

Filière d'Ingénieur CI1 :Big Data & AI

Fondamentaux du Big Data

# **Analyse des Retards Aériens via le Big Data**

Encadré par : Professeur Faouzi Marzouki

Préparé par :  
YAHYA Zakariae  
KINDA Abdoul Latif  
MAHARAVO Tefinjanahary Anicet

## Table des matières

Dédicace

Remerciements

I. Introduction

II. Technologie utilisées :

1. Compréhension de l'infrastructure :

1.1 Utilisation de d'infrastructure de données

1.2 Ingestion et service des données

2. Hadoop et BigData :

2.1 Présentation de Hadoop

2.2 Fonctionnement de HDFS

3. Apache Pig:

3.1 fonctionnalités et composant d'Apache Pig

4. Apache Hive :

4.1 Présentation et utilisation

4.2 Composant de Hive

4.3 Exécution des requêtes HiveQL

5. Cloudera Quick-Start et Impala :

5.1 Présentation de Cloudera

5.2 Fonctionnement d'Impala avec CDH

III. Projet :

1. Jeux de données :

2. Mise en place de l'entrepôt de données :

3. Préparation des données :

3.1. Prétraitement des données :

4. Gestion des fichiers

5. Création de la table de données :

IV. Comparaison Hive et Impala :

1. Introduction à Impala :

2. Fonctionnement d'Impala :

3. Limitation d'Impala

4. Compression et Optimisation des requêtes

VI. Partitionnement et Clustering :

1. Fondamentaux du partitionnement :

2. Partitionnement statistique et dynamique :

3. Partitionnement dans Impala :

4. Calcule de partitionnement :

5. Clustering avec Hive:

Conclusion :

Annexes :

## Dédicace

Nous tenons à exprimer notre sincère reconnaissance en dédiant ce projet à nos proches, qu'il s'agisse de nos parents attentionnés, de nos amis loyaux ou de tous ceux qui nous ont toujours encouragés. Nous tenons également à remercier notre école, qui a été bien plus qu'un simple lieu d'apprentissage. Nous accordons une mention spéciale à toutes les personnes qui ont contribué de manière significative à la conception et à la réussite de ce projet. Leur engagement, leur passion et leur précieuse aide ont été déterminants pour notre succès, et nous leur sommes profondément reconnaissants pour leur soutien indéfectible. Ainsi, nous dédions ce projet à nos proches, à notre école et à toutes les personnes qui ont réellement participé à sa réalisation et à son succès. Leur influence positive sur notre parcours est inestimable, et nous leur sommes infiniment reconnaissants pour leur soutien constant.

## Remerciements

Avant tout, rendons grâce à ALLAH le Tout-Puissant, qui nous a donné la force et la sagesse nécessaires pour mener à bien ce projet de Gestion des Données de Retards Aériens : Approche Hadoop et Écosystème Apach. Avant d'entrer dans les détails techniques de cette application, il est crucial d'exprimer notre gratitude envers toutes les personnes qui nous ont guidés, encouragés et soutenus durant cette aventure.

Nous tenons à exprimer notre profonde reconnaissance à Monsieur, notre encadrant de projet, pour son accompagnement constant, ses précieux conseils, et sa patience inébranlable tout au long de semestre. Ses orientations et recommandations ont été des éléments clés dans l'avancement de ce projet.

Nous souhaitons également remercier nos collègues étudiants, dont la disponibilité et l'entraide ont joué un rôle essentiel dans le succès de ce projet. Leur générosité en termes de temps et de connaissances a été précieuse.

Enfin, nous souhaitons exprimer notre gratitude à tous ceux qui, de près ou de loin, ont contribué à ce projet. Leur soutien, leurs conseils et leur confiance ont été des piliers importants pour la réussite de cette étape. Nous remercions chaleureusement toutes les personnes qui nous ont aidés à mener à bien cette mission et qui ont enrichi notre expérience par leur engagement et leur bienveillance.

## **I- Introduction**

Il y a beaucoup d'excitation autour du terme Big Data. En termes simples, le Big Data peut être défini comme des données à grande échelle qui n'ont pas une structure bien définie. La taille des données est si énorme qu'il n'est pas pratiquement facile pour un seul ordinateur de stocker et de traiter toutes les données. Dans l'approche informatique traditionnelle, il y a de nombreux problèmes différents et l'objectif a toujours été d'augmenter la vitesse de traitement et la puissance de l'ordinateur. À mesure que les données croissent de manière exponentielle, la puissance de traitement d'un seul ordinateur devient un goulot d'étranglement et, par conséquent, une nouvelle approche est nécessaire pour résoudre le problème à portée de main. Une nouvelle méthode a été développée dans laquelle de nombreux ordinateurs peu coûteux travaillaient ensemble en harmonie pour stocker et traiter ces grandes données en parallèle. Cela nous permet d'extraire des informations significatives d'un grand ensemble de données. De plus, en utilisant la technologie cloud, il est facile de créer un cluster, de calculer et de libérer les ressources informatiques lorsqu'elles ne sont pas nécessaires. Ainsi, grâce à la technologie cloud, nous obtenons la puissance de calcul du cluster d'ordinateurs avec un investissement minimal

## **II- Technologies utilisées**

### **1. Compréhension de l'infrastructure :**

#### **1.1. Utilisation de d'infrastructure de données**

L'infrastructure est la pierre angulaire de l'architecture Big Data. Posséder les bons outils pour stocker, traiter et analyser vos données est crucial dans tout projet Big Data.

#### **1.2. Ingestion et service des données**

Deux principaux utilisateurs de l'infrastructure de données : Humain et Système. Les façons dont les données peuvent être ingérées/servies dans l'infrastructure de données : Couche d'ingestion

### **2. Hadoop et Big Data :**

#### **2.1 Présentation de Hadoop**

Hadoop est l'un des outils conçus pour gérer le Big Data. Hadoop et d'autres produits logiciels travaillent à interpréter ou analyser les résultats des recherches de Big Data à travers des algorithmes et méthodes propriétaires spécifiques. Hadoop est un programme open source sous la licence Apache qui est maintenu par une communauté mondiale d'utilisateurs. Apache Hadoop est 100% open source et a été pionnier d'une nouvelle manière fondamentale de stocker et de traiter les données, au lieu de s'appuyer sur du matériel coûteux et propriétaire ainsi que sur différents systèmes pour stocker et traiter les données.

#### **2.2 Fonctionnement de HDFS**

HDFS : HDFS est le principal stockage distribué sur Hadoop pour gérer des pools de Big Data qui s'étendent sur de grands clusters de serveurs peu coûteux. HDFS est considéré comme le réservoir de l'écosystème Hadoop, où les données sont déposées et restent jusqu'à ce que l'utilisateur veuille les exporter vers un autre outil pour analyser les données stockées. Toute machine qui supporte le langage de programmation Java peut exécuter HDFS.

Fonctionnement de HDFS : HDFS utilise une architecture maître-esclave où chaque cluster a un NameNode pour gérer les opérations du système de fichiers et des DataNodes de support pour gérer le stockage des données sur des nœuds informatiques individuels (généralement, il existe un DataNode par nœud dans le cluster Hadoop). HDFS stocke les données dans des

fichiers qui sont divisés en un ou plusieurs segments et sont stockés dans des DataNodes particuliers. Ces petits segments de fichiers sont appelés blocs et la quantité minimale de données qui peut être lue ou écrite est appelée un bloc unique. Par défaut, la taille d'un bloc est de 64 Mo, ce qui peut être modifié dans le fichier de configuration.



### **3. Apache Pig**

Apache Pig est un langage de flux de données procédural de haut niveau basé sur Hadoop pour traiter et analyser de grandes quantités de données sans avoir à écrire du code MapReduce en Java. Apache Pig possède des fonctionnalités similaires à celles des SGBDR, telles que les jointures, les clauses distinctes, l'union, etc. pour traiter de gros fichiers contenant des données semi-structurées ou non structurées.

Les composants d'Apache Pig sont :

**Pig Latin** : un langage de flux de données similaire au SQL pour joindre, grouper et agréger des ensembles de données distribués avec facilité.

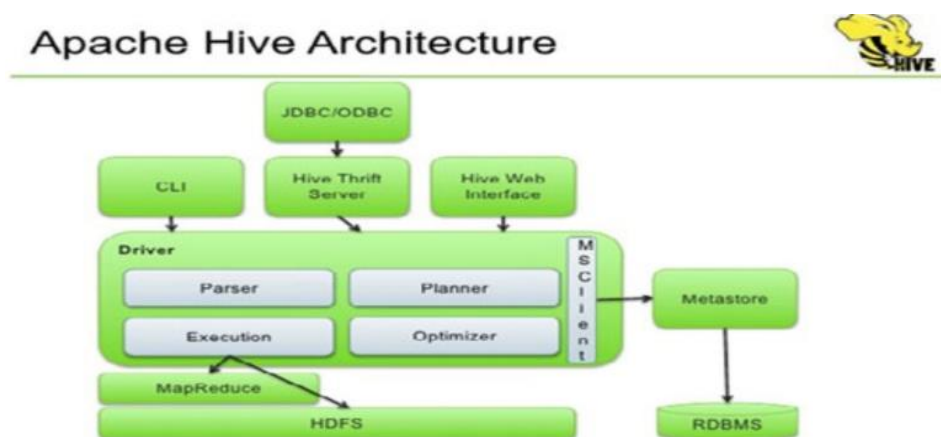
**Pig Engine** : le moteur Pig prend les scripts Pig Latin écrits par les utilisateurs, les analyse, les optimise et les exécute ensuite sous forme de séries de tâches MapReduce sur un cluster Hadoop.

### **4. Apache Hive : Une solution Data Warehousing pour le Big Data sur Hadoop**

#### **4.1. Présentation et utilisation :**

Hive est une solution Data Warehousing développée sur Hadoop pour répondre aux défis du Big Data liés au stockage, à la gestion et au traitement de grands ensembles de données sans avoir à écrire des programmes complexes basés sur MapReduce en Java. Hive est un modèle de programmation familier pour les professionnels du Big Data qui connaissent SQL mais qui ne maîtrisent pas bien la programmation. Hive n'est pas une base de données relationnelle ni une architecture pour le traitement des transactions en ligne (OLTP). Il est particulièrement conçu pour les systèmes de traitement analytique en ligne (OLAP). Le compilateur Hive convertit les requêtes écrites en HiveQL en tâches MapReduce afin que les développeurs Hadoop n'aient pas à se soucier du code de programmation complexe au-delà du traitement et puissent se concentrer sur le problème commercial. Les trois fonctions importantes effectuées par Hive incluent : la synthèse de données, l'interrogation de données et l'analyse de données. Apache Hive est largement utilisé par les data scientists et les analystes de données pour l'exploration de données, la création de pipelines de données et le traitement des requêtes ad-hoc.

## 4.2. Composants de Hive :



=>. CLI, JDBC, ODBC ou toute autre interface Web GUI constituent les interfaces externes du cadre Hive pour créer une interaction entre l'utilisateur et HDFS.

=>. Metastore Thrift API suit les données stockées dans les différentes parties de HDFS. C'est comme un catalogue de système.

=>. Driver est le cœur de l'architecture Hive, responsable de la compilation, de l'optimisation et de l'exécution des instructions HiveQL.

=>. Thrift Server est une API côté client pour exécuter des instructions HiveQL.

## 4.3. Exécution des requêtes HiveQL

Comment une requête HiveQL est-elle exécutée dans Apache Hive ? : Lorsqu'un utilisateur soumet une requête HiveQL, elle est d'abord compilée. La requête compilée est ensuite exécutée par un moteur d'exécution tel que Hadoop MapReduce ou Apache Tez. Les données dans divers formats comme ORC, AVRO, Parquet ou Text résident dans HDFS sur lesquelles la requête doit être exécutée. YARN alloue ensuite les ressources nécessaires à travers le cluster Hadoop pour l'exécution. Les résultats de l'exécution de la requête sont envoyés via une connexion JDBC ou ODBC.

## 5. Cloudera Quick-Start et Impala

### 5.1. Présentation de Cloudera

Cloudera offre une plateforme évolutive, flexible et intégrée qui facilite la gestion des volumes et des variétés de données en augmentation rapide dans votre entreprise. Les produits et solutions Cloudera vous permettent de déployer et de gérer Apache Hadoop et les projets connexes, de manipuler et d'analyser vos données, et de garantir la sécurité et la protection de

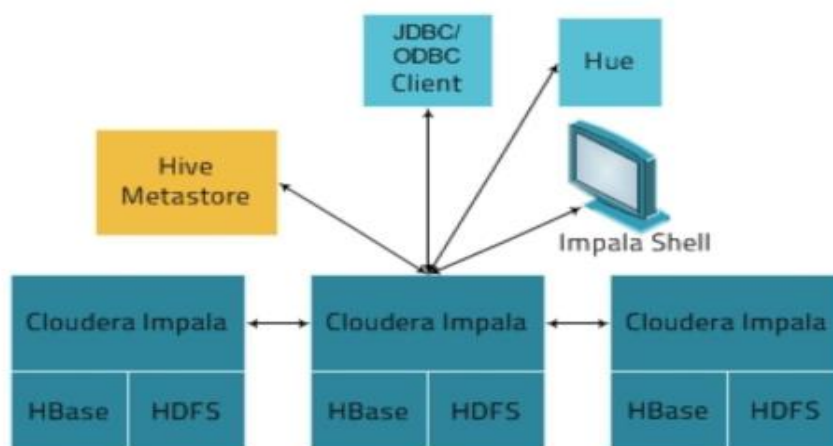
ces données. Cloudera propose de nombreux produits et outils. Nous utilisons CDH et Apache Impala.

CDH : La distribution la plus complète, testée et populaire d'Apache Hadoop et d'autres projets open-source connexes, y compris Apache Impala et Cloudera Search. CDH offre également flexibilité, sécurité et intégration avec de nombreuses solutions matérielles et logicielles.

Apache Impala : Un moteur SQL de traitement massivement parallèle pour les analyses interactives et l'intelligence d'affaires. Son architecture hautement optimisée le rend idéalement adapté aux requêtes de style BI traditionnel avec jointures, agrégations et sous-requêtes. Il peut interroger les fichiers de données Hadoop provenant de diverses sources, y compris ceux produits par les tâches MapReduce chargées dans les tables Hive. Le composant de gestion des ressources YARN permet à Impala de coexister sur des clusters exécutant des charges de travail par lots simultanément avec les requêtes SQL d'Impala. Vous pouvez gérer Impala aux côtés d'autres composants Hadoop via l'interface utilisateur de Cloudera Manager et sécuriser ses données grâce au cadre d'autorisation Sentry.

## **5.2.Fonctionnement d'Impala avec CDH :**

Le graphique suivant illustre comment Impala est positionné dans l'environnement plus large de Cloudera :



### III- PRATIQUE

#### 1. Jeux de données

La ressource web de benchmarking est disponible à l'adresse suivante :

[Data Expo 2009: Airline on time data - ASA Statistical Computing Dataverse \(harvard.edu\)](https://dataexpo2009.asa.harvard.edu/).

Le jeu de données est librement téléchargeable à partir de ce site web. Selon le site, les données comprennent les détails des arrivées et des départs des vols commerciaux aux États-Unis, d'octobre 1987 à avril 2008. Cependant, nous utilisons les données de 2003 à 2008 pour notre projet, et tous les fichiers CSV pèsent entre 100 Mo et 125 Mo. Le jeu de données utilisé principalement est "flights.csv" qui contient un total de 29 variables avec une dimension de 31 colonnes et 5 819 079 lignes. Nous avons tiré des conclusions en nous concentrant principalement sur les variables du retard à l'arrivée et du retard au départ. Lorsqu'il y avait un retard à l'arrivée des vols, il y avait également un retard au départ pour la plupart des vols, sauf pour quelques-uns.

#### 2. Mise en place de l'entrepôt de données :

Définir toutes les données sur Quickstart Hue comme indiqué dans la figure ci-dessous.

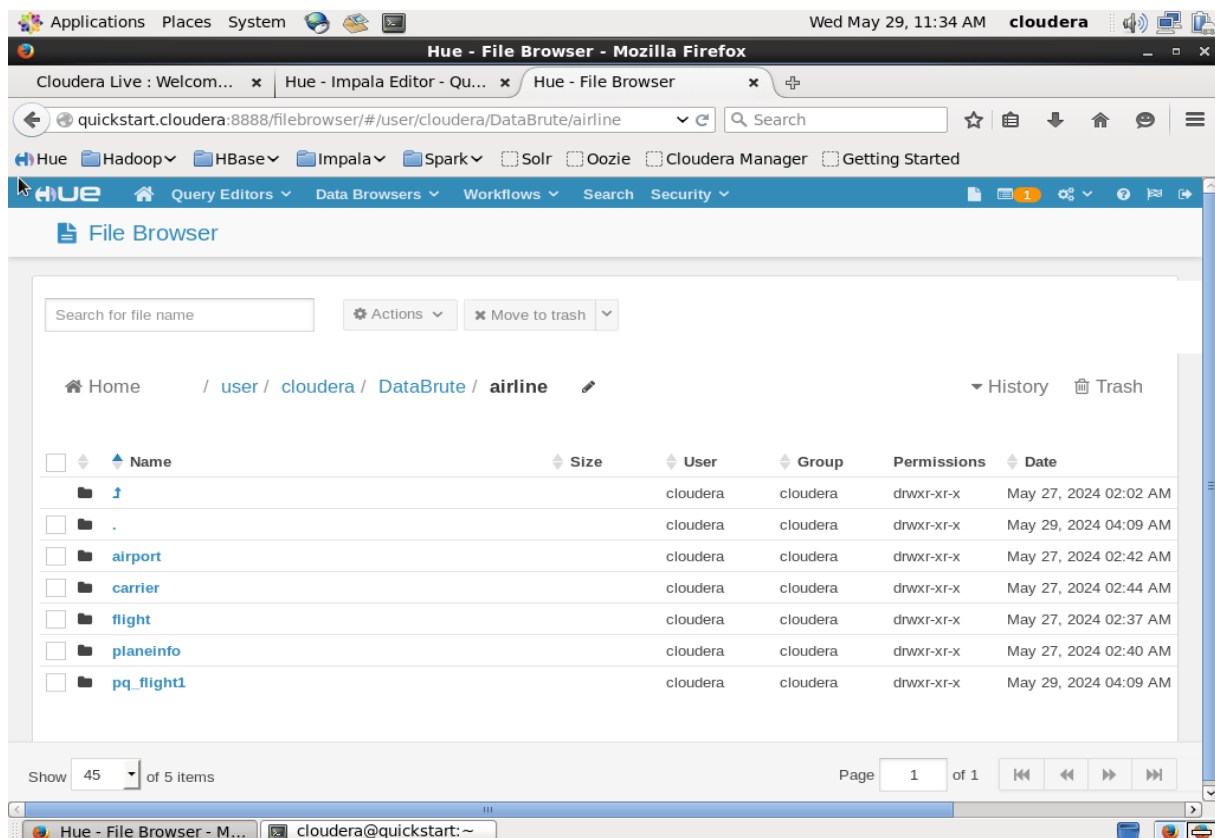


fig: Configuration des données sur Cloudera

### **3. Préparation des données**

La préparation des données est essentielle pour le succès du Big Data. L'un des principaux obstacles au succès du Big Data est l'absence d'une stratégie de préparation des données. La préparation des données comprend toutes les étapes nécessaires pour acquérir, préparer, rendre précises et gérer les actifs de données de l'organisation. Voici quelques étapes les plus importantes pour le projet :

#### **Prétraitement des données, extraction et chargement :**

Objectif : En général, le but du prétraitement des données est de donner une structure aux données non structurées, d'intégrer les données avec des valeurs provenant de systèmes externes et de prétraiter le contenu binaire ; écrire directement dans l'entrepôt de données.

- Charger les données brutes à partir du fichier spécifié en utilisant PigStorage avec une virgule comme séparateur
- Filtrer les données pour exclure les en-têtes (lignes où l'année est 'Year')
- Générer les données filtrées avec des valeurs par défaut pour les champs manquants ('NA')

Le stockage des données est moins problématique que la récupération efficace des données.

À cette étape, nous téléchargeons tous les jeux de données requis sur la machine locale et extrayons tous les jeux de données.

Pig ne peut pas écrire dans une table HCatalog au format Parquet, mais Spark peut écrire dans des tables Hive et aussi écrire des fichiers Parquet dans des répertoires HDFS.

### **4. Gestion des fichiers sur HDFS**

Les requêtes pour cela sont données dans le fichier code.

```

Applications Places System Mon May 27, 6:03 AM cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help
2024-05-27 05:37:14,430 [main] INFO org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalAp
plicationStatus=SUCCESS. Redirecting to job history server
2024-05-27 05:37:35,620 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.reduce.tasks is deprecated. In
stead, use mapreduce.job.reduces
2024-05-27 05:37:36,264 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 100% co
mplete
2024-05-27 05:37:36,360 [main] INFO org.apache.pig.tools.pigstats.SimplePigStats - Script Statistics:

HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.4.2 0.12.0-cdh5.4.2 cloudera 2024-05-27 04:09:23 2024-05-27 05:37:36 FILTER

Success!

Job Stats (time in seconds):
JobId Maps Reduces MaxMapTime MinMapTime AvgMapTime MedianMapTime MaxReduceTime MinReduceTime AvgRe
duceTime MedianReduceTime Alias Feature Outputs n/a n/a n/a n/a Data,headless_data,re
al_data MAP_ONLY /user/cloudera/out/airline/flight,

Input(s):
Successfully read 37742766 records (3576751062 bytes) from: "/user/cloudera/DataBrute/airline/flight"

Output(s):
Successfully stored 37742760 records (3884235043 bytes) in: "/user/cloudera/out/airline/flight"

Counters:
Total records written : 37742760
Total bytes written : 3884235043
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1716777574023_0001

2024-05-27 05:37:37,507 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success
!
grunt>
cloudera@quickstart:~

```

```

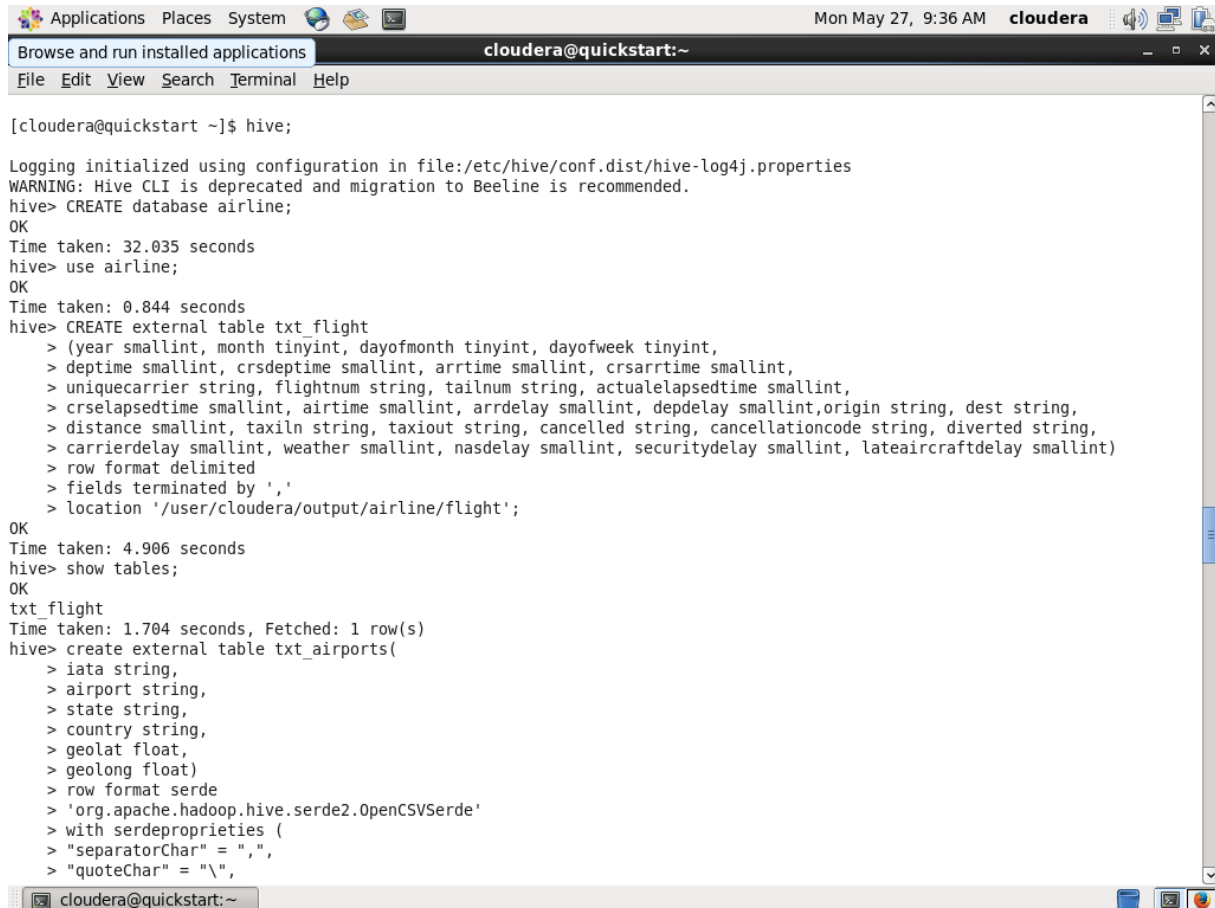
Applications Places System Mon May 27, 8:03 AM cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help
Found 28 items
-rw-r--r-- 1 cloudera cloudera 0 2024-05-27 05:37 /user/cloudera/out/airline/flight/_SUCCESS
-rw-r--r-- 1 cloudera cloudera 136633063 2024-05-27 04:30 /user/cloudera/out/airline/flight/part-m-00000
-rw-r--r-- 1 cloudera cloudera 225536771 2024-05-27 04:29 /user/cloudera/out/airline/flight/part-m-00001
-rw-r--r-- 1 cloudera cloudera 224146024 2024-05-27 04:28 /user/cloudera/out/airline/flight/part-m-00002
-rw-r--r-- 1 cloudera cloudera 135023375 2024-05-27 04:29 /user/cloudera/out/airline/flight/part-m-00003
-rw-r--r-- 1 cloudera cloudera 135060052 2024-05-27 04:30 /user/cloudera/out/airline/flight/part-m-00004
-rw-r--r-- 1 cloudera cloudera 135557565 2024-05-27 04:29 /user/cloudera/out/airline/flight/part-m-00005
-rw-r--r-- 1 cloudera cloudera 135054587 2024-05-27 04:47 /user/cloudera/out/airline/flight/part-m-00006
-rw-r--r-- 1 cloudera cloudera 135285024 2024-05-27 04:46 /user/cloudera/out/airline/flight/part-m-00007
-rw-r--r-- 1 cloudera cloudera 135551499 2024-05-27 04:47 /user/cloudera/out/airline/flight/part-m-00008
-rw-r--r-- 1 cloudera cloudera 135953258 2024-05-27 04:46 /user/cloudera/out/airline/flight/part-m-00009
-rw-r--r-- 1 cloudera cloudera 135047171 2024-05-27 04:47 /user/cloudera/out/airline/flight/part-m-00010
-rw-r--r-- 1 cloudera cloudera 135420869 2024-05-27 04:47 /user/cloudera/out/airline/flight/part-m-00011
-rw-r--r-- 1 cloudera cloudera 135517107 2024-05-27 05:04 /user/cloudera/out/airline/flight/part-m-00012
-rw-r--r-- 1 cloudera cloudera 135374073 2024-05-27 05:04 /user/cloudera/out/airline/flight/part-m-00013
-rw-r--r-- 1 cloudera cloudera 135002675 2024-05-27 05:04 /user/cloudera/out/airline/flight/part-m-00014
-rw-r--r-- 1 cloudera cloudera 135337248 2024-05-27 05:03 /user/cloudera/out/airline/flight/part-m-00015
-rw-r--r-- 1 cloudera cloudera 135343779 2024-05-27 05:04 /user/cloudera/out/airline/flight/part-m-00016
-rw-r--r-- 1 cloudera cloudera 136276387 2024-05-27 05:04 /user/cloudera/out/airline/flight/part-m-00017
-rw-r--r-- 1 cloudera cloudera 135429688 2024-05-27 05:21 /user/cloudera/out/airline/flight/part-m-00018
-rw-r--r-- 1 cloudera cloudera 135727764 2024-05-27 05:22 /user/cloudera/out/airline/flight/part-m-00019
-rw-r--r-- 1 cloudera cloudera 135181518 2024-05-27 05:21 /user/cloudera/out/airline/flight/part-m-00020
-rw-r--r-- 1 cloudera cloudera 135268554 2024-05-27 05:21 /user/cloudera/out/airline/flight/part-m-00021
-rw-r--r-- 1 cloudera cloudera 190494853 2024-05-27 05:20 /user/cloudera/out/airline/flight/part-m-00022
-rw-r--r-- 1 cloudera cloudera 135140088 2024-05-27 05:21 /user/cloudera/out/airline/flight/part-m-00023
-rw-r--r-- 1 cloudera cloudera 134223236 2024-05-27 05:36 /user/cloudera/out/airline/flight/part-m-00024
-rw-r--r-- 1 cloudera cloudera 174981839 2024-05-27 05:37 /user/cloudera/out/airline/flight/part-m-00025
-rw-r--r-- 1 cloudera cloudera 90666976 2024-05-27 05:35 /user/cloudera/out/airline/flight/part-m-00026
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera/out/airline/flight/part-m-00000
-rw-r--r-- 1 cloudera cloudera 136633063 2024-05-27 04:30 /user/cloudera/out/airline/flight/part-m-00000
[cloudera@quickstart ~]$ hdfs dfs -cat /user/cloudera/out/airline/flight/part-m-00000 | head -5
2006,10,30,1,655,655,754,750,MQ,3740,N620AE,59,55,38,4,0,TKX,DFW,181,14,7,0,,0,0,0,0,0
2006,10,31,2,652,655,740,750,MQ,3740,N620AE,48,55,35,-10,-3,TKX,DFW,181,6,7,0,,0,0,0,0,0
2006,10,1,7,1958,1945,2101,2040,MQ,3741,N639AE,63,55,34,21,13,DFW,TKX,181,5,24,0,,0,13,0,8,0,0
2006,10,2,1,1946,1945,2041,2040,MQ,3741,N650AE,55,55,36,1,1,DFW,TKX,181,6,13,0,,0,0,0,0,0
2006,10,3,2,1940,1945,2030,2040,MQ,3741,N697AB,50,55,30,-10,-5,DFW,TKX,181,6,14,0,,0,0,0,0,0
cat: Unable to write to output stream.
[cloudera@quickstart ~]$
cloudera@quickstart:~

```

figure: Gestion des fichiers sur Hadoop

## 5. Création de la table de données :

Pour analyser tous les jeux de données, nous créons un format de table pour les jeux de données des aéroports, des transporteurs et des données des avions, et analysons différentes requêtes. Les requêtes correspondantes sont données dans la Figure ci dessous et un exemple de sortie pour l'année 2006 est donné dans la Figure qui le suivie.



```

[cloudera@quickstart ~]$ hive;

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> CREATE database airline;
OK
Time taken: 32.035 seconds
hive> use airline;
OK
Time taken: 0.844 seconds
hive> CREATE external table txt_flight
> (year smallint, month tinyint, dayofmonth tinyint, dayofweek tinyint,
> deptime smallint, crsdeptime smallint, arrtime smallint, crsarrrtime smallint,
> uniquecarrier string, flightnum string, tailnum string, actualelapsedtime smallint,
> crselapsedtime smallint, airtime smallint, arrdelay smallint, depdelay smallint, origin string, dest string,
> distance smallint, taxiin string, taxiout string, cancelled string, cancellationcode string, diverted string,
> carrierdelay smallint, weather smallint, nasdelay smallint, securitydelay smallint, lateaircraftdelay smallint)
> row format delimited
> fields terminated by ','
> location '/user/cloudera/output/airline/flight';
OK
Time taken: 4.906 seconds
hive> show tables;
OK
txt_flight
Time taken: 1.704 seconds, Fetched: 1 row(s)
hive> create external table txt_airports(
> iata string,
> airport string,
> state string,
> country string,
> geolat float,
> geolong float)
> row format serde
> 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
> with serdeproperties (
> "separatorChar" = ",",
> "quoteChar" = "\"",

```

fig: Création d'un tableau



```

cloudera@quickstart:~
File Edit View Search Terminal Help
hive> select * from txt_carriers limit 5;
OK
Code      Description
020       Titan Airways
040       Tradewind Aviation
050       Comlux Aviation, AG
060       Master Top Linhas Aereas Ltd.
Time taken: 0.409 seconds, Fetched: 5 row(s)
hive> select * from txt_airports where iata<>'iata' limit 6;
OK
00M       Thigpen      Bay Springs  MS      USA      31.95376472  -89.23450472
00R       Livingston Municipal  Livingston  TX      USA      30.68586111  -95.01792778
00V       Meadow Lake  Colorado Springs  CO      USA      38.94574889  -104.5698933
01G       Perry-Warsaw  Perry  NY      USA      42.74134667  -78.05208056
01J       Hilliard Airpark  Hilliard  FL      USA      30.6880125   -81.90594389
01M       Tishomingo County  Belmont  MS      USA      34.49166667  -88.20111111
Time taken: 0.377 seconds, Fetched: 6 row(s)
hive> select * from txt_flight limit 10;
OK
2006      10      30      1      655      655      754      750      MQ      3740      N620AE  59      55      38      4      0      T
XK      DFW      181      14      7      0      754      750      MQ      3740      N620AE  48      55      35      -10      -3      T
2006      10      31      2      652      655      740      750      MQ      3740      N620AE  55      55      36      1      1      D
XK      DFW      181      6      7      0      2030      2040      MQ      3741      N639AE  63      55      34      21      13      D
FW      TXK      181      5      24      0      2041      2040      MQ      3741      N650AE  55      55      36      1      1      D
2006      10      2      1      1946      1945      2041      2040      MQ      3741      N650AE  55      55      36      1      1      D
FW      TXK      181      6      13      0      2030      2040      MQ      3741      N697AB  50      55      30      -10      -5      D
2006      10      3      2      1940      1945      2030      2040      MQ      3741      N697AB  50      55      30      -10      -5      D
FW      TXK      181      6      14      0      2044      2040      MQ      3741      N671AE  48      55      33      4      11      D
2006      10      4      3      1956      1945      2044      2040      MQ      3741      N671AE  48      55      33      4      11      D
FW      TXK      181      2      13      0      2028      2040      MQ      3741      N632AE  48      55      31      -12      -5      D
2006      10      5      4      1940      1945      2028      2040      MQ      3741      N632AE  48      55      31      -12      -5      D
FW      TXK      181      3      14      0      2045      2040      MQ      3741      N663AR  51      55      33      5      9      D
2006      10      6      5      1954      1945      2045      2040      MQ      3741      N663AR  51      55      33      5      9      D
FW      TXK      181      3      15      0      2031      2040      MQ      3741      N633AE  54      55      35      -9      -8      D
2006      10      7      6      1937      1945      2031      2040      MQ      3741      N633AE  54      55      35      -9      -8      D
FW      TXK      181      3      16      0      2032      2040      MQ      3741      N931AE  52      55      34      -8      -5      D
2006      10      8      7      1940      1945      2032      2040      MQ      3741      N931AE  52      55      34      -8      -5      D
FW      TXK      181      2      16      0      2040      2040      MQ      3741      N931AE  52      55      34      -8      -5      D

```

fig : Exemple de txt\_airport et output de vol de l'année 2006

## IV. Comparaison Hive et Impala

### 1. Introduction à Impala

Le serveur Impala est un moteur d'exécution de requêtes SQL pour Hadoop. Certaines des caractéristiques de l'architecture d'Impala sont : un moteur de traitement massivement parallèle (MPP) pour un environnement de clustering distribué. Il est open source et utilise les données sur HDFS. Il se compose de divers processus daemon qui s'exécutent sur des hôtes spécifiques au sein de votre cluster Hadoop. Les trois principaux composants d'Impala sont le daemon Impala, le statestore Impala et le service de catalogue Impala, représentés respectivement par les daemons impalad, statestored et catalog. Cependant, le composant central d'Impala est le processus daemon s'exécutant sur chaque nœud du cluster Impala.

### 2. Fonctionnement d'Impala :

Le processus impalad lit et écrit dans les fichiers de données.

Il divise logiquement une requête en requêtes parallèles plus petites et les distribue à



différents nœuds dans le cluster Impala. Lorsque vous soumettez une requête au daemon Impala s'exécutant sur n'importe quel nœud, ce nœud sert de nœud coordinateur pour cette requête.

Impala transmet les résultats intermédiaires de la requête au nœud coordinateur central. Le coordinateur construit le résultat final de la requête. Lorsque vous exécutez une expérience en utilisant la commande Impala-shell, il peut vous connecter au même processus daemon Impala pour plus de commodité.

### 3. Limitation d'impala

Lorsque nous soumettons une requête Hive, elle mappe le contexte à une tâche MapReduce qui se lance ensuite dans le gestionnaire de ressources. Le gestionnaire de ressources travaille sur la localisation des données. Le gestionnaire de ressources assigne un gestionnaire de nœud pour effectuer une tâche. Dans Impala, n'importe quel gestionnaire de ressources ou de nœuds peut être connecté à n'importe quel daemon Impala, à condition qu'un daemon soit disponible pour lui. Cependant, il a échoué car il ne connaît pas les serdes (serializers/deserializers) dans n'importe quel objet de base de données, qui peuvent être vus dans Hive avec des bibliothèques externes, car ce sont des bibliothèques très centrées sur Hive qui ne peuvent pas être utilisées dans Impala, comme illustré dans la Figure ci dessous.

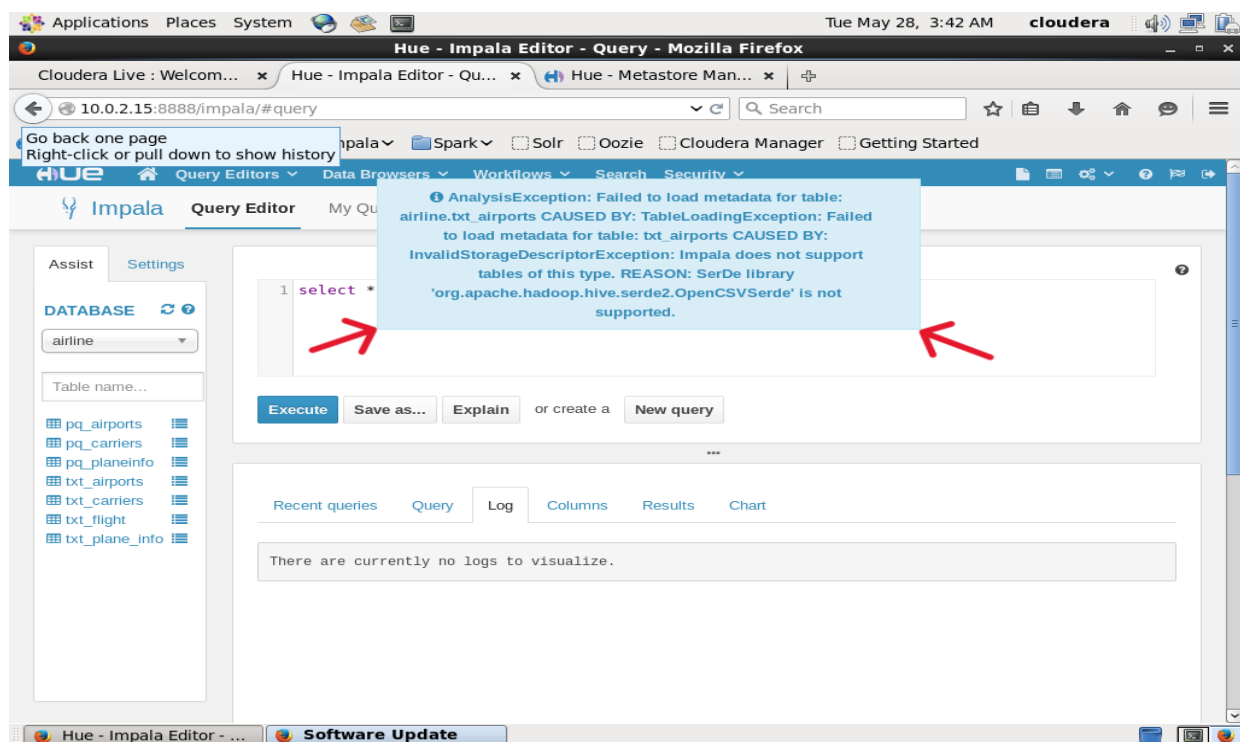


fig:Erreur que nous obtenons

Une UDF personnalisée est créée dans Hive. Impala prend en charge les formats de fichiers Text, RC, Sequence, Avro et Parquet. Nous utilisons principalement des fichiers Parquet. Impala vous aide à créer, gérer et interroger des tables du Parquet. Parquet est un format de fichier binaire orienté colonnes conçu pour être très efficace pour les types de requêtes à grande échelle pour lesquelles Impala excelle. Parquet est bon pour effectuer des opérations d'agrégation telles que SUM() et AVG() qui nécessitent de traiter la plupart ou la totalité des valeurs d'une colonne. Il y aura une amélioration considérable dans les requêtes lorsque nous utiliserons le Parquet.

#### 4. Compression et Optimisation des requêtes

Nous obtenons une énorme différence entre Impala et Hive. Hive prend environ 3 minutes pour s'exécuter tandis qu'Impala prend environ 19 secondes. Impala donne une réponse plus rapide. Ce temps peut même être réduit en utilisant la compression et l'optimisation des requêtes. Il est facile de requêter la version Parquet que la version SerDes. Le principal objectif est de rendre les données disponibles, accessibles et efficaces de manière conviviale. De plus, le deuxième passage de Parquet est inférieur au deuxième passage du fichier texte. Compression des données Parquet pour les fichiers texte et Parquet : Avec Parquet, il y a une réduction de 17 fois la vitesse. Toutes les requêtes Impala avec une sortie sont présentées dans la figure ci-dessous.

```

Applications Places System cloudera@quickstart:~ Wed May 29, 2:28 AM cloudera
File Edit View Search Terminal Help

[quickstart.cloudera:21000] > use airline;
Query: use airline
[quickstart.cloudera:21000] > show tables;
Query: show tables
+-----+
| name |
+-----+
| pq_airports |
| pq_carriers |
| pq_flight |
| pq_planeinfo |
| txt_airports |
| txt_carriers |
| txt_flight |
| txt_flight1 |
| txt_plane_info |
+-----+
Fetched 9 row(s) in 0.08s
[quickstart.cloudera:21000] > select * from txt_flight1 limit 1;
Query: select * from txt_flight1 limit 1
+-----+
| year | month | dayofmonth | dayofweek | deptime | crsdeptime | arrtime | crsarrrtime | uniquecarrier | flightnum | tailnum |
| actualelapsed | crselapsed | airtime | arrdelay | depdelay | origin | dest | distance | taxiin | taxiout | cancelled |
| cancellationcode | diverted | carrierdelay | weatherdelay | nasdelay | securitydelay | lateaircraftdelay |
+-----+
| 2004 | 10 | 31 | 7 | 1250 | 1250 | 1345 | 1355 | MQ | 3099 | N394AE |
| 55 | 65 | 33 | -10 | 0 | 0 | LAX | PSP | 110 | 3 | 19 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
+-----+
Fetched 1 row(s) in 10.64s
[quickstart.cloudera:21000] > select year , count(1) no_of_flights from txt_flight1 group by year;

```

fig :Affichage du tableau utilisant Impala et output txt\_flight

The screenshot shows a terminal window titled "cloudera@quickstart:~". The window contains the following text:

```

+-----+
Fetchd 1 row(s) in 10.64s
[quickstart.cloudera:21000] > select year , count(1) no_of_flights from txt_flight1 group by year;
Query: select year , count(1) no_of_flights from txt_flight1 group by year
+-----+
| year | no_of_flights |
+-----+
| 2008 | 2389217       |
| 2007 | 7453215       |
| 2005 | 7140596       |
| 2003 | 6488540       |
| 2006 | 7141922       |
| 2004 | 7129270       |
+-----+
Fetchd 6 row(s) in 274.64s
[quickstart.cloudera:21000] > select year , count(1) no_of_flights from txt_flight1 group by year;
Query: select year , count(1) no_of_flights from txt_flight1 group by year
+-----+
| year | no_of_flights |
+-----+
| 2008 | 2389217       |
| 2007 | 7453215       |
| 2005 | 7140596       |
| 2003 | 6488540       |
| 2006 | 7141922       |
| 2004 | 7129270       |
+-----+
Fetchd 6 row(s) in 233.50s

```

fig : Output de group by txt\_flight texte première et deuxième fois

L'Impala met environ 274 et 233 seconde pour exécution du fichier txt

Exécution de la requête avec la version impala, exemple créé de fichiers Parquet et avec des fichiers Parquet le temps nécessaire est suffisamment inférieur.

The screenshot shows a terminal window titled 'cloudera@quickstart:~'. The terminal output is as follows:

```

+-----+
| name |
+-----+
| pq_airports |
| pq_carriers |
| pq_flight |
| pq_planeinfo |
| txt_airports |
| txt_carriers |
| txt_flight |
| txt_plane_info |
+-----+

```

Fetches 8 rows in 0.19s.

```

[quickstart.cloudera:21000] > select year, count(1) no_of_flight from pq_flight group by year;
Query: select year, count(1) no_of_flight from pq_flight group by year
+-----+
| year | no_of_flight |
+-----+
| 2008 | 2389217 |
| 2007 | 7453215 |
| 2005 | 7140596 |
| 2003 | 6488540 |
| 2006 | 7141922 |
| 2004 | 7129270 |
+-----+

```

Fetches 6 rows in 55.19s.

```

[quickstart.cloudera:21000] > select year, count(1) no_of_flight from pq_planeinfo group by year;
Query: select year, count(1) no_of_flight from pq_planeinfo group by year
+-----+
| year | no_of_flight |
+-----+
| 2008 | 2389217 |
| 2007 | 7453215 |
| 2005 | 7140596 |
| 2003 | 6488540 |
| 2006 | 7141922 |
| 2004 | 7129270 |
+-----+

```

Fetches 6 rows in 41.29s.

```

[quickstart.cloudera:21000] >

```

fig : Output de group by pq\_flight parquet premier et deuxième fois

La deuxième passe de parquet est inférieure à la deuxième passe du fichier texte. Comme si ça devenait 41.29 seconde et sa réduction de 6 fois.

Compression de données Parquet pour les fichiers texte et parquet : Avec Parquet, c'est 6 fois moins de vitesse.

## **VI. Partitionnement et regroupement(clustering) dans Hive/Impala**

### **1. Fondamentaux du partitionnement :**

Objectif : Obtenir une répartition efficace des données afin que les données ne soient pas biaisées par le hasard. Par défaut, tous les fichiers de données d'une table sont situés dans un seul répertoire. Le partitionnement est une technique permettant de diviser physiquement les données lors du chargement, en fonction des valeurs d'une ou plusieurs colonnes, pour accélérer les requêtes qui testent ces colonnes. Hive est utilisé pour le partitionnement.

### **2. Partitionnement Statistique et Dynamique**

Deux méthodes de partitionnement, dynamique et statique, sont toutes deux mises en œuvre sur l'ensemble de données pour connaître la différence. Nous utilisons le partitionnement statique car nous obtenons des données année par année. Le partitionnement dynamique est également présenté pour l'ensemble de données, car avec le partitionnement dynamique, nous pouvons ajouter un nombre quelconque de partitions avec une seule exécution SQL.

Partitionnement statique : Utilisé lorsque l'utilisateur souhaite spécifier le nom de la partition lors de l'insertion de données dans la table et que les valeurs distinctes dans la colonne partitionnée sont très peu nombreuses, et plus lorsque les données sont chargées de manière incrémentielle, partitionnées sur une période spécifique. Chargement : une portion est chargée à la fois ; Bon pour une opération continue ; Non adapté aux chargements initiaux.

Partitionnement dynamique : Utilisé lors du chargement massif de données dans une table et que vous souhaitez déduire automatiquement les valeurs de colonne, et utilisé lorsque les valeurs distinctes dans la colonne partitionnée sont très élevées. Chargement : Les données sont réparties entre les partitions de manière dynamique.

Où effectuer le traitement des données : Hive ou Impala ? Normalement, nous pouvons utiliser l'un ou l'autre, Hive ou Impala. Mais nous constatons qu'Impala n'a pas autant d'intégration avec l'écosystème Hadoop que Hive.

### 3. Partitionnement Impala :

Analyse : Impala est plus rapide qu'Apache Hive, mais cela ne signifie pas que c'est la solution SQL unique pour tous les problèmes de Big Data. Impala est intensif en mémoire et a une faible latence, mais il ne fonctionne pas efficacement pour les opérations de données lourdes comme les jointures, car il n'est pas possible de tout stocker en mémoire. C'est là que Hive intervient pour sauver la situation. Si une application a des besoins de traitement par lots sur de gros volumes de données, les organisations doivent opter pour Hive. Si elles ont besoin d'un traitement en temps réel des requêtes ad-hoc sur un sous-ensemble de données, alors Impala est un meilleur choix. Impala prend en charge le système de fichiers distribué Hadoop (HDFS) et Apache HBase.

Les caractéristiques suivantes sont différentes pour Hive et Impala :



### 4. Calcul de partitionnement :

Pour le partitionnement, une utilisation efficace des ressources du nœud de nom est nécessaire. Ainsi, dans le cluster, ce fichier sélectionné doit être répliqué trois fois, comme indiqué dans l'image et la taille de chaque fichier.

<input type="checkbox"/>	Name	Size	User	Group	Perm
<input type="checkbox"/>			cloudera	supergroup	drwxr
<input type="checkbox"/>			impala	supergroup	drwxr
<input type="checkbox"/>	<code>_impala_insert_staging</code>		impala	supergroup	drwxr
<input type="checkbox"/>	<code>df4c5b98bd9e30be-815826c26922d588_776687520_data.0.parq</code>	252.7 MB	impala	supergroup	-rw-r--r--
<input type="checkbox"/>	<code>df4c5b98bd9e30be-815826c26922d588_776687520_data.1.parq</code>	252.7 MB	impala	supergroup	-rw-r--r--

fig : Impala

Les fichiers impala sont très volumineux et doivent être répliqués. Chacun de ces fichiers fait environ 252,7 Mo, donc au total, cela représente environ 760 Mo. Maintenant, à partir de la figure ci-dessous, on peut remarquer que le total des vols chaque année est de 7 millions, et si nous divisons par 12 pour calculer par mois, nous obtenons finalement une donnée d'environ 800 Mo, ce qui est trop volumineux, donc nous ne pourrions pas exécuter la requête. Nous supposons donc que l'hypothèse pour chaque bloc pour chaque mois n'est pas bonne, et donc Hive est un meilleur choix pour le projet. Hive n'utilise pas plus d'espace, comme le montre la Figure ci-dessous montre.

De plus, pour réduire ces trois fichiers en un seul, la commande est "set mapred.reduce.tasks=1".

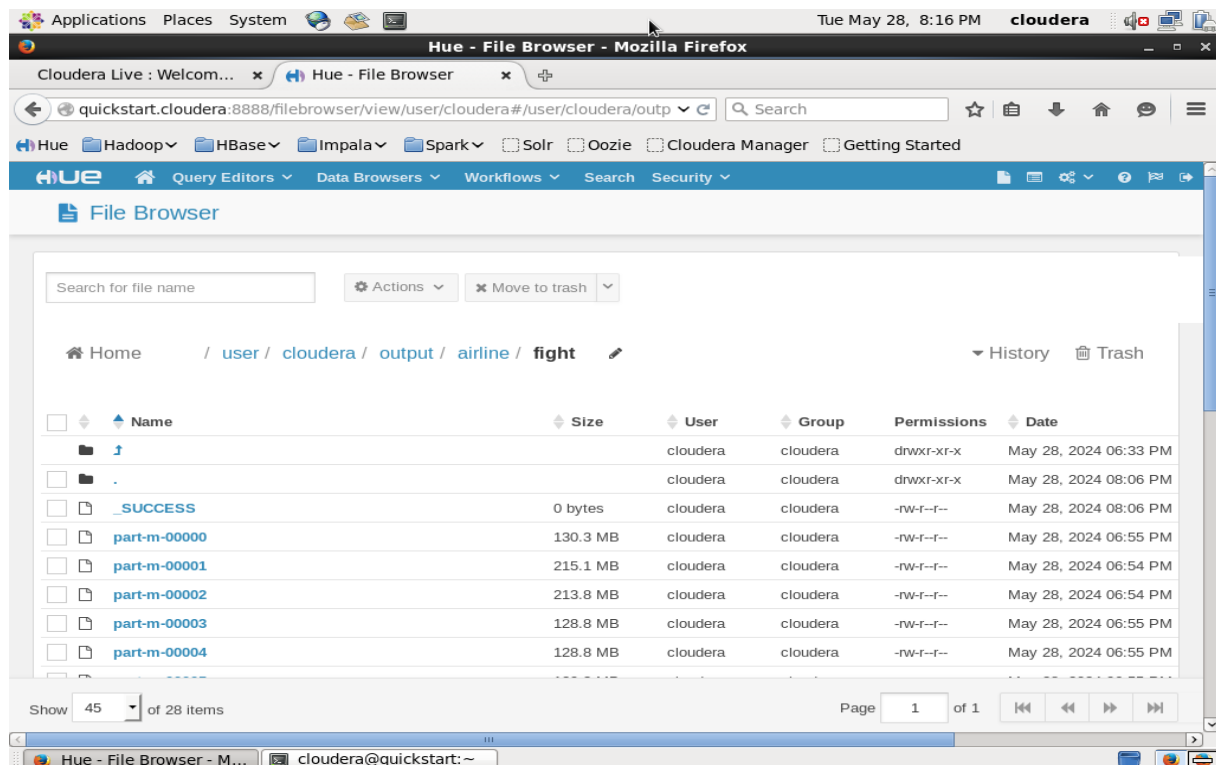
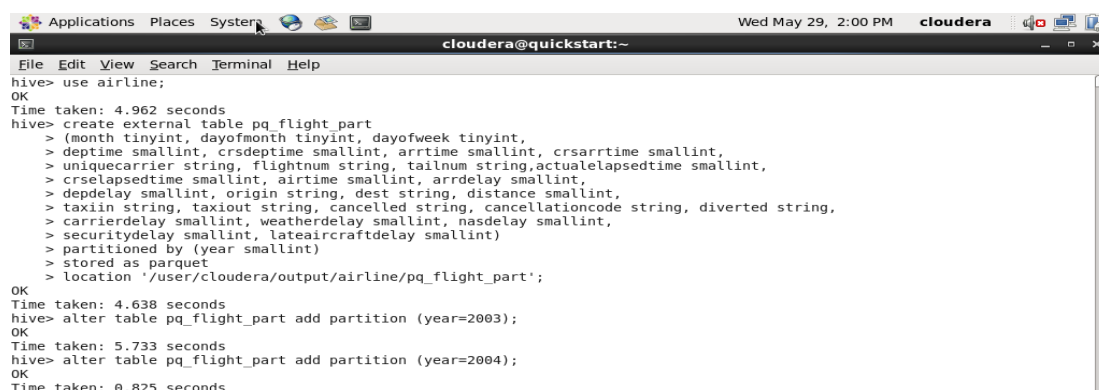


fig : Fichiers Hive : moins de taille qu'Impala

Implémentation Hive pour le partitionnement : les résultats sont donnés dans les images ci-dessous pour l'année 2003, 2006, 2007 et 2008.

En raison de contrainte de temps on a implementer seulement 2003 et 2004 dans pq\_flight\_part



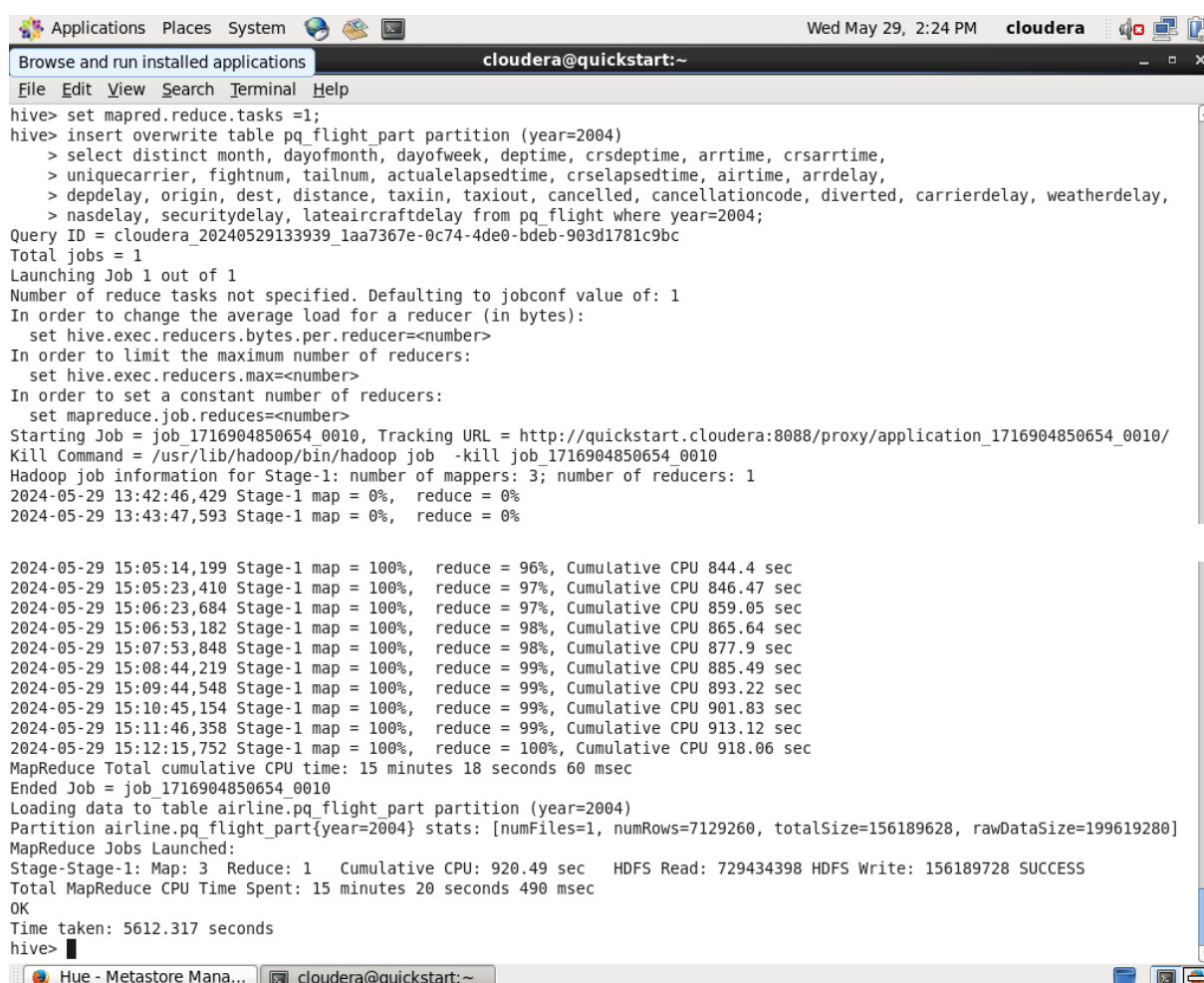
```

Applications Places System cloudera@quickstart:~
File Edit View Search Terminal Help
hive> use airline;
OK
Time taken: 4.962 seconds
hive> create external table pq_flight part
> (month tinyint, dayofmonth tinyint, dayofweek tinyint,
> deptime smallint, crsdeptime smallint, arrtime smallint, crsarrrtime smallint,
> uniquecarrier string, flightnum string, tailnum string,actualelapsedtime smallint,
> crselapsedtime smallint, airtime smallint, arrdelay smallint,
> depdelay smallint, origin string, dest string, distance smallint,
> taxiin string, taxiout string, cancelled string, cancellationcode string, diverted string,
> carrierdelay smallint, weatherdelay smallint, nasdelay smallint,
> securitydelay smallint, lateaircraftdelay smallint)
> partitioned by (year smallint)
> stored as parquet
> location '/user/cloudera/output/airline/pq_flight_part';
OK
Time taken: 4.638 seconds
hive> alter table pq_flight_part add partition (year=2003);
OK
Time taken: 5.733 seconds
hive> alter table pq_flight_part add partition (year=2004);
OK
Time taken: 0.825 seconds

```

Fig : Creation de la table pq\_flight\_part et ajouter des partitions

(Vous trouverez le code complet pour toutes les partitions dans le fichier code)



```

Applications Places System cloudera@quickstart:~
Browse and run installed applications cloudera@quickstart:~
File Edit View Search Terminal Help
hive> set mapred.reduce.tasks=1;
hive> insert overwrite table pq_flight part partition (year=2004)
> select distinct month, dayofmonth, dayofweek, deptime, crsdeptime, arrtime, crsarrrtime,
> uniquecarrier, flightnum, tailnum, actualelapsedtime, crselapsedtime, airtime, arrdelay,
> depdelay, origin, dest, distance, taxiin, taxiout, cancelled, cancellationcode, diverted, carrierdelay, weatherdelay,
> nasdelay, securitydelay, lateaircraftdelay from pq_flight where year=2004;
Query ID = cloudera_20240529133939_1aa7367e-0c74-4de0-bdeb-903d1781c9bc
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Defaulting to jobconf value of: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1716904850654_0010, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1716904850654_0010/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1716904850654_0010
Hadoop job information for Stage-1: number of mappers: 3; number of reducers: 1
2024-05-29 13:42:46,429 Stage-1 map = 0%, reduce = 0%
2024-05-29 13:43:47,593 Stage-1 map = 0%, reduce = 0%

2024-05-29 15:05:14,199 Stage-1 map = 100%, reduce = 96%, Cumulative CPU 844.4 sec
2024-05-29 15:05:23,410 Stage-1 map = 100%, reduce = 97%, Cumulative CPU 846.47 sec
2024-05-29 15:06:23,684 Stage-1 map = 100%, reduce = 97%, Cumulative CPU 859.05 sec
2024-05-29 15:06:53,182 Stage-1 map = 100%, reduce = 98%, Cumulative CPU 865.64 sec
2024-05-29 15:07:53,848 Stage-1 map = 100%, reduce = 98%, Cumulative CPU 877.9 sec
2024-05-29 15:08:44,219 Stage-1 map = 100%, reduce = 99%, Cumulative CPU 885.49 sec
2024-05-29 15:09:44,548 Stage-1 map = 100%, reduce = 99%, Cumulative CPU 893.22 sec
2024-05-29 15:10:45,154 Stage-1 map = 100%, reduce = 99%, Cumulative CPU 901.83 sec
2024-05-29 15:11:46,358 Stage-1 map = 100%, reduce = 99%, Cumulative CPU 913.12 sec
2024-05-29 15:12:15,752 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 918.06 sec
MapReduce Total cumulative CPU time: 15 minutes 18 seconds 60 msec
Ended Job = job_1716904850654_0010
Loading data to table airline.pq_flight part partition (year=2004)
Partition airline.pq_flight part{year=2004} stats: [numFiles=1, numRows=7129260, totalSize=156189628, rawDataSize=199619280]
MapReduce Job Launch:
Stage-Stage-1: Map: 3 Reduce: 1 Cumulative CPU: 920.49 sec HDFS Read: 729434398 HDFS Write: 156189728 SUCCESS
Total MapReduce CPU Time Spent: 15 minutes 20 seconds 490 msec
OK
Time taken: 5612.317 seconds
hive>

```

fig : Requête pour réduire trois fichiers à un pour les données de l'année 2004



The screenshot shows the Hue Metastore Manager interface. The breadcrumb navigation is 'Databases > airline > pq\_flight\_part'. The 'Sample' tab is selected, displaying a table with the following columns: month, dayofmonth, dayofweek, deptime, crsdeptime, arrtime, crsarrrtime, and unique. The table contains 12 rows of data for the year 2004.

	month	dayofmonth	dayofweek	deptime	crsdeptime	arrtime	crsarrrtime	unique
0	1	1	4	0	550	0	658	MQ
1	1	1	4	0	600	0	721	UA
2	1	1	4	0	600	0	728	AA
3	1	1	4	0	600	0	825	OO
4	1	1	4	0	600	0	855	OO
5	1	1	4	0	610	0	738	OO
6	1	1	4	0	617	0	953	AA
7	1	1	4	0	620	0	827	AS
8	1	1	4	0	625	0	739	AA
9	1	1	4	0	626	0	857	OO
10	1	1	4	0	629	0	1126	AA
11	1	1	4	0	630	0	720	OO

fig : Output pour l'année 2004 – Mapreduce (Visualiser dans browsers)

## 5. Clustering:

Hive est utilisé pour le regroupement(clustering). Le regroupement est utilisé pour échantillonner et pour le tri/jointement par compartiments. L'échantillonnage de données est effectué sur le transporteur, l'origine et le temps en utilisant des tables par compartiments. Les tables par compartiments sont fantastiques car elles permettent un échantillonnage beaucoup plus efficace que les tables sans compartiments, et elles peuvent permettre des opérations d'économie de temps telles que les jointures côté mappage. Le flux de cette méthode consiste d'abord à créer de petits compartiments et à y placer certaines valeurs. Ces valeurs sont appelées échantillons et à partir de chaque échantillon par valeur particulière, nous pouvons créer un cluster pour les ensembles de données. Si nous ne faisons pas de regroupement, cela se fera avec un échantillonnage aléatoire et un regroupement.

Pour connaître la raison du retard des vols, dans notre projet, nous pouvons faire un échantillonnage par année et par transporteur, et pour cela, une requête est donnée, et si nous faisons un échantillonnage par pays, nous pouvons obtenir un échantillonnage efficace.

regroupé par (unique carrier, année) en 6 compartiments

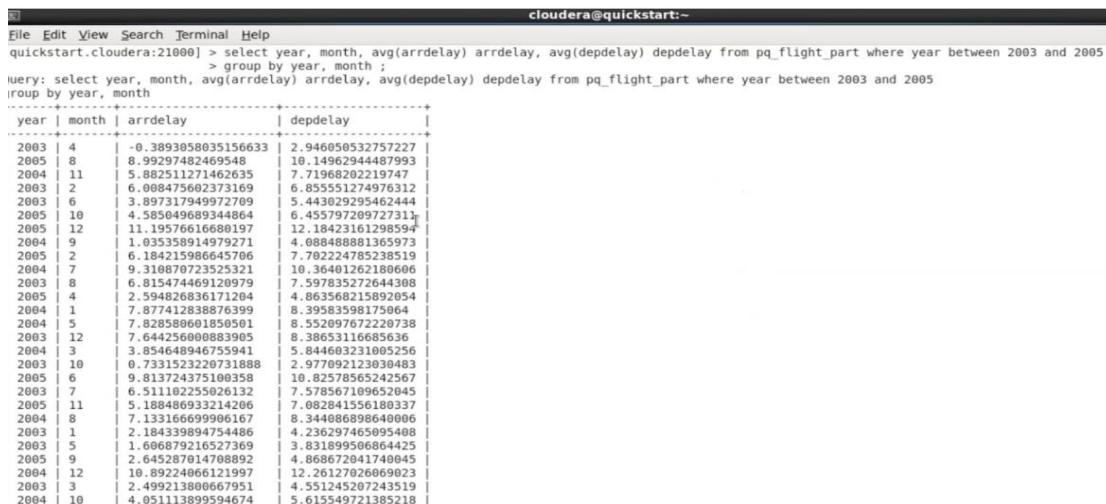
Impala ne prend pas en charge l'échantillonnage et le regroupement.

Compression des données, ajustement et optimisation des requêtes

Hive :



Compression des données : Hive utilise MapReduce et donc les données sont compressées à l'aide de Hive pour économiser de l'espace sur le disque et le réseau. Après que Hive ait terminé l'exécution de la requête, le résultat est soumis au JobTracker, qui réside sur YARN. Le JobTracker se compose de tâches Map/Reduce qui exécutent la tâche de mappage et de réduction pour stocker le résultat final dans le HDFS. La tâche de mappage déséréalise (reading) les données du HDFS et la tâche de réduction sérialise (writing) les données en tant que résultat de la requête Hive. MapReduce est le cadre utilisé pour traiter les données qui sont stockées dans le HDFS, ici le langage natif Java est utilisé pour écrire des programmes MapReduce. Hive est un cadre de traitement par lots. Ce composant traite les données en utilisant un langage appelé langage de requête Hive (HQL). Hive empêche l'écriture de programmes MapReduce en Java. Au lieu de cela, on peut utiliser un langage similaire à SQL pour effectuer ses tâches quotidiennes. Pour HIVE, il n'y a aucun processus pour communiquer directement avec les tâches Map/Reduce. Il communique avec le gestionnaire de tâches (Application Master dans YARN) uniquement pour les choses liées au traitement des tâches une fois qu'elles sont planifiées.



year	month	arrdelay	depdelay
2003	4	-0.3893858035156633	2.946050532757227
2005	8	8.99297482469548	10.14962944487993
2004	11	5.882511271462635	7.71968202219747
2003	2	6.008475602373169	6.855551274976312
2003	6	3.897317949972709	5.443029295462444
2005	10	4.585049689344864	6.455797209727311
2005	12	11.19576616680197	12.18423161298594
2004	9	1.035358914979271	4.08848881365973
2005	2	6.184215986645706	7.702224785238519
2004	7	9.310870723525321	10.36401262189606
2003	8	6.815474469120979	7.597835272644308
2005	4	2.594826836171204	4.863568215892054
2004	1	7.877412838876399	8.39583598175064
2004	5	7.828580661850501	8.55209767220738
2003	12	7.644256008083905	8.38653116685636
2004	3	3.854648946755941	5.844683231005256
2003	10	0.7331523220731888	2.977092123030483
2005	6	9.813724375100358	10.82578565242567
2003	7	6.511102255026132	7.578567109652045
2005	11	5.188480933214206	7.082841556180337
2004	8	7.133166699906167	8.344806898640006
2003	1	2.184339894754486	4.236297465095408
2003	5	1.606879216527369	3.831899506864425
2005	9	2.645287014708892	4.868672041740045
2004	12	10.89224066121997	12.26127026069823
2003	3	2.49921380667951	4.551245207243519
2004	10	4.051113899594674	5.615549721385218

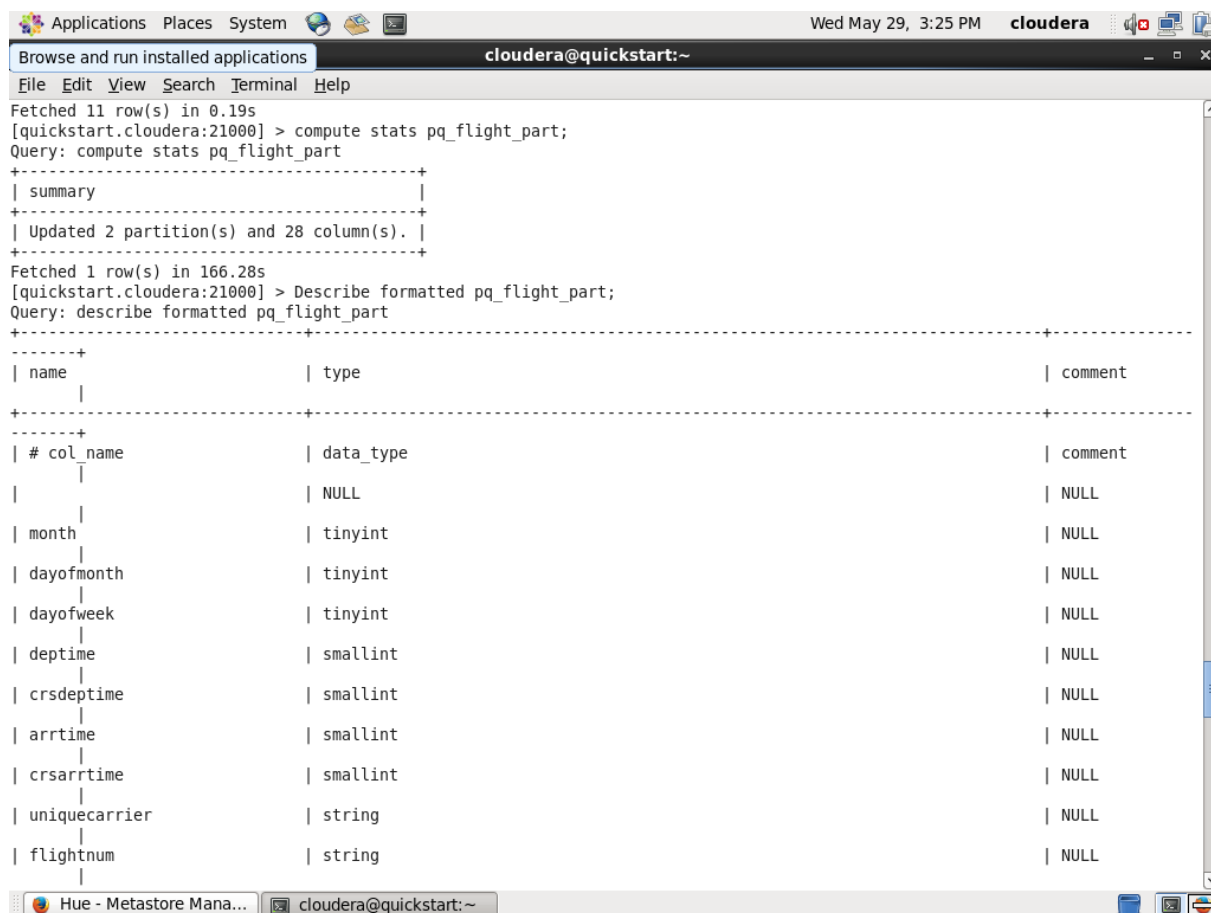
fig : Output de retard de vol de 2003 à 2005

Impala :

UTILISER STATS pour améliorer les performances des requêtes lors de l'exécution de jointures.

Formats de fichier : Les statistiques sont calculées sur différents formats de fichier Impala pour comparer les performances. Nous calculons sur six fichiers, nous créons une portion et ajoutons des données à la partition. Toutes les requêtes et les résultats sont présentés dans les images ci-dessous, où lorsque vous calculez des statistiques, nous pouvons connaître le nombre de colonnes et de lignes. De plus, grâce au caching HDFS, nous pouvons également améliorer les performances.

Les résultats de l'exécution Impala sont donnés dans les images ci-dessous, y compris compute stats et describe.

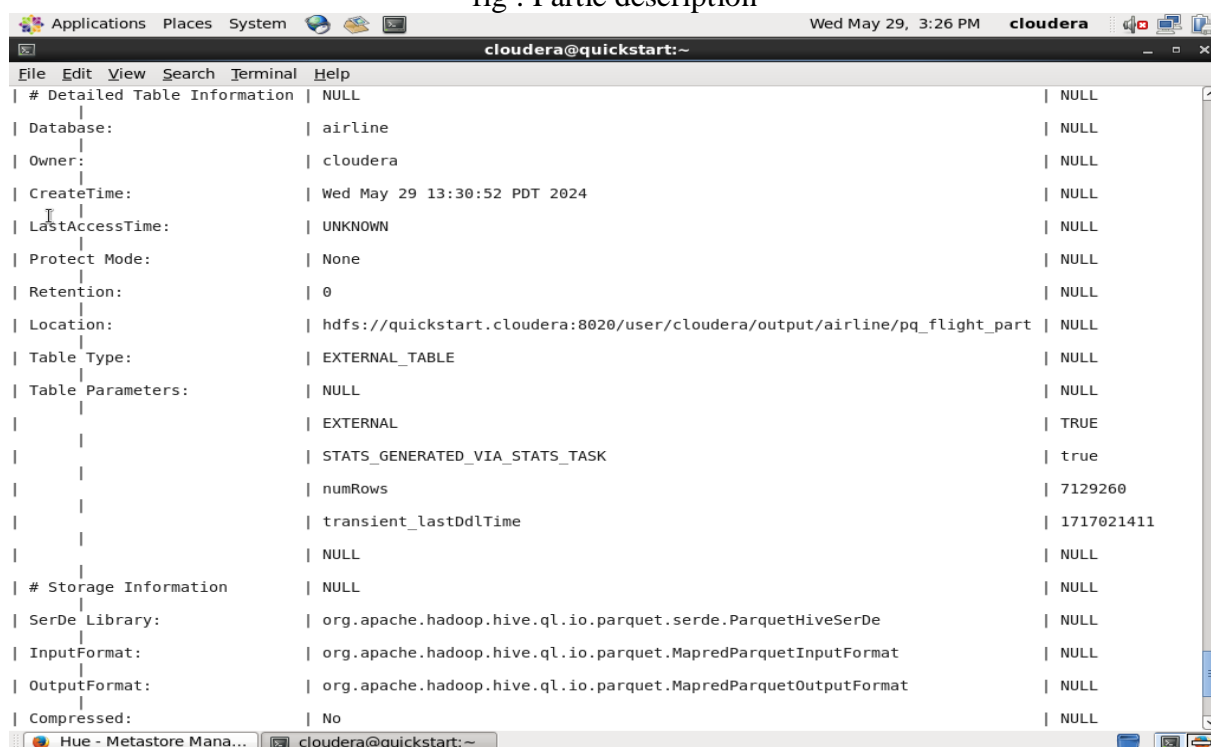


```

Applications Places System cloudera Wed May 29, 3:25 PM cloudera
Browse and run installed applications cloudera@quickstart:~
File Edit View Search Terminal Help
Fetched 11 row(s) in 0.19s
[quickstart.cloudera:21000] > compute stats pq_flight_part;
Query: compute stats pq_flight_part
+-----+
| summary |
+-----+
| Updated 2 partition(s) and 28 column(s). |
+-----+
Fetched 1 row(s) in 166.28s
[quickstart.cloudera:21000] > Describe formatted pq_flight_part;
Query: describe formatted pq_flight_part
+-----+
| name | type | comment |
+-----+
+-----+
| # col_name | data_type | comment |
+-----+
| month | tinyint | NULL |
| dayofmonth | tinyint | NULL |
| dayofweek | tinyint | NULL |
| deptime | smallint | NULL |
| crsdeptime | smallint | NULL |
| arrtime | smallint | NULL |
| crsarrrtime | smallint | NULL |
| uniquecarrier | string | NULL |
| flightnum | string | NULL |
+-----+

```

fig : Partie description



```

Applications Places System cloudera Wed May 29, 3:26 PM cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help
| # Detailed Table Information | NULL | NULL |
+-----+
| Database: | airline | NULL | |
| Owner: | cloudera | NULL |
| CreateTime: | Wed May 29 13:30:52 PDT 2024 | NULL |
| LastAccessTime: | UNKNOWN | NULL |
| Protect Mode: | None | NULL |
| Retention: | 0 | NULL |
| Location: | hdfs://quickstart.cloudera:8020/user/cloudera/output/airline/pq_flight_part | NULL |
| Table Type: | EXTERNAL_TABLE | NULL |
| Table Parameters: | NULL | NULL |
| | | |
| | | EXTERNAL | TRUE |
| | | |
| | | STATS_GENERATED_VIA_STATS_TASK | true |
| | | |
| | | numRows | 7129260 |
| | | |
| | | transient_lastDdlTime | 1717021411 |
| | | |
| | | NULL | NULL |
+-----+
| # Storage Information | NULL | NULL |
+-----+
| SerDe Library: | org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe | NULL |
| InputFormat: | org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat | NULL |
| OutputFormat: | org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat | NULL |
| Compressed: | No | NULL |
+-----+

```

fig : Output Impala décrit tout - partie statistiques (sélectionnée)

```
[quickstart.cloudera:21000] > show table stats pq_flight_part;
Query: show table stats pq_flight_part
+-----+-----+-----+-----+-----+-----+-----+-----+
| year | #Rows | #Files | Size | Bytes Cached | Cache Replication | Format | Incremental stats |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 2003 | 0      | 0      | 0B   | NOT CACHED   | NOT CACHED        | PARQUET | false             |
| 2004 | 7129260 | 1      | 148.95MB | NOT CACHED   | NOT CACHED        | PARQUET | false             |
| Total | 7129260 | 1      | 148.95MB | 0B           |                    |         |                   |
+-----+-----+-----+-----+-----+-----+-----+-----+
Fetched 3 row(s) in 2.26s
[quickstart.cloudera:21000] > compute stats pq_flight;
Query: compute stats pq_flight
+-----+-----+
| summary |
+-----+-----+
| Updated 1 partition(s) and 29 column(s). |
+-----+-----+
Fetched 1 row(s) in 298.28s
[quickstart.cloudera:21000] >
```

fig : Affichage des statistiques du tableau et calculer les statistiques, y compris la taille du fichier (sélectionné)

La raison du zéro dans la ligne 2003 est qu'on a déjà ajouté la partition mais on n'a pas encore écrit dans la table pq\_flight\_part

Utilisation des vues de base de données pour représenter les données

Objectif : L'objectif de la visualisation des données est de projeter et de masquer les complexités et la sécurité. La sécurité est nécessaire en raison du calcul de certains accidents d'avion. Toutes les commandes pour visualiser des données et les résultats correspondants sont donnés dans la figure ci-dessous et dans le code.

```

cloudera@quickstart:~
File Edit View Search Terminal Help
Query: describe pq_airports
+-----+-----+-----+
| name   | type   | comment |
+-----+-----+-----+
| iata    | string |          |
| airport | string |          |
| city    | string |          |
| state   | string |          |
| country | string |          |
| geolat  | float  |          |
| geolong | float  |          |
+-----+-----+-----+
Fetched 7 row(s) in 0.16s
[quickstart.cloudera:21000] > compute stats pq_airports;
Query: compute stats pq_airports
+-----+-----+-----+
| summary |
+-----+-----+-----+
| Updated 1 partition(s) and 7 column(s). |
+-----+-----+-----+
Fetched 1 row(s) in 5.58s
[quickstart.cloudera:21000] > compute stats pq_carriers;
Query: compute stats pq_carriers
+-----+-----+-----+
| summary |
+-----+-----+-----+
| Updated 1 partition(s) and 2 column(s). |
+-----+-----+-----+
Fetched 1 row(s) in 8.26s
[quickstart.cloudera:21000] > desc pq_carriers;
Query: describe pq_carriers
+-----+-----+-----+
| name      | type      | comment |
+-----+-----+-----+
| ccde      | varchar(4) |          |
| description | varchar(30) |          |
+-----+-----+-----+
Fetched 2 row(s) in 0.15s
[quickstart.cloudera:21000] >

```

fig : Tableau de description

À partir du tableau ci-dessus, nous pouvons écrire n'importe quelle requête pour afficher les données différemment. Ensuite, il est nécessaire de recalculer toutes les tables, et cette requête de visualisation est destinée à des fins de sécurité, afin que vous puissiez identifier immédiatement les accidents d'avion.

```

1 create view v_flights_denom as
2 select year,month, dayofmonth,dayofweek, flightnum, deptime, crsdeptime, arrtime, crsarrrtime,
3         actualelapsedtime,
4         crselapsedtime, airtime, arrdelay, depdelay,
5         origin, dest, distance, taxiin, taxiout,
6         cancelled, cancellationcode, diverted, carrierdelay,
7         weatherdelay, nasdelay, securitydelay, lateaircraftdelay
8         ,pao.airport origin_airport, pao.city origin_city, pao.state origin_state, pao.country origi
9         ,pad.airport dest_airport, pad.city dest_city, pad.state dest_state, pad.country dest_countr
10        ,pf.uniquecarrier, pc.description carrier
11        ,pf.tailnum, ppi.type plane_type, ppi.manufacturer, ppi.issue_date, ppi.model, ppi.status, s
12        from pq_flight pf
13 join pq_airports pao on pao.iata = pf.origin
14 join pq_airports pad on pad.iata = pf.dest
15
16 join pq_carriers pc on pc.ccde = pf.uniquecarrier
17 join pq_plane_info ppi on ppi.tailnum = pf.tailnum
18
19 INFO : Starting task [Stage-0:DDL] in serial mode
20 INFO : Completed executing command(queryId=hive_20170115093838_4565124c-7874-4b2a-bedd-a55926257e52); Time taki
21 INFO : OK

```

fig : Requête pour joindre des tables différemment

## **CONCLUSION**

Ce projet pratique porte sur l'analyse de jeux de données sur les compagnies aériennes. L'objectif principal du projet pratique était, après le prétraitement des ensembles de données, de résumer toutes les tables et de visualiser également les données en fonction de différents énoncés de problèmes. De plus, nous avons comparé les requêtes et les résultats de Hive et Impala et analysé les retards des vols. En conclusion, nous avons constaté qu'Impala n'a pas autant d'intégration avec l'écosystème Hadoop que Hive et nous supposons qu'Impala ne prend pas en charge HDFS et Hbase. Cependant, nous avons constaté que la fonction de calcul des statistiques d'Impala était utilisée pour améliorer les performances ; le temps d'exécution pour Impala est beaucoup moins que le temps d'exécution pour Hive. Finalement nous avons conclu que la plupart du temps, chaque fois que des retards d'arrivée se produisaient, le départ était également retardé, sauf pour un certain nombre de vols, où ce n'était pas le cas selon le jeu de données considéré.

## RÉFÉRENCES

Les exposées de nos collègues

[1]<https://www.cloudera.com/>

[2] [Data Expo 2009: Airline on time data - ASA Statistical Computing Dataverse \(harvard.edu\)](#)

[3]<https://www.ijcsmc.com/docs/papers/June2017/V6I6201764.pdf>

[4]<https://www.cloudera.com/documentation/enterprise/latest/PDF/cloudera-quickstart.pdf>

[5]<https://www.cloudera.com/documentation/enterprise/5-9-x/PDF/cloudera-introduction.pdf>

[6]<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManual>

DDL-Dynamic Partitions

[7]<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL+BucketedTables>

[8][https://www.cloudera.com/documentation/enterprise/5-9-](https://www.cloudera.com/documentation/enterprise/5-9-x/topics/introduction_compression.html)

[x/topics/introduction\\_compression.h](https://www.cloudera.com/documentation/enterprise/5-9-x/topics/introduction_compression.html)

tml

[9][https://www.cloudera.com/documentation/enterprise/5-9-](https://www.cloudera.com/documentation/enterprise/5-9-x/topics/impala_compute_stats.html)

[x/topics/impala\\_compute\\_stats.html](https://www.cloudera.com/documentation/enterprise/5-9-x/topics/impala_compute_stats.html)

[10][http://hadoopilluminated.com/hadoop\\_illuminated/hadoop-illuminated.pdf](http://hadoopilluminated.com/hadoop_illuminated/hadoop-illuminated.pdf)

[11][http://cidrdb.org/cidr2015/Papers/CIDR15\\_Paper28.pdf](http://cidrdb.org/cidr2015/Papers/CIDR15_Paper28.pdf)

[12]<https://www.cloudera.com/documentation/enterprise/5-5-x/PDF/cloudera-impala.pdf>