

PROJECT REPORT

Under Guidance of

Mr. Kuldeep Singh

Hadoop Developer and Trainer

At

CETPA Pvt. Ltd

D-58, Red FM Road, D Block,

Sector 2, Noida

Uttar Pradesh - 201301

Project Title

Aviation Data Analysis

Submitted By

Shiva Dev

Roll : 2015ETCS024

Enroll No : (AU/2015/02/0076)

4th Year CSE Department

Name and Address of the Institution

Adamas University

Barasat-Barrackpore Road, Barbaria

P.O.-Jagannathpur, District-24 Parganas(North)

Kolkata, West Bengal - 700126

Date of Submission

31/07/2018

Abstract

In airline environments, information is collected for archival and historical purposes from a wide variety of sources. This creates a repository of data that is, perhaps, the most valuable asset for an airline. However, the challenges of managing and extracting information to aid an airline is increasingly difficult to resolve due to customer centricity. In other words, effective data analytics helps in analyzing the data of any business system. But it is the big data which helps and axial rates the process of analysis of data paving way for a success of any business intelligence system. With the expansion of the industry, the data of the industry also expands. Then, it is increasingly difficult to handle huge amount of data that gets generated no matter what's the business is like, range of fields from social media to finance, flight data. In this paper we uses an open source solution to handle such Big Data called hadoop which is used to gathered, stored, processed and analyzed it to turn the raw data information to support decision making. We can analyze the airlines data by apache hive and pig and also compare the performance of these two.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Mr. Kuldeep Singh of the department Hadoop Developer & Trainer, whose role as project guide was invaluable for the project.

We are extremely thankful for the keen interest he took in advising us, for the books and reference materials provided for the moral support extended to us.

I am highly indebted to Mr. Kuldeep Singh for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards my parents & member of CETPA Pvt. Ltd for their kind co-operation and encouragement which help me in completion of this project.

I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

.....

(Shiva Dev)

School of Engg. and Technology, B.Tech, CS

Contents

| | |
|---|----|
| 1. Abstraction | 1 |
| 2. Acknowledgement | 2 |
| 3. Contents | 3 |
| 4. Introduction | 4 |
| 5. Literature Review | 6 |
| 6. Current State | 7 |
| 7. BigData | 8 |
| 7.1 Volume | |
| 7.2 Velocity | |
| 7.3 Variety | |
| 7.4 Value | |
| 7.5 Veracity | |
| 8. Hadoop | 10 |
| 8.1 Hadoop Ecosystem | |
| 9. Sketching Out the HDFS Architecture | 11 |
| 9.1 HDFS | |
| 9.2 YARN | |
| 9.3 Hadoop Framework | |
| 10. Hadoop Data Analysis Technologies | 15 |
| 10.1 MapReduce | |
| 10.2 Hive | |
| 10.3 Apache PIG | |
| 11. Aviation Data Analysis Using Apache | 17 |
| 12. Conclusion | 24 |
| 13. References | 25 |

Introduction

About ten years ago, Continental Airlines was faltering and ranked among the lowest of the major US airlines. The problems the airline faced were not unique. But, siloed, unstructured data and incomplete information made determining the root cause analysis a difficult task. Continental Airlines needed a more comprehensive view of the business in order to reduce operating expenses, increase commercial insights and improve time-to-market. Here, too, legacy systems were hampering the growth and profitability story. In the past decade, there has been a substantial investment by airlines to mothball their legacy systems and move toward a modernization strategy. Some airlines realized substantial benefits by making investments in real-time data warehousing and analytical solutions.

Notably, Continental Airlines produced an ROI of over 1,000% with \$500 million in cost savings and revenue generation over a six year period. The challenges for implementing a real-time data warehouse are not alien within an airline environment. Real-time flight data feeds from flight operating systems and either flat-file or XML data feeds from the CRS are two solid examples of velocity and variety-related challenges that real-time Data Warehouse and Business Intelligence (DWBI) systems have to surpass. For many airlines, their current infrastructure is legacy in nature and moving to a new generation of data integration and business intelligence is an added level of complexity. Scandinavian Airlines had to move from a legacy mainframe and DB2 infrastructure in order to implement a 4th generation data integration and business intelligence application.

Even though there is tremendous complexity around the large volume of data airlines produce, real-time DWBI implementation is the need of the hour. With real-time DWBI systems, historical reporting is enabled over a longer period of time, with an increased level of granularity.

According to Wikipedia, "Big Data is a term for data sets that are so large or complex that traditional data processing applications are inadequate to deal with them. Analysis of data sets can find new correlations to spot business trends, prevent diseases, combat crime and so on." [1] With millions of people using Twitter to tweet about their most recent brand experience or hundreds of thousands of check-ins on Yelp, thousands of people talking about a recently released movie on Facebook and millions of views on YouTube for a recently released movie trailer, we are at a stage wherein we are heading into a social media data explosion. Companies are already facing challenges getting useful information from the transactional data from their customers (for e.g. data captured by the companies when a customer opens a new account or sign up for a credit card or a service). This type of data is structural in nature and still manageable. However, social media data is primarily unstructured in nature. The very unstructured nature of the data makes it very hard to analyze and very interesting at the same time.

Whereas RDBMS is designed to handle structured data and that to only certain limit, RDBMS fails to handle this kind of unstructured and huge amount of data called Big Data. This inability of RDBMS has given birth to new database management system called NoSQL management system.

Some of the key concepts used in Big Data Analysis are

1. Data Mining: Data mining is incorporation of quantitative methods. Using powerful mathematical techniques applied to analyze data and how to process that data. It is used to extract data and find actionable information which is used to increase productivity and efficiency.
2. Data Warehousing: A data warehouse is a database as the name implies. It is a kind of central repository for collecting relevant information. It has centralized logic which reduces the need for manual data integration.
3. MapReduce: MapReduce is a data processing paradigm for condensing large volumes of data into useful aggregated results. Suppose we have a large volume of data for particular users or employees etc. to handle. For that we need MapReduce function to get the aggregated result as per the query.
4. Hadoop: Anyone holding a web application would be aware of the problem of storing and retrieving data every minute. The adaptive solution created for the same was the use of Hadoop including Hadoop Distributed File System or HDFS for performing operations of storing and retrieving data. Hadoop framework has a scalable and highly accessible architecture.
5. Hive: Hive is a data warehouse system for Hadoop that facilitates ad hoc queries and analysis of large data sets stored in Hadoop
6. HQL: Hive uses a SQL like language called Hive. HQL is a popular choice for Hadoop analytics.

Literature Review

In the research work is carried out using Apache pig and hadoop on a crime dataset. It describes the large volume of data yielded from multiple sources and termed it as voluminous data. Crime and crime related datasets with ever growing population has raised to a higher extent and is a attention seeking subject to government for taking strict measures by prevailing law and procedure. Bigdata analytics using pig and hadoop has been applied on this crime dataset with the idea behind it as the optimal improvement for analysing some trends that needs to figure out, so among the citizens of the country there could be a feel of security and safety. Also it could help the government to furnish law and procedure and welfare among the people of the country.

Analysis results shows the total number of crimes occurred in every state, crimes that took place against women, type of crime and from year 2000 to 2014 the total number of crimes that took place. Experimental setup was pseudo distributed mode of hadoop and it was concluded that scripting language Pig Latin has fewer lines of code as compared to mapreduce program but the execution time increases in pig as compared to mapreduce. In the author describes that Big data analytics has attracted intense interest from all academia and industry recently for its attempt to extract knowledge, information and wisdom form big data.

Big data and cloud computing, two of the most important trends that are defining the new emerging analytical tools. Big data analytical capabilities using cloud delivery models could ease adoption for many industry, and most important thinking to cost saving, it could simplify useful insights that could providing them with different kinds of competitive advantage. Many companies to provide online Big Data analytical tools some of the top most companies like Amazon Big data Analytics Platform ,HIVE web based Interface, SAP Big data Analytics, IBM InfoSphere BigInsights, TERADATA Big Data Analytics, 1010data Big Data Platform, Cloudera Big Data Solution etc. Those companies analyze huge amount of data with help of different type of tools and also provide easy or simple user interface for analyzing data. In, Information technology gives utmost importance to processing of data.

Some petabytes of data is not sufficient for storing large amount of data. Large volume of unstructured and structured data that gets created from various sources such as Emails, web logs, social media like Twitter, Facebook etc. The major obstacles with processing Big Data include capturing, storing, searching, sharing and analysis. Hadoop enables to explore complex data. It is an open source framework written in Java which supports parallel and distributed data processing and is used for reliable storage of data. With the help of big data analytics, many enterprises are able to improve customer retention, help with product development and gain competitive advantage, speed and reduce complexity. E-commerce companies study traffic on web sites or navigation patterns to determine probable views, interests and dislikes of a person or a group as a whole depending on the previous purchases. In this paper, they compare some typically used data analytic tools.

Current State

Traditional systems that are deployed for airlines around the world are soloed and very ad-hoc in nature. Rarely do these systems share data in an economical manner. If they do, they are piecemeal extracts after the fact. Ideally, data generated by all operational enterprise systems and partners across an enterprise should be automatically archived and indexed. Regardless of the application or platform that created the data. Airlines should also be capable of searching the entire corporate database to retrieve the relevant data. Airlines need the right information at the right time, with the right degree of accuracy. For the airlines at this new frontier of innovation, competition and productivity, it's about analyzing masses of unstructured or semi-structured data. This, until recently, was considered too difficult, too time consuming and / or too expensive. But, as airlines get closer to their customers, they gain insight from looking at interaction patterns throughout the customer journey.

Typical attributes of airline data can be identified by four main attributes:-

1. **Volume** – Airline data is massive. Typical tier 1 carriers and Global Distribution Systems (GDS) have Passenger Name Record (PNR) data measured in terabytes.
2. **Velocity** – Airline data is real-time and arrives quickly, making timely decisions difficult.
3. **Variety** – Ad-hoc systems traditionally have different structures and shapes. Data analysis on such data is difficult because of rigid schemas.
4. **Value** – Airline data on its own is low value until it is rolled up and analyzed. At this point data becomes information which, in turn, becomes knowledge for the broader - market.

BIGDATA

Big data is a structured and unstructured data video, audio, pictures and information emails etc. it is a very large amount of data provided by social sites and daily activities of social media like news and news channels or new technology, television, mobile, and computers and industries all are big data. That's we can say it is more than thousands of information storage for the growth of the industries. Now if we have information or history of previous data then it's very easy for the next new changes for the industries or business.

Today's competitive world in this time industries and business are growing very fast by the help of the storage of the previous data which is known as big data. Big data is very hard to process and analysis the data easily. But with the help of HADOOP data is easy to process and analysis the data easily. Big data is a different collection of complex data sets. Big data is produced by different kinds of sources like television, mobile and other sources like industries data records.

It has three characteristics of big data:-

1. Volume
2. Velocity
3. Variety.
4. Value
5. Veracity

• Volume

Volume mainly defined is amount of data. Volume of data is growing exponentially megabytes, gigabytes, zeta bytes and petabytes. It's a very large amount of data and also hard to the process and storage.

Like Some earlier estimates suggested by the websites that 20 petabytes of storage its very large space was used to store 260 billion Facebook photos and messages or tag. In 2010, it was some newly reported by one million photographs were processed by Facebook per second.

Twitter is generating the data 12 terabytes of data per day it's newly research. Now the Facebook in 2012 stated that 2.7 billion "likes" and "comments" and messages were registered per day by the peoples.

• Velocity

Social media is one the major factor to provide the data exponentially. Social sites is continuous generated a complex data unstructured and semi-structured form of data. There are currently generated 90% data in last two years.

Increase the velocity of big data with help of mobile, televisions more advance technology. Internet is main factor to collecting of huge data. According to the user requirement which is save somewhere. It is known as velocity.

- **Variety**

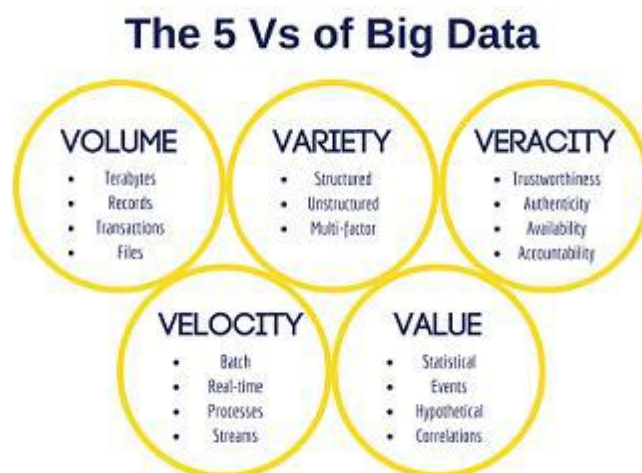
It is different kind of data are structured and Unstructured format. Structured data is always fixed format everywhere there is no possibility of changes of this data like tabular data, ERP, backup storage for large volume of data. But Unstructured data always no fixed format like text, audio, video, images and many social sites' data like Facebook, twitter, LinkedIn, logs file web chats etc. All companies and industries are having up to 85% of the data semi-structured and unstructured and structured format of data.

- **Value**

refers to our ability turn our data into value. It is important that businesses make a case for any attempt to collect and leverage big data. Value is that Big Data tends to have low value density. There are at least four additional characteristics that pop up in the literature from time to time. Big Data stores that they could not find in small stores. Certainly it is true that if in the past we were storing data about groups of customers an now storing data

- **Veracity**

refers to the messiness or trustworthiness of the data. With many forms of big data, quality and accuracy are less controllable, What is the provenance of the data? Does it come from a reliable source? It is accurate and by extension, complete. Big Data Veracity refers to the biases,noise and abnormality in Data. Inderpal feel veracity in data analysis is the biggest challange when compare to things like volume and veracity figur shows the 5Vs as:



HADOOP

As organizations are getting flooded with massive amount of raw data, the challenge here is that traditional tools are poorly equipped to deal with the scale and complexity of such kind of data. That's where Hadoop comes in. Hadoop is well suited to meet many Big Data challenges, especially with high volumes of data and data with a variety of structures.

At its core, Hadoop is a framework for storing data on large clusters of commodity hardware — everyday computer hardware that is affordable and easily available — and running applications against that data. A cluster is a group of interconnected computers (known as nodes) that can work together on the same problem. Using networks of affordable compute resources to acquire business insight is the key value proposition of Hadoop.

Hadoop consists of two main components

1. A distributed processing framework named MapReduce (which is now supported by a component called YARN(Yet Another Resource Negotiator) and
2. A distributed file system known as the Hadoop Distributed File System, or HDFS. In Hadoop you can do any kind any kind of aggregation of data whether it is one- month old data or one-year-old data. Hadoop provides a mechanism called MapReduce model to do distributed processing of large data which internally takes care of data even if one machine goes down.

Hadoop Ecosystem

Hadoop is a shared nothing system where each node acts independently throughout the system. A framework where a piece of work is divided among several parallel MapReduce task. Each task operated independently on cheap commodity servers. This enables businesses to generate values from data that was previously considered too expensive to be stored and processed in a traditional data warehouse or OLTP (Online Transaction Processing) environment. In the old paradigm, companies would use a traditional enterprise data warehouse system and would buy the biggest data warehouse they could afford and store the data on a single machine. However, with the increasing amount of data, this approach is no longer affordable nor practical.

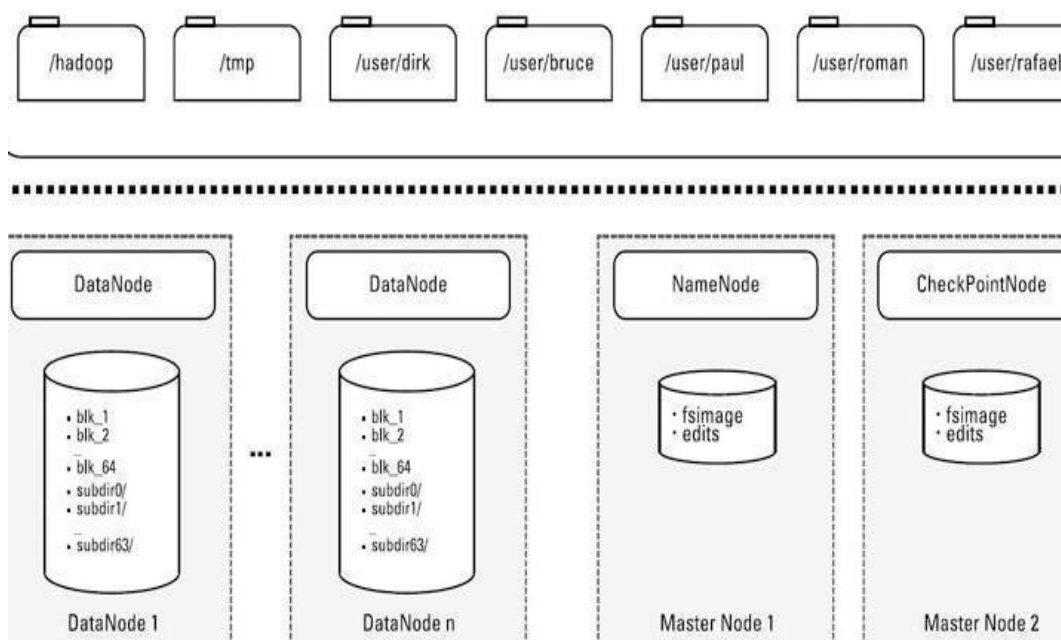
Some of the components of Hadoop ecosystem are HDFS (Hadoop Distributed File System), MapReduce, Yarn, Hive and Hbase. Hadoop has two core components. 'Storage' part to store the data and 'Processing' part to process the data. The storage part is called 'HDFS' and the processing part is called as 'YARN'.

Sketching Out the HDFS Architecture

Storage Component: Hadoop Distributed File System (HDFS)

As stated above, the Hadoop Distributed File System (HDFS) is the storage component of the core Hadoop Infrastructure. HDFS provides a distributed architecture for extremely large scale storage, which can easily be extended by scaling out. It is important to mention the difference between scale up and scale out. In its initial days, Google was facing challenges to store and process not only all the pages on the internet but also its users' web log data. At that time, Google was using scale up architecture model where you can increase the system capacity by adding CPU cores, RAM etc to the existing server. But such kind of model had not only been expensive but also had structural limitations. So instead, Google engineers implemented Scale out architecture model by using cluster of smaller servers which can be further scaled out if they require more power and capacity. Google File System (GFS) was developed based on this architectural model. HDFS is designed based on similar concept.

The core concept of HDFS is that it can be made up of dozens, hundreds, or even thousands of individual computers, where the system's files are stored in directly attached disk drives. Each of these individual computers is a self-contained server with its own memory, CPU, disk storage, and installed operating system (typically Linux, though Windows is also supported). Technically speaking, HDFS is a user-space-level file system because it lives on top of the file systems that are installed on all individual computers that make up the Hadoop cluster.



HDFS as a user-space-level file system

The above figure shows that a Hadoop cluster is made up of two classes of servers: slave nodes, where the data is stored and processed and master nodes, which govern the management of the Hadoop cluster. On each of the master nodes and slave nodes, HDFS runs special services and stores raw data to capture the state of the file system. In the case of the slave nodes, the raw data consists of the blocks stored on the node, and with the master nodes, the raw data consists of metadata that maps data blocks to the files stored in HDFS.

HDFS is a system that allows multiple commodity machines to store data from a single source. HDFS consists of a NameNode and a DataNode. HDFS operates as master slave architecture as opposed to peer to peer architecture. NameNode serves as the master component while the DataNode serves as a slave component. NameNode comprises of only the Meta data information of HDFS that is the blocks of data that are present on the Data Node.

- ✓ How many times the data file has been replicated?
- ✓ When does the NameNode start?
- ✓ How many DataNodes constitute a NameNode, capacity of the NameNode and space utilization?
- ✓ The DataNode comprises of data processing, all the processing data that is stored on the DataNode and deployed on each machine.
- ✓ The actual storage of the files being processed and serving read and write request for the clients

In the earlier versions of Hadoop there was only one NameNode attached to the DataNode which was a single point of failure. Hadoop version 2.x provides multiple NameNode where secondary NameNode can take over in the event of a primary nameNode failure. Secondary NameNode is responsible for performing periodic check points in the event of a primary NameNode failure. You can start secondary NameNode by providing checkpoints that provide high availability within HDFS.

Let's take look at a data warehouse structure example where we have one machine and with HDFS we can distribute the data into more than one machine. Let's say we have 100 GB of file that takes 20 minutes to process on a machine with a given number of channel and hard drive. If you add four machines of exactly the same configuration on a Hadoop cluster, the processing time reduces to approximately one fourth of the original processing time or about 5 minutes. But what happens if one of these four machines fails?

HDFS creates a self-healing architecture by replicating the same data across multiple nodes. So it can process the data in a high availability environment. For example, if we have three DataNodes and one NameNode, the data is transferred from the client environment into HDFS DataNode. The replication factor defines the number of times a data block is replicated in a clustered environment. Let's say we have a file that is split into two data blocks across three

DataNodes. If we are processing these files to a three DataNode cluster and we set the replication factor to three. If one of the nodes fails, the data from the failed nodes is redistributed among the remaining active nodes and the other nodes will complete the processing function.

Processing Component: Yet Another Resource Negotiator (YARN)

YARN (Yet Another Resource Negotiator) is a resource manager that identifies on which machine a particular task is going to be executed. The actual processing of the task or program will be done by Node Manager. In Hadoop 2.2, YARN augments the MapReduce platform and serves as the Hadoop operating system. Hadoop 2.2 separates the resource management function from data processing allowing greater flexibility.

This way MapReduce only performs data processing while resource management is isolated in YARN. Being the primary resource manager in HDFS, YARN enables enterprises to store data in a single place and interact with it in multiple ways with consistent levels of service.

In Hadoop 1.0 the NameNode used job tracker and the DataNode used task tracker to manage resources. In Hadoop 2.x, YARN splits up into two major functionalities of the job tracker - the resource management and job scheduling. The client reports to the resource manager and the resource manager allocates resources to jobs using the resource container, Node Manager and app master. The resource container splits memory, CPU, network bandwidth among other hardware constraints into a single unit. The Node Manager receives updates from the resource containers which communicate with the app master. The Node Manager is the framework for containers, resource monitoring and for reporting data to the resource manager and scheduler.

Hadoop Framework

Hadoop Framework comprises of Hadoop Distributed File System and the MapReduce framework. The Hadoop framework divides the data into smaller chunks and stores each part of the data on a separate node within the cluster. For example, if we have 4 terabytes of data, the HDFS divides this data into 4 parts of 1TB each. By doing this, the time taken to store the data onto the disk is significantly reduced. The total time to store this entire data onto the disk is equal to storing 1 part of the data as it will store all the parts of the data simultaneously on different machines.

In order to provide high availability what Hadoop does is replicate each part of the data onto other machines that are present within the cluster. The number of copies it will replicate depends on the replication factor. By default the replication factor is 3, in such a case there will be 3 copies of each part of the data on three different machines. In order to reduce the bandwidth and latency time, it will store two copies on the same rack and third copy on a different rack.

For example, in the above example, NODE 1 and NODE 2 are on rack one and NODE 3 and NODE 4 are on rack two. Then the first two copies of part 1 will be stored on NODE 1 and third copy will be stored either on NODE 3 or NODE 4. Similar process is followed in storing remaining parts of the data. The HDFS takes care of the networking required by these nodes in order to communicate.

Hadoop Data Analysis Technologies

While Hadoop provides the ability to collect data on HDFS (Hadoop Distributed File System), there are many applications available in the market (like MapReduce, Pig and Hive) that can be used to analyze the data.

Let us first take a closer look at all three applications and then analyze which application is better suited for YouTube Data Analysis project.

- **MapReduce**

MapReduce is a set of Java classes run on YARN with the purpose of processing massive amounts of data and reducing this data into output files. HDFS works with MapReduce to divide the data in parallel fashion on local or parallel machines. Parallel structure requires that the data is immutable and cannot be updated. It begins with the input files where the data is initially stored typically residing in HDFS. These input files are then split up into input format which selects the files, defines the input splits, breaks the file into tasks and provides a place for record reader objects. The input format defines the list of tasks that makes up the map phase. The tasks are then assigned to the nodes in the system based on where the input files chunks are physically resident. The input split describes the unit of work that comprises a single map task in a MapReduce program. The record reader loads the data and converts it into key value pairs that can be read by the Mapper. The Mapper performs the first phase of the MapReduce program. Given a key and a value the mappers export key and value pairs and send these values to the reducers. The process of moving mapped outputs to the reducers is known as shuffling. Partitions are the inputs to reduce tasks, the partitioner determines which key and value pair will be stored and reduced. The set of intermediate keys are automatically stored before they are sent to the reduce function. A reducer instance is created for each reduced task to create an output format. The output format governs the way objects are written, the output format provided by Hadoop writes the files to HDFS.

- **Hive**

After congregating the tweets into HDFS they are analyzed by queries using Hive. Apache Hive data warehouse software facilitates querying and managing large datasets residing in distributed storage. Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL. In opinion mining system, hive is used to query out interested part of the tweets which can be an opinion, comments related to a specific topic or a trending hash tag.

Twitter API loads the HDFS with tweets which are represented as JSON blobs. Processing twitter data in relational database such as SQL requires significant transformations due to nested data structures. Hive facilitates an interface that provides easy access to tweets using HiveQL that supports nested data structures. Hive compiler converts the HiveQL queries into map reduce jobs. Partition feature in hive allows tweet tables to split into different directories. By constructing queries that includes partitions, hive can determine the partition comprising the result. The location of twitter tables are explicitly specified in "Hive External Table" which are partitioned. Hive uses SerDe (Serializer-Deserializer) interface in determining record processing steps. Deserializer interface

intakes string of tweets and translates it into a Java Object that Hive can manipulate on. The Serializer interface intakes a java Object that Hive has worked on and converts it into required data to be written on HDFS.

- **Apache Pig**

Pig comes from the language Pig Latin. Pig Latin is a procedural programming language and fits very naturally in the pipeline paradigm. When queries become complex with most of joins and filters then Pig is strongly recommended. Pig Latin allows pipeline developers to decide where to checkpoint data in the pipeline. That is storing data in between operations has the advantage of check pointing data in the pipeline. This ensures the whole pipeline does not has to be rerun in the event of a failure. Pig Latin allows users to store data at any point in the pipeline without disturbing the pipeline execution.

The advantage that Pig Latin provides is that pipelines developers decide where appropriate checkpoints are in the pipeline rather than being forced to checkpoint wherever the schematics of SQL impose it. Pig Latin supports splits in the pipeline. Common features of data pipelines is that they are often graphics and not linear pipelines since disk's read and write scan time and intermediate results usually dominate processing of large datasets reducing the number of times data must be written to and read from disk is crucial for good performance.

Pig Latin allows developers to insert their own code almost anywhere in the data pipeline which is useful for pipeline development. This is accomplished through a user defined functions UDFS (User Defined Functions). UDFS allows user to specify how data is loaded, how data is stored and how data is processed. Streaming allows users to include executable at any point in the data flow. Pipeline also often includes user defined columns transformation functions and user defined aggregations. Pig Latin supports writing both of these types of functions in java.

Aviation Data Analysis Using Apache Pig

This blog will help you in gaining some insights on the U.S. Airline data using Apache Pig. I hope readers of this blog are aware of what Apache Pig is and various operations that can be performed using it.

The U.S. Department of Transportation's (DOT) Bureau of Transportation Statistics (BTS) tracks the on-time performance of domestic flights operated by large air carriers. Summary information on the number of on-time, delayed, canceled, and diverted flights appears in DOT's monthly Air Travel Consumer Report, published about 30 days after the month's end, as well as in summary tables posted on this website. Summary statistics and raw data are made available to the public at the time the Air Travel Consumer Report is released.

These are the two files:

[Delayed_Flights.csv](#)

[Airports.csv](#)

These are 2 different datasets, i.e., Delayed_Flights.csv and Airports.csv. Let us understand one at a time.

Delayed_Flights.csv Datasets

There are 29 columns in this dataset. Some of them have been mentioned below:

- Year: 1987 – 2008
- Month: 1 – 12
- FlightNum: Flight number Canceled: Was the flight canceled?
- CancellationCode: The reason for cancellation.

Airports.csv Datasets

- iata: the international airport abbreviation code name of the airport
- city and country in which airport is located.
- lat and long: the latitude and longitude of the airport

Now, using Apache pig, we will try to gain more insights from these datasets.

Problem Statement 1

Find out the top 5 most visited destinations.

Source Code

```
REGISTER '/home/acadgild/airline_usecase/piggybank.jar';
A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING B
= foreach A generate (int)$1 as year, (int)$10 as flight_num, (ch C =
filter B by dest is not null;
D = group C by dest;
E = foreach D generate group, COUNT(C.dest); F = order E by $1 DESC;
Result = LIMIT F 5;
A1 = load '/home/acadgild/airline_usecase/airports.csv' USING org.a
A2 = foreach A1 generate (chararray)$0 as dest, (chararray)$2 as ci
joined_table = join Result by $0, A2 bydest;
dump joined_table;
```

In Line 1: We are registering the *piggybank* jar in order to use the CSVExcelStorage class.

In relation **A**, we are loading the dataset using CSVExcelStorage because of its effective technique to handle double quotes and headers.

In relation **B**, we are generating the columns that are required for processing and explicitly typecasting each of them.

In relation **C**, we are filtering the null values from the “dest” column. In relation **D**, we are grouping relation C by “dest.”

In relation **E**, we are generating the grouped column and the count of each. Relation **F** and **Result** is used to order and limit the result to top 5.

These are the steps to find the top 5 most visited destinations. However, adding few more steps in this process, we will be using another table to find the city name and country as well.

In relation **A1**, we are loading another table to which we will look-up and find the city as well as the country.

In relation **A2**, we are generating dest, city, and country from the previous relation.

In relation **joined_table**, we are joining Result and A2 based on a common column, i.e., “dest”

Finally, using dump, we are printing the result.

```
grunt> REGISTER '/home/acadgild/airline_usecase/piggybank.jar';
grunt> A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage('','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER');
2016-11-13 11:48:37,955 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapreduce.job.counters.limit is deprecated. Instead, use mapreduce.job.counters.max
2016-11-13 11:48:37,956 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2016-11-13 11:48:37,956 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> B = foreach A generate (int)$1 as year, (int)$10 as flight_num, (chararray)$17 as origin, (chararray) $18 as dest;
grunt> C = filter B by dest is not null;
grunt> D = group C by dest;
grunt> E = foreach D generate group, COUNT(C.dest);
grunt> F = order E by $1 DESC;
grunt> Result = LIMIT F 5;
2016-11-13 11:49:25,773 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapreduce.job.counters.limit is deprecated. Instead, use mapreduce.job.counters.max
2016-11-13 11:49:25,773 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2016-11-13 11:49:25,776 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> A1 = load '/home/acadgild/airline_usecase/airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage('','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER');
2016-11-13 11:49:25,776 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapreduce.job.counters.limit is deprecated. Instead, use mapreduce.job.counters.max
2016-11-13 11:49:25,776 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2016-11-13 11:49:25,776 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> A2 = foreach A1 generate (chararray)$0 as dest, (chararray)$2 as city, (chararray)$4 as country;
grunt> joined_table = join Result by $0, A2 by dest;
```

```
max
2016-11-13 11:52:07,979 [main] WARN or
2016-11-13 11:52:07,995 [main] INFO or
2016-11-13 11:52:07,995 [main] INFO or
(ATL,106898,ATL,Atlanta,USA)
(DEN,63003,DEN,Denver,USA)
(DFW,70657,DFW,Dallas-Fort Worth,USA)
(LAX,59969,LAX,Los Angeles,USA)
(ORD,108984,ORD,Chicago,USA)
```

Problem Statement 2

Which month has seen the most number of cancellations due to bad weather?

Source code

```
REGISTER '/home/acadgild/airline_usecase/piggybank.jar';
A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING B
= foreach A generate (int)$2 as month,(int)$10 as flight_num,(int) C =
filter B by cancelled == 1 AND cancel_code=='B';
D = group C by month;
E = foreach D generate group, COUNT(C.cancelled); F= order E by $1
DESC;
Result = limit F 1; dump Result;
```

In Line 1: We are registering *piggybank* jar in order to use the CSVExcelStorage class.

In relation **A**, we are loading the dataset using CSVExcelStorage because of its effective technique to handle double quotes and header.

In relation **B**, we are generating the columns which are required for processing and explicitly typecasting each of them.


In relation **C**, we are filtering the data based on cancellation and cancellation code, i.e., canceled = 1 means flight have been canceled and cancel_code = 'B' means the reason for cancellation is "weather." So relation C will point to the data which consists of canceled flights due to bad weather.

In relation **D**, we are grouping the relation C based on every month. In relation **E**, we are finding the count of canceled flights every month.

Relation **F** and **Result** is for ordering and finding the top month based on cancellation.

```
grunt> REGISTER '/home/acadgild/airline_usecase/piggybank.jar';
grunt> A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage('','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER'
2
2016-11-13 11:56:42,492 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapreduce.job.counters.limit is deprecated. Instead, use mapreduce.job.counters
max
2016-11-13 11:56:42,493 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2016-11-13 11:56:42,493 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> B = foreach A generate (int)$2 as month,(int)$10 as flight_num,(int)$22 as cancelled,(chararray)$23 as cancel_code;
grunt> C = filter B by cancelled == 1 AND cancel_code == 'B';
grunt> D = group C by month;
grunt> E = foreach D generate group, COUNT(C.cancelled);
grunt> F= order E by $0 DESC;
grunt> Result = limit F 1;
```

```
2016-11-13 11:58:45,906 [main] INFO
2016-11-13 11:58:45,906 [main] INFO
max
2016-11-13 11:58:45,907 [main] WARN
2016-11-13 11:58:45,922 [main] INFO
2016-11-13 11:58:45,922 [main] INFO
(12,250)
```



Problem Statement 3

Top ten origins with the highest AVG departure delay

Source code

```
REGISTER '/home/acadgild/airline_usecase/piggybank.jar';

A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING
B1 = foreach A generate (int)$16 as dep_delay, (chararray)$17 as or
C1 = filter B1 by (dep_delay is not null) AND (origin is not null);
D1 = group C1 by origin;

E1 = foreach D1 generate group, AVG(C1.dep_delay);
Result = order E1 by $1 DESC;

Top_ten = limit Result 10;

Lookup = load '/home/acadgild/airline_usecase/airports.csv' USING o
Lookup1 = foreach Lookup generate (chararray)$0 as origin, (chararr
Joined = join Lookup1 by origin, Top_ten by $0;

Final = foreach Joined generate $0,$1,$2,$4;
```

Explanation of first 3 lines are the same as explained in the previous 2 problem statements.

In relation **C1**, we are removing the null values fields present if any. In relation **D1**, we are grouping the data based on column “origin.” In relation **E1**, we are finding average delay from each unique origin.

Relations named **Result** and **Top_ten** are ordering the results in descending order and printing the top ten values.

These steps are good enough to find the top ten origins with the highest

average departure delay.

However, rather than generating just the code of origin, we will be following a few more steps to find some more details like country and city.

In the relation **Lookup**, we are loading another table to which we will look up and find the city as well as the country.

In the relation **Lookup1**, we are generating the destination, city, and country from the previous relation.

In the relation **Joined**, we are joining relation Top_ten and Lookup1 based on common a column, i.e., “origin.”

In the relation **Final**, we are generating required columns from the Joined table.

Finally, we are ordering and printing the results.

```
grunt> REGISTER /home/acadgild/airline_usecase/piggybank.jar;
grunt> A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE', 'UNIX', 'SKIP_INPUT_HEADER')
;
2016-11-13 12:20:54,240 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapreduce.job.counters.limit is deprecated. Instead, use mapreduce.job.counters.
max
2016-11-13 12:20:54,240 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2016-11-13 12:20:54,240 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> B1 = foreach A generate (int)$16 as dep_delay, (chararray)$17 as origin;
grunt> C1 = filter B1 by (dep_delay is not null) AND (origin is not null);
grunt> D1 = group C1 by origin;
grunt> E1 = foreach D1 generate group, AVG(C1.dep_delay);
grunt> Result = order E1 by $1 DESC;
grunt> Top_ten = limit Result 10;
grunt> LookUp = load '/home/acadgild/airline_usecase/airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE', 'UNIX', 'SKIP_INPUT_HEADER')
;
2016-11-13 12:21:32,073 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapreduce.job.counters.limit is deprecated. Instead, use mapreduce.job.counters.
max
2016-11-13 12:21:32,074 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2016-11-13 12:21:32,074 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> LookUp1 = foreach LookUp generate (chararray)$0 as origin, (chararray)$2 as city, (chararray)$4 as country;
grunt> Joined = join LookUp1 by origin, Top_ten by $0;
grunt> Final = foreach Joined generate $0,$1,$2,$4;
grunt> Final Result = ORDER Final by $3 DESC;
```

```
2016-11-13 12:21:11,025 [main] INFO org.apache.hadoop.mapreduce.lib.output.TextOutputFormat -
(CMX, Hancock, USA, 116.1470588235294)
(PLN, Pellston, USA, 93.76190476190476)
(SPI, Springfield, USA, 83.84873949579831)
(ALO, Waterloo, USA, 82.2258064516129)
(MQT, NA, USA, 79.55665024630542)
(ACY, Atlantic City, USA, 79.3103448275862)
(MOT, Minot, USA, 78.66165413533835)
(HHH, NA, USA, 76.53005464480874)
(EGE, Eagle, USA, 74.12891986062718)
(BGM, Binghamton, USA, 73.15533980582525)
```

Problem Statement 4

Which route (origin & destination) has seen the maximum diversion?

Source code

```
REGISTER '/home/acadgild/airline_usecase/piggybank.jar';
A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING B
= FOREACH A GENERATE (chararray)$17 as origin, (chararray)$18 as C
= FILTER B BY (origin is not null) AND (dest is not null) AND (di D =
GROUP C by (origin,dest);
E = FOREACH D generate group, COUNT(C.diversion); F = ORDER E BY
$1 DESC;
Result = limit F 10; dump Result;
```

In Line 1: We are registering *piggybank* jar in order to use CSVExcelStorage class.

In relation **A**, we are loading the dataset using CSVExcelStorage because of its effective technique to handle double quotes and headers.

In relation **B**, we are generating the columns which are required for processing and explicitly type-casting each of them.

In relation **C**, we are filtering the data based on “not null” and diversion =1. This will remove the null records, if any, and give the data corresponding to the diversion taken.

In relation **D**, we are grouping the data based on origin and destination. Relation **D** finds the count of diversion taken per unique origin and destination.

Relations **F** and **Result** orders the result and produces top 10 results.

```
grunt> REGISTER '/home/acadgild/airline_usecase/piggybank.jar';
grunt> A = load '/home/acadgild/airline_usecase/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage('','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER');
2016-11-13 12:29:00,015 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapreduce.job.counters.limit is deprecated. Instead, use mapreduce.job.counters.max
2016-11-13 12:29:00,015 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2016-11-13 12:29:00,015 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> B = FOREACH A GENERATE (chararray)$17 as origin, (chararray)$18 as dest, (int)$24 as diversion;
grunt> C = FILTER B BY (origin is not null) AND (dest is not null) AND (diversion == 1);
grunt> D = GROUP C by (origin,dest);
grunt> E = FOREACH D generate group, COUNT(C.diversion);
grunt> F = ORDER E BY $1 DESC;
grunt> Result = limit F 10;
```

Conclusion

Hadoop Mapreduce is now a popular choice for performing large-scale data analytics. Bigdata analytics using pig and hive sheds light on significant issues faced by aviation data and we can find the numbers of flight cancelled per year and based on the parameters like execution time, , lines of code it has been examined that hive holds better and efficient than pig.

The thesis has given a brief introduction to the core thechnology of Hadoop but there are still many applications and projects developed on Hadoop. In conclusion, the Hadoop, which is based on the Hadoop HDFS and MapReduce has provided a distributed data processing platform. The high fault tolerance and high scalability allow its users to apply Hadoop on cheap hardware. The MapReduce distributed programming mode allows the users to develop their own applications without the users having to know the bottom layer of the MapReduce. Because of the advantages of Hadoop, the users can easily manage the computer resources and build their own distributed data processing platform.

Above all, it is obvious to notice the convenience that the Hadoop has brought in Big Data processing. It also should be pointed out that since Google published the first paper on the distributed file system till now, the history of Hadoop is only 10-year old. With the advancement of the computer science and the Internet technology, Hadoop has rapidly solved key problems and been widely used in real life. In spite of this, there are still some problems in facing the rapid changes and the ever increasing demand of analysis. To solve these problems, Internet companies, such as Google also introduced the newer technologies. It is predictable that with the key problems being solved, Big Data processing based on Hadoop will have a wider application prospect.

References

- [1] Arushi Jain, Vishal Bhatnagar, "Crime Data Analysis Using Pig with Hadoop" in International Conference on Information Security & Privacy (ICISP2015), 11-12 December 2015, Nagpur, INDIA, in ELSEVIER 2015.
- [2] Rahul Kumar Chawda, Dr. Ghanshyam Thakur, "Big Data and Advanced Analytics Tools", 2016 Symposium on Colossal Data Analysis and Networking (CDAN), IEEE 2016, ISSN: 978-1-5090-0669-4/16.
- [3] Mrunal Sogodekar, Shikha Pandey, Isha Tupkari, Amit Manekar, "BIG DATA ANALYTICS: HADOOP AND TOOLS", in 978-1- 5090-2730-9/16, 2016 IEEE
- [4] Jurmo Mehine, Satish Srirama, Pelle Jakovits "Large Scale Data Analysis Using Apache Pig"
- [5] Dave Jaffe "Three Approaches to Data Analysis with Hadoop"
- [6] <http://hadoop.apache.org/>
- [7] <https://pig.apache.org/>
- [8] Aditya B. Patel, Manashvi Birla, Ushma Nair, "Addressing Big Data Problem Using Hadoop and Map Reduce", 6-8 Dec. 2012.
- [9] Michael G. Noll, Applied Research, Big Data, Distributed Systems, Open Source, "Running Hadoop on Ubuntu Linux (Single-Node Cluster)", [online], available at <http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/>
- [10] Sagioglu, S., & Sinanc, D, "Big data: A review", IEEE International Conference on Collaboration Technologies and Systems (CTS), 2013, pp 42-47.
- [11] <https://hive.apache.org/>
- [12] Dave Jaffe "Three Approaches to Data Analysis with Hadoop"
- [13] Jurmo Mehine, Satish Srirama, Pelle Jakovits "Large Scale Data Analysis Using Apache Pig.