Formation Hadoop

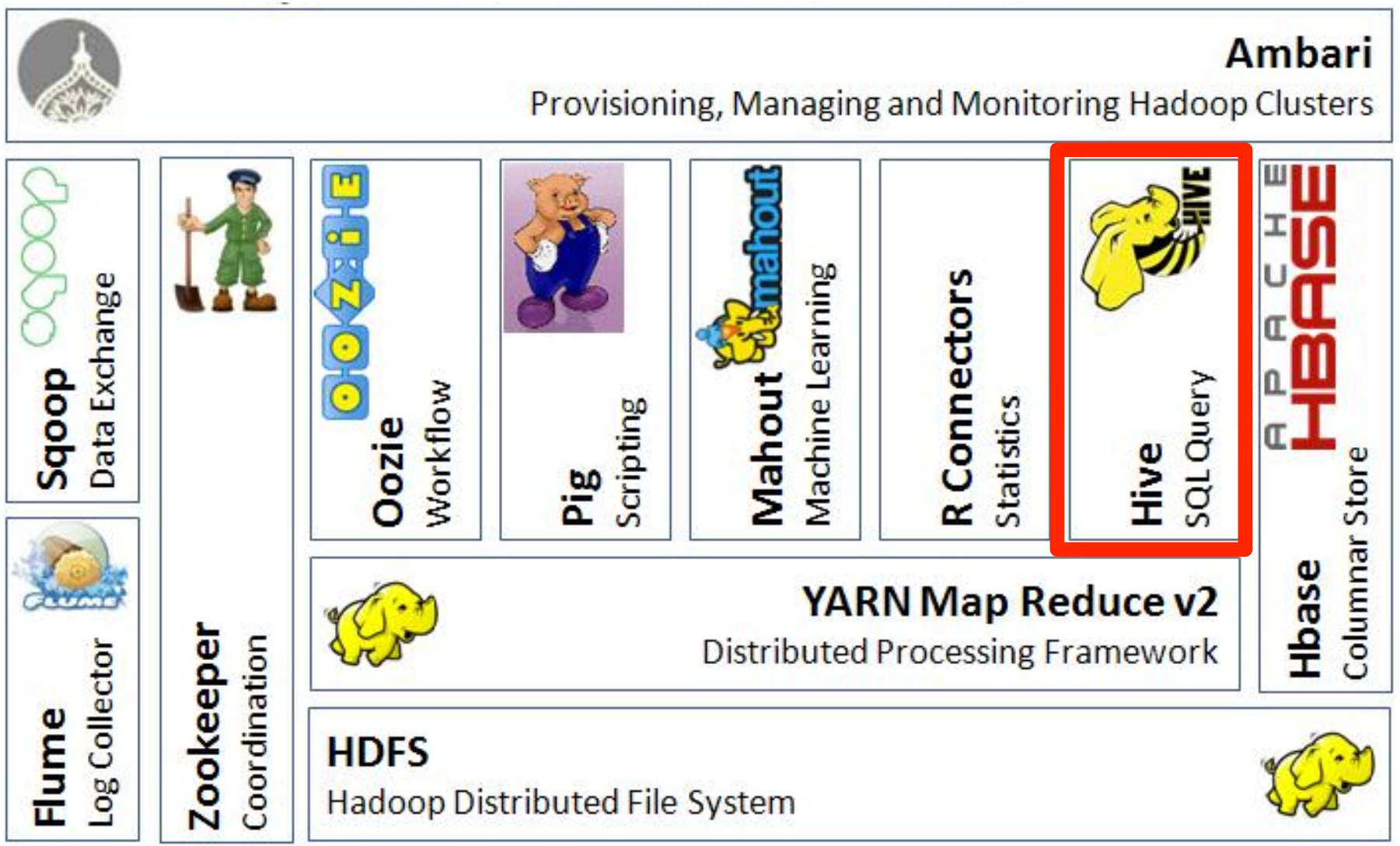
Chapitre Hive

HIVE

Hive

- Introduction de Hive
- Gestion des tables
- Gestion des données
- Gestion des requêtes (Query)
- Architecture
- Jointures

Hive


Ambari
Provisioning, Managing and Monitoring Hadoop Clusters
Sqoop
Data Exchange
Flume
Log Collector
Zookeeper
Coordination
Oozie
Workflow
Pig
Scripting
Mahout
Machine Learning
R Connectors
Statistics
Hive
SQL Query
Hbase
Columnar Store
YARN Map Reduce v2
Distributed Processing Framework
HDFS
Hadoop Distributed File System

Qu’est que Hive ?

- Entrepôt de données pour Hadoop Langage semblable à SQL, appelé HiveQL (HQL)

N’est pas conçu pour :

- Le traitement des transactions en ligne et des requêtes à temps réel

HIVE - Concept

- HIVE a été développé par **Facebook** en 2007 et permet d'utiliser les « données » sur le HDFS sans plus de connaissance que l'on a sur une base de données classique.

- HIVE propose un dialecte de SQL (HQL, Hive Query Language) pour définir/ interroger des données dans un entrepôt de données stocké dans le HDFS.

- HIVE ne fournit pas toutes les fonctionnalités d'un OLTP comme la mise à jour des données.

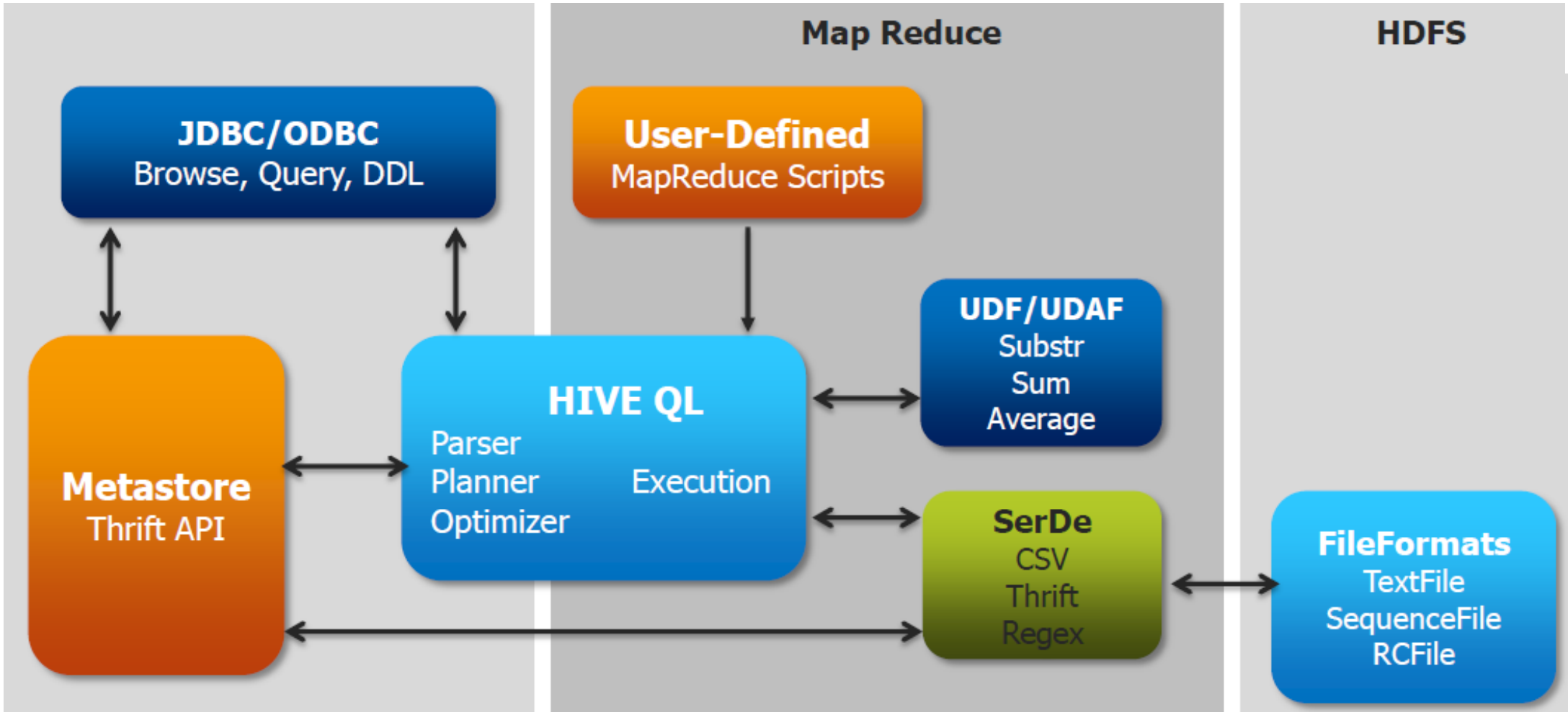
- HIVE permet d'utiliser HDFS comme source de données et le MapReduce comme moteur.

- Une connexion permet d'utiliser HIVE pour interroger les données présentent dans Hbase en créant une vue HIVE

HIVE - SGBDR vs HIVE

Dans un SGBDR	Dans HIVE
❑ Le schéma de la table est établi lors de la création de la table (schema on write).	❑ Le schéma de la table est établi lors de l'interrogation (schema on read).
❑ Si les données ne sont pas conformes aux schémas, elles sont rejetées.	❑ Lors du chargement, les données sont déplacées/copiées.
❑ Lors du chargement, les données sont lues, parsées et sérialisées dans le format interne du SGBD.	❑ Le chargement est plus rapide.

HIVE - Architecture



HIVE - Utilisation

HiveQL supporte

- DDL (Create, Alter, Drop)
- DML (Load, Insert, Select)
- Fonctions utilisateurs
- Appel à des programmes externe MapReduce

HIVE - Gestion des tables

Utilisation d'opérations Hive Data Definition Language (DDL)

Opération	Commande
Création	CREATE TABLE db_name;
Description	DESCRIBE db_name;
Liste	SHOW TABLES;
Modification	ALTER TABLE db_name ADD COLUMNS (field1 STRING);
Suppression	DROP TABLE db_name;

HIVE - Gestion des données

Utilisation d'opérations Hive Data Manipulation Language (DML)

Opération	Commande
Chargement des données locales	LOAD DATA LOCAL INPATH './files/data_db_name.txt' OVERWRITE INTO TABLE db_name;
Chargement des données HDFS	LOAD DATA INPATH '/user/myname/data_db_name.txt' OVERWRITE INTO TABLE db_name;

Utilisation d'opérations SQL

Opération	Commande
Retrieving information	SELECT from_columns FROM table WHERE conditions;
All values	SELECT * FROM table;
Some values	SELECT * FROM table WHERE rec_name = "value";
Multiple criteria	SELECT * FROM TABLE WHERE rec1 = "value1" AND rec2 = "value2";
Selecting specific columns	SELECT column_name FROM table;
Retrieving unique output records	SELECT DISTINCT column_name FROM table;
Sorting	SELECT col1, col2 FROM table ORDER BY col2;
Counting rows	SELECT COUNT(*) FROM table;
Grouping with counting	SELECT owner, COUNT(*) FROM table GROUP BY owner;

HIVE - Exemple création de table

Création de la table airlinetable

```
CREATE DATABASE IF NOT EXISTS airlinedb ;
USE airlinedb ;
CREATE EXTERNAL TABLE IF NOT EXISTS airlinetable
(AirlineID int
,AirlineName string
,AirlineAlias string
,IATA string
,ICAO string
,CallSign string
,Country string
,Active string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',‘
LOCATION '/user/hive/warehouse/demo';
```

HIVE - Créer une table

hive> **CREATE TABLE employes (id INT, prenom STRING, nom STRING)**

1ere ligne : créer une table de 3 colonnes

> **ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;**

2eme ligne : indique que le séparateur entre les champs est une virgule

La commande doit finir par un point virgule pour l'exécuter

---

hive> **DESCRIBE employes;** Affiche la structure de la table

id	int
prenom	string
nom	string

HIVE - Charger les données dans la table

hive> **LOAD DATA LOCAL INPATH 'data/data-employes.txt**

Copier les valeurs à partir d'un fichier local data-employes.txt

> **OVERWRITE INTO TABLE employes;**

Les enregistrements éventuels de la table employes sont supprimés

---

$ hdfs dfs -cat /user/hive/warehouse/employes/data-employes.txt

100,jean,martin
200,robert,poulain
300,michel,dunot

Afficher les enregistrements copiés.
Les tables Hive sont stockés par défaut dans le répertoire /user/hive/warehouse

HIVE - Effectuer des requêtes simples

hive> **SELECT COUNT(*) FROM employes;**

Compter le nombre d'enregistrements de la table employes

hive> **SELECT * FROM employes WHERE prenom = "Robert";**

Afficher les enregistrements satisfaisant un critère

HIVE - Supprimer une table

hive> **DROP TABLE employes;**

Supprimer la table sélectionnée

$ hdfs dfs -ls /user/hive/warehouse/

La table a été supprimée

HIVE - Optimisation

- Record Columnar File format (RCFile)
  - ✓ Format row-columnar hybride qui permet une analyse efficace lorsque seul un sous-ensemble des données est nécessaire
- Partition: définir ses propres colonnes, son stockage et sa sérialisation
  - ✓ Les partitions permettent d'accélérer les requêtes es partitions sont physiquement stockées dans des répertoires séparés dans HDFS
- Buckets: découpage des partitions
  - ✓ Optimisation pour les jointures

HIVE - Exemple de partitions


```
CREATE EXTERNAL TABLE clicks (
  hms          STRING,
  hostname     STRING,
  process      STRING,
  pid          INT,
  uid          INT,
  message      STRING)
PARTITIONED BY (
  year         INT,
  month        INT,
  day          INT);
```

HIVE - Exécution

- Sous forme d'un script :

```
$ hive -f marequette.hql
```

- A l'aide d'un shell:

```
$ hive

$ hive> source marequette.hql
```

HIVE - Logs

Les logs sont disponibles :

- Hive Metastore

```
/var/log/hive/
hive.*
```

- HiveServer2

```
/var/log/hive/hive-
server2.*
```

- Cloudera utilise Log4j pour formater les logs, 6 niveaux de détails sont disponibles (TRACE, INFO, FATALE…).

HIVE - Interactions avec JAVA

- HiveServer2 est recommandé pour des clients Java.
- Il est livré avec un driver qui s'interface en mode embarqué et distant.
- Le mode distant repose sur un serveur Thrift d'Apache.
- Le mode distant est recommandé en production pour des raisons de sécurité.
- Configuration minimal au niveau de HiveServer2:

Propriété	Description
hive.metastore.uris	Uri1,ur2
hive.metastore.warehouse.dir	Path vers le Metastore
hive.server2.use.SSL	True/False

HIVE - Interactions avec JAVA

1. Création d'une table

```
// Connexion
Connection con = DriverManager.getConnection("jdbc:hive2://localhost:
10000/default", "", "");
Statement stmt = con.createStatement();
// On renseigne les types de la clé et de la valeur
String query = "create table testHiveTable (key int, value string)";
// Execution
ResultSet res = stmt.executeQuery(query);
```

2. Insertion de données

```
// A partir d'un fichier texte sur le hdfs contenant 2 champs par
ligne
String filePath = "/tmp"a.txt";
String query = "load data local inpath '" + filePath + "' into table
testHiveTable";
res = stmt.executeQuery(query);
```

3. Sélection de données

```
String query = "select * from testHiveTable";
res = stmt.executeQuery(query);
while (res.next()) {
      System.out.println(String.valueOf(res.getInt(1)) + "\t" +
res.getString(2));
  }
```

HIVE - User defined functions

Qu'est ce que les UDF ?

- Parfois les fonctions de Hive ne sont pas suffisantes pour répondre au besoin de l'utilisateur.
- Hive offre la possibilité à l'utilisateur de développer ses propres fonctions en Java.

Comment écrire des UDF pour Hive ?

- Hive fournit une API pour écrire des UDF.
- La classe Java doit étendre org.apache.hadoop.hive.ql.exec.UDF
- Le code de la fonction est contenu dans la méthode « evaluate ».

HIVE - User defined functions

Exemple:

Entête de l'UDF:

```
@Description(

name="reverse",

value="_FUNC_(string) - reverse the input string.
\n",

extended="select _FUNC_(string_a, string_b) FROM
src;"

)
```

Corps de l'UDF

```
public final class ReverseUdf extends UDF {

 public Text evaluate(final Text s){

 if(s==null) {return null;}

 String reverse = new
StringBuilder(s.toString()).reverse().toString();

                    return new Text(reverse);

         }
```

HIVE - User defined functions

Ajout de l'UDF :

- Ajouter le jar de l'Udf
- Créer la fonction associé

```
// Adding the UDF jar

Hive> ADD JAR target/jar-with-dependencies.jar

// The path to UDF class must be complete

Hive> CREATE TEMPORARY FUNCTION REVERSEUDF as
'com.trimane.hive.udf.ReverseUdf';

// Use

Hive> select REVERSEUDF(field) from table limit 10;
```

HIVE - User defined functions

- **Exemple UDF: Qualifier un commentaire**

On dispose des données de notation et commentaires sur le films du VOD, on souhaite qualifier les commentaires par un avis :

- ✓ Satisfait
- ✓ Insatisfait

Exemple d'analyse : analyser les mots clefs présents dans les commentaires.


Exercice