER TABLE table_name
ADD PARTITION
(partition1=…,partition2=…)
location ‘hdfs_directory_path';
```

Supprimer une partition

```
hive> ALTER TABLE table_name DROP [IF EXISTS] PARTITION
(partition1=xxx,partition2=yyy) ;
```

Hive: partitionnement des tables (4)

- Avantage
  - Le partitionnement permet de distribuer la charge horizontallement
  - Accélération des requêtes surt des volumes de données faibles

    Inconvénient: Risque d'avoir plusieurs petites partitions, donc plusieurs répertoires.
- Inconvénient
  - Le partitionnement est efficace pour de petits volumes, mais risque que certaines requêtes, telles que les GROUP BY, soient lentes au vu de la taille des données à charger

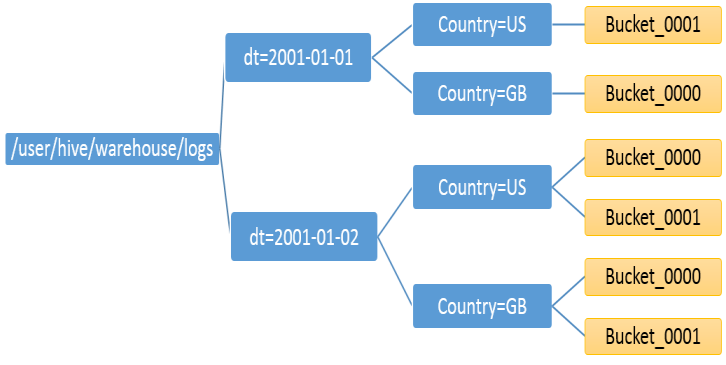
Hive: Bucketing

- C'est un mécanisme qui permet de diviser les données d'une table ou partition en plusieurs sous-ensembles
- La répartition des données sur les bucket se fait:
  - Identification du bucket accueillant une donnée se fait par: (function de hashage sur une colonne) % (nombre de buckets)
  - Les tuples ayant les mêmes hash atterissent dans la même bucket
  - Physiquement, chaque bucket est stocké dans un fichier

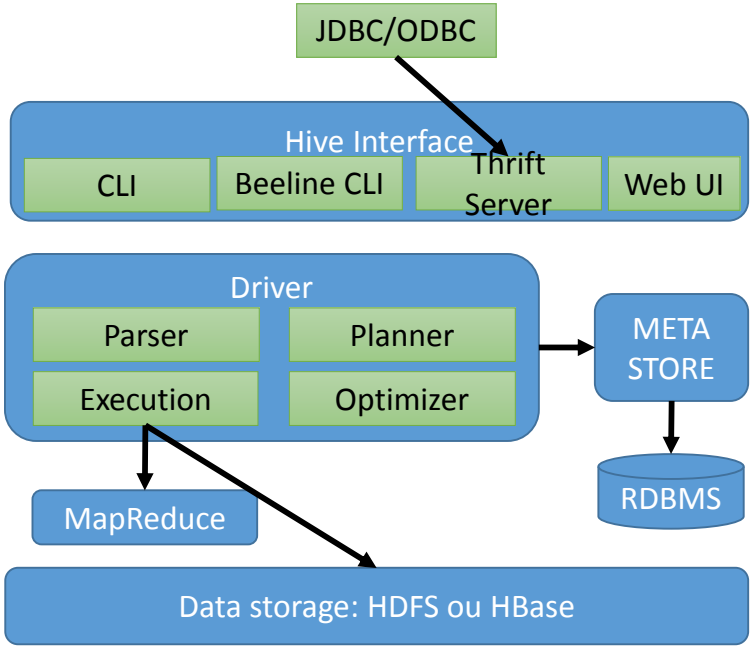
Création de tables buckettées

```
Hive> CREATE TABLE table_name (column1 data_type, column2 data_type, …)
CLUSTERED BY column_name INTO X BUCKETS
```

Arborescence HDFS d'une table buckettée

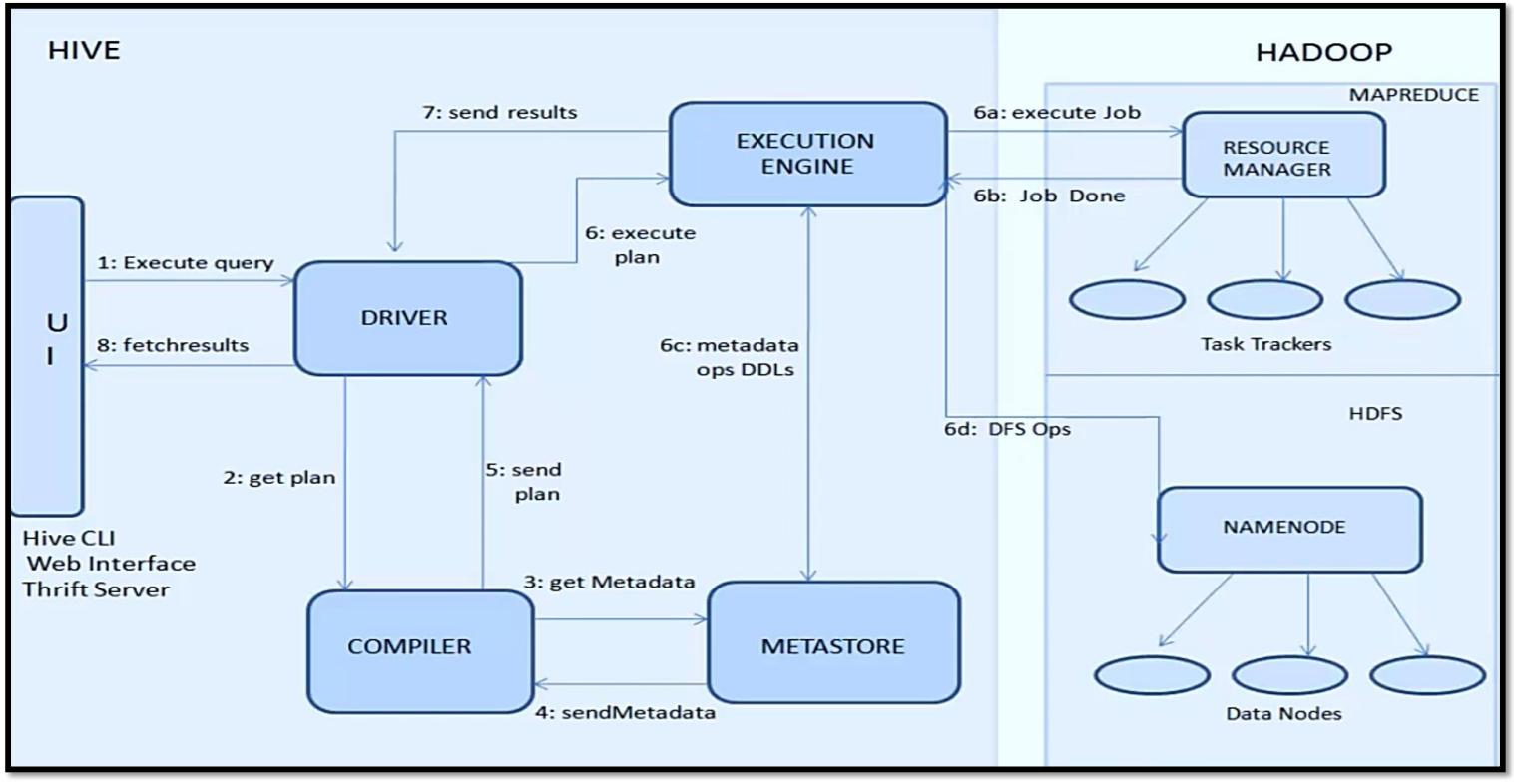


Apache Hive: Architecture



- Drivers:
  - Il gère les sessions HQL
  - Il est composé de 4 composants:
    - *Parser*: vérifie que la requête est s yntaxiquement correcte
    - *Planner*: génère le plan d'exécution de la requête
    - *Optimizer*: optimise le plan d'exécution
    - *Execution*: exécute et coordonne les tâches répondant à la reqûete
- Metastore:
  - Point central dans l'architecture Hive
  - Base de données relationnelle pour stocker les informations des tables Hive
    - Schéma des tables, localisation des données dans HDFS, format des fichiers, …
  - La base doit être backupée assez souvent
    - La perte de cette base entraine la perte du datawarehouse
  - Il est composé de 2 services:
    - Base de données relationnel pour stocker toutes les infos nécessaires
    - D'une API pour donner ces informations aux autres services HIVE
  - Les méta-données stockées sont relatives:
    - Databases, Tables, Partitions, Buckets
- Interfaces d'interaction avec Hive:
  - CLI (Command Line Interface): intéragir avec Hive en utilisant le shell
  - ThriftServer: permet de se connecter à Hive avec des clients JDBC/ODBC

Apache Hive: Architecture



Formats de fichier supportés par Hive

- Formats de fichiers supportés
  - Text File
  - Sequence File
  - RC File
  - Avro File
  - ORC File
  - Parquet File
  - Format propriétaire en définissant INPUTFORMAT et OUTPUTFORMAT

Les transactions sur Hive

- Apparues à partir de Hive 0.13
- Hive supporte les transactions au niveau d'une seule table à la fois
- Les tables doivent:
  - Etre marquees comme étant des tables transactionnelle
  - Avoir ORC comme format de stockage
  - Etre bucketisées
- Les opérations ACID: les commandes SQL standard pour l'insertion, mise à jour et suppression sont supportées
- Il n'y a pas de notion de clé primaire:
  - Hive ne vérifie pas l'unicité des clés
  - Il faut simuler une clé primaire en utilisant une convention au niveau applicatif et verifier son unicité à l'insertion
- Les données peuvent écrites en mode straming dans une table HIVE transactionnelle
  - Hive Streamng API, Flume, Storm, Spark, …
- Le modèle de concurrence de HIVE est optimiste:
  - Le premier committeur gagne la partieCompactions
- Compaction des tables transactionnnelles
  - Automatiquement (géré par Hive) ou programmé par un utilisateur

Création d'une table transactionnelle

```
Hive> CREATE TABLE table_acid_name (column1 data_type, column2 data_type, …, column N typeN)
CLUSTERED BY (column1) INTO X BUCKETS
STORED AS ORC
TBLPROPERTIES ("transactional"="true" );
```

Opérations supportées

```
SELECT * from table_acid_name where ….
INSERT INTO table_acid_name VALUES (val1, val2, …valN);
DELETE FROM table_acid_name WHERE …;
UPDATE hello_acid SET value = 10 WHERE …;
```

TP Hive

- Charger une table Hive à partir d'un fichier stocké dans HDFS
- Créer une table à partir d'une autre table
- Créer une table partitionnée
- Faire des requêtes