

Airline Dataset Analysis using Hadoop, Hive, Pig and Impala

Illinois Institute of Technology, Chicago
by Ayushi Patel and Bhargavi Deshpande

Abstract — *This paper is about the analysis of the airline data set which is performed using Cloudera, it delivers the modern platform for analytics optimised for the cloud. The project is performing big data analysis on airline dataset using Hadoop and tools like Pig, Hive and Impala. At the end, analysis of delay will be visualised using Excel spreadsheet.*

Keywords : *Hadoop, HDFS, Pig, Hive and Impala, Data Analysis*

INTRODUCTION

There is a lot of excitement that exists with the term Big Data. In simple words, Big Data can be large-scale data which does not have a well-defined structure. The size of the data is so huge that it is not practically easy for a single computer to store and process all the data. In traditional computing approach there are many problems in a different way and the focus was always to increase the processing speed and power of the computer. As the data grows exponentially, the processing power of the single computer becomes a bottleneck and thus a new approach was needed to address the issue at hand[3]. A new way was developed wherein many non-expensive commodity computers were working together in harmony with each other, in order to store and process this big data in parallel. This allows us to extract meaningful information from a large data set. In addition, by using the cloud technology, it is easy to create cluster, compute and release the computing resources when it is not needed. So, from the cloud technology we get the computing power of the cluster of computers with minimal investment[10]. The draft mainly contains details about Hadoop, Hive, Pig and Impala and gives vague idea of the project flow.

TECHNOLOGY USED

- **Understanding of Infrastructure**

Infrastructure is the cornerstone of Big Data architecture. Possessing the right tools for storing, processing and analysing your data is crucial in any Big Data project.

Two major Users of data infrastructure : Human and System. Ways that data can be ingested/served in data infrastructure : Messaging Layer(message oriented middleware) , Data Warehousing(Batch) and Backend Services.

- **Hadoop and Big Data**

Hadoop is one of the tools designed to handle big data. Hadoop and other software products work to interpret or parse the results of big data searches through specific proprietary algorithms

and methods. Hadoop is an open-source program under the Apache license that is maintained by a global community of users. Apache Hadoop is 100% open source, and pioneered to be a fundamentally new way of storing and processing data instead of relying on expensive, proprietary hardware and different systems to store and process data.

HDFS : HDFS is the primary distributed storage on Hadoop for managing pools of big data, that spans across large clusters of commodity servers. HDFS is regarded as the bucket of the hadoop ecosystem, where data is dumped and sits there until the user wants to export it to another tool, for running analysis on the stored data. Any machine that supports Java programming language can run HDFS[10].

Working of HDFS : HDFS uses a master slave architecture where each cluster has a NameNode for managing the file system operations and supporting DataNodes for managing data storage on individual computing nodes (usually there exists one DataNode per node in the Hadoop cluster). HDFS stores data in files which are divided into one or more segments and are stored in particular DataNodes. These small segments of files are referred to as block and the minimum amount of data that can be read or written is referred to as a single block. By default the size of a block is 64MB which can be changed in the configuration file.

- **Apache Pig**

Apache Pig is a high level procedural dataflow language on top of Hadoop for processing and analysing big data without having to write Java based MapReduce code. Apache Pig has RDBMS like features- joins, distinct clause, union, etc. For crunching large files containing semi-structured or unstructured data[10]. **Apache pig components:** 1. Pig Latin: It is a SQL like data flow language to join, group and aggregate distributed data sets with ease. 2. Pig Engine: Pig engine takes the Pig Latin scripts written by users, parses them, optimizes them and then executes them as a series of MapReduce jobs on a Hadoop Cluster.

- **Apache Hive: A Data Warehousing Solution for Big Data on Hadoop**

Hive is a data warehousing solution developed on top of Hadoop to meet the big data challenges of storing, managing and processing large data sets without having to write complex Java based MapReduce programs. Hive is a familiar programming model for big data professionals who know SQL but do not have a good grip in programming. Hive is not a relational database or an architecture for online transaction processing[11]. It is particularly designed for online analytical processing systems (OLAP). Hive compiler converts the queries written in HiveQL into MapReduce jobs so that Hadoop developers need not worry much about the complex programming code beyond the processing and they can focus on the business problem. The three important functions performed by Hive include - data summarization, data querying and data analysis.

Apache Hive is extensively used by data scientists and data analysts for data exploration, building data pipelines and for processing ad-hoc queries.

Hive Components :

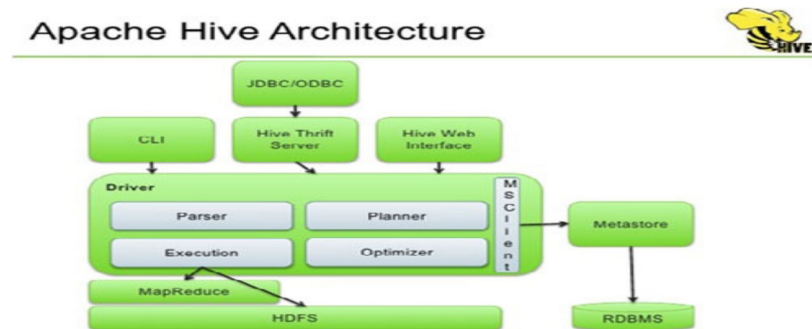


Fig.1 Apache Hive Architecture

=>. CLI, JDBC, ODBC or any other Web GUI form the external interfaces to the Hive framework interface for creating interaction between user and HDFS.

=>. Metastore Thrift API keeps track of what data is stored in which part of the HDFS .It is like a system catalog.

=>. Driver is heart of the Hive architecture responsible for compilation, optimization and execution of HiveQL statements.

=>. Thrift Server is a client side API for executing HiveQL statements.

How a HiveQL query is executed in Apache Hive? : Whenever a user submits a HiveQL query, it is first compiled. The compiled query is then executed by an execution engine like Hadoop MapReduce or Apache Tez. Data in diverse formats like ORC, AVRO, Parquet, or Text reside in HDFS on which the query is to be executed. YARN then allocates desired resources across the Hadoop cluster for execution. The results of the query execution are sent over a JDBC or ODBC connection.

- **Cloudera quick-start and Impala**

Cloudera provides a scalable, flexible, integrated platform that makes it easy to manage rapidly increasing volumes and varieties of data in your enterprise. Cloudera products and solutions enable you to deploy and manage Apache Hadoop and related projects, manipulate and analyze your data, and keep that data secure and protected[4]. Cloudera provides many products and tools. We are using CDH and Apache Impala.

CDH: The most complete, tested and popular distribution of Apache Hadoop and other related open-source projects, including Apache Impala and Cloudera Search. CDH also provides flexibility, security and integration with numerous hardware and software solutions[5].

Apache Impala: A massively parallel processing SQL engine for interactive analytics and business intelligence. Its highly optimized architecture makes it ideally suited for traditional BI-style queries with joins, aggregations, and subqueries. It can query Hadoop data files from a variety of sources, including those produced by MapReduce jobs overloaded into Hive tables[11]. The YARN resource management component lets Impala coexist on clusters running batch workloads concurrently with Impala SQL queries. You can manage Impala alongside other Hadoop components through the Cloudera Manager user interface and secure its data through the Sentry authorization framework.

How Impala Works with CDH.

The following graphic illustrates how Impala is positioned in the broader Cloudera environment[11]:

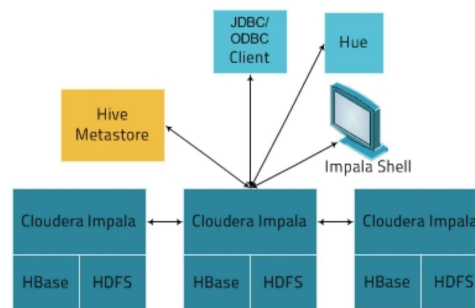


Fig.2 Working of Impala

PRACTICUM

Working of the Project :

1. Datasets

The benchmarking web resource is available at <http://stat-computing.org/dataexpo/2009/>. Dataset is freely available for download from this website . According to the website, the data consists of flight arrival and departure details for all commercial flights within the USA, from October 1987 to April 2008[2]. However, we are using dataset from 2003 to 2008 for our project, and all csv files are around 100 MB to 125 MB. The dataset used mainly is flights.csv which has a total of 29 variables with dimension of 31 columns and 58,19,079 rows. We have drawn conclusion by mainly focusing on variables arrival delay and departure delay. When there was a delay in the arrival of flights, then there was a delay in departure too for most flights except for few.

2. Data Preparation

Data preparation is the key to big data success. One of the primary barriers to big data success is the lack of a data preparation strategy[10]. Data preparation includes all the steps necessary to acquire, prepare, accurate, and manage the data assets of the organization. Some most important steps are given for the project.

=>. Data Pre-processing and Extraction and loading :

Purpose: In general the purpose of the data preprocessing to give some structure to unstructured data, Integrated data with values from external system and preprocess binary content; Writing directly to the data warehouse.

=>. Data storage is less of a problem than efficient data retrieval.

=>. In this step we download all required datasets in local machine and extracted all datasets.

=>. Pig cannot write to a HCatalog table in parquet format but Spark can write to Hive Tables and also write parquet files to hdfs directories

=>. Setting up the Data-warehouse :

Managing files on hdfs; Queries for that are given in Fig.3 and Fig.4.



```
cloudera [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Fri Nov 23, 7:04 PM cloudera
cloudera@quickstart:~/Downloads

2018-11-23 18:10:07,418 [JobControl] INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl - Submitted application application_1543021518145_0002
2018-11-23 18:10:07,527 [JobControl] INFO org.apache.hadoop.mapreduce.Job - The url to track the job: http://quickstart.cloudera:8088/proxy/application_1543021518145_0002/
2018-11-23 18:10:07,527 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - HadoopJobId: job_1543021518145_0002
2018-11-23 18:10:07,527 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Processing aliases headless_data,raw_data,rel_data
2018-11-23 18:10:07,527 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - detailed locations: M: raw_data[67,11],raw_data[-1,-1],headless_data[80,16],rel_data[81,11] C: R:
2018-11-23 18:10:07,527 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - More information at: http://localhost:50030/jobdetails.jsp?jobid=job_1543021518145_0002
2018-11-23 18:10:07,769 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 0% complete
2018-11-23 18:14:03,853 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 4% complete
2018-11-23 18:17:00,490 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 8% complete
2018-11-23 18:21:01,331 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 12% complete
2018-11-23 18:23:53,911 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 16% complete
2018-11-23 18:28:02,708 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 20% complete
2018-11-23 18:30:39,172 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 24% complete
2018-11-23 18:33:18,314 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 28% complete
2018-11-23 18:37:00,151 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 32% complete
2018-11-23 18:39:22,779 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 36% complete
2018-11-23 18:43:12,371 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 40% complete
2018-11-23 18:45:25,950 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 44% complete
2018-11-23 18:47:51,432 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 49% complete
2018-11-23 18:48:14,899 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
2018-11-23 18:48:14,239 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 100% complete
2018-11-23 18:48:14,241 [main] INFO org.apache.pig.tools.pigstats.SimplePigStats - Script Statistics:

HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera 2018-11-23 18:10:00 2018-11-23 18:48:14 FILTER

Success!

Job Stats (Time in seconds):
JobId Maps Reduces MaxMapTime MinMapTime AvgMapTime MedianMapTime MaxReduceTime MinReduceTime AvgReduceTime MedianReduceTime Alias Feature Outputs
job_1543021518145_0002 31 0 513 94 431 432 n/a n/a n/a n/a headless_data,raw_data,rel_data MAP_ONLY /user/cloudera/output/airline/flight,

Input(s):
Successfully read 42363277 records (4832130119 bytes) from: "/user/cloudera/rawdata/airline/flight"

Output(s):
Successfully stored 42363277 records (3934400919 bytes) in: "/user/cloudera/output/airline/flight"

Counters:
Total records written : 42363277
Total bytes written : 3934400919
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1543021518145_0002
```

Fig.3 Managing files on Hadoop

```

[cloudera@quickstart ~]$ pwd
/home/cloudera
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera/output/airline/flight
Found 32 items
-rw-r--r-- 1 cloudera cloudera 0 2018-11-23 18:48 /user/cloudera/output/airline/flight/_SUCCESS
-rw-r--r-- 1 cloudera cloudera 134789428 2018-11-23 18:18 /user/cloudera/output/airline/flight/part-m-00000
-rw-r--r-- 1 cloudera cloudera 117708203 2018-11-23 18:17 /user/cloudera/output/airline/flight/part-m-00001
-rw-r--r-- 1 cloudera cloudera 118030734 2018-11-23 18:17 /user/cloudera/output/airline/flight/part-m-00002
-rw-r--r-- 1 cloudera cloudera 133990169 2018-11-23 18:18 /user/cloudera/output/airline/flight/part-m-00003
-rw-r--r-- 1 cloudera cloudera 133979045 2018-11-23 18:18 /user/cloudera/output/airline/flight/part-m-00004
-rw-r--r-- 1 cloudera cloudera 133836522 2018-11-23 18:18 /user/cloudera/output/airline/flight/part-m-00005
-rw-r--r-- 1 cloudera cloudera 133988803 2018-11-23 18:25 /user/cloudera/output/airline/flight/part-m-00006
-rw-r--r-- 1 cloudera cloudera 133910040 2018-11-23 18:25 /user/cloudera/output/airline/flight/part-m-00007
-rw-r--r-- 1 cloudera cloudera 133838700 2018-11-23 18:25 /user/cloudera/output/airline/flight/part-m-00008
-rw-r--r-- 1 cloudera cloudera 133724183 2018-11-23 18:26 /user/cloudera/output/airline/flight/part-m-00009
-rw-r--r-- 1 cloudera cloudera 133982417 2018-11-23 18:26 /user/cloudera/output/airline/flight/part-m-00010
-rw-r--r-- 1 cloudera cloudera 133995824 2018-11-23 18:33 /user/cloudera/output/airline/flight/part-m-00011
-rw-r--r-- 1 cloudera cloudera 133848801 2018-11-23 18:33 /user/cloudera/output/airline/flight/part-m-00012
-rw-r--r-- 1 cloudera cloudera 133889712 2018-11-23 18:32 /user/cloudera/output/airline/flight/part-m-00013
-rw-r--r-- 1 cloudera cloudera 133995824 2018-11-23 18:33 /user/cloudera/output/airline/flight/part-m-00014
-rw-r--r-- 1 cloudera cloudera 133901505 2018-11-23 18:34 /user/cloudera/output/airline/flight/part-m-00015
-rw-r--r-- 1 cloudera cloudera 133898623 2018-11-23 18:33 /user/cloudera/output/airline/flight/part-m-00016
-rw-r--r-- 1 cloudera cloudera 133831748 2018-11-23 18:34 /user/cloudera/output/airline/flight/part-m-00017
-rw-r--r-- 1 cloudera cloudera 133874122 2018-11-23 18:39 /user/cloudera/output/airline/flight/part-m-00018
-rw-r--r-- 1 cloudera cloudera 133790677 2018-11-23 18:40 /user/cloudera/output/airline/flight/part-m-00019
-rw-r--r-- 1 cloudera cloudera 133945990 2018-11-23 18:41 /user/cloudera/output/airline/flight/part-m-00020
-rw-r--r-- 1 cloudera cloudera 133919802 2018-11-23 18:41 /user/cloudera/output/airline/flight/part-m-00021
-rw-r--r-- 1 cloudera cloudera 123318524 2018-11-23 18:40 /user/cloudera/output/airline/flight/part-m-00022
-rw-r--r-- 1 cloudera cloudera 123877812 2018-11-23 18:41 /user/cloudera/output/airline/flight/part-m-00023
-rw-r--r-- 1 cloudera cloudera 123855883 2018-11-23 18:46 /user/cloudera/output/airline/flight/part-m-00024
-rw-r--r-- 1 cloudera cloudera 122586552 2018-11-23 18:47 /user/cloudera/output/airline/flight/part-m-00025
-rw-r--r-- 1 cloudera cloudera 122598611 2018-11-23 18:47 /user/cloudera/output/airline/flight/part-m-00026
-rw-r--r-- 1 cloudera cloudera 133877337 2018-11-23 18:48 /user/cloudera/output/airline/flight/part-m-00027
-rw-r--r-- 1 cloudera cloudera 132663392 2018-11-23 18:48 /user/cloudera/output/airline/flight/part-m-00028
-rw-r--r-- 1 cloudera cloudera 121296663 2018-11-23 18:47 /user/cloudera/output/airline/flight/part-m-00029
-rw-r--r-- 1 cloudera cloudera 10686454 2018-11-23 18:48 /user/cloudera/output/airline/flight/part-m-00030
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera/output/airline/flight/part-m-00000
-rw-r--r-- 1 cloudera cloudera 134789428 2018-11-23 18:18 /user/cloudera/output/airline/flight/part-m-00000
[cloudera@quickstart ~]$ hdfs dfs -cat /user/cloudera/output/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
2006,10,31,2,652,655,748,750,MQ,3740,N620AE,48,55,35,-10,-3,TKX,DFW,181,6,7,0,-,0,0,0,0,0
2006,10,1,7,1956,1945,2181,2040,MQ,3741,N630AE,63,55,34,21,13,DFW,TKX,181,5,24,0,-,0,13,0,0,0
2006,10,2,1,1946,1945,2041,2040,MQ,3741,N658AE,55,55,36,1,1,DFW,TKX,181,6,13,0,-,0,0,0,0,0
2006,10,3,2,1946,1945,2038,2040,MQ,3741,N697AB,50,55,38,-10,-5,DFW,TKX,181,6,14,0,-,0,0,0,0,0

```

Fig.4 Managing files on Hadoop

Set all data on Quickstart Hue as shown in below figure.

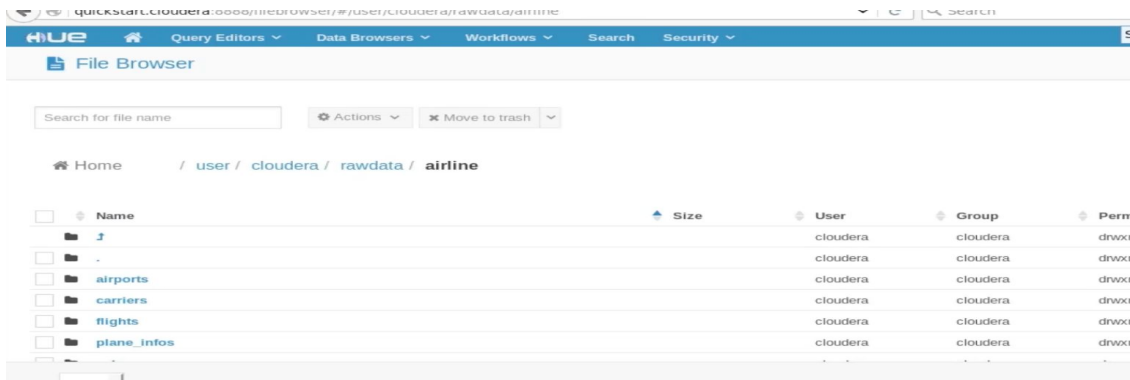
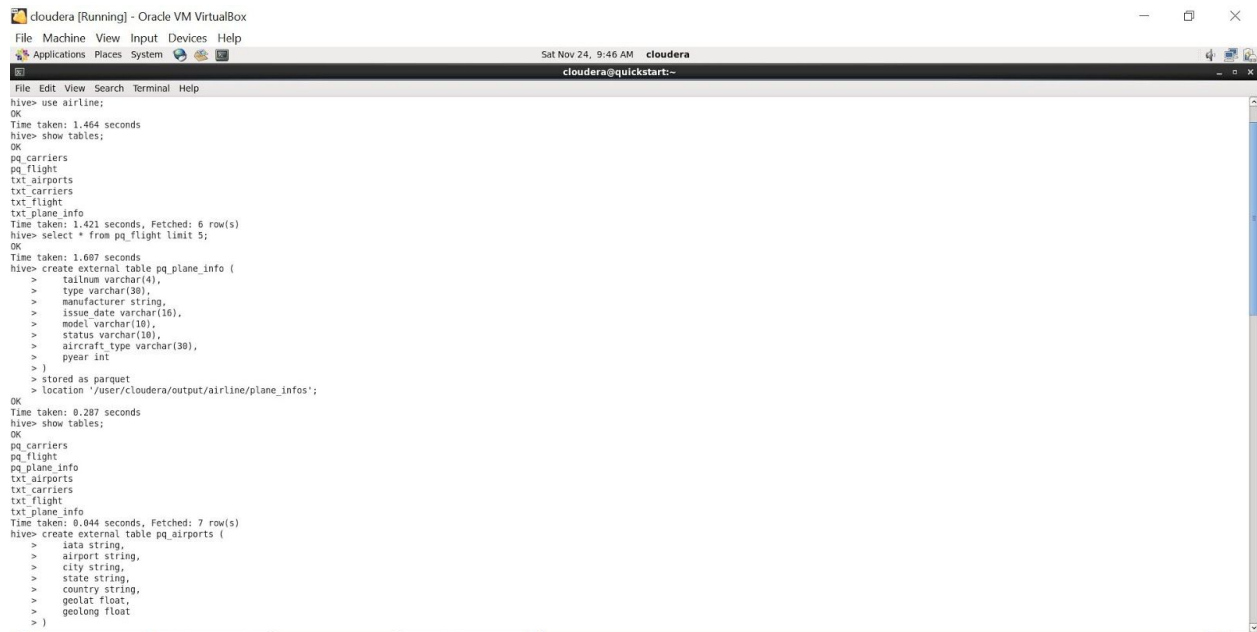


Fig.5 Data setup on Cloudera

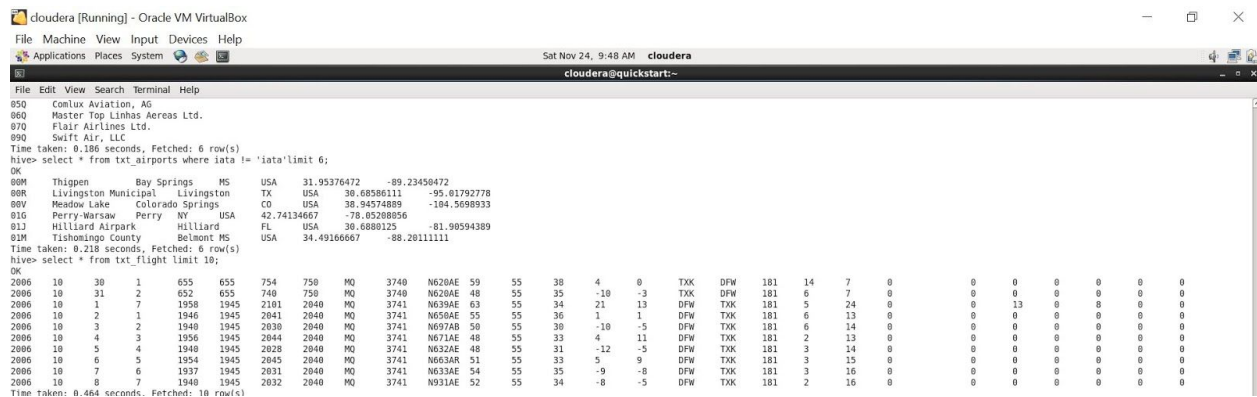
=>. Creating Data Table :

To analyze all datasets we create a table format for Datasets of airports, carriers and plane-data and analyze different queries. The queries for same are given in Fig.6 and example output for year 2006 is given in Fig.7.



```
cloudera [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Sat Nov 24, 9:46 AM cloudera
cloudera@quickstart:~$
File Edit View Search Terminal Help
hive> use airline;
OK
Time taken: 1.464 seconds
hive> show tables;
OK
pq_carriers
pq_flight
txt_airports
txt_carriers
txt_flight
txt_plane_info
Time taken: 1.421 seconds, Fetched: 6 row(s)
hive> select * from pq_flight limit 5;
OK
Time taken: 1.697 seconds
hive> create external table pq_plane_info (
  > tailnum varchar(4),
  > type varchar(30),
  > manufacturer string,
  > issue_date varchar(16),
  > model varchar(10),
  > status varchar(10),
  > aircraft_type varchar(30),
  > pyear int
  > )
  > stored as parquet
  > location '/user/cloudera/output/airline/plane_infos';
OK
Time taken: 0.287 seconds
hive> show tables;
OK
pq_carriers
pq_flight
pq_plane_info
txt_airports
txt_carriers
txt_flight
txt_plane_info
Time taken: 0.044 seconds, Fetched: 7 row(s)
hive> create external table pq_airports (
  > iata string,
  > airport string,
  > city string,
  > state string,
  > country string,
  > geolat float,
  > geolong float
  > )
```

Fig.6 Creating a Table



```
cloudera [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Sat Nov 24, 9:48 AM cloudera
cloudera@quickstart:~$
File Edit View Search Terminal Help
050 Comlux Aviation, AG
060 Master Top Linhas Aereas Ltd.
070 Flair Airlines Ltd.
090 Swift Air, LLC
Time taken: 0.186 seconds, Fetched: 6 row(s)
hive> select * from txt_airports where iata != 'iata' limit 6;
OK
00M Thippen 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-Marshaw 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: 0.218 seconds, Fetched: 6 row(s)
hive> select * from txt_flight limit 10;
OK
2006 10 30 1 655 655 754 750 MQ 3740 N620AE 50 55 30 4 0 TXK DFW 181 14 7 0 0 0 0 0 0 0
2006 10 31 2 652 655 740 750 MQ 3740 N620AE 48 55 35 -10 -3 TXK DFW 181 6 7 0 0 0 0 0 0 0
2006 10 1 7 1958 1945 2101 2040 MQ 3741 N639AE 63 55 34 21 13 DFW TXK 181 5 24 0 0 13 0 0 0 0
2006 10 2 1 1946 1945 2041 2040 MQ 3741 N650AE 55 55 36 1 1 DFW TXK 181 6 13 0 0 0 0 0 0 0
2006 10 3 2 1940 1945 2030 2040 MQ 3741 N657AB 50 55 30 -10 -5 DFW TXK 181 6 14 0 0 0 0 0 0 0
2006 10 4 3 1956 1945 2044 2040 MQ 3741 N671AE 48 55 33 4 11 DFW TXK 181 2 13 0 0 0 0 0 0 0
2006 10 5 4 1940 1945 2028 2040 MQ 3741 N632AE 48 55 31 -12 -5 DFW TXK 181 3 14 0 0 0 0 0 0 0
2006 10 6 5 1954 1945 2045 2040 MQ 3741 N663AR 51 55 33 5 9 DFW TXK 181 3 15 0 0 0 0 0 0 0
2006 10 7 6 1937 1945 2031 2040 MQ 3741 N633AE 54 55 35 -9 -8 DFW TXK 181 3 16 0 0 0 0 0 0 0
2006 10 8 7 1940 1945 2032 2040 MQ 3741 N931AE 52 55 34 -8 -5 DFW TXK 181 2 16 0 0 0 0 0 0 0
Time taken: 0.464 seconds, Fetched: 10 row(s)
```

Fig.7 Example output of flight of year 2006

3. Working with Hive vs. Impala or both

Impala server is a SQL query execution engine of Hadoop. Some of the features of Impala architecture are: A massively parallel processing or MPP engine for distributed clustering environment. It is open source and uses data on HDFS[6]. Consists of various daemon processes that run on specific hosts within your Hadoop cluster. The three main components of Impala are Impala daemon, Impala statestore and Impala catalog service, represented by the

daemons `impalad` , `statestored` and `catalog` respectively. This architecture is shown in Fig.2. However core component of Impala is the daemon process running on each Impala cluster node.

Functions of Impala are as follows:

=>. The `impalad` process reads and writes to data files

=>. It logically divides a query into smaller parallel queries and distributes them to different nodes in the Impala cluster. When you submit a query to the Impala daemon running on any node, the node serves as the coordinator node for that query.

=>. Impala transmits intermediate query results back to the central coordinator node. The coordinator constructs the final query output. When you run an experiment using the `Impala-shell` command, it may connect you to the same Impala daemon process for convenience[12].

When we submit a Hive query,It maps the context to MAPReduce job which in turn jumps into resource manager .Resource manager works on data localization. Resource Manager assigns node manager to do a job. In Impala any resource manager or node manager can be connected to any Impala Daemon, just that all should have one daemon available for it. Impala was written in `c++`.But it failed as it doesn't know what serdes in any database object can be viewed in hive with external libraries as they are very hive centric libraries which cannot be used in impala, shown in Fig.8.

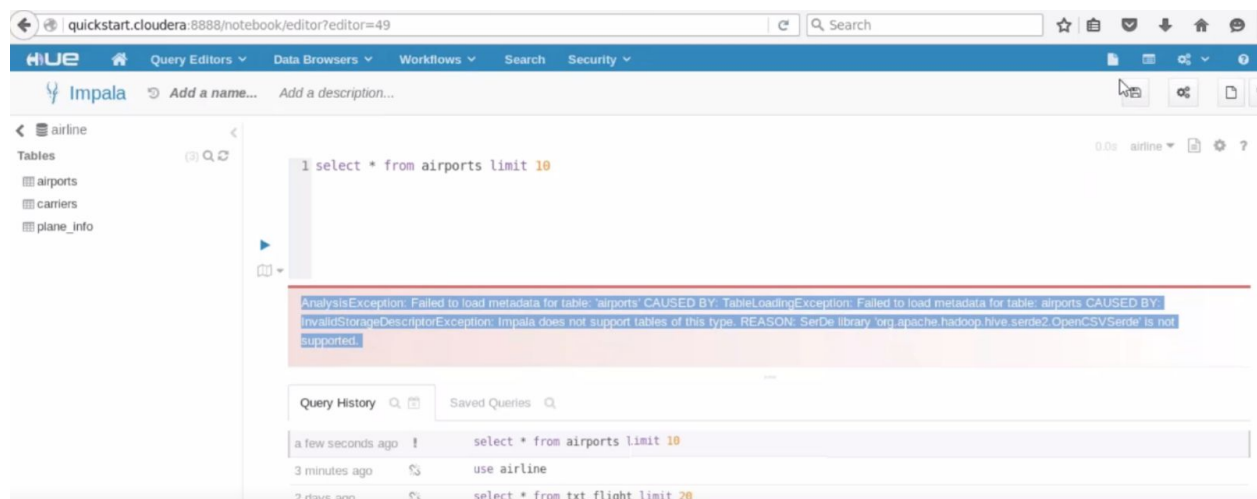


Fig.8 Error we're getting

Custom udf is created in hive.Impala supports Text,RC file Sequence file,Avro and Parquet formats. We are mainly using Parquet files. Impala helps you to create, manage, and query Parquet tables. Parquet is a column-oriented binary file format intended to be highly efficient for the types of large-scale queries that Impala is best at. Parquet is good to perform aggregation

operations such as SUM() and AVG() that need to process most or all of the values from a column. There will be tremendous improvement in queries when we use parquet[12]. Time comparison with hive and Impala for same query: we get huge difference between impala and hive. Hive takes around 3 minutes to execute while Impala around 19 seconds. Impala gives a faster response. This time can even be reduced by using compression and query optimization. It is easy to query parquet version then serdes version. Main motto is to make data available, accessible and efficient in a friendly way. Moreover, second pass of parquet is less than the second pass of txt file. Parquet Data Compression for text and parquet files : With Parquet it is 17 times reduction in speed. Every Impala queries with an output are shown in below figures.

```

cloudera@quickstart:~
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ impala-shell -i quickstart.cloudera:21000
Starting Impala Shell without Kerberos authentication
Connected to quickstart.cloudera:21000
Server version: impalad version 2.6.0-cdh5.8.0 RELEASE (build 5464d1750381b40a7e7163b12b09f11b891b4de3)
*****
Welcome to the Impala shell. Copyright (c) 2015 Cloudera, Inc. All rights reserved.
(Impala Shell v2.6.0-cdh5.8.0 (5464d17) built on Thu Jun 16 12:35:00 PDT 2016)

You can run a single query from the command line using the '-q' option.
*****
[quickstart.cloudera:21000] > use airline;
Query: use airline
[quickstart.cloudera:21000] > show tables;
Query: show tables
+-----+
| name |
+-----+
| airports |
| avro_flight |
| carriers |
| plane_info |
| pq_flight |
| txt_flight |
+-----+
Fetched 6 row(s) in 0.01s
[quickstart.cloudera:21000] >

```

Fig.9 Showing table using Impala

```

cloudera@quickstart:~
File Edit View Search Terminal Help
[quickstart.cloudera:21000] > show tables;
Query: show tables
+-----+
| name |
+-----+
| airports |
| carriers |
| plane_info |
| txt_flight |
+-----+
Fetched 4 row(s) in 0.01s
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] > select * from txt_flight limit 1;
Query: select * from txt_flight limit 1
+-----+
| year | month | dayofmonth | dayofweek | deptime | crsdeptime | arrtime | crsarrrtime | uniquecarrier | flightnum | tailnum | actualelapsedtime | crselapsedtime | airti |
me | arrdelay | depdelay | origin | dest | distance | taxiin | taxiout | cancelled | cancellationcode | diverted | carrierdelay | weatherdelay | nasdelay | securitydela |
y | lateaircraftdelay |
+-----+
| 2003 | 3 | 23 | 7 | 1406 | 1405 | 1601 | 1621 | DL | 610 | N698DL | 115 | 136 | 99 |
| -20 | 1 | ATL | PVD | 903 | 3 | 13 | 0 | | 0 | NULL | NULL | NULL | NULL |
| NULL |
+-----+
Fetched 1 row(s) in 5.85s

```

Fig.10 txt_flight output

```
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from txt_flight group by year;
Query: select year, count(1) no_of_flights from txt_flight group by year
+-----+-----+
| year | no_of_flights |
+-----+-----+
| 2008 | 7009728       |
| 2007 | 7453215       |
| 2005 | 7140596       |
| 2003 | 6488540       |
| 2006 | 7141922       |
| 2004 | 7129270       |
+-----+-----+
Fetched 6 row(s) in 18.95s
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from txt_flight group by year;
Query: select year, count(1) no_of_flights from txt_flight group by year
```

Fig.11 Impala taking around 19 s to run

```
cloudera@quickstart:~
File Edit View Search Terminal Help
+-----+-----+
| 2004 | 7129270       |
+-----+-----+
Fetched 6 row(s) in 57.16s
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from txt_flight group by year;
Query: select year, count(1) no_of_flights from txt_flight group by year
+-----+-----+
| year | no_of_flights |
+-----+-----+
| 2008 | 7009728       |
| 2007 | 7453215       |
| 2005 | 7140596       |
| 2003 | 6488540       |
| 2006 | 7141922       |
| 2004 | 7129270       |
+-----+-----+
Fetched 6 row(s) in 10.09s
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from txt_flight group by year;
Query: select year, count(1) no_of_flights from txt_flight group by year
+-----+-----+
| year | no_of_flights |
+-----+-----+
| 2008 | 7009728       |
| 2007 | 7453215       |
| 2005 | 7140596       |
| 2003 | 6488540       |
| 2006 | 7141922       |
| 2004 | 7129270       |
+-----+-----+
Fetched 6 row(s) in 8.89s
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from pq_flight group by year;
Query: select year, count(1) no_of_flights from pq_flight group by year
```

Fig.12 Output of group by

Running the query with impala version created example of Parquet files and with Parquet files time taken is sufficiently less.

```

cloudera@quickstart:~
File Edit View Search Terminal Help
| 2003 | 6488540 |
| 2006 | 7141922 |
| 2004 | 7129270 |
+-----+
Fetched 6 row(s) in 10.09s
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from txt_flight group by year;
Query: select year, count(1) no_of_flights from txt_flight group by year
+-----+
| year | no_of_flights |
+-----+
| 2008 | 7009728 |
| 2007 | 7453215 |
| 2005 | 7140596 |
| 2003 | 6488540 |
| 2006 | 7141922 |
| 2004 | 7129270 |
+-----+
Fetched 6 row(s) in 8.89s
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from pq_flight group by year;
Query: select year, count(1) no_of_flights from pq_flight group by year
+-----+
| year | no_of_flights |
+-----+
| 2008 | 7009728 |
| 2007 | 7453215 |
| 2005 | 7140596 |
| 2003 | 6488540 |
| 2006 | 7141922 |
| 2004 | 7129270 |
+-----+
Fetched 6 row(s) in 6.93s
[quickstart.cloudera:21000] >

```

Fig.13 Comparison of txt and Parquet flight file running time

The second pass of parquet is less than the second pass of the text file. Like it becomes 2.23 s and its 17 times reduction.

Parquet Data Compression for text and parquet files : With Parquet it is 17 times reduction in speed.

```

Fetched 6 row(s) in 6.93s
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from pq_flight group by year;
Query: select year, count(1) no_of_flights from pq_flight group by year
+-----+
| year | no_of_flights |
+-----+
| 2008 | 7009728 |
| 2007 | 7453215 |
| 2005 | 7140596 |
| 2003 | 6488540 |
| 2006 | 7141922 |
| 2004 | 7129270 |
+-----+
Fetched 6 row(s) in 2.23s
[quickstart.cloudera:21000] >

```

Fig.14 Output of second time of the Parquet flight file (too less)

4. Hive/Impala partitioning and clustering : Hive vs Impala Theory

Partitioning Basics :

Purpose : To get efficient spread of the data so that data does not get skewed by randomness.

By default, all the data files for a table are located in a single directory. Partitioning is a technique for physically dividing the data during loading, based on values from one or more columns, to speed up queries that test those columns[12]. Hive is used for partitioning. Two ways of partitioning dynamic and static partitioning both are being implemented on dataset to know the difference[4]. We are using Static partitioning as we get data year by year.

Dynamic partitioning is also shown for the

dataset as with dynamic partitioning we can add any number of partitions with single SQL execution[6].

Static partition : Used when user wants to specify partition name while inserting data in table and distinct values in portioned column are very few and more when data is incrementally loaded portioned on specific time. Loading : one portion is loaded at a time; Good for continuous operation; Not suitable for initial loads[5].

Dynamic partition : Used when you are bulk loading data in table and want to automatically deduce column values and used when distinct values in partitioned column are very high. Loading : Data is distributed between partitions dynamically[6].

Where to do data processing : Hive or Impala? : Normally we can do any of them Hive or Impala. But, we analyse that Impala does not have as much integration with order to Hadoop ecosystem as much as Hive.

Impala Partitioning :

Analysis : Impala is faster than Apache Hive but that does not mean that it is the one stop SQL solution for all big data problems. Impala is memory intensive and having a low latency and does not run effectively for heavy data operations like joins because it is not possible to push everything into the memory. This is when Hive comes to the rescue[12]. If an application has batch processing kind of needs over big data then organizations must opt for Hive. If they need real time processing of ad-hoc queries on subset of data then Impala is a better choice. Impala does support for Hadoop Distributed File System (HDFS) and Apache HBase.

Following features are different for Hive and Impala :



Fig.15 Impala features name different from Hive

Partitioning Calculation :

For partitioning, need efficient utilisation of name node resources and so in the cluster this selected file have to be replicated three times as shown in the picture and the size of each files

Home / user / hive / warehouse / airline.db / pq_flight

Name	Size	User	Group	Perm
.		cloudera	supergroup	drwxr
_impala_insert_staging		impala	supergroup	drwxr
df4c5b98bd9e30be-815826c26922d588_776687520_data.0.parq	252.7 MB	impala	supergroup	-rw-r--
df4c5b98bd9e30be-815826c26922d588_776687520_data.1.parq	252.7 MB	impala	supergroup	-rw-r--
df4c5b98bd9e30be-815826c26922d588_776687520_data.2.parq	221.5 MB	impala	supergroup	-rw-r--

Fig.16 Impala files have to replicated

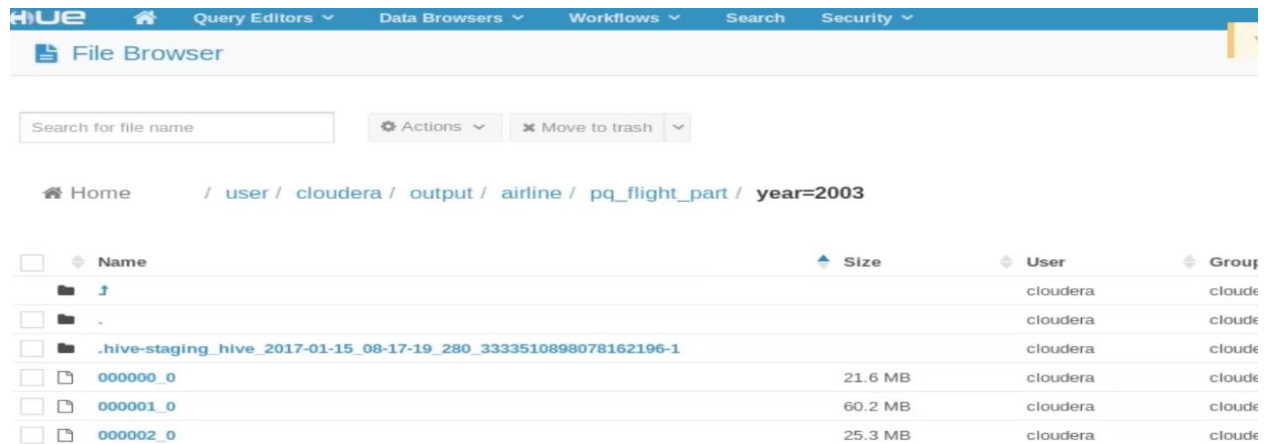
```

[quickstart.cloudera:21000] > use airline;
Query: use airline
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] > select year, count(1) from txt_flight group by year;
Query: select year, count(1) from txt_flight group by year
+----+-----+
| year | count(1) |
+----+-----+
| 2008 | 7009728 |
| 2007 | 7453215 |
| 2005 | 7140596 |
| 2003 | 6488540 |
| 2006 | 7141922 |
| 2004 | 7129270 |
+----+-----+
Fetched 6 row(s) in 105.77s
[quickstart.cloudera:21000] > select year, count(1) from pq_flight group by year;
Query: select year, count(1) from pq_flight group by year
+----+-----+
| year | count(1) |
+----+-----+
| 2008 | 7009728 |
| 2007 | 7453215 |
| 2005 | 7140596 |
| 2003 | 6488540 |
| 2006 | 7141922 |
| 2004 | 7129270 |
+----+-----+
Fetched 6 row(s) in 6.86s

```

Fig.17 Total 7 million flights

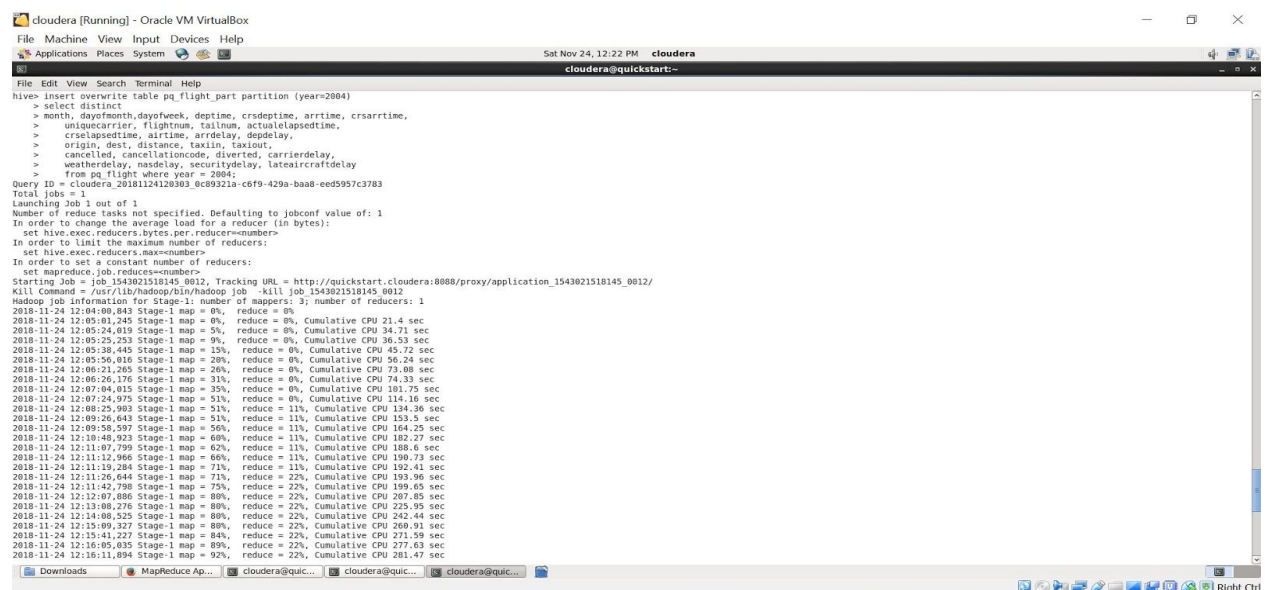
are very large. Each of these files are around 252.7 MB and so total it is around 760 MB. Now from the below figure. It can be noticed that every year total flights are 7 million and if we divide by 12, to calculate as per month then at last we are getting a data about 800 MB and it is too large so we won't be able to run query and so we are assuming that assumption for every block for each month is not good and so Hive is better choice for the project. Hive does not utilise more space, as shown in Fig.18. Moreover to reduce these three files to one, command is “set mapred.reduce.tasks=1”.



Name	Size	User	Group
.		cloudera	cloudera
.		cloudera	cloudera
.hive-staging_hive_2017-01-15_08-17-19_280_3333510898078162196-1		cloudera	cloudera
000000_0	21.6 MB	cloudera	cloudera
000001_0	60.2 MB	cloudera	cloudera
000002_0	25.3 MB	cloudera	cloudera

Fig.18 Hive files : less size than Impala

Hive Implementation for partitioning : Output are given in below pictures for the year 2003, 2006, 2007 and 2008.



```

cloudera [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Sat Nov 24, 12:22 PM cloudera
cloudera@quickstart:~$
hive> insert overwrite table pq_flight_part partition (year=2003)
> select distinct
> month, dayofmonth, dayofweek, deptime, crsdeptime, arrtime, crsarrrtime,
> uniquecarrier, flightnum, tailnum, actualelapsedtime,
> crselapsedtime, airline, arddelay, depdelay,
> origin, dest, distance, taxiin, taxiout,
> cancelled, cancellationcode, diverted, carrierdelay,
> weatherdelay, nasdelay, securitydelay, lateaircraftdelay
> from pq_flight where year = 2003;
Query ID = cloudera_20181124120303_0c09321a-c6f9-429a-ba08-eed5957c3783
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_1543021518145_0012, Tracking URL = http://quickstart.cloudera:8080/proxy/application_1543021518145_0012/
Kill Command = /usr/lib/hadoop/hive/hadoop job -kill job_1543021518145_0012
Hadoop job information for Stage-1: number of mappers: 3; number of reducers: 1
2018-11-24 12:04:00,643 Stage-1 map = 0%, reduce = 0%
2018-11-24 12:05:01,245 Stage-1 map = 0%, reduce = 0%, Cumulative CPU 21.4 sec
2018-11-24 12:05:24,019 Stage-1 map = 5%, reduce = 0%, Cumulative CPU 34.71 sec
2018-11-24 12:05:25,253 Stage-1 map = 9%, reduce = 0%, Cumulative CPU 36.53 sec
2018-11-24 12:05:30,445 Stage-1 map = 15%, reduce = 0%, Cumulative CPU 45.72 sec
2018-11-24 12:05:56,016 Stage-1 map = 20%, reduce = 0%, Cumulative CPU 56.24 sec
2018-11-24 12:06:21,265 Stage-1 map = 26%, reduce = 0%, Cumulative CPU 73.08 sec
2018-11-24 12:06:26,176 Stage-1 map = 31%, reduce = 0%, Cumulative CPU 74.33 sec
2018-11-24 12:07:04,015 Stage-1 map = 35%, reduce = 0%, Cumulative CPU 101.75 sec
2018-11-24 12:07:24,975 Stage-1 map = 51%, reduce = 0%, Cumulative CPU 114.16 sec
2018-11-24 12:08:25,903 Stage-1 map = 51%, reduce = 11%, Cumulative CPU 134.36 sec
2018-11-24 12:09:26,643 Stage-1 map = 51%, reduce = 11%, Cumulative CPU 153.5 sec
2018-11-24 12:09:58,597 Stage-1 map = 56%, reduce = 11%, Cumulative CPU 164.25 sec
2018-11-24 12:10:40,923 Stage-1 map = 60%, reduce = 11%, Cumulative CPU 182.27 sec
2018-11-24 12:11:07,799 Stage-1 map = 62%, reduce = 11%, Cumulative CPU 188.6 sec
2018-11-24 12:11:12,966 Stage-1 map = 66%, reduce = 11%, Cumulative CPU 190.73 sec
2018-11-24 12:11:13,284 Stage-1 map = 71%, reduce = 11%, Cumulative CPU 192.41 sec
2018-11-24 12:11:20,644 Stage-1 map = 71%, reduce = 22%, Cumulative CPU 193.96 sec
2018-11-24 12:11:42,788 Stage-1 map = 75%, reduce = 22%, Cumulative CPU 199.65 sec
2018-11-24 12:12:07,886 Stage-1 map = 80%, reduce = 22%, Cumulative CPU 207.85 sec
2018-11-24 12:13:06,278 Stage-1 map = 80%, reduce = 22%, Cumulative CPU 225.95 sec
2018-11-24 12:14:00,525 Stage-1 map = 80%, reduce = 22%, Cumulative CPU 242.44 sec
2018-11-24 12:15:09,327 Stage-1 map = 80%, reduce = 22%, Cumulative CPU 260.91 sec
2018-11-24 12:15:41,227 Stage-1 map = 84%, reduce = 22%, Cumulative CPU 271.59 sec
2018-11-24 12:16:05,635 Stage-1 map = 89%, reduce = 22%, Cumulative CPU 277.63 sec
2018-11-24 12:16:11,894 Stage-1 map = 92%, reduce = 22%, Cumulative CPU 281.47 sec

```

Fig.19 Query to reduce three files to one for data of year 2003


```
cloudera [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Sat Nov 24, 1:40 PM cloudera
cloudera@quickstart:~$
File Edit View Search Terminal Help
2018-11-24 13:36:48,118 Stage-1 map = 100%, reduce = 86%, Cumulative CPU 338.55 sec
2018-11-24 13:36:54,645 Stage-1 map = 100%, reduce = 87%, Cumulative CPU 341.2 sec
2018-11-24 13:37:09,151 Stage-1 map = 100%, reduce = 88%, Cumulative CPU 343.84 sec
2018-11-24 13:37:12,158 Stage-1 map = 100%, reduce = 89%, Cumulative CPU 349.19 sec
2018-11-24 13:37:18,688 Stage-1 map = 100%, reduce = 90%, Cumulative CPU 351.85 sec
2018-11-24 13:37:25,219 Stage-1 map = 100%, reduce = 91%, Cumulative CPU 354.52 sec
2018-11-24 13:37:37,255 Stage-1 map = 100%, reduce = 92%, Cumulative CPU 359.96 sec
2018-11-24 13:37:42,788 Stage-1 map = 100%, reduce = 93%, Cumulative CPU 363.12 sec
2018-11-24 13:37:54,846 Stage-1 map = 100%, reduce = 94%, Cumulative CPU 365.54 sec
2018-11-24 13:38:01,455 Stage-1 map = 100%, reduce = 95%, Cumulative CPU 371.22 sec
2018-11-24 13:38:13,438 Stage-1 map = 100%, reduce = 96%, Cumulative CPU 376.52 sec
2018-11-24 13:38:18,991 Stage-1 map = 100%, reduce = 97%, Cumulative CPU 379.19 sec
2018-11-24 13:38:25,536 Stage-1 map = 100%, reduce = 98%, Cumulative CPU 381.84 sec
2018-11-24 13:38:37,488 Stage-1 map = 100%, reduce = 99%, Cumulative CPU 387.2 sec
2018-11-24 13:38:45,168 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 390.8 sec
MapReduce Total cumulative CPU time: 6 minutes 30 seconds 880 msec
Ended Job = job_1543821518145_0015
Loading data to table airline.pq.flight part partition (year=2006)
Partition airline.pq.flight part(year=2006) stats: (numFiles=1, numRows=7141915, totalSize=163172934, rawDataSize=199973620)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3 Reduce: 1 Cumulative CPU: 390.8 sec HDFS Read: 823731029 HDFS Write: 163173034 SUCCESS
Total MapReduce CPU Time Spent: 6 minutes 30 seconds 880 msec
OK
Time taken: 1841.486 seconds
hive> select * from pq.flight part where year=2006 limit 20;
OK
1 1 1 NULL 100 NULL 605 MW 166 N351NB NULL 185 NULL NULL NULL SEA MSP 1399 0 0 1 A 0 0 0 0 0 0 2006
1 1 1 NULL 545 NULL 631 YV 2892 N8393J NULL 46 NULL NULL NULL CFE CLT 88 0 0 1 A 0 0 0 0 0 0 2006
1 1 1 NULL 600 NULL 722 UA 353 000000 NULL 152 NULL NULL NULL ORD DEN 888 0 0 1 A 0 0 0 0 0 0 2006
1 1 1 NULL 680 NULL 737 UA 1555 000000 NULL 157 NULL NULL NULL MDW DEN 895 0 0 1 C 0 0 0 0 0 0 2006
1 1 1 NULL 600 NULL 742 UA 1834 000000 NULL 182 NULL NULL NULL PHX DEN 602 0 0 1 A 0 0 0 0 0 0 2006
1 1 1 NULL 600 NULL 815 EV 4468 N881AS NULL 75 NULL NULL NULL GPT ATL 352 0 0 1 B 0 0 0 0 0 0 2006
1 1 1 NULL 610 NULL 819 OO 6674 0 NULL 131 NULL NULL NULL FAT DEN 844 0 0 1 A 0 0 0 0 0 0 2006
1 1 1 NULL 610 NULL 756 UA 518 000000 NULL 186 NULL NULL NULL SFO PDX 558 0 0 1 A 0 0 0 0 0 0 2006
1 1 1 NULL 612 NULL 728 UA 162 000000 NULL 76 NULL NULL NULL SFO LAX 337 0 0 1 A 0 0 0 0 0 0 2006
1 1 1 NULL 615 NULL 1218 UA 1236 000000 NULL 243 NULL NULL NULL OAK ORD 1835 0 0 1 A 0 0 0 0 0 0 2006
1 1 1 NULL 630 NULL 748 US 235 0 NULL 78 NULL NULL NULL PHL PIT 267 0 0 1 A 0 0 0 0 0 0 2006
1 1 1 NULL 630 NULL 834 DL 1484 N918DL NULL 64 NULL NULL NULL MDR ATL 362 0 0 1 B 0 0 0 0 0 0 2006
1 1 1 NULL 630 NULL 828 UA 297 000000 NULL 358 NULL NULL NULL BMT SFO 2437 0 0 1 A 0 0 0 0 0 0 2006
1 1 1 NULL 630 NULL 1080 UA 163 000000 NULL 398 NULL NULL NULL BOS LAX 2611 0 0 1 A 0 0 0 0 0 0 2006
1 1 1 NULL 635 NULL 758 NW 391 N526NW NULL 75 NULL NULL NULL GPT MDM 324 0 0 1 B 0 0 0 0 0 0 2006
1 1 1 NULL 635 NULL 941 UA 526 000000 NULL 126 NULL NULL NULL ORD BOS 887 0 0 1 A 0 0 0 0 0 0 2006
1 1 1 NULL 640 NULL 725 OO 6618 0 NULL 45 NULL NULL NULL COS DEN 72 0 0 1 A 0 0 0 0 0 0 2006
1 1 1 NULL 643 NULL 928 OO 3918 0 NULL 97 NULL NULL NULL PSP SLC 541 0 0 1 A 0 0 0 0 0 0 2006
Time taken: 0.159 seconds, Fetched: 20 row(s)
hive>
```

Fig.20 Output for year 2006 - Mapreduce

```
cloudera [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Sat Nov 24, 2:07 PM cloudera
cloudera@quickstart:~$
File Edit View Search Terminal Help
2018-11-24 14:04:19,837 Stage-1 map = 100%, reduce = 86%, Cumulative CPU 333.0 sec
2018-11-24 14:04:25,356 Stage-1 map = 100%, reduce = 87%, Cumulative CPU 335.7 sec
2018-11-24 14:04:37,357 Stage-1 map = 100%, reduce = 88%, Cumulative CPU 341.03 sec
2018-11-24 14:04:43,873 Stage-1 map = 100%, reduce = 89%, Cumulative CPU 343.66 sec
2018-11-24 14:04:49,342 Stage-1 map = 100%, reduce = 90%, Cumulative CPU 346.32 sec
2018-11-24 14:05:01,477 Stage-1 map = 100%, reduce = 91%, Cumulative CPU 351.65 sec
2018-11-24 14:05:08,000 Stage-1 map = 100%, reduce = 92%, Cumulative CPU 354.17 sec
2018-11-24 14:05:19,917 Stage-1 map = 100%, reduce = 93%, Cumulative CPU 359.54 sec
2018-11-24 14:05:26,425 Stage-1 map = 100%, reduce = 94%, Cumulative CPU 362.24 sec
2018-11-24 14:05:31,924 Stage-1 map = 100%, reduce = 95%, Cumulative CPU 364.92 sec
2018-11-24 14:05:44,951 Stage-1 map = 100%, reduce = 96%, Cumulative CPU 370.57 sec
2018-11-24 14:05:50,406 Stage-1 map = 100%, reduce = 97%, Cumulative CPU 373.22 sec
2018-11-24 14:06:02,438 Stage-1 map = 100%, reduce = 98%, Cumulative CPU 376.61 sec
2018-11-24 14:06:08,977 Stage-1 map = 100%, reduce = 99%, Cumulative CPU 381.25 sec
2018-11-24 14:06:19,798 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 386.17 sec
MapReduce Total cumulative CPU time: 6 minutes 26 seconds 178 msec
Ended Job = job_1543821518145_0016
Loading data to table airline.pq.flight part partition (year=2007)
Partition airline.pq.flight part(year=2007) stats: (numFiles=1, numRows=7453188, totalSize=171003054, rawDataSize=2086899264)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3 Reduce: 1 Cumulative CPU: 386.17 sec HDFS Read: 823731029 HDFS Write: 171003154 SUCCESS
Total MapReduce CPU Time Spent: 6 minutes 26 seconds 178 msec
OK
Time taken: 1502.529 seconds
hive> select * from pq.flight part where year=2007 limit 20;
OK
1 1 1 NULL 510 NULL 553 AQ 82 0 NULL 43 NULL NULL NULL HNL KOA 163 0 0 1 A 0 0 0 0 0 0 2007
1 1 1 NULL 545 NULL 988 YV 2821 N914FJ NULL 143 NULL NULL NULL EUG PHX 952 0 0 1 B 0 0 0 0 0 0 2007
1 1 1 NULL 600 NULL 760 AS 30 N742AS NULL 60 NULL NULL NULL ANC ADO 252 0 0 1 B 0 0 0 0 0 0 2007
1 1 1 NULL 600 NULL 715 MQ 4682 0 NULL 75 NULL NULL NULL BOS JFK 187 0 0 1 A 0 0 0 0 0 0 2007
1 1 1 NULL 600 NULL 750 MQ 4446 0 NULL 110 NULL NULL NULL ORD LIT 552 0 0 1 A 0 0 0 0 0 0 2007
1 1 1 NULL 605 NULL 753 9E 2807 0 NULL 113 NULL NULL NULL SWF DTW 479 0 0 1 A 0 0 0 0 0 0 2007
1 1 1 NULL 600 NULL 834 OH 5111 0 NULL 154 NULL NULL NULL MHT CVG 741 0 0 1 A 0 0 0 0 0 0 2007
1 1 1 NULL 600 NULL 840 AA 1430 0 NULL 180 NULL NULL NULL ABQ DFW 569 0 0 1 B 0 0 0 0 0 0 2007
1 1 1 NULL 605 NULL 650 MQ 4126 0 NULL 105 NULL NULL NULL CHA ORD 501 0 0 1 A 0 0 0 0 0 0 2007
1 1 1 NULL 605 NULL 731 9E 5710 0 NULL 86 NULL NULL NULL PMS MEM 355 0 0 1 C 0 0 0 0 0 0 2007
1 1 1 NULL 605 NULL 750 AA 1729 0 NULL 165 NULL NULL NULL LGA STL 887 0 0 1 A 0 0 0 0 0 0 2007
1 1 1 NULL 605 NULL 754 9E 2938 0 NULL 109 NULL NULL NULL ABE DTW 424 0 0 1 A 0 0 0 0 0 0 2007
1 1 1 NULL 610 NULL 714 OO 5781 0 NULL 64 NULL NULL NULL SBA SJC 234 0 0 1 A 0 0 0 0 0 0 2007
1 1 1 NULL 610 NULL 723 9E 3793 0 NULL 73 NULL NULL NULL PLN DTW 243 0 0 1 A 0 0 0 0 0 0 2007
1 1 1 NULL 610 NULL 732 US 201 N667AW NULL 82 NULL NULL NULL AAR PHX 328 0 0 1 B 0 0 0 0 0 0 2007
1 1 1 NULL 610 NULL 739 9E 2934 0 NULL 29 NULL NULL NULL SCE DTW 800 0 0 1 A 0 0 0 0 0 0 2007
1 1 1 NULL 610 NULL 908 9E 3758 0 NULL 118 NULL NULL NULL DSM DTW 534 0 0 1 A 0 0 0 0 0 0 2007
1 1 1 NULL 610 NULL 931 AA 495 0 NULL 141 NULL NULL NULL MIA SJU 1045 0 0 1 A 0 0 0 0 0 0 2007
1 1 1 NULL 615 NULL 700 MQ 3118 0 NULL 45 NULL NULL NULL SAN LAX 109 0 0 1 A 0 0 0 0 0 0 2007
1 1 1 NULL 615 NULL 705 MQ 4258 0 NULL 50 NULL NULL NULL DBO ORD 147 0 0 1 A 0 0 0 0 0 0 2007
Time taken: 0.202 seconds, Fetched: 20 row(s)
hive>
```

Fig.21 Output for year 2007 - Mapreduce

```
cloudera [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Sat Nov 24, 2:30 PM cloudera
cloudera@quickstart:~$
File Edit View Search Terminal Help
2018-11-24 14:21:06,018 Stage-1 map = 100%, reduce = 88%, Cumulative CPU 315.38 sec
2018-11-24 14:21:13,387 Stage-1 map = 100%, reduce = 89%, Cumulative CPU 318.02 sec
2018-11-24 14:21:18,883 Stage-1 map = 100%, reduce = 90%, Cumulative CPU 320.67 sec
2018-11-24 14:21:25,388 Stage-1 map = 100%, reduce = 91%, Cumulative CPU 323.31 sec
2018-11-24 14:21:37,337 Stage-1 map = 100%, reduce = 92%, Cumulative CPU 326.04 sec
2018-11-24 14:21:42,776 Stage-1 map = 100%, reduce = 93%, Cumulative CPU 331.3 sec
2018-11-24 14:21:49,467 Stage-1 map = 100%, reduce = 94%, Cumulative CPU 334.8 sec
2018-11-24 14:22:01,464 Stage-1 map = 100%, reduce = 95%, Cumulative CPU 339.48 sec
2018-11-24 14:22:07,996 Stage-1 map = 100%, reduce = 96%, Cumulative CPU 342.15 sec
2018-11-24 14:22:13,438 Stage-1 map = 100%, reduce = 97%, Cumulative CPU 344.84 sec
2018-11-24 14:22:19,948 Stage-1 map = 100%, reduce = 98%, Cumulative CPU 347.49 sec
2018-11-24 14:22:32,102 Stage-1 map = 100%, reduce = 99%, Cumulative CPU 352.76 sec
2018-11-24 14:22:37,629 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 355.48 sec
MapReduce Total cumulative CPU time: 5 minutes 55 seconds 970 msec
Ended Job = job_1543821518145_0017
Loading data to table airline.pq_flight_part partition (year=2008)
Partition airline.pq_flight_part(year=2008) stats: (numFiles=1, numRows=7009724, totalSize=158052022, rawDataSize=19627272)
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3 Reduce: 1 Cumulative CPU: 355.97 sec
Total MapReduce CPU Time Spent: 5 minutes 55 seconds 970 msec
OK
Time taken: 810.256 seconds
hive> select * from pq_flight_part where year=2008 limit 2;
FAILED: ParseException line 1:53 extraneous input '*' expecting EOF near '<EOF>'
hive> select * from pq_flight_part where year=2008 limit 20;
OK
1 1 2 NULL 295 NULL 620 AS 154 N309AS NULL 195 NULL NULL NULL ANC SEA 1449 1 A 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 595 NULL 620 9E 2880 NULL 75 NULL NULL NULL CID MSP 221 1 A 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 540 NULL 983 YV 2816 N982FJ NULL 143 NULL NULL NULL EUG PHX 952 1 B 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 550 NULL 907 YV 2841 N77286 NULL 137 NULL NULL NULL HFR PHX 854 1 B 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 600 NULL 609 UA 1285 NULL 69 NULL NULL NULL DTW ORD 235 1 B 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 600 NULL 630 YV 7203 N651ML NULL 98 NULL NULL NULL CAK ORD 344 1 A 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 600 NULL 717 OO 6749 N975SW NULL 77 NULL NULL NULL RAP DEN 381 1 A 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 600 NULL 728 YV 7186 N456YV NULL 88 NULL NULL NULL CPR DEN 238 1 A 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 600 NULL 750 9E 3753 NULL 110 NULL NULL NULL FNT LGA 530 1 B 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 600 NULL 753 UA 1245 NULL 233 NULL NULL NULL IAD DEN 1452 1 A 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 600 NULL 805 MO 4181 NULL 125 NULL NULL NULL OKC ORD 693 1 A 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 600 NULL 851 YV 2684 N933LR NULL 111 NULL NULL NULL MSY CLT 651 1 A 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 600 NULL 909 OO 6820 N946SW NULL 129 NULL NULL NULL BFL DEN 845 1 A 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 600 NULL 914 UA 1438 NULL 134 NULL NULL NULL ONT DEN 819 1 A 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 600 NULL 1157 UA 618 NULL 237 NULL NULL NULL LAX ORD 1745 1 A 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 605 NULL 706 YV 7170 N511MJ NULL 61 NULL NULL NULL ORF IAD 157 1 C 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 605 NULL 725 9E 5889 NULL 88 NULL NULL NULL LEX DTW 296 1 A 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 605 NULL 752 UA 739 NULL 167 NULL NULL NULL DEN SFO 967 1 A 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 605 NULL 755 MO 3954 NULL 110 NULL NULL NULL MEM ORD 491 1 B 0 NULL NULL NULL NULL NULL 2008
1 1 2 NULL 605 NULL 983 9E 5827 NULL 118 NULL NULL NULL DLH DTW 541 1 A 0 NULL NULL NULL NULL NULL 2008
Time taken: 0.12 seconds, Fetched: 20 row(s)
hive>
```

Fig.22 Output for year 2008 - Mapreduce

Clustering:

Hive is used for clustering. Clustering is used to do sampling and for bucket sort/join. Data sampling is done on carrier, origin and time using bucketed tables. Bucketed tables are fantastic in that they allow much more efficient sampling than do non-bucketed tables, and they may allow for time saving operations such as map side joins. Flow of this method is to create first small buckets and put some values in those buckets. Those values are called as samples and from each sample by particular value we can make cluster for the data sets. If we don't do bucketing then, it will be go with random sampling and clustering.

To know a reason of delay flight, in our project can do sample by year and carrier and for that query is given and if we do sampling by country we can get efficient sampling.

> clustered by (uniquecarrier, year) into 6 buckets

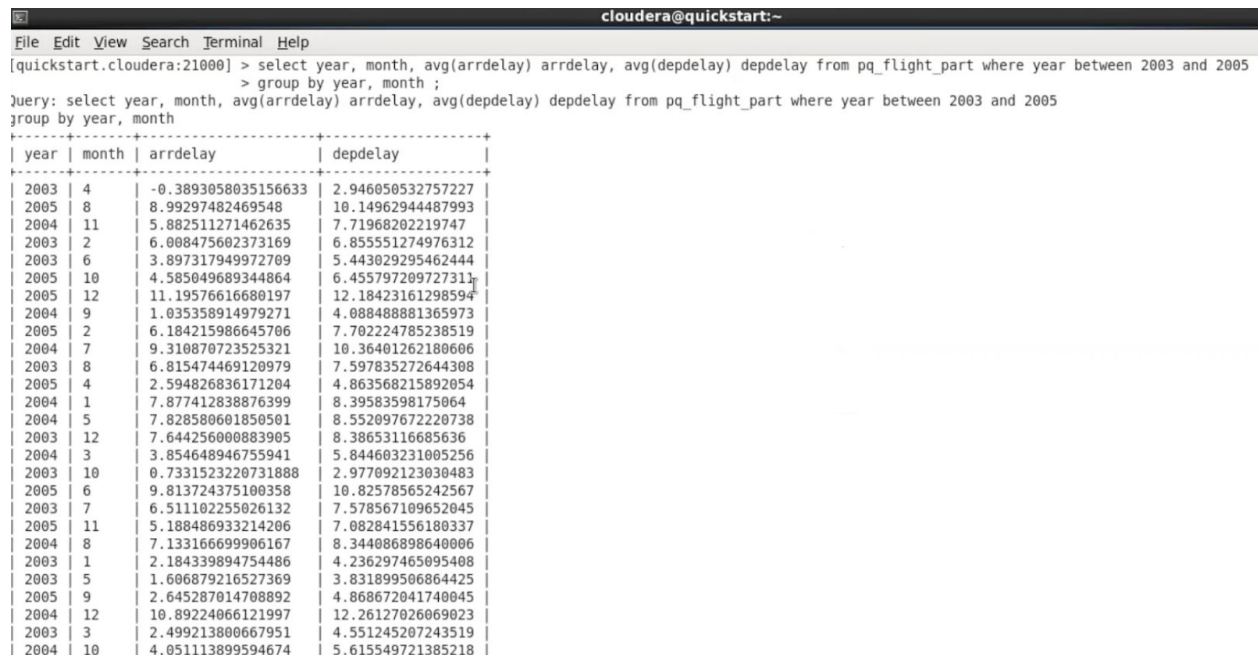
Impala does not support sampling and Clustering.

5. Data Compression, tuning and query optimization

=>. Hive :

Data Compression : Hive uses MapReduce and so data is compressed using Hive to save space on disk and network. After the Hive finishes the query execution, the result is submitted to the JobTracker, which resides on YARN. The JobTracker consists of Map/Reduce tasks which runs the mapper and reducer job to store the final result in the HDFS. The Map task

deserializes(reading) the data from the HDFS and the Reduce task serializes(writing) the data as the result of the Hive query[8]. Map Reduce is the framework used to process the data which is stored in the HDFS, here java native language is used to writing Mapreduce programs. Hive is a batch processing framework. This component process the data using a language called Hive Query Language(HQL). Hive prevents writing MapReduce programs in Java. Instead one can use SQL like language to do their daily tasks. For HIVE there is no process to communicate Map/Reduce tasks directly. It communicates with Job tracker(Application Master in YARN) only for job processing related things once it got scheduled.



```

cloudera@quickstart:~
File Edit View Search Terminal Help
[quickstart.cloudera:21000] > select year, month, avg(arrdelay) arrdelay, avg(depdelay) depdelay from pq_flight_part where year between 2003 and 2005
> group by year, month ;
Query: select year, month, avg(arrdelay) arrdelay, avg(depdelay) depdelay from pq_flight_part where year between 2003 and 2005
group by year, month

```

year	month	arrdelay	depdelay
2003	4	-0.3893058035156633	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	4.088488881365973
2005	2	6.184215986645706	7.702224785238519
2004	7	9.310870723525321	10.36401262180606
2003	8	6.815474469120979	7.597835272644308
2005	4	2.594826836171204	4.863568215892054
2004	1	7.877412838876399	8.39583598175064
2004	5	7.828580601850501	8.552097672220738
2003	12	7.644256000883905	8.38653116685636
2004	3	3.854648946755941	5.844603231005256
2003	10	0.7331523220731888	2.977092123030483
2005	6	9.813724375100358	10.82578565242567
2003	7	6.511102255026132	7.578567109652045
2005	11	5.188486933214206	7.082841556180337
2004	8	7.133166699906167	8.344086898640006
2003	1	2.184339894754486	4.236297465095408
2003	5	1.606879216527369	3.831899506864425
2005	9	2.645287014708892	4.868672041740045
2004	12	10.89224066121997	12.26127026069023
2003	3	2.499213800667951	4.551245207243519
2004	10	4.051113899594674	5.615549721385218

Fig.23 Output of flight delay from year 2003 to 2005

=>. Impala :

COMPUTE STATS to improve query performance while performing joins.

File Formats : Statistic is computed on various impala file formats to compare performance. We compute on six files, we create a portion and add data to the partition. All queries and output are shown in pictures below where when you compute a statistics we can know about the numbers of columns and rows. Moreover, by HDFS caching we can also improve performance[12].

Output for Impala execution are given in below pictures including compute stats and describe.

```

alter table pq_flight_part add partition(year=2003)
alter table pq_flight_part add partition(year=2004)
alter table pq_flight_part add partition(year=2005)
alter table pq_flight_part add partition(year=2006)
alter table pq_flight_part add partition(year=2007)
alter table pq_flight_part add partition(year=2008)

set mapred.reduce.tasks=1]
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

insert overwrite table pq_flight_part partition (year)
select
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, year
from pq_flight where year != 2003

```

Fig.24 Creating and Adding all six files

```

cloudera@quickstart:~
File Edit View Search Terminal Help
| 2003 | 11 | 4.716544401284412 | 5.758203332027559 |
| 2005 | 7 | 13.85383111122728 | 14.29684095683184 |
| 2004 | 2 | 5.953804922643387 | 7.049601851596726 |
| 2004 | 6 | 11.3532444130866 | 11.48183952992779 |
| 2005 | 3 | 7.621900809412867 | 9.150015394215398 |
| 2003 | 9 | 0.9105020298429279 | 2.761916299855968 |
| 2005 | 5 | 3.039095726310181 | 5.274917272242066 |
+-----+
Fetched 36 row(s) in 1.69s
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] > compute stats pq_flight_part;
Query: compute stats pq_flight_part
WARNINGS:
Failed to open HDFS file hdfs://quickstart.cloudera:8020/user/cloudera/output/airline/pq_flight_part/year=2004/000001_0
Error(2): No such file or directory

[quickstart.cloudera:21000] > invalidate metadata;
Query: invalidate metadata

Fetched 0 row(s) in 4.03s
[quickstart.cloudera:21000] > compute stats pq_flight_part;
Query: compute stats pq_flight_part
+-----+
| summary |
+-----+
| Updated 6 partition(s) and 28 column(s). |
+-----+
Fetched 1 row(s) in 33.78s
[quickstart.cloudera:21000] > describe formatted pq_flight_part;

```

Fig.25 Execution Part


```

cloudera@quickstart:~
File Edit View Search Terminal Help
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
42363271 | 3 | 726.85MB | NOT CACHED | NOT CACHED | PARQUET | false | hdfs://quickstart.cloudera:8020/user/hive/wa
it |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
Fetched 1 row(s) in 0.01s
[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 | Location
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
2003 | 6488540 | 3 | 107.12MB | NOT CACHED | NOT CACHED | PARQUET | false | hdfs://quickstart.cloudera:8020/user
flight_part/year=2003 |
2004 | 7129260 | 1 | 155.95MB | NOT CACHED | NOT CACHED | PARQUET | false | hdfs://quickstart.cloudera:8020/user
flight_part/year=2004 |
2005 | 7140596 | 3 | 133.18MB | NOT CACHED | NOT CACHED | PARQUET | false | hdfs://quickstart.cloudera:8020/user
flight_part/year=2005 |
2006 | 7141922 | 3 | 136.93MB | NOT CACHED | NOT CACHED | PARQUET | false | hdfs://quickstart.cloudera:8020/user
flight_part/year=2006 |
2007 | 7453215 | 3 | 143.88MB | NOT CACHED | NOT CACHED | PARQUET | false | hdfs://quickstart.cloudera:8020/user
flight_part/year=2007 |
2008 | 7009728 | 3 | 131.34MB | NOT CACHED | NOT CACHED | PARQUET | false | hdfs://quickstart.cloudera:8020/user
flight_part/year=2008 |
Total | 42363261 | 16 | 808.40MB | 0B | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
Fetched 7 row(s) in 0.01s
[quickstart.cloudera:21000] >

```

Fig.28 Including number of rows

6. Using database views to represent data

Purpose : Purpose of viewing data is to project and to hide complexities and security. Security because of computing some airplane accident. All command for viewing a data and outputs for that are given in below figures.

```

[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] > show tables;
Query: show tables
+-----+
| name |
+-----+
| airports |
| carriers |
| plane info |
| pq_airports |
| pq_carriers |
| pq_flight |
| pq_flight_part |
| pq_plane_info |
| txt_flight |
+-----+
Fetched 9 row(s) in 0.01s
[quickstart.cloudera:21000] >

```

Fig.29 Showing Tables

```

Fetched 1 row(s) in 5.22s
[quickstart.cloudera:21000] > compute stats pq_airports
> ;
Query: compute stats pq_airports
comput+-----+
| summary |
+-----+
| Updated 1 partition(s) and 7 column(s). |
+-----+
Fetched 1 row(s) in 1.32s
[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 5.19s
[quickstart.cloudera:21000] > compute stats pq_plane_info;
Query: compute stats pq_plane_info

```

Fig.30 Computing stats of tables

```

cloudera@quickstart:~
File Edit View Search Terminal Help
+-----+
| Updated 1 partition(s) and 2 column(s). |
+-----+
Fetched 1 row(s) in 5.19s
[quickstart.cloudera:21000] > compute stats pq_plane_info;
Query: compute stats pq_plane_info
+-----+
| summary |
+-----+
| Updated 1 partition(s) and 8 column(s). |
+-----+
Fetched 1 row(s) in 5.58s
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] > desc pq_airports;
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 0.03s
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] > desc pq_carriers

```

Fig.31 Describing Table

From above table we can write any query to view data differently. Query example is shown in Fig.32, which will add more table and that's shown in Fig.33. After that need to compute all tables again and this query to view is for security purpose that you can identify airplane accident immediately.

quickstart.cloudera:8888/notebook/editor?editor=97

HUE Query Editors Data Browsers Workflows Search Security

Hive Add a name... Add a description...

airline

Tables (10) Q

- airports
- carriers
- plane_info
- pq_airports
- pq_carriers
- pq_flight
- pq_flight_part
- pq_plane_info
- txt_flight
- v_flights_denom

v_flights_denom

```

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 origin_country
9        ,pad.airport dest airport, pad.city dest_city, pad.state dest_state, pad.country dest_country
10       ,pf.uniquecarrier, pc.description carrier
11       ,pf.tailnum, ppi.type plane_type, ppi.manufacturer, ppi.issue_date, ppi.model, ppi.status, ppi.registration
12       from pq_flight pf
13 join pq_airports pao on pao.iata = pf.origin
14 join pq_carriers pc on pc.edde = pf.uniquecarrier
15 join pq_plane_info ppi on ppi.tailnum = pf.tailnum

```

INFO : Starting task [Stage-0:DDL] in serial mode
INFO : Completed executing command(queryId=hive_20170115093838_4565124c-7874-4b2a-bedd-a55926257e52); Time taken for query: 0.01s
INFO : OK

Fig.32 Query to join tables differently

```

[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] > invalidate metadata;
Query: invalidate metadata

Fetched 0 row(s) in 4.34s
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] > show tables;
Query: show tables
+-----+
| name |
+-----+
| airports |
| carriers |
| plane_info |
| pq_airports |
| pq_carriers |
| pq_flight |
| pq_flight_part |
| pq_plane_info |
| txt_flight |
| v_flights_denom |
+-----+
Fetched 10 row(s) in 0.01s
[quickstart.cloudera:21000] >

```

Fig.33 One more table add after

7. Visualizing data using Microsoft Excel

In this project we visualized details about airplane delay in Microsoft Excel graph which is shown below. And from this visualization we analyzed that normally for every arrival delay, departure was also delayed. So in our graph the number of flights for arrival and departure

delay were same for most of the points according to data. As per dataset considered, we also observed that it was not the case for one point wherein the number of flights for both delay were different and it is for some number of flights where arrival delay happened but departure was not delayed and vice versa which can be seen in Fig.35.

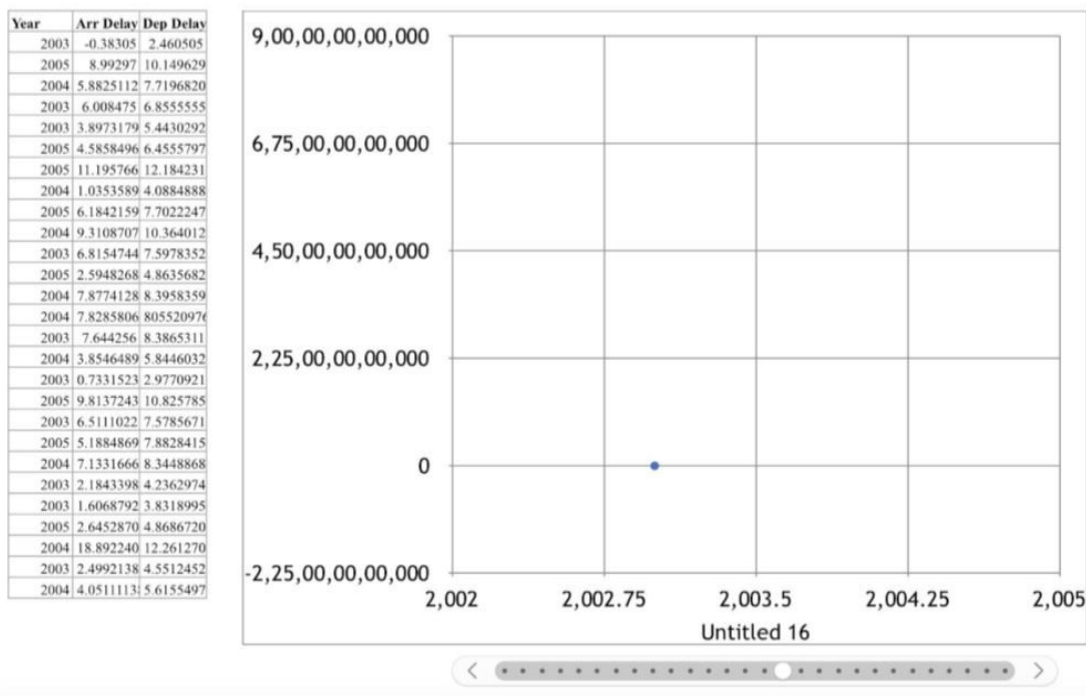


Fig.34 Arrival and Departure delays most of the time at same point

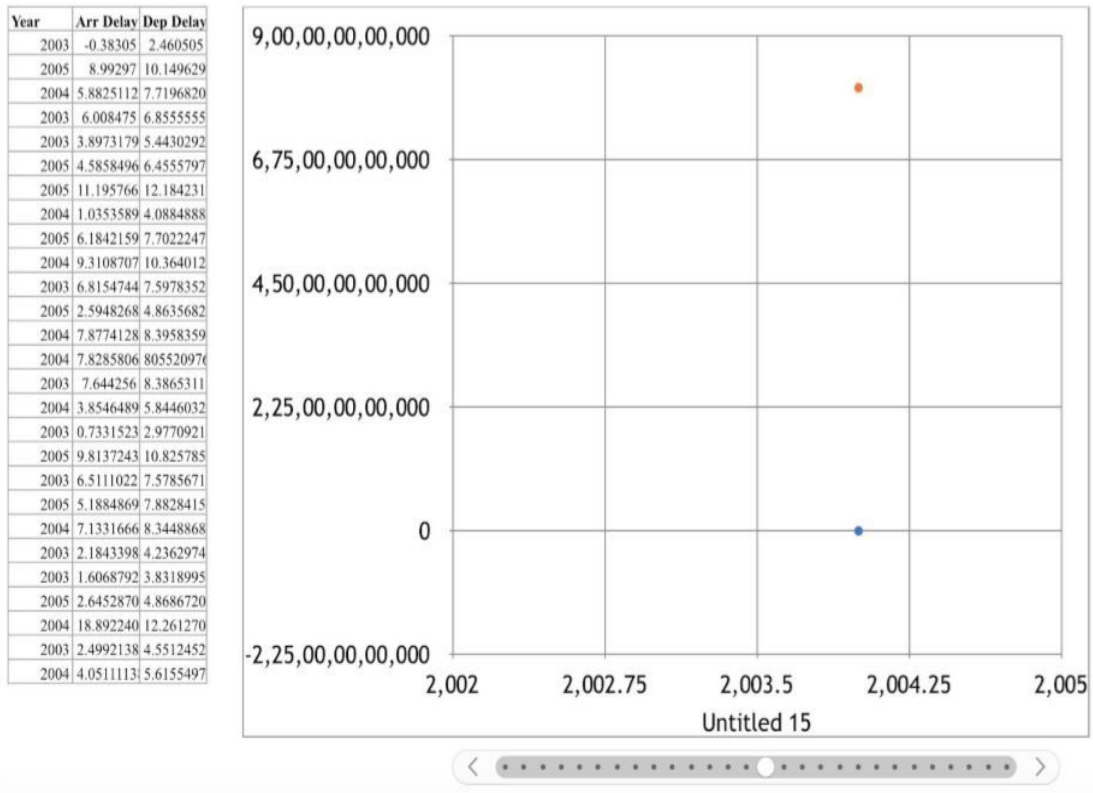


Fig.35 Arrival and Departure delays are for different num of flight

CONCLUSION

This practicum is all about the analysis of airline dataset. The main purpose of the practicum was - after preprocessing the data sets we summarized all tables and also viewed data based on various problem statements differently. Moreover, we compared queries and output from Hive and Impala and analyzed delays of flights. In conclusion, we analyzed that Impala does not have as much integration with order to Hadoop ecosystem as much as Hive and we assume that Impala does not support HDFS and Hbase. But, we have seen that Impala's - compute stats feature was used to improve performance; an execution time for Impala is very less than Hive execution time. From the visualization part, we finally concluded that for most of the time whenever arrival delays happened, departure was delayed too except some number of flights, when it was not the case as per the dataset considered.

REFERENCES

- [1]<https://www.cloudera.com/>
- [2]<http://stat-computing.org/dataexpo/2009/>
- [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#Language_Manual+DDL-Dynamic Partitions](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#Language_Manual+DDL-Dynamic_Partitions)
- [7]<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL+BucketedTables>
- [8]https://www.cloudera.com/documentation/enterprise/5-9-x/topics/introduction_compression.html
- [9]https://www.cloudera.com/documentation/enterprise/5-9-x/topics/impala_compute_stats.html
- [10]http://hadoopilluminated.com/hadoop_illuminated/hadoop-illuminated.pdf
- [11]http://cidrdb.org/cidr2015/Papers/CIDR15_Paper28.pdf
- [12]<https://www.cloudera.com/documentation/enterprise/5-5-x/PDF/cloudera-impala.pdf>