**Project On**

# Airline Analysis

A Report for the Evaluation

**Bachelor of Technology in Computer Science and Engineering**


**Under the Guidance of**

Name of the Guide

Ms. N. Gayathri

**Submitted by**

Name of the Student :

NAITIK SHARMA

16SCSE101059

1613101424

**Department of Computer Science and Engineering**

**GALGOTIAS UNIVERSITY GREATER NOIDA UP.**

## Table of contents


| Content | Page No. |
|---|---|
| Front page | 1 |
| Abstract | 3 |
| Introduction<br>1.1 overall description<br>1.2 purpose<br>1.3 motivation and scope | 4 |
| Literature Survey | 6 |
| Proposed Approach | 7 |
| References | 10 |

## Abstract

In the contemporary world, Data analysis is a challenge in the era of varied inters- disciplines though there is a specialization in the respective disciplines.

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, environment and health.

Big Data can be used to assess risk in the insurance industry and to track reactions to products in real time. Big Data is also used to monitor things as diverse as wave movements, flight data, traffic data, financial transactions, health and crime. The challenge of Big Data is how to use it to create something that is value to the user.

How can it be gathered, stored, processed and analyzed it to turn the raw data information to support decision making. In this paper Big Data is depicted in a form of case study for Airline data.

The proposed method is made by considering following scenario under consideration

An Airport has huge amount of data related to number of flights, data and time of arrival and dispatch, flight routes, No. of airports operating in each country, list of active airlines in each country. The problem they faced till now it's, they have ability to analyze limited data from databases. The Proposed model intension is to develop a model for the airline data to provide platform for new analytics based on the following queries.

# Introduction

## 1.1 Overall description

Big Data is not only a Broad term but also a latest approach to analyze a complex and huge amount of data; there is no single accepted definition for Big Data. But many researchers working on Big Data have defined Big Data in different ways. One such approach is that it is characterized by the widely used 4 V's approach. The first "V" is Volume, from which the Big Data comes from. This is the data which is difficult to handle in conventional data analytics. For example, Volume of data created by the BESCOM (Bangalore Electricity Supply Company) in the process of the power supply and its consumption for Bangalore city or for the entire Karnataka State generates a huge volume of data. To analyze such data, it is the Big data that comes to aid of data analytics; the second "V" is velocity, the high speed at which the data is created, processed and analyzed; the third "V" is variety which helps to analyze the data like face book data which contains all types of variety, like text messages, attachments, images, photos and so on; the forth "V" is Veracity, that is cleanliness and accuracy of the data with the available huge amount of data which is being used for processing.

Researchers working in the structured data face many challenges in analyzing the data. For instance the data created through social media, in blogs, in Facebook posts or Snap chat. These types of data have different structures and formats and are more difficult to store in a traditional business data base. The data in big data comes in all shapes and formats including structured. Working with big data means handling a variety of data formats and structures. Big data can be a data created from sensors which track the movement of objects or changes in the environment such as temperature fluctuations or astronomy data. In the world of the internet

of things, where devices are connected and these wearables create huge volume of data. Thus big data approaches are used to manage and analyze this kind of data. Big Data include data from a whole range of fields such as flight data, population data, financial and health data such data brings as to another V, value which has been proposed by a number of researcher i.e., Veracity.

Most of the time social media is analyzed by advertisers and used to promote produces and events but big data has many other uses. It can also been used to assess risk in the insurance industry and to track reaction to products in real time. Big Data is also used to monitor things as diverse as wave movements, flight data, traffic data, financial transactions, health and crime. The challenge of Big Data is how to use it to create something that is value to the user. How to gather it, store it, process it and analyze it to turn the raw data information to support decision making.

Hadoop allows to store and process Big Data in a distributed environment across group of computers using simple programming models. It is intended to scale up starting with solitary machines and will be scaled to many machines. But now since huge amount of data in Terabytes which is injected into Hadoop Distributed File System files and processed by HDFS Tool.

An Airport has huge amount of data related to number of flights, data and time of arrival and dispatch, flight routes, No. of airports operating in each country, list of active airlines in each country. The problem they faced till now it's, they have ability to analyze limited data from databases. The Proposed model intension is to develop a model for the airline data to provide platform for new analytics based on the following queries.

## 1.1 Problem Statement

- ✓ Big amount of data generated on hourly basis.
- ✓ A single twin engine aircraft with an average 12 hour flight time can produce up to 844 TB of data
- ✓ There are many active users of flights
- ✓ Many flights are scheduled everyday

- ✓ User varies from common man to celebrities

The proposed method is made by considering following scenario under consideration .An Airport has huge amount of data related to number of flights, data and time of arrival and dispatch, flight routes, No. of airports operating in each country, list of active airlines in each country. The problem they faced till now it's, they have ability to analyze limited data from databases. The Proposed model intension is to develop a model for the airline data to provide platform for new analytics based on the following queries.

1. Extract unstructured data using python language.

2. Make unstructured data into structured using hadoop.

3. Analyse data for the following queries

a) List of airports operating in the country India?

b) How many active airlines in United State.?

c) List of airlines operating with code share?

d) Which country having highest Airport?

e) How many flight having same air code for flight which uses code share?

## 1.2 Purpose

The main purpose of the project to explore detailed analysis on airline data sets such as listing airports operating in the India, list of airlines having zero stops, list of airlines operating with code share which country has highest airports and list of active airlines in united states. The main objective of project is the processing the big data sets using map reduce component of hadoop ecosystem in distributed environment.

## 1.3 Motivation and scope

### Product Perspective

The main purpose of the project to explore detailed analysis on airline data sets such as listing airports operating in the India, list of airlines having zero stops, list of airlines operating with code share which country has highest airports and list of active airlines in united states. The main objective of project is the processing the big data sets using map reduce component of Hadoop ecosystem in distributed environment.

### Product Features

Airline data analysis can provide a solution for businesses to collect and optimize large datasets, improve performance, improve their competitive advantage, and make faster and better decisions.

- ✓ By using airline data analysis, we can save time of users.
- ✓ The data could even be structured, semi-structured or unstructured.
- ✓ Cost savings
- ✓ Implementing new strategies
- ✓ Fraud can be detected the moment it happens

## 1.4 Operating Environment or Software Environment

Software environment is the term commonly used to refer to support an application. A software environment for a particular application could include the operating system, the database system, specific analysis tools.

The software and hardware that we are using in our project Airline data analysis are:

1.4.1 Intel core i3 and above

1.4.2 Windows 10

1.4.3 Windows subsystem for Linux

1.4.4 Ubuntu

1.4.5 Java JDK 1.8

1.4.6 Hadoop 3.0.0

1.4.7 Map reduce

1.4.8 Microsoft Excel

1.4.9 Minimum RAM 4GB and above

## 1.5 Assumptions and Dependencies

Constraints are limitations which are outside the control of the project. The Project must be managed within these constraints.

Assumptions are made about events, or facts outside the control of project. External dependencies are activities which need to be completed before an internal activity can proceed.

Constraints, assumptions and dependencies can create risks that the project may be delayed because access is not provided to the site (assumption).

Assumption will be that the complexity may arise due to large unstructured data set.

## 1.6 Constraints

Hardware limitation and timing constraints.

High feature may not correspond to semantic similarity.

System Environment

Windows subsystem for Linux with Ubuntu operating system will be required to run the application

## Proposed Model

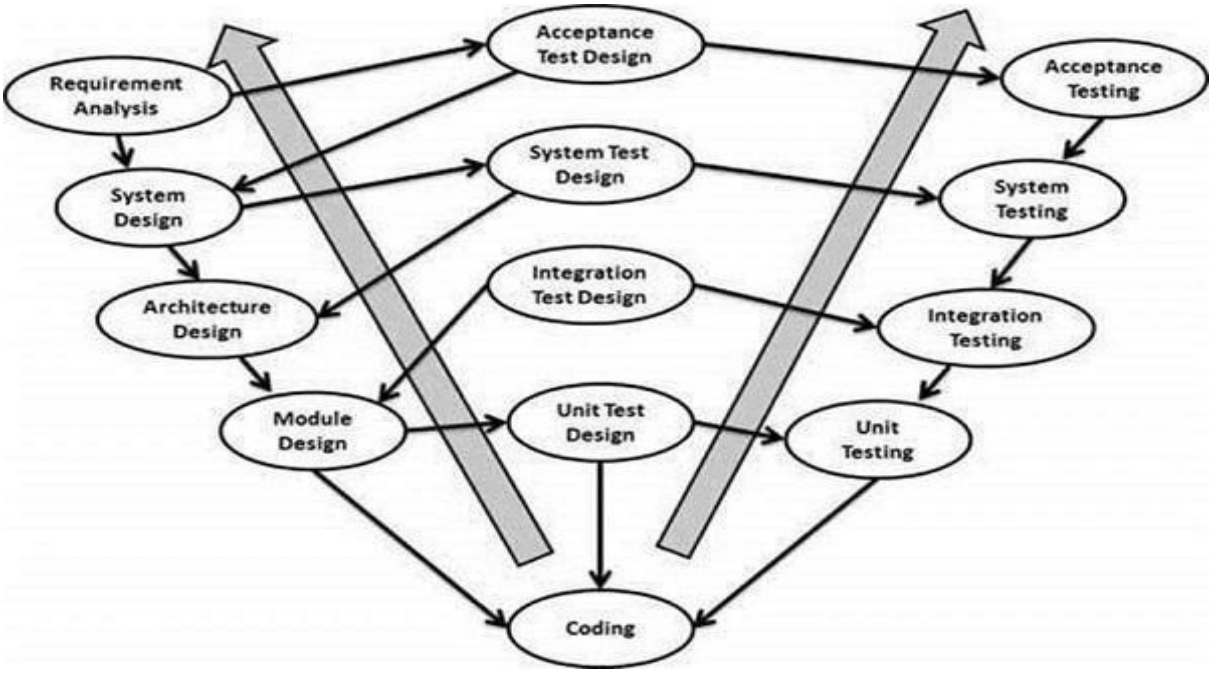
This Project is based on V-model **SDLC** (Software Development Life Cycle)

The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as Verification and Validation model.

The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

Under the V-Model, the corresponding testing phase of the development phase is planned in parallel. So, there are Verification phases on one side of the 'V' and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model.

The following illustration depicts the different phases in a V-Model of the SDLC.


Requirement Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Unit Test Design
Unit Testing
Coding

# DISTRIBUTED FILE SYSTEM

## Introduction

A distributed file system (DFS) is a file system with data stored on a server. The data is accessed and processed as if it was stored on the local client machine. The DFS makes it convenient to share information and files among users on a network in a controlled and authorized way. The server allows the client users to share files and store data just like they are storing the information locally. However, the servers have full control over the data and give access control to the clients.

There has been exceptional growth in network-based computing recently and client/server-based applications have brought revolutions in this area. Sharing storage resources and information on the network is one of the key elements in both local area networks (LANs) and wide area networks (WANs). Different technologies have been developed to bring convenience to sharing resources and files on a network; a distributed file system is one of the processes used regularly.

One process involved in implementing the DFS is giving access control and storage management controls to the client system in a centralized way, managed by the servers. Transparency is one of the core processes in DFS, so files are accessed, stored, and managed on the local client machines while the process itself is actually held on the servers. This transparency brings convenience to the end user on a client machine because the network file system efficiently manages all the processes. Generally, a DFS is used in a LAN, but it can be used in a WAN or over the Internet. A DFS allows efficient and well-managed data and storage sharing options on a network compared to other options. Another option for users in network-based computing is a shared disk file system. A shared disk file system puts the

access control on the client's systems so the data is inaccessible when the client system goes offline. DFS is fault-tolerant and the data is accessible even if some of the network nodes are offline.

**Client**

Client more than one client may access the same data simultaneously, the server must have a mechanism in place (such as maintaining information about the times of access) to organize updates so that the client always receives the most current version of data and that data conflicts do not arise.

**Server**

Server is a system which receives request or commands from client and gives back the response according to the request. Server can run on any type computer.

**Challenges in HDFS**

- ✓ DFS due to failure of hardware components data do not reach the destination point.
- ✓ Data in node can get altered or corrupted.
- ✓ Lack of performance and scalability.
- ✓ Lack of flexible resource management.
- ✓ Lack of application deployment support.
- ✓ Lack of quality of service.
- ✓ Lack of multiple data source support.

## HADOOP DISTRIBUTED FILE SYSTEM – HDFS [3]

### Introduction

Apache Hadoop is good choice for airline data analysis as it works for distributed big data. Apache Hadoop is an open source software framework for distributed storage and large-scale distributed processing of data-sets on clusters. Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes. In short, Hadoop framework is capable enough to develop applications capable of running on clusters of computers and they could perform complete statistical analysis for huge amounts of data. Hadoop MapReduce is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

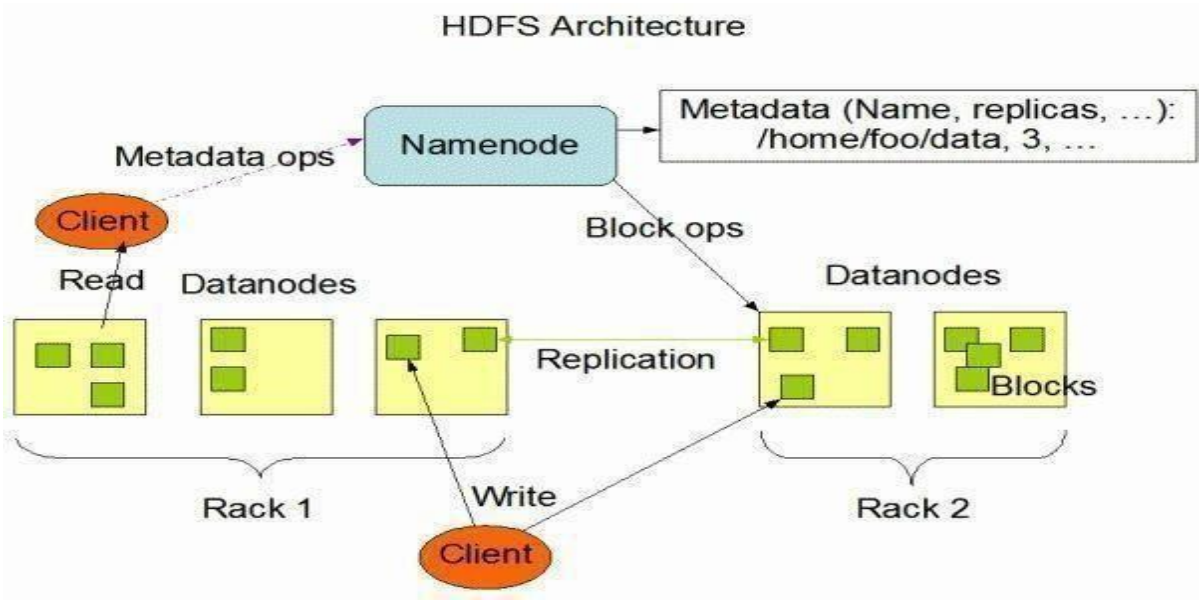



Figure : HDFS architecture

### Assumptions and Goals

**Hardware Failure** – In HDFS hardware failure is very common. HDFS instance has hundred thousand of servers which contain data. So, because of this large network there is a probability that failure will occur. Thus, HDFS error and fault control with automatic recovery should be our main goal.

**Streaming Data Access** – Streaming data means is shifting/transferring of data at constant rate (high speed) in order to carry out various functions. By data streaming, HDFS can provide High Definition TV services or constant back up to storage medium. Therefore, data is read in continuously with constant data rate rather reading in form of blocks/packets.

**Latency-** Latency is defined as time delay caused due to various operations during the process. In Hadoop, initial time is spent in various activities for example – resource distribution, job submission and split creation. Thus, in Hadoop latency is very high.

**Large Data Sets** – In Hadoop, applications which are running require considerable data sets. Memory requirement can vary from gigabytes to terabytes.

**Moving Computation Vs Moving Data** – In HDFS, computation is moved to data. In Hadoop taking computation toward data is more efficient. HFDS provides interface which transfer application to data where it is located

### Name Node and Data Node

Hadoop Distributed File system follows Master-Slave architecture. Cluster is made in Hadoop, and cluster consists of single Name node which acts as master server which is user for managing file system namespace and it provides regulation for accessing files by client.

## Difference between Name Node and Data Node

Names node is used for executing file system namespace operations like closing, renaming files and directories whereas data node is responsible for reading and writing data. Name node

is responsible for mapping of blocks to data node while data node is used for creation, replication and deletion.

In HDFS file is divided into one or more blocks.

**Hard Link**

Hard link is a file that links a name with a file in distributed file system. There can be multiple hard links for a same file, we can create multiple names for same file and create aliasing effect for example if contents of file 1 are altered then these effects will be visible when the same file is opened with another name.

**Soft Link, Symbolic Link**

In HDFS, reference for another or directory is there in target file. Reference is in the form of relative path. If the link is deleted, target will not get affected. Also, if target is shifted or removed, even then it will point to old target and non- existing target will be broken.

**Replication Factor**

Replication factor is defined as number of copies should be maintained for particular file. Replication factor is stored in Name Node which maintains file system namespace.

**Data Replication**

Data replication is a main feature of HDFS. Data replication makes HDFS very reliable system that can store large files. In this, files are broken into blocks which are stored. All the blocks have same size except the last block. In order to provide reliability blocks are replicated. In HDFS block size and replication factor specified during creation, are not fixed and they can be changed. Name node receives block report and heartbeat in periodic intervals, thus ensuring data nodes are working properly. Block report contains list of all blocks in data node. Files can be written only once and name node makes decisions for replication of blocks.

**Replication Placement**

Optimization replica replacement distinguishes Hadoop distributed file system from other DFS.

The main goal of rack-aware replica placement policy is increase network bandwidth utilization, fault tolerance and data reliability and availability.

Rack-It is a combination of data nodes. In large networks, HDFS is run on cluster of computers which spread across multiple racks.

Two nodes at different racks communicate each other through switches.

Network bandwidth between machines of different racks is less than network bandwidth of machines in same rack.

In HDFS, policy of placing replicas on different racks is followed. This policy prevents loss of data during rack failure and it also allows use of bandwidth from multiple racks during reading of data. But this policy increases the cost of writing as multiple writes for different are required.

In HDFS Name node determines the rack ID, where each data node belongs to.

**Sample Case**

Let replication factor is 3. First replica is on one node in local rack. Second replica will be in other node in the same rack. While third replica will be on different rack.

**Advantages of Replica Placement Policy**

- ✓ It helps in increasing write performance.
- ✓ It ensures data reliability as the probability of rack failure is very less than chance of node failure.

**Replication Selection**

HDFS follows the policy of minimum distance rack policy, that is, it responds to the read request of the user by finding replica that is closest to the reader. If it finds replica and reader on the same rack, then it selects that replica. In HDFS cluster is spanned across multiple data. Centers and a replica which is present in local data center is preferred over remote replica (if present).

**Safe Mode**

When HDFS is started, Name node uses a special stage called safe mode.

When name node is in the safe mode then no replication occurs. In safe mode, name node receives heartbeat and block report from data nodes.

A block is safe if minimum numbers of replicas for that block are checked. In HDFS each block has minimum number of specified replicas.

Name node is said to be in safe mode when configurable percent of its replicated data blocks are verified.

**Persistence of File System Metadata**

In HDFS, name node stores the namespace.

Edit log is used by name node. Edit is basically a transaction log and it is stored in local host as file system.

For example- changing of replication factor, creating a new file.

Fsimage is used store Namespace plus file system property and mapping of blocks to file. Fsimage is stored in local file system of name nodes.

**Role of Data node during Start-up**

During start-up, data scans all its local file system and then it generates a list of all HDFS data blocks which represent each of the local files. After that, it sends the report to Name none. Report generated is called as Block Report.

In HDFS various events occurs: -

- ✓ It stores data in the local file system.

- ✓ Local file system contains separate files each containing a block.
- ✓ In HDFS all are not in single directory. It tries to find minimum number of files per directory and then creates a sub directory
- ✓ It is not efficient to create all local files in same directory as efficiency gets reduced.
- ✓ Replication and Data Disk Failure

- ✓ In HDFS, many times data done becomes unavailable due to which data is lost or replica may also get corrupted. Due to these reasons there is a need for Re- replication. Rereplication is also required when replication factor of file gets increased or hard disk on data node gets failed.

**Secondary Name Node**

Secondary name node is used for connecting with name node and builds snapshot of directory of primary name nodes.

**Advantages of Hadoop**

- ✓ In Hadoop, a code for ten 10 nodes can work for thousands nodes with little requirement of re-work.
- ✓ Hadoop uses easy programming model that enables clients to quickly perform their operations.
- ✓ Hadoop provides reliable data storage.
- ✓ It also provides efficient and dynamic of data.
- ✓ Hadoop can work across machines

Apache Hadoop Framework consists of:

**Hadoop Common** – It utilities and libraries which are needed by Hadoop components.

**Hadoop Distributed File System** – HDFS- The Hadoop Distributed File System (HDFS) is

designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers both host directly attached storage and execute user application tasks

**Hadoop YARN** – Yet another Resource Negotiator (YARN) is used to manage computer resource in cluster. These resources are used for scheduling user's application.

**Hadoop Map Reduce** – It is programming technique used for large scale processing of data. Map Reduce consists of one job tracker. In this clients submit map reduce tasks to job tracker.

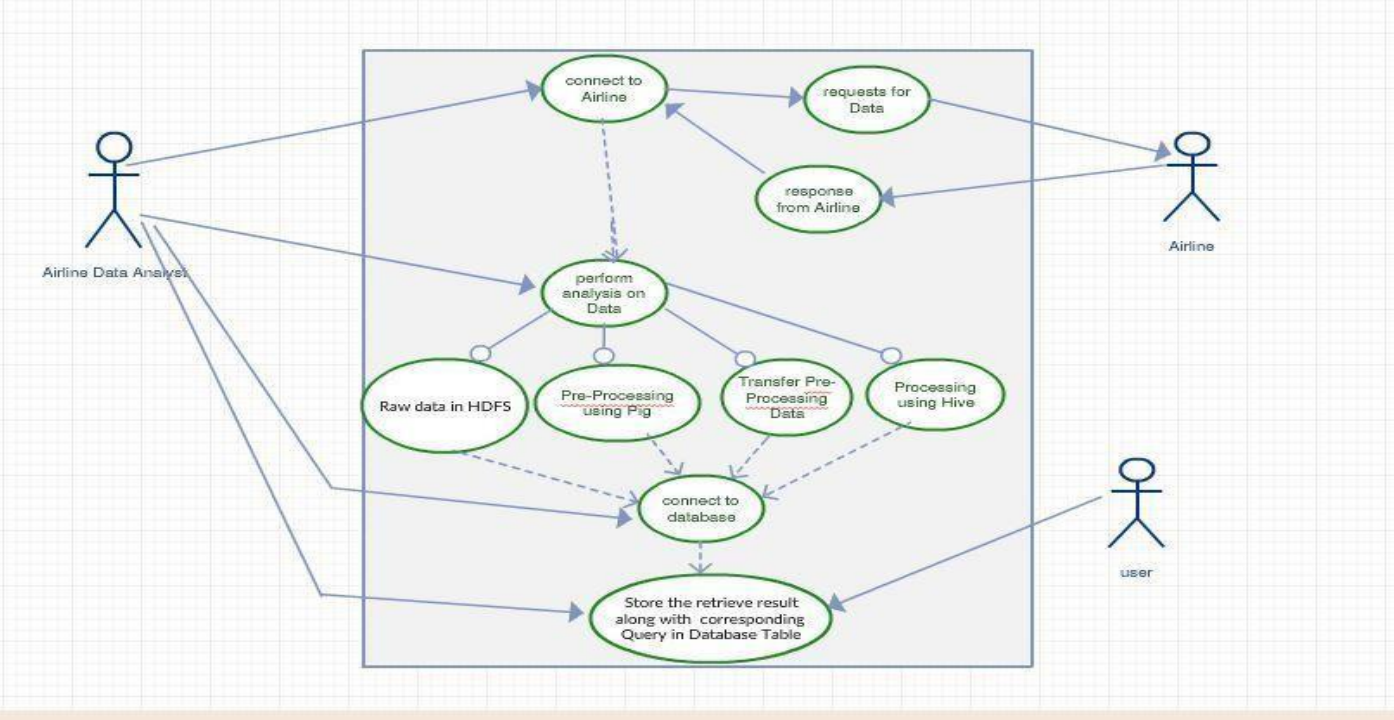
**Mappers and Reducer--**Mappers are the tasks which are used for processing records in isolation. In map-reduce architecture, output from mapper, combined together, is fed to second set of tasks called reducer. In reducer results of various mappers can be together combined.

# UML DIAGRAMS

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997. OMG is continuously making efforts to create a truly industry standard. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard.

## Use case Diagram:

There are three actors in our project first one is the airline data analyst, second is airline and the third one is the user. The role of the analyst is to connect to the airline and then create an API which give the access to the to extract the data from airline. After getting access from airline using API .we can extract the airline data. Afterwards we will put the data into a excel table and insert it into HDFS after which the analysis one the particular topic. The analyst will receive the output by which the client will use the particular data.




**Sequence Diagram:**

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. There are four objects in our project which are airline data analyst, system interface, airline and client. first of all the process starts by the analyst by creating an API for airline data . the analyst request for the excess of airline data and then the access grated by the airline. Now the role of airline is done here, after we will program the raw data into HDFS and then insert into the excel table than it will show the extract view of the data. Now we are ready to fire the command and then we get the particular data as our output and provide to the clients.

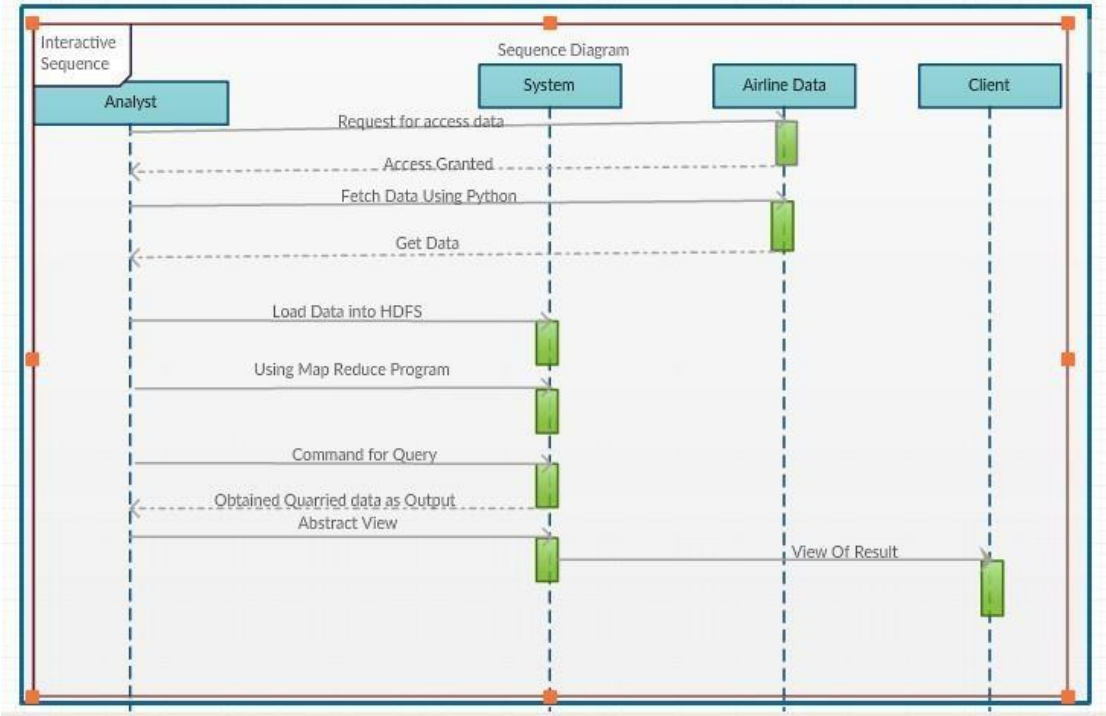
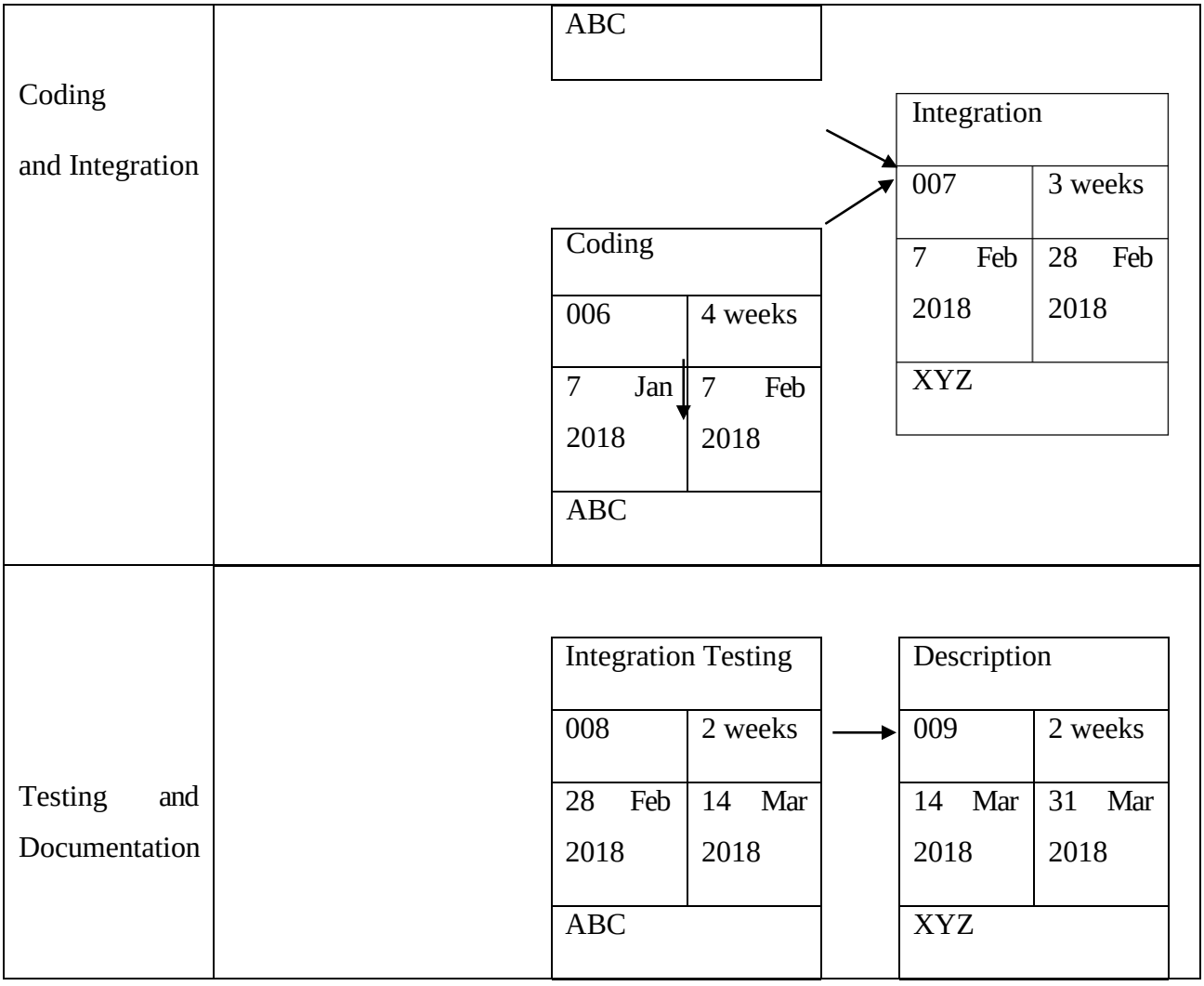



Figure: Sequence Diagram of Airline Analysis

## Gantt Chart:-

Gantt chart is a graphical depiction of a project schedule. A Gantt chart is a type of bar chart that shows the start and finish dates of several elements of a project that include resources, milestones, tasks and dependencies.

<table>
<tr>
<td>Planning</td>
<td>
<table>
<tr><td colspan="2">Analysis</td></tr>
<tr><td>001</td><td>3 weeks</td></tr>
<tr><td>10 Oct 2017</td><td>31 Oct 2017</td></tr>
<tr><td colspan="2">ABC</td></tr>
</table>
<table>
<tr><td colspan="2">Requirement</td></tr>
<tr><td>002</td><td>2 weeks</td></tr>
<tr><td>31 Oct 2017</td><td>15 Nov 2017</td></tr>
<tr><td colspan="2">XYZ</td></tr>
</table>
</td>
</tr>
<tr>
<td>Designing</td>
<td>
<table>
<tr><td colspan="2">High Level Design</td></tr>
<tr><td>003</td><td>2 weeks</td></tr>
<tr><td>15 Nov 2017</td><td>31 Nov 2017</td></tr>
<tr><td colspan="2">XYZ</td></tr>
</table>
<table>
<tr><td colspan="2">Detailed Design</td></tr>
<tr><td>004</td><td>3 weeks</td></tr>
<tr><td>31 Nov 2017</td><td>21 Dec 2017</td></tr>
<tr><td colspan="2">ABC</td></tr>
</table>
</td>
</tr>
<tr>
<td></td>
<td>
<table>
<tr><td colspan="2">Coding</td></tr>
<tr><td>005</td><td>3 weeks</td></tr>
<tr><td>21 Dec 2017</td><td>7 Jan 2018</td></tr>
</table>
</td>
</tr>
</table>


Coding
and Integration
ABC
Integration
007
3 weeks
7 Feb 2018
28 Feb 2018
XYZ
Coding
006
4 weeks
7 Jan 2018
7 Feb 2018
ABC
Testing and Documentation
Integration Testing
008
2 weeks
28 Feb 2018
14 Mar 2018
ABC
Description
009
2 weeks
14 Mar 2018
31 Mar 2018
XYZ

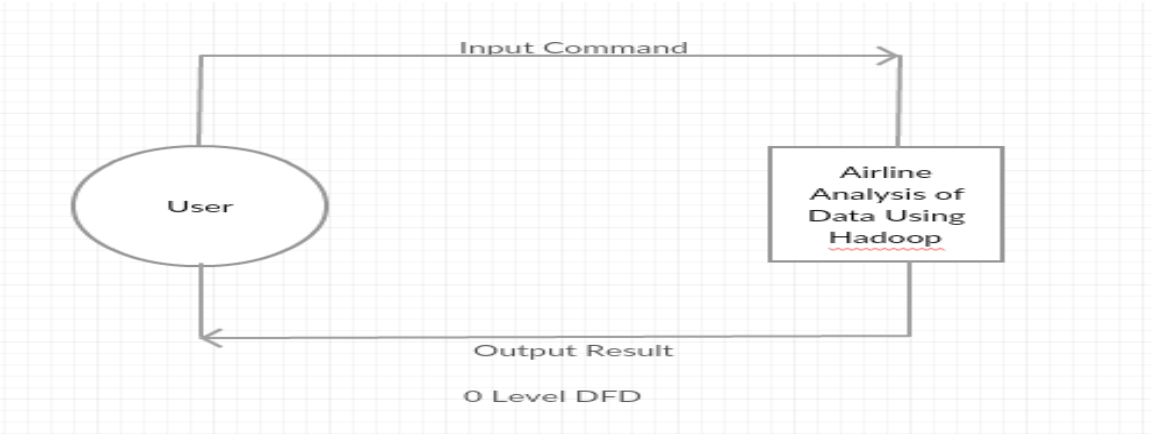
# DATA FLOW DIAGRAMS

## Introduction

Data flow diagrams are the basic building blocks that define the flow of data in a system to the particular destination and difference in the flow when any transformation happens. It makes whole procedure like a good document and makes simpler and easy to understand for both programmers and non-programmers by dividing into the sub process. The data flow diagrams are the simple blocks that reveal the relationship between various components of the system and provide high level overview, boundaries of particular system as well as provide detailed overview of system elements.

The data flow diagrams start from source and ends at the destination level i.e., it decomposes from high level to lower levels. The important things to remember about data flow diagrams are: it indicates the data flow for one way but not for loop structures and it doesn't indicate the time factors.

**Level 0 data flow diagram**



# HADOOP INSTALLATION AND SIMULATION[2]

### Supported Platforms

- ✓ GNU/Linux is supported as a development and production platform
- ✓ Windows is also a supported platform.

### Required Software

- ✓ Dataset
- ✓ UBUNTU - LINUX operating system
- ✓ APACHE HADOOP FRAMEWORK.
- ✓ Map Reduce

### Modes of working of Hadoop:

**STANDALONE MODE:** By default Hadoop is configured to run in a non- distributed mode, as a single Java process. This is useful for debugging.

**PSEUDO DISTRIBUTED MODE:** Hadoop can also be run on a single- node in a pseudo-distributed mode where each Hadoop daemon runs in a separate Java process.

**SIMULATIONS:** Very first code is to find and displays every match of the given regular ex-pression. Output is written to the output directory.

### Steps for installing of Hadoop:

Step 1 — Installing Java

To get started, we'll update our package list:

sudo apt-get update

SNext, we'll install Open JDK, the default Java Development Kit on Ubuntu 16.04.

sudo apt-get install default-jdk

Once the installation is complete, let's check the version.

java –version

Output

Openjdk version "1.8.0_91"

OpenJDK Runtime Environment (build 1.8.0_91-8u91-b14-3ubuntu1~16.04.1-b14)

OpenJDK 64-Bit Server VM (build 25.91-b14, mixed mode)

This output verifies that OpenJDK has been successfully installed.

Step 2 — Installing Hadoop

With Java in place, we'll visit the Apache Hadoop Releases page to find the most recent stable release. Follow the binary for the current release:

On the server, we'll use wget to fetch it:

wget http://apache.mirrors.tds.net/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz

In order to make sure that the file we downloaded hasn't been altered; we'll do a quick check using SHA-256. Return the releases page, and then follow the Apache link:

Dist –Revision 16478:/release/hadoop/common En-

ter the directory for the version you downloaded:

Hadoop-3.0.0

Finally, locate the .mds file for the release you downloaded, then copy the link for the corresponding file:hadoop-3.0.0.tar.gz.mds

Again, we'll right-click to copy the file location, then use wget to transfer the file:

we get https://dist.apache.org/repos/dist/release/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz.mds Then run the verification:shasum -a 256 hadoop-2.7.3.tar.gz

Output

d489df3808244b906eb38f4d081ba49e50c4603db03efd5e594a1e98b09259c2

hadoop-2.7.3.tar.gz

Compare this value with the SHA-256 value in the .mds file:

cat hadoop-2.7.3.tar.gz.mds

~/hadoop-2.7.3.tar.gz.mds

hadoop-2.7.3.tar.gz: SHA256 = D489DF38 08244B90 6EB38F4D 081BA49E 50C4603D B03EFD5E 594A1E98 B09259C2

You can safely ignore the difference in case and the spaces. The output of the command we ran against the file we downloaded from the mirror should match the value in the file we downloaded from apache.org.

Now that we've verified that the file wasn't corrupted or changed, we'll use the tar command with the -xflag to extract, -z to uncompress, -v for verbose output, and -f to specify that we're extracting from a file. Use tab-completion or substitute the correct version number in the command below:

tar -xzvf hadoop-2.7.3.tar.gz

Finally, we'll move the extracted files into /usr/local, the appropriate place for locally installed software. Change the version number, if needed, to match the version you downloaded.

sudo mv hadoop-2.7.3 /usr/local/hadoop

With the software in place, we're ready to configure its environment.

Step 3 — Configuring Hadoop's Java Home

Hadoop requires that you set the path to Java, either as an environment variable or in the Hadoop configuration file.

The path to Java, /usr/bin/java is a symlink to /etc/alternatives/java, which is in turn a symlink to default Java binary. We will use readlink with the -f flag to follow every symlink in every part of the path, recursively. Then, we'll use sed to trim bin/java from the output to give us the correct value for JAVA_HOME.

To find the default Java path

readlink -f /usr/bin/java | sed "s:bin/java::" Out-

put

/usr/lib/jvm/java-8-openjdk-amd64/jre/

You can copy this output to set Hadoop's Java home to this specific version, which ensures that if the default Java changes, this value will not. Alternatively, you can use the readlink command dynamically in the file so that Hadoop will automatically use whatever Java version is set as the system default.

To begin, open hadoop-env.sh:

sudonano /usr/local/hadoop/etc/hadoop/hadoop-env.sh

Then, choose one of the following options:

Option 1: Set a Static Value

/usr/local/hadoop/etc/hadoop/hadoop-env.sh

. . .

#export JAVA_HOME=${JAVA_HOME}

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre/

. . .

Option 2: Use Readlink to Set the Value Dynamically

/usr/local/hadoop/etc/hadoop/hadoop-env.sh #ex-

port JAVA_HOME=${JAVA_HOME}

export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")

Step 4 — Running Hadoop

Now we should be able to run Hadoop:

/usr/local/hadoop/bin/hadoop

Output

Usage: hadoop [OPTIONS] SUBCOMMAND [SUBCOMMAND OPTIONS]

Or hadoop [OPTIONS] CLASSNAME [CLASSNAME OPTIONS]

**CLASSNAME** is a user-provided Java class

**OPTIONS** is none or any of:

- ✓ buildpaths attempt to add class files from build tree
- ✓ --configdir Hadoop config directory
- ✓ --debug turn on shell script debug mode
- ✓ --help usage information
- ✓ hostnames list[of,host,names] hosts to use in slave mode
- ✓ hosts filename list of hosts to use in slave mode
- ✓ loglevel level set the log4j level for this command
- ✓ workers turn on worker mode

SUBCOMMAND is one of:

Admin Commands:

- ✓ daemonlog get/set the log level for each daemon

Client Commands:

- ✓ archivecreate a Hadoop archive

- ✓ checknativecheck native Hadoop and compression libraries availability
- ✓ classpathprints the class path needed to get the Hadoop jar
- ✓ conftestvalidate configuration XML files
- ✓ credentialinteract with credential providers
- ✓ distchdistributed metadata changer
- ✓ distcpcopy file or directories recursively
- ✓ dtutiloperations related to delegation tokens
- ✓ envvars display computed Hadoop environment variables
- ✓ fsrun a generic filesystem user client
- ✓ gridmixsubmit a mix of synthetic job, modeling a profiled
- ✓ jar<jar> run a jar file
- ✓ jnipathprints the java.library.path
- ✓ kerbname show auth_to_local principal conversion
- ✓ key manage keys via the KeyProvider
- ✓ rumenfolderscale a rumen input trace
- ✓ rumentrace convert logs into a rumen trace
- ✓ s3guard manage metadata on S3
- ✓ trace view and modify Hadoop tracing settings
- ✓ version print the version

Daemon Commands:

- ✓ KMS run KMS, the Key Management Server
- ✓ SUBCOMMAND may print help when invoked w/o parameters or with -h.

The help means we've successfully configured Hadoop to run in stand-alone mode. We'll ensure that it is functioning properly by running the example MapReduce program it ships with. To do so, create a directory called input in our home directory and copy Hadoop's configuration files into it to use those files as our data.

mkdir ~/input

cp /usr/local/hadoop/etc/hadoop/*.xml ~/input

Next, we can use the following command to run the MapReducehadoop-mapreduce-examples program, a Java archive with several options. We'll invoke its grep program, one of many examples included in hadoop-mapreduce-examples, followed by the input directory, input and the output directory grep_example. The MapReduce grep program will count the matches of a literal word or regular expression. Finally, we'll supply a regular expression to find occurrences of the word principal within or at the end of a declarative sentence. The expression is case-sensitive, so we wouldn't find the word if it were capitalized at the beginning of a sentence:

/usr/local/hadoop/bin/hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.0.jar grep ~/input ~/grep_example'principal[.]*'

When the task completes, it provides a summary of what has been processed and errors it has encountered, but this doesn't contain the actual results.

Output atjava.lang.reflect.Method.in-

voke(Method.java:498) atorg.apache.hadoop.util.Run-

Jar.run(RunJar.java:221) atorg.apache.hadoop.util.Run-

Jar.main(RunJar.java:136)

# MAP-REDUCE

## Introduction

Map-reduce refer to two distinct things; the programming model and the specific implementation of the framework. It is a programming model for data processing. The model is simple, yet not too simple to express useful programs in Hadoop can run map reduce programs written in various languages like java, ruby, python and C++.

Map-reduce programs are inherently parallel, thus putting very large scale data analysis into the hands of anyone with enough machines at her disposal. Map-reduce comeinto its own for large datasets.

Map-reduce works by breaking the processing into two phases: the map phase and the reduce phase. Each phase has key value as input, the types of which may be chosen by the programmer. The programmer also specifies two functions: the map function and the reduce function.

The input to our map phase is the raw NCDC data. We choose a text input format that gives us each line in the dataset as a text value. The key is offset of the beginning of the line from the beginning of the file, but as we have no need for this, we ignore it.

## Map function: -

Our map function is simple. we simply used the values of the datasets in which we are interested. The map function is just a data preparation phase setting up the data in such a way that the reducer function can do its work on it: for example, Finding the maximum temperature of the year. The map function is also good place to drop bad records; here we filter out temperatures that are missing, suspects, or erroneous.

**Reduce function: -**

Reduces a set of intermediate values which share a key to a smaller set of values.

Reducer has 3 primary phases:

Shuffle

Reducer is input the grouped output of a mapper. In the phase the framework, for each Reducer, fetches the relevant partition of the output of all the Mappers.

Sort

The framework groups Reducer inputs by keys (since different Mappers may have output the same key) in this stage. The shuffle and sort phases occur simultaneously i.e. while outputs are being fetched they are merged.

Reduce

In this phase the reduce (object, iterator, outputcollector, reporter) method is called for each < (key, (list of values)> pair in the grouped inputs

**Map-reduce architecture**[5]

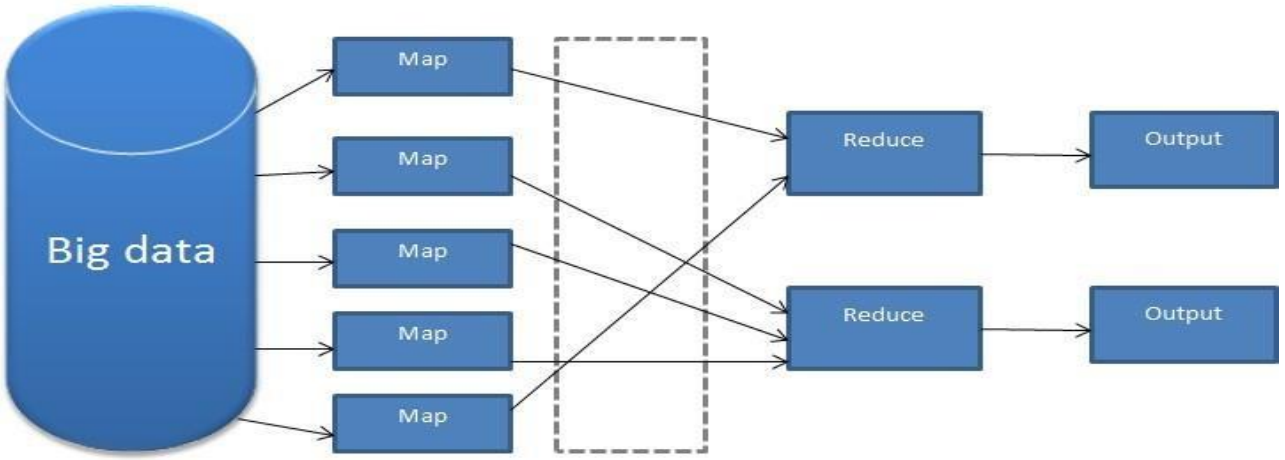



Figure 8.4: Diagram of Mapper and Reducer

There are basically two phases here mapper and reducer. The number of mapper used here is depends on the no. of blocks used in the HDFS. Firstly, mappers do the portioning of the blocks. A partitioner works like a condition in processing an input dataset. The partition phase takes place after the Map phase and before the Reduce phase. The number of partitioner is equal to the number of reducers. That means a partitioner will divide the data according to the number of reducers. After the Map phase and before the beginning of the Reduce phase is a handoff process, known as shuffle and sort. When the mapper task is complete, the results are sorted by key, partitioned if there are multiple reducers, and then written to disk. The reduce phase will then sum up the number of times each word was seen and write that sum count tog

## Benefits of map-reduce

Map Reduce is useful for batch processing on terabytes or petabytes of data stored in Apache Hadoop.

The following tables describe some of MapReduce's key benefits:

- ✓ Simplicity: - Developers can write their application in their language of own choice, such as java, C++ or python and map-reduce jobs are easy to run.
- ✓ Scalability: - Map-reduce can process petabytes of data, stored in HDFS of one cluster. Speed: - Parallel processing means that Map-reduce can take problems that used to take days to solve and solve them in few hours.
- ✓ Recovery: -Map-reduce takes care of failures if a machine with one copy of the data is unavailable, another machine has a copy of the same value/key pair, which can be used to solve the same sub task. The job tracker keep track of it all.
- ✓ Minimal data motion: - Map-reduce moves compute processes to the data on HDFS and not the other way round. Processing tasks can occur on the physical node where the data resides. This significantly reduces the network I/O patterns and contributes to Hadoop's processing speed. Ether with the word as output.

# FLUME[6]

## Introduction

Apache Flume is a tool/service/data ingestion mechanism for collecting aggregating and transporting large amounts of streaming data such as log files, events (etc...) from various sources to a centralized data store. Flume is a highly reliable, distributed, and configurable tool. It is principally designed to copy streaming data (log data) from various web servers to HDFS.

It is a very use case is collecting log data from one system-a bank of web servers, for example-and aggregating it in HDFS for later analysis. Flume supports a large variety of sources, some of the more commonly used ones include tail (which pipes data from a local file being written into the flume, just like Unix tail), syslog, and Apache log4j (allowing java applications to write events to files in HDFS via flume).

Flume nodes can be arranged in arbitrary topologies. Typically, there is a node running on each source machine (each web server, for example), with tiers of aggregating nodes that the data flows through on its way to HDFS. Flumes offers different levels of delivery reliability, from best effort delivery, which doesn't tolerate any flume node failures, to end to end, which guarantees delivery even in the event of multiple flume node failures between the source and HD.

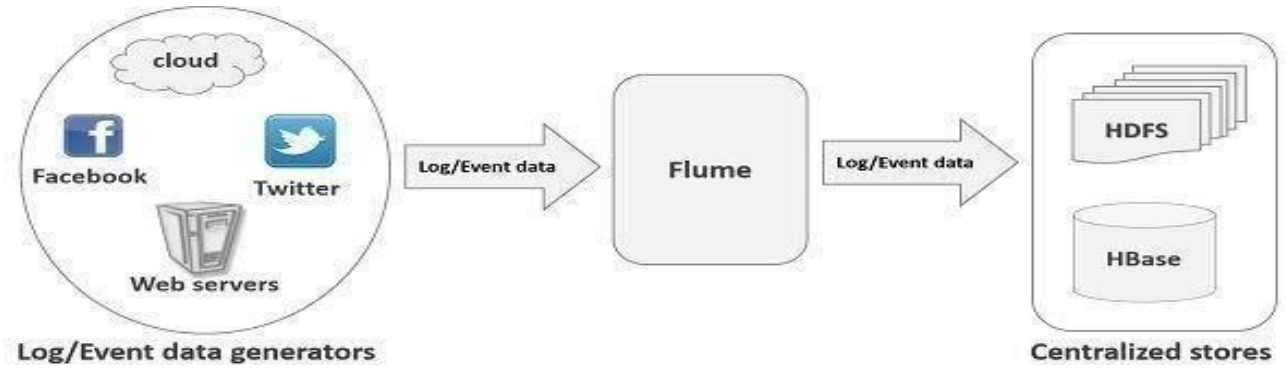



Figure: Structure of Apache Flume

## Applications of Flume

Assume an e-commerce web application wants to analyze the customer behavior from a particular region. To do so, they would need to move the available log data in to Hadoop for analysis. Here, Apache Flume comes to our rescue. Flume is used to move the log data generated by application servers into HDFS at a higher speed.

## Advantages of Flume

Here are the advantages of using Flume −

- ✓ Using Apache Flume we can store the data in to any of the centralized stores (HBase, HDFS).
- ✓ When the rate of incoming data exceeds the rate at which data can be written to the destination, Flume acts as a mediator between data producers and the centralized stores and provides a steady flow of data between them.
- ✓ Flume provides the feature of contextual routing.
- ✓ The transactions in Flume are channel-based where two transactions (one sender and one receiver) are maintained for each message. It guarantees reliable message delivery.
- ✓ Flume is reliable, fault tolerant, scalable, manageable, and customizable.

## What flume does

Flume lets Hadoop users ingest high-volume streaming data into HDFS for storage.

Specifically, Flume allows users to:

**Stream data:** - and analysis Ingest streaming data from multiple sources into Hadoop for storage.

**Insulate system:** - Buffer storage platform from transient spikes, when the rate of incoming data exceeds the rate at which data can be written to the destination.

**Guarantee data delivery**: - Flume NG uses channel-based transactions to guarantee reliable message delivery. When a message moves from one agent to another, two transactions are started; one on the agent that delivers the event and the other on the agent that receives the event. This ensures guaranteed delivery semantics

**Scale horizontally**: - To ingest new data streams and additional volume as needed.

## Features of Flume

Some of the notable features of Flume are as follows −

- ✓ Flume ingests log data from multiple web servers into a centralized store (HDFS, HBase) efficiently.
- ✓ Using Flume, we can get the data from multiple servers immediately into Hadoop. Along with the log files, Flume is also used to import huge volumes of event data of airlines and flights.
- ✓ Flume supports a large set of sources and destinations types.
- ✓ Flume supports multi-hop flows, fan-in fan-out flows, contextual routing, etc..Flume can be scaled horizontally.

# METHODOLOGY[8]

This Project is based on V-model **SDLC** (Software Development Life Cycle)

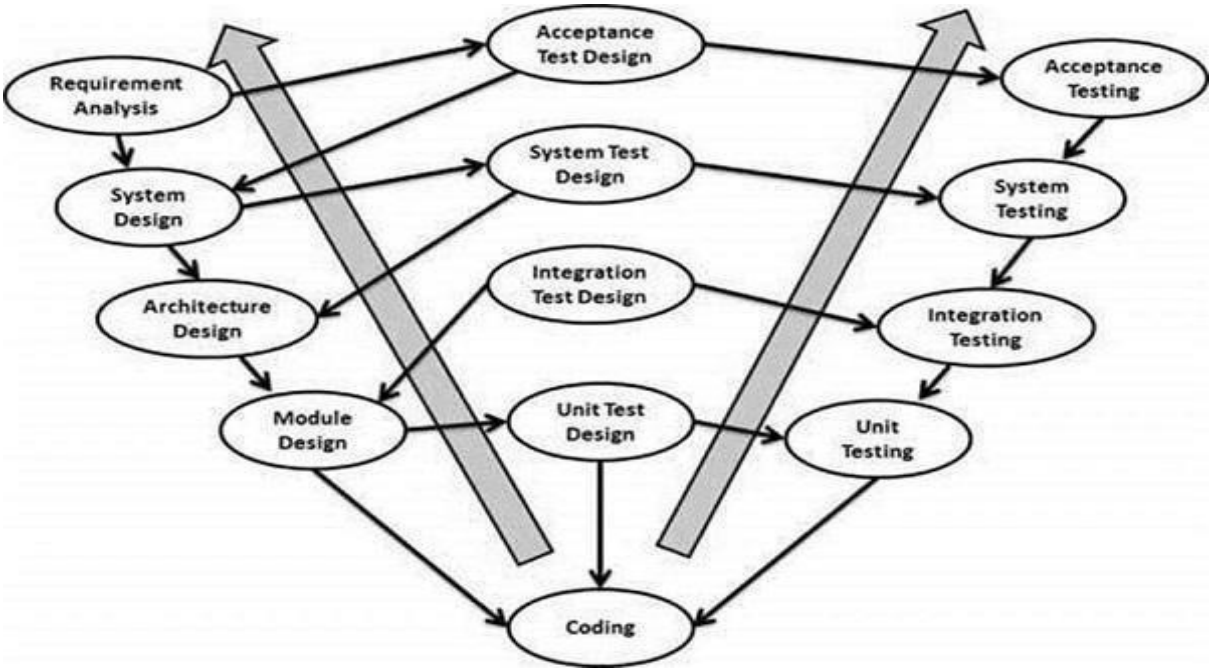
The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as Verification and Validation model.

The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

## V-Model -Design

Under the V-Model, the corresponding testing phase of the development phase is planned in parallel. So, there are Verification phases on one side of the 'V' and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model.

The following illustration depicts the different phases in a V-Model of the SDLC.




**V-Model - VerificationPhases**

There are several Verification phases in the V-Model, each of these are explained in detail below.

- ✓ Business Requirement Analysis

  This is the first phase in the development cycle where the product requirements are understood from the customer's perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and needs to be managed well, as most of the customers are not sure about what exactly they need. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.
- ✓ System Design

  Once you have the clear and detailed product requirements, it is time to design the complete system. The system design will have the understanding and detailing the complete hardware and communication setup for the product under development. The system test plan is developed based on the system design. Doing this at an earlier stage leaves more time for the actual test execution later.
- ✓ Architectural Design

Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. The system design is broken down further into modules taking up different functionality. This is also referred to as High Level Design (HLD).

The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.

- ✓ Module Design

  In this phase, the detailed internal design for all the system modules is specified, referred to as Low Level Design (LLD). It is important that the design is compatible with the other modules in the system architecture and the other external systems. The unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. These unit tests can be designed at this stage based on the internal module designs.

## Coding Phase

The actual coding of the system modules designed in the design phase is taken up in the Coding phase. The best suitable programming language is decided based on the system and architectural requirements.

The coding is performed based on the coding guidelines and standards. The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.

## ValidationPhases

The different Validation Phases in a V-Model are explained in detail below.

Unit Testing

Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.

### Integration Testing

Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

### System Testing

System testing is directly associated with the system design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during this system test execution.

### Acceptance Testing

Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non- functional issues such as load and performance defects in the actual user environment.

## V- Model– Application

V- Model application is almost the same as the waterfall model, as both the models are of sequential type. Requirements have to be very clear before the project starts, because it is usually expensive to go back and make changes. This model is used in the medical development field, as it is strictly a disciplined domain.

The following pointers are some of the most suitable scenarios to use the V-Model application.

- ✓ Requirements are well defined, clearly documented and fixed.
- ✓ Product definition is stable.
- ✓ Technology is not dynamic and is well understood by the project team.
- ✓ There are no ambiguous or undefined requirements.
- ✓ The project is short.

## V-Model-ProsandCons

The advantage of the V-Model method is that it is very easy to understand and apply. The simplicity of this model also makes it easier to manage. The disadvantage is that the model is not flexible to changes and just in case there is a requirement change, which is very common in today's dynamic world, it becomes very expensive to make the change.

The advantages of the V-Model method are as follows −

- ✓ This is a highly-disciplined model and Phases are completed one at a time.
- ✓ Works well for smaller projects where requirements are very well understood.
- ✓ Simple and easy to understand and use.
- ✓ Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.

The disadvantages of the V-Model method are as follows −

- ✓ High risk and uncertainty.
- ✓ Not a good model for complex and object-oriented projects.
- ✓ Poor model for long and ongoing projects.
- ✓ Not suitable for the projects where requirements are at a moderate to high risk of changing.
- ✓ Once an application is in the testing stage, it is difficult to go back and change functionality.

In this paper the tools used for the proposed method is Hadoop,map reduce which is mainly used for structured data. Assuming all the Hadoop tools have been installed and having semi structured information on airport data. The above mentioned queries have to be addressed Methodology used is as follows:

1. Create tables with required attributes

2. Extract semi structured data into table using the load a command

3. Analyse data for the following queries

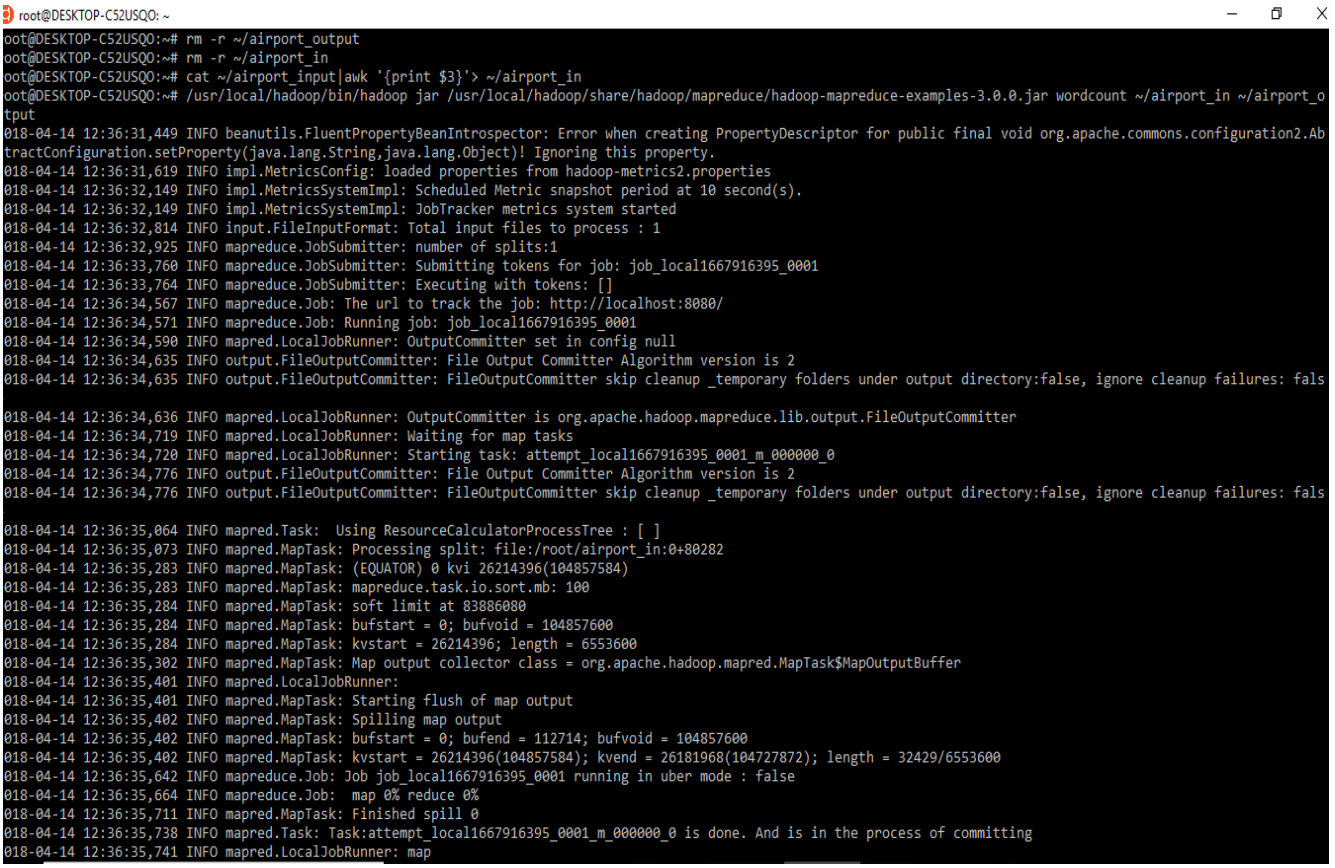
Q1. Find list of Airports operating in the Country India

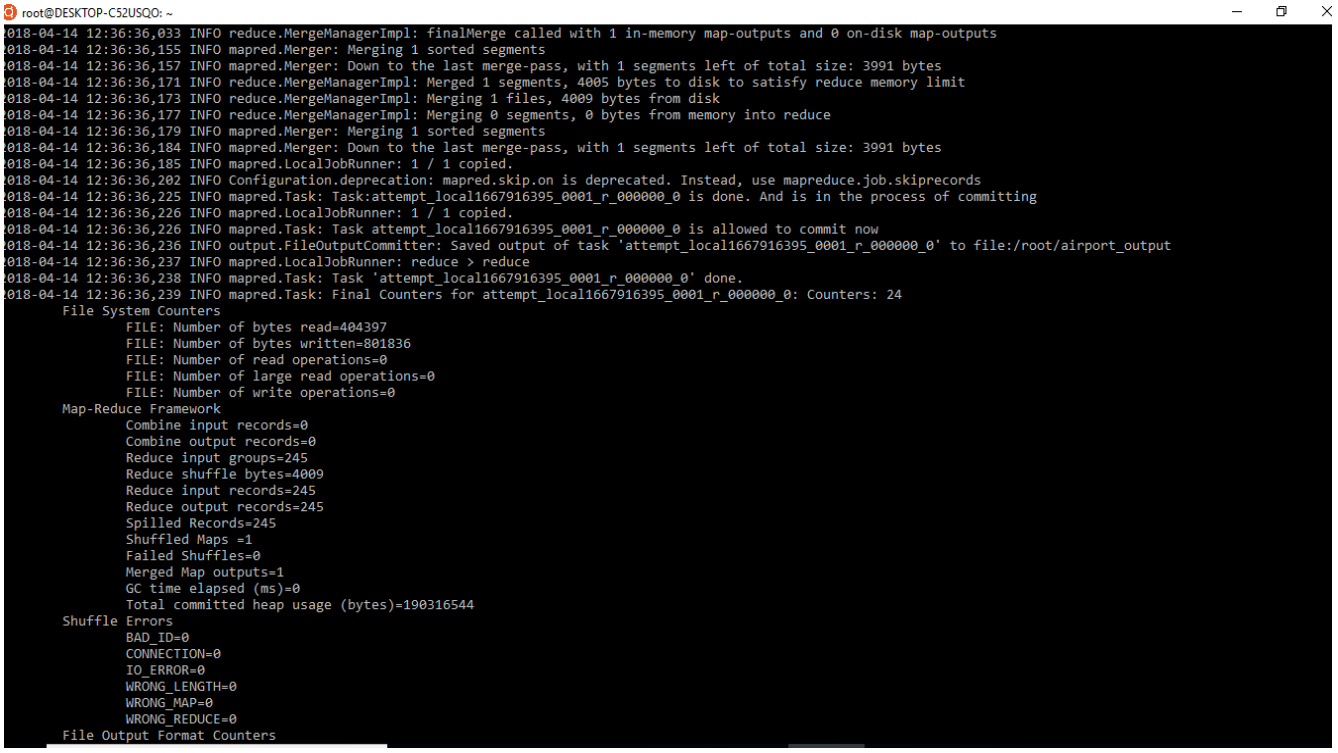
Step 1 rm -r ~/airport_output

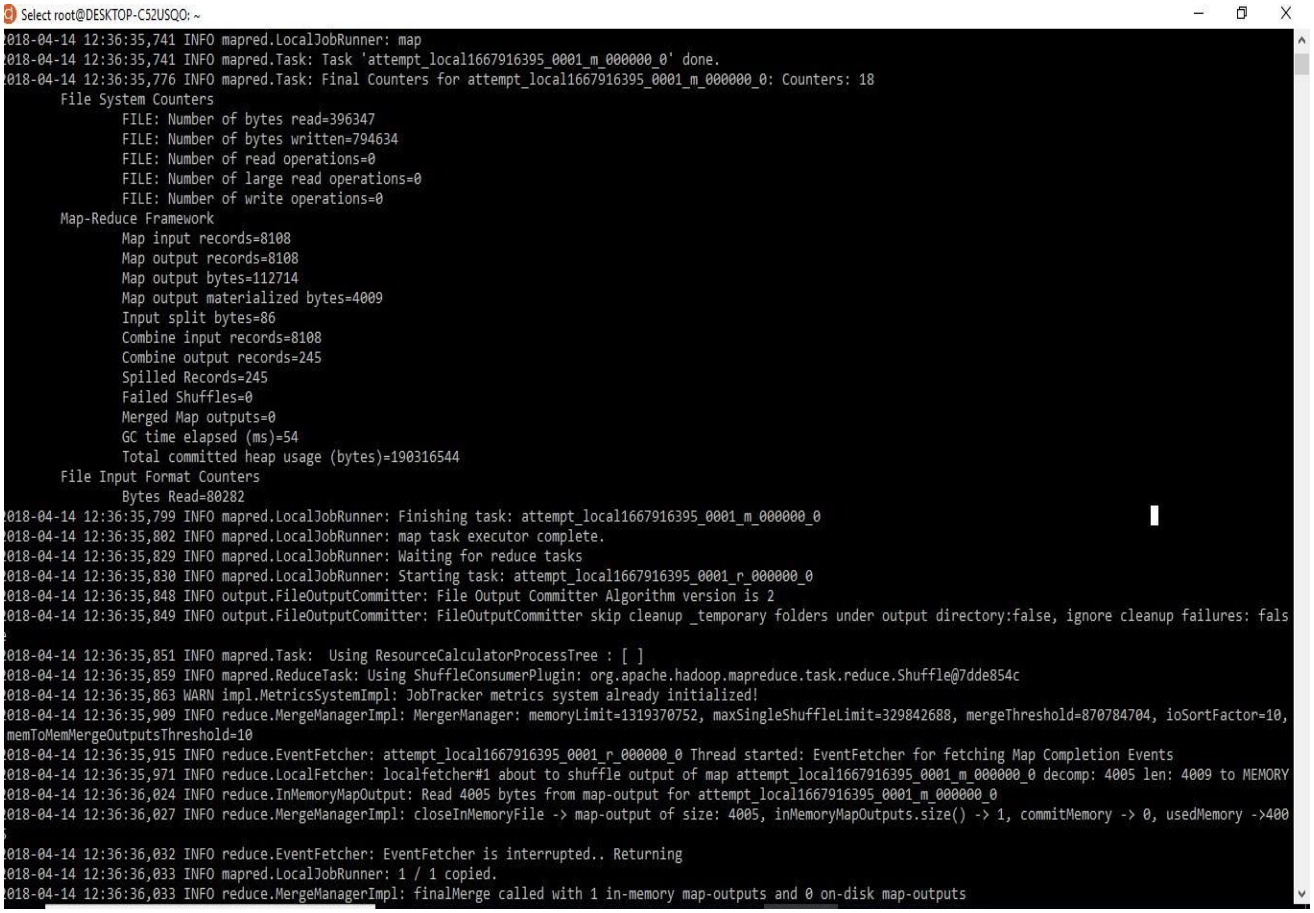
Step 2 rm -r ~/airport_in

Step 3　cat ~/airport_input|awk '{print $3}'> ~/airport_in

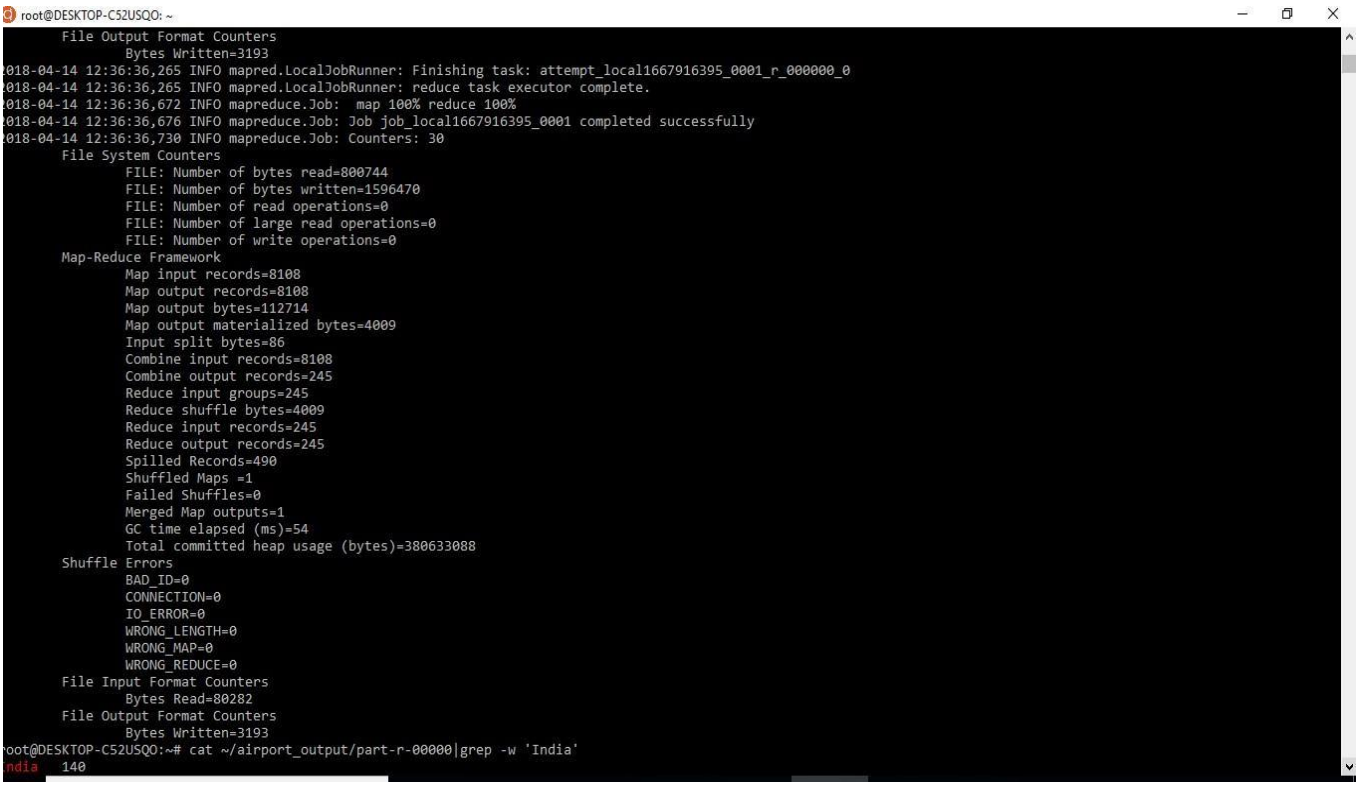
Step 4 /usr/local/hadoop/bin/hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.0.jar wordcount ~/airport_in ~/airport_output




root@DESKTOP-C52USQO: ~
2018-04-14 12:36:36,033 INFO reduce.MergeManagerImpl: finalMerge called with 1 in-memory map-outputs and 0 on-disk map-outputs
2018-04-14 12:36:36,155 INFO mapred.Merger: Merging 1 sorted segments
2018-04-14 12:36:36,157 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 3991 bytes
2018-04-14 12:36:36,171 INFO reduce.MergeManagerImpl: Merged 1 segments, 4005 bytes to disk to satisfy reduce memory limit
2018-04-14 12:36:36,173 INFO reduce.MergeManagerImpl: Merging 1 files, 4009 bytes from disk
2018-04-14 12:36:36,177 INFO reduce.MergeManagerImpl: Merging 0 segments, 0 bytes from memory into reduce
2018-04-14 12:36:36,179 INFO mapred.Merger: Merging 1 sorted segments
2018-04-14 12:36:36,184 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 3991 bytes
2018-04-14 12:36:36,185 INFO mapred.LocalJobRunner: 1 / 1 copied.
2018-04-14 12:36:36,202 INFO Configuration.deprecation: mapred.skip.on is deprecated. Instead, use mapreduce.job.skiprecords
2018-04-14 12:36:36,225 INFO mapred.Task: Task:attempt_local1667916395_0001_r_000000_0 is done. And is in the process of committing
2018-04-14 12:36:36,226 INFO mapred.LocalJobRunner: 1 / 1 copied.
2018-04-14 12:36:36,226 INFO mapred.Task: Task attempt_local1667916395_0001_r_000000_0 is allowed to commit now
2018-04-14 12:36:36,236 INFO output.FileOutputCommitter: Saved output of task 'attempt_local1667916395_0001_r_000000_0' to file:/root/airport_output
2018-04-14 12:36:36,237 INFO mapred.LocalJobRunner: reduce > reduce
2018-04-14 12:36:36,238 INFO mapred.Task: Task 'attempt_local1667916395_0001_r_000000_0' done.
2018-04-14 12:36:36,239 INFO mapred.Task: Final Counters for attempt_local1667916395_0001_r_000000_0: Counters: 24
File System Counters
FILE: Number of bytes read=404397
FILE: Number of bytes written=801836
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
Map-Reduce Framework
Combine input records=0
Combine output records=0
Reduce input groups=245
Reduce shuffle bytes=4009
Reduce input records=245
Reduce output records=245
Spilled Records=245
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=0
Total committed heap usage (bytes)=190316544
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Output Format Counters


Select root@DESKTOP-C52USQO: ~
2018-04-14 12:36:35,741 INFO mapred.LocalJobRunner: map
2018-04-14 12:36:35,741 INFO mapred.Task: Task 'attempt_local1667916395_0001_m_000000_0' done.
2018-04-14 12:36:35,776 INFO mapred.Task: Final Counters for attempt_local1667916395_0001_m_000000_0: Counters: 18
File System Counters
FILE: Number of bytes read=396347
FILE: Number of bytes written=794634
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
Map-Reduce Framework
Map input records=8108
Map output records=8108
Map output bytes=112714
Map output materialized bytes=4009
Input split bytes=86
Combine input records=8108
Combine output records=245
Spilled Records=245
Failed Shuffles=0
Merged Map outputs=0
GC time elapsed (ms)=54
Total committed heap usage (bytes)=190316544
File Input Format Counters
Bytes Read=80282
2018-04-14 12:36:35,799 INFO mapred.LocalJobRunner: Finishing task: attempt_local1667916395_0001_m_000000_0
2018-04-14 12:36:35,802 INFO mapred.LocalJobRunner: map task executor complete.
2018-04-14 12:36:35,829 INFO mapred.LocalJobRunner: Waiting for reduce tasks
2018-04-14 12:36:35,830 INFO mapred.LocalJobRunner: Starting task: attempt_local1667916395_0001_r_000000_0
2018-04-14 12:36:35,848 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2018-04-14 12:36:35,849 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: fals
e
2018-04-14 12:36:35,851 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]
2018-04-14 12:36:35,859 INFO mapred.ReduceTask: Using ShuffleConsumerPlugin: org.apache.hadoop.mapreduce.task.reduce.Shuffle@7dde854c
2018-04-14 12:36:35,863 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2018-04-14 12:36:35,909 INFO reduce.MergeManagerImpl: MergerManager: memoryLimit=1319370752, maxSingleShuffleLimit=329842688, mergeThreshold=870784704, ioSortFactor=10,
memToMemMergeOutputsThreshold=10
2018-04-14 12:36:35,915 INFO reduce.EventFetcher: attempt_local1667916395_0001_r_000000_0 Thread started: EventFetcher for fetching Map Completion Events
2018-04-14 12:36:35,971 INFO reduce.LocalFetcher: localfetcher#1 about to shuffle output of map attempt_local1667916395_0001_m_000000_0 decomp: 4005 len: 4009 to MEMORY
2018-04-14 12:36:36,024 INFO reduce.InMemoryMapOutput: Read 4005 bytes from map-output for attempt_local1667916395_0001_m_000000_0
2018-04-14 12:36:36,027 INFO reduce.MergeManagerImpl: closeInMemoryFile -> map-output of size: 4005, inMemoryMapOutputs.size() -> 1, commitMemory -> 0, usedMemory ->400
5
2018-04-14 12:36:36,032 INFO reduce.EventFetcher: EventFetcher is interrupted.. Returning
2018-04-14 12:36:36,033 INFO mapred.LocalJobRunner: 1 / 1 copied.
2018-04-14 12:36:36,033 INFO reduce.MergeManagerImpl: finalMerge called with 1 in-memory map-outputs and 0 on-disk map-outputs

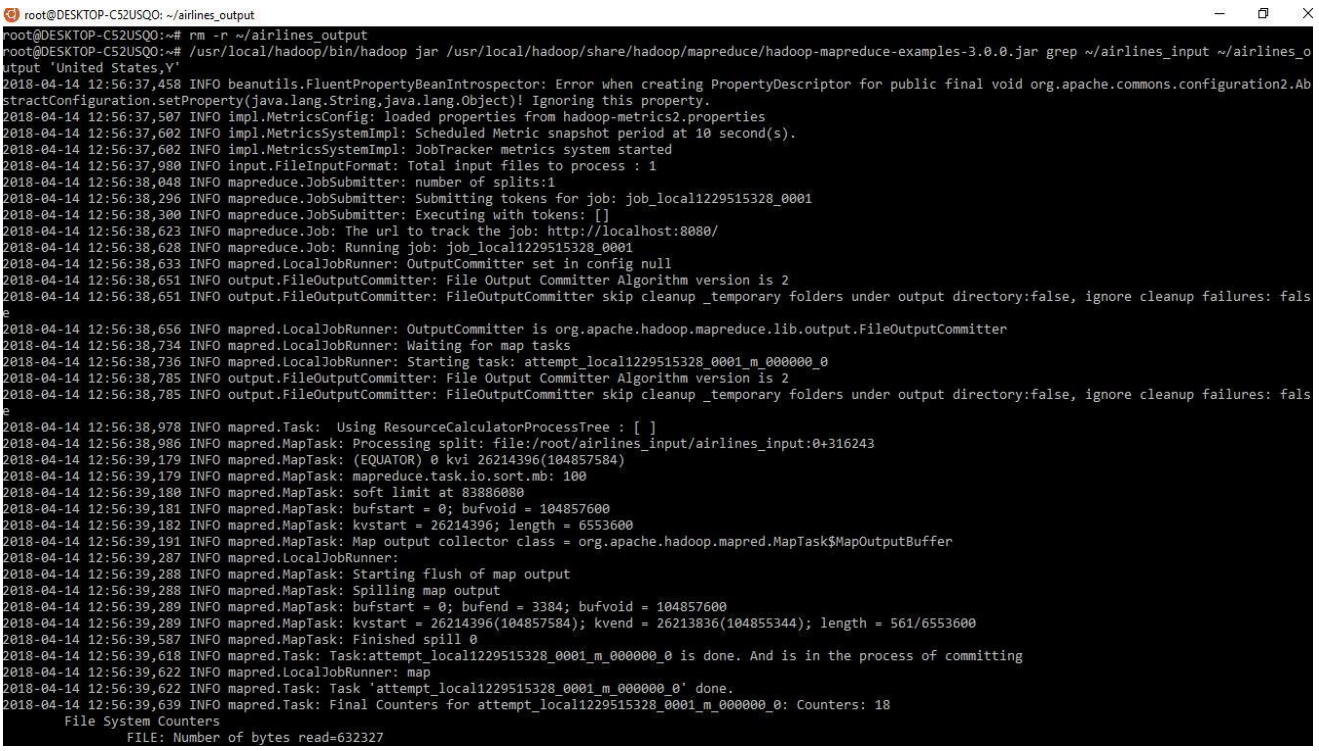
Step 5 cat ~/airport_output/part-r-00000|grep -w 'India'

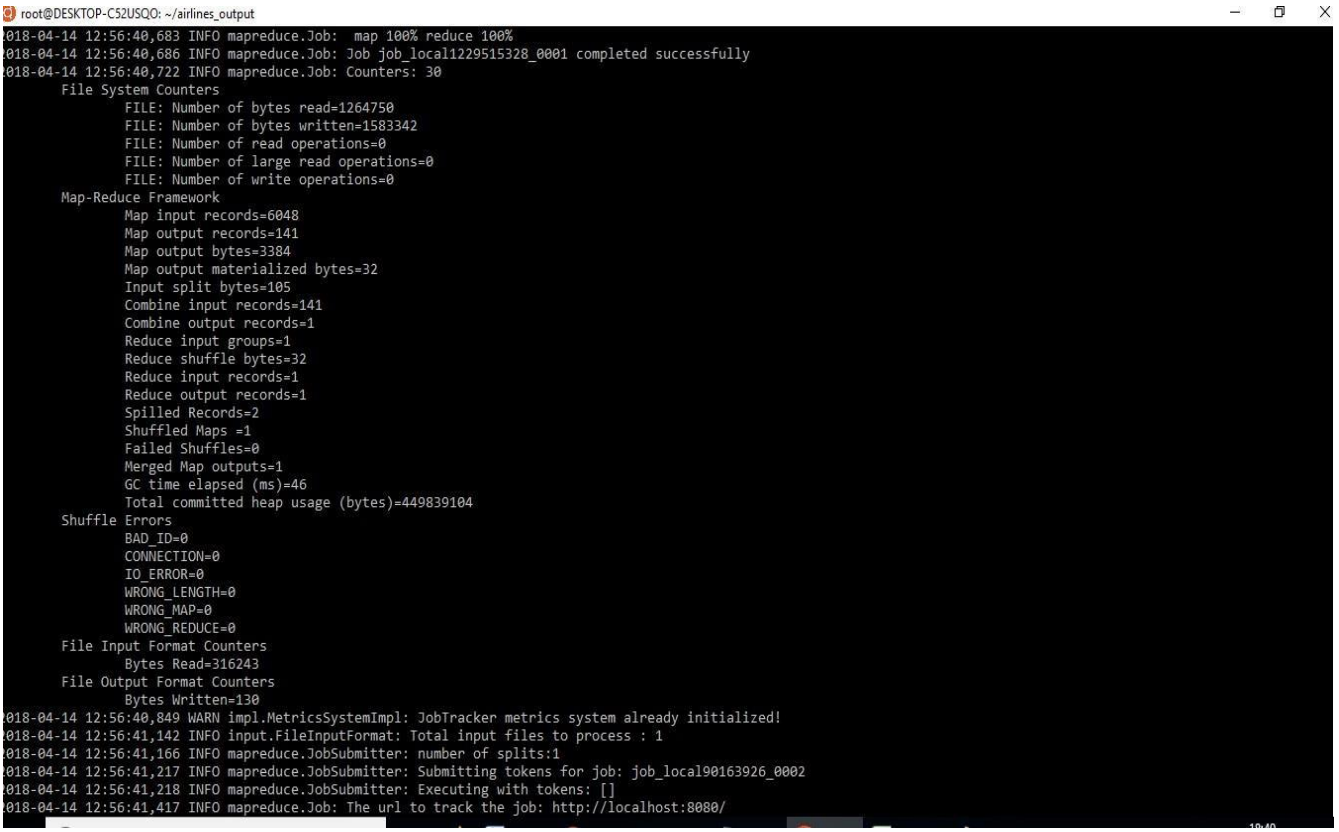


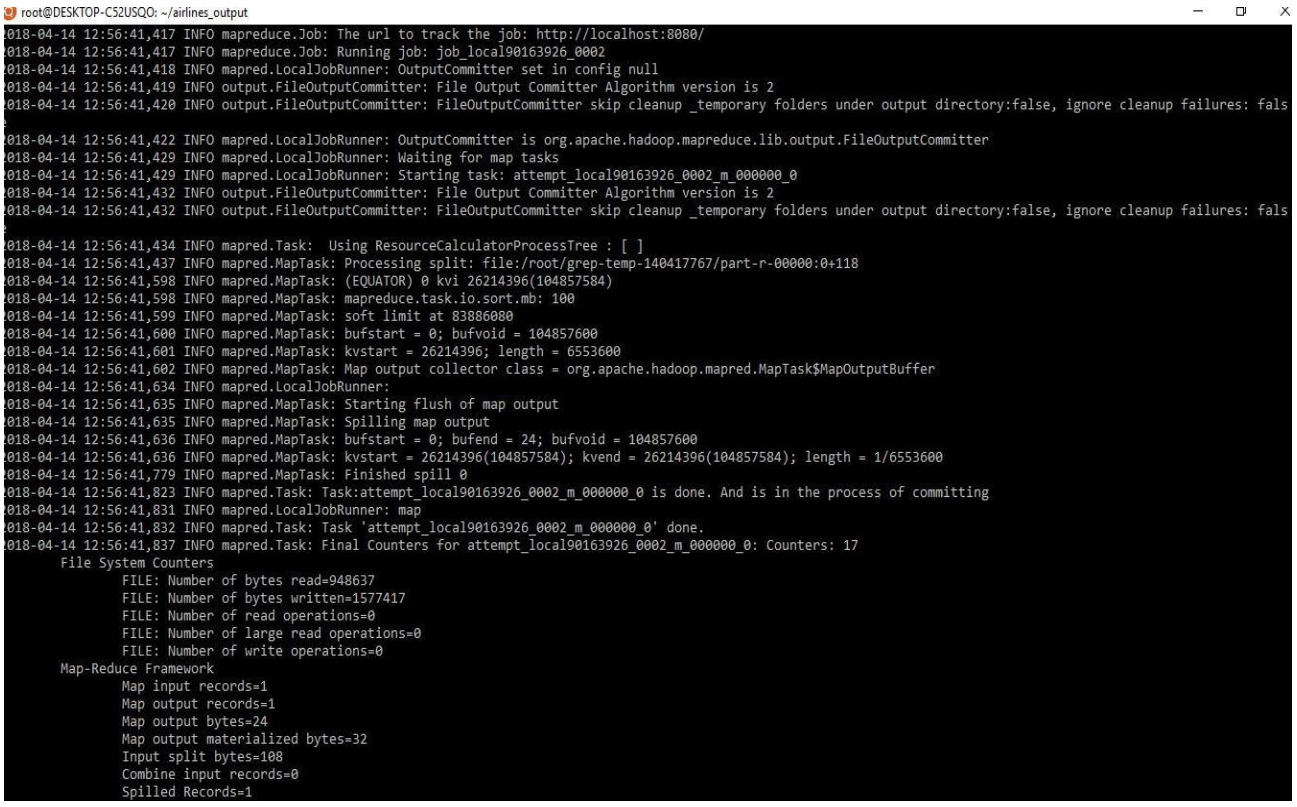
Q2. How many Active Airlines in United state.

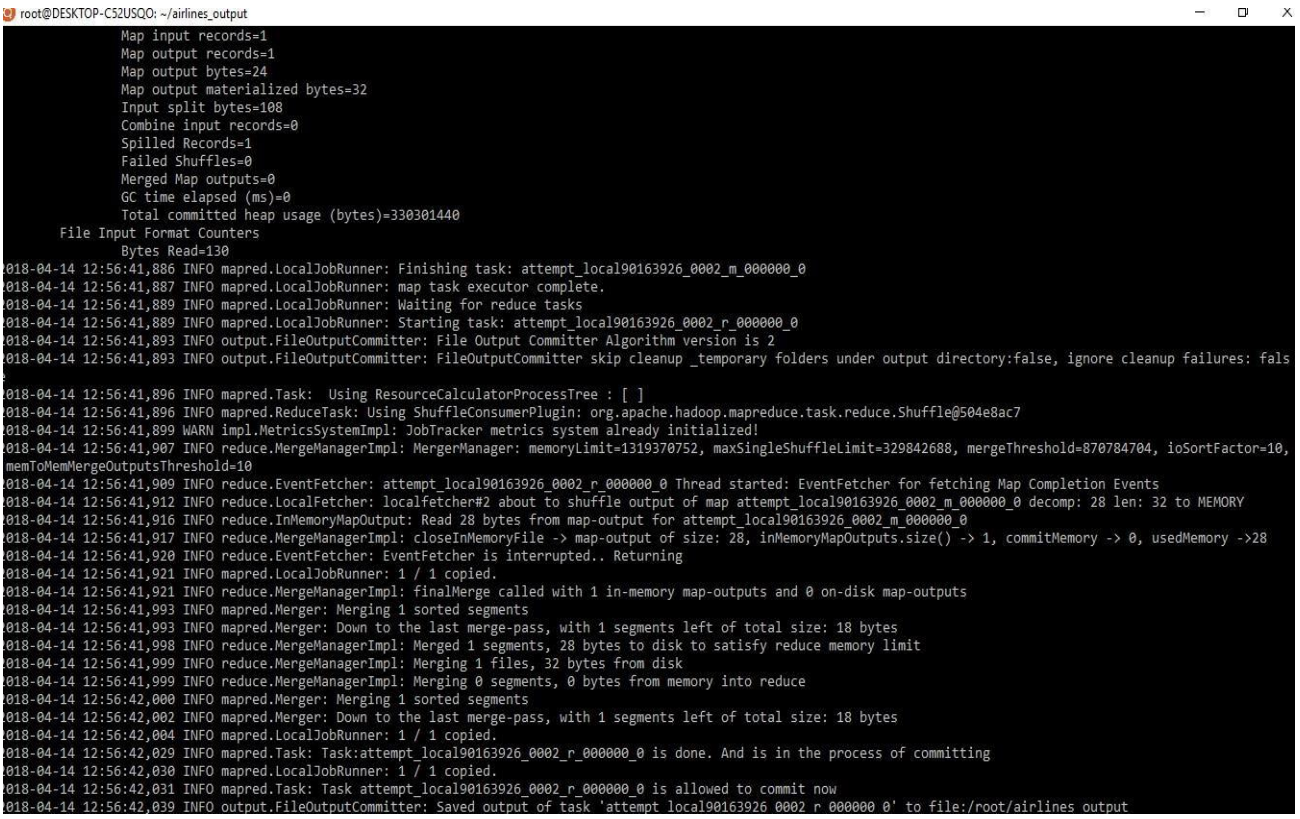
Step 1 rm -r ~/airlines_output

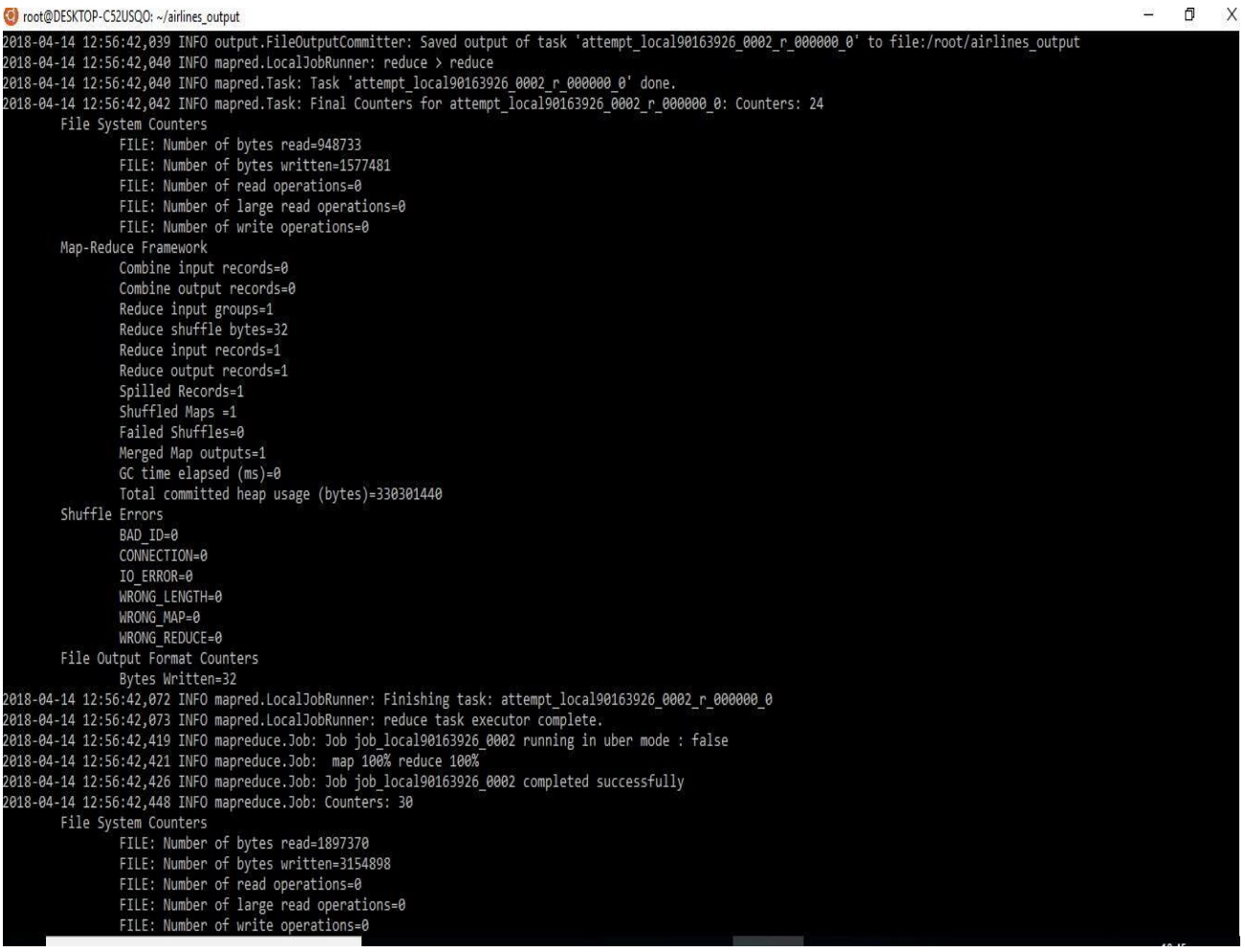
Step 2 /usr/local/hadoop/bin/hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.0.jar grep ~/airlines_input ~/airlines_output 'United States,Y'




root@DESKTOP-C52USQO: ~/airlines_output
2018-04-14 12:56:40,683 INFO mapreduce.Job:  map 100% reduce 100%
2018-04-14 12:56:40,686 INFO mapreduce.Job: Job job_local1229515328_0001 completed successfully
2018-04-14 12:56:40,722 INFO mapreduce.Job: Counters: 30
File System Counters
FILE: Number of bytes read=1264750
FILE: Number of bytes written=1583342
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
Map-Reduce Framework
Map input records=6048
Map output records=141
Map output bytes=3384
Map output materialized bytes=32
Input split bytes=105
Combine input records=141
Combine output records=1
Reduce input groups=1
Reduce shuffle bytes=32
Reduce input records=1
Reduce output records=1
Spilled Records=2
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=46
Total committed heap usage (bytes)=449839104
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=316243
File Output Format Counters
Bytes Written=130
2018-04-14 12:56:40,849 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2018-04-14 12:56:41,142 INFO input.FileInputFormat: Total input files to process : 1
2018-04-14 12:56:41,166 INFO mapreduce.JobSubmitter: number of splits:1
2018-04-14 12:56:41,217 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local90163926_0002
2018-04-14 12:56:41,218 INFO mapreduce.JobSubmitter: Executing with tokens: []
2018-04-14 12:56:41,417 INFO mapreduce.Job: The url to track the job: http://localhost:8080/


root@DESKTOP-C52USQO: ~/airlines_output
2018-04-14 12:56:41,417 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2018-04-14 12:56:41,417 INFO mapreduce.Job: Running job: job_local90163926_0002
2018-04-14 12:56:41,418 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2018-04-14 12:56:41,419 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2018-04-14 12:56:41,420 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: fals
e
2018-04-14 12:56:41,422 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2018-04-14 12:56:41,429 INFO mapred.LocalJobRunner: Waiting for map tasks
2018-04-14 12:56:41,429 INFO mapred.LocalJobRunner: Starting task: attempt_local90163926_0002_m_000000_0
2018-04-14 12:56:41,432 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2018-04-14 12:56:41,432 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: fals
e
2018-04-14 12:56:41,434 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2018-04-14 12:56:41,437 INFO mapred.MapTask: Processing split: file:/root/grep-temp-140417767/part-r-00000:0+118
2018-04-14 12:56:41,598 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2018-04-14 12:56:41,598 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2018-04-14 12:56:41,599 INFO mapred.MapTask: soft limit at 83886080
2018-04-14 12:56:41,600 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2018-04-14 12:56:41,601 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2018-04-14 12:56:41,602 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2018-04-14 12:56:41,634 INFO mapred.LocalJobRunner:
2018-04-14 12:56:41,635 INFO mapred.MapTask: Starting flush of map output
2018-04-14 12:56:41,635 INFO mapred.MapTask: Spilling map output
2018-04-14 12:56:41,636 INFO mapred.MapTask: bufstart = 0; bufend = 24; bufvoid = 104857600
2018-04-14 12:56:41,636 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 26214396(104857584); length = 1/6553600
2018-04-14 12:56:41,779 INFO mapred.MapTask: Finished spill 0
2018-04-14 12:56:41,823 INFO mapred.Task: Task:attempt_local90163926_0002_m_000000_0 is done. And is in the process of committing
2018-04-14 12:56:41,831 INFO mapred.LocalJobRunner: map
2018-04-14 12:56:41,832 INFO mapred.Task: Task 'attempt_local90163926_0002_m_000000_0' done.
2018-04-14 12:56:41,837 INFO mapred.Task: Final Counters for attempt_local90163926_0002_m_000000_0: Counters: 17
File System Counters
FILE: Number of bytes read=948637
FILE: Number of bytes written=1577417
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
Map-Reduce Framework
Map input records=1
Map output records=1
Map output bytes=24
Map output materialized bytes=32
Input split bytes=108
Combine input records=0
Spilled Records=1

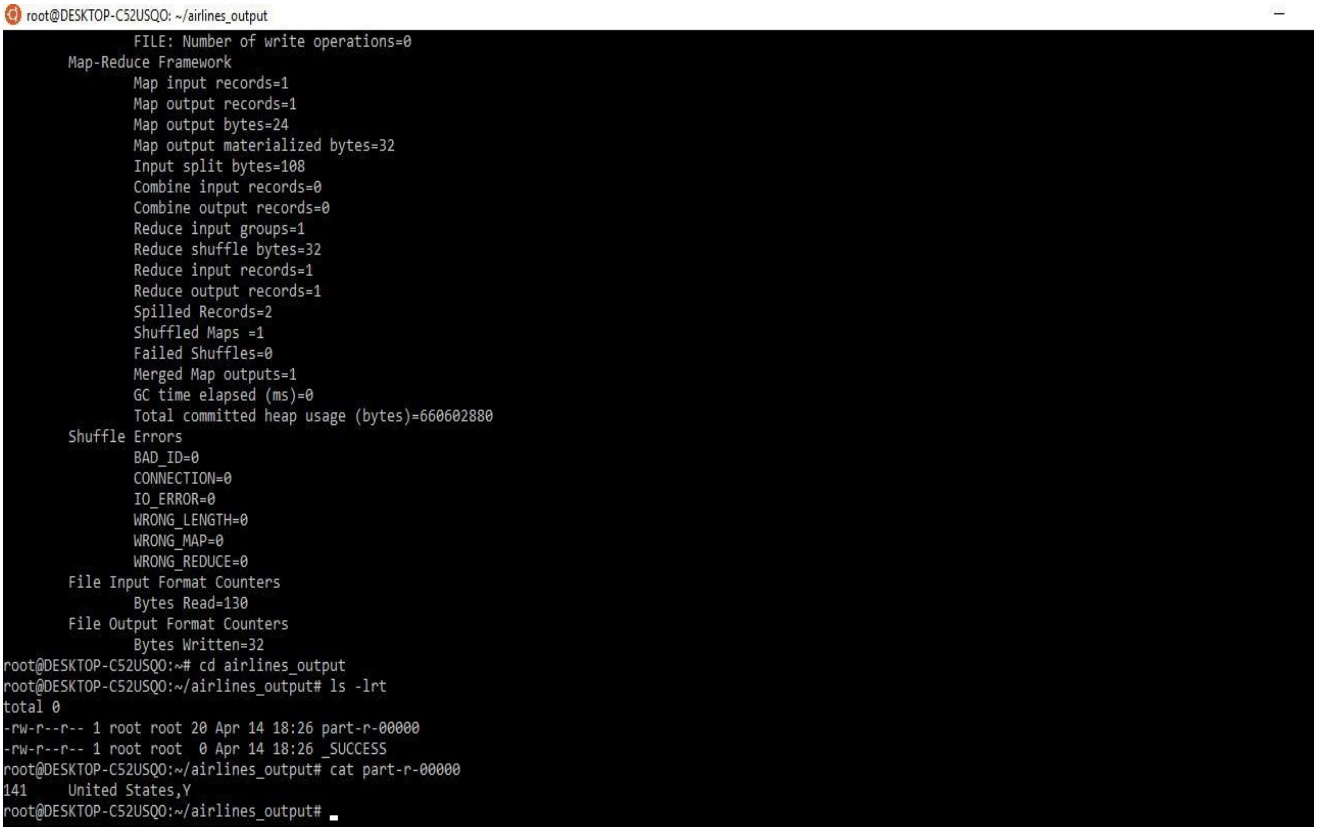

root@DESKTOP-C52USQO: ~/airlines_output
Map input records=1
Map output records=1
Map output bytes=24
Map output materialized bytes=32
Input split bytes=108
Combine input records=0
Spilled Records=1
Failed Shuffles=0
Merged Map outputs=0
GC time elapsed (ms)=0
Total committed heap usage (bytes)=330301440
File Input Format Counters
Bytes Read=130
2018-04-14 12:56:41,886 INFO mapred.LocalJobRunner: Finishing task: attempt_local90163926_0002_m_000000_0
2018-04-14 12:56:41,887 INFO mapred.LocalJobRunner: map task executor complete.
2018-04-14 12:56:41,889 INFO mapred.LocalJobRunner: Waiting for reduce tasks
2018-04-14 12:56:41,889 INFO mapred.LocalJobRunner: Starting task: attempt_local90163926_0002_r_000000_0
2018-04-14 12:56:41,893 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2018-04-14 12:56:41,893 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: fals
e
2018-04-14 12:56:41,896 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]
2018-04-14 12:56:41,896 INFO mapred.ReduceTask: Using ShuffleConsumerPlugin: org.apache.hadoop.mapreduce.task.reduce.Shuffle@504e8ac7
2018-04-14 12:56:41,899 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2018-04-14 12:56:41,907 INFO reduce.MergeManagerImpl: MergerManager: memoryLimit=1319370752, maxSingleShuffleLimit=329842688, mergeThreshold=870784704, ioSortFactor=10,
memToMemMergeOutputsThreshold=10
2018-04-14 12:56:41,909 INFO reduce.EventFetcher: attempt_local90163926_0002_r_000000_0 Thread started: EventFetcher for fetching Map Completion Events
2018-04-14 12:56:41,912 INFO reduce.LocalFetcher: localfetcher#2 about to shuffle output of map attempt_local90163926_0002_m_000000_0 decomp: 28 len: 32 to MEMORY
2018-04-14 12:56:41,916 INFO reduce.InMemoryMapOutput: Read 28 bytes from map-output for attempt_local90163926_0002_m_000000_0
2018-04-14 12:56:41,917 INFO reduce.MergeManagerImpl: closeInMemoryFile -> map-output of size: 28, inMemoryMapOutputs.size() -> 1, commitMemory -> 0, usedMemory ->28
2018-04-14 12:56:41,920 INFO reduce.EventFetcher: EventFetcher is interrupted.. Returning
2018-04-14 12:56:41,921 INFO mapred.LocalJobRunner: 1 / 1 copied.
2018-04-14 12:56:41,921 INFO reduce.MergeManagerImpl: finalMerge called with 1 in-memory map-outputs and 0 on-disk map-outputs
2018-04-14 12:56:41,993 INFO mapred.Merger: Merging 1 sorted segments
2018-04-14 12:56:41,993 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 18 bytes
2018-04-14 12:56:41,998 INFO reduce.MergeManagerImpl: Merged 1 segments, 28 bytes to disk to satisfy reduce memory limit
2018-04-14 12:56:41,999 INFO reduce.MergeManagerImpl: Merging 1 files, 32 bytes from disk
2018-04-14 12:56:41,999 INFO reduce.MergeManagerImpl: Merging 0 segments, 0 bytes from memory into reduce
2018-04-14 12:56:42,000 INFO mapred.Merger: Merging 1 sorted segments
2018-04-14 12:56:42,002 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 18 bytes
2018-04-14 12:56:42,004 INFO mapred.LocalJobRunner: 1 / 1 copied.
2018-04-14 12:56:42,029 INFO mapred.Task: Task:attempt_local90163926_0002_r_000000_0 is done. And is in the process of committing
2018-04-14 12:56:42,030 INFO mapred.LocalJobRunner: 1 / 1 copied.
2018-04-14 12:56:42,031 INFO mapred.Task: Task attempt_local90163926_0002_r_000000_0 is allowed to commit now
2018-04-14 12:56:42,039 INFO output.FileOutputCommitter: Saved output of task 'attempt_local90163926_0002_r_000000_0' to file:/root/airlines_output


root@DESKTOP-C52USQO: ~/airlines_output
2018-04-14 12:56:42,039 INFO output.FileOutputCommitter: Saved output of task 'attempt_local90163926_0002_r_000000_0' to file:/root/airlines_output
2018-04-14 12:56:42,040 INFO mapred.LocalJobRunner: reduce > reduce
2018-04-14 12:56:42,040 INFO mapred.Task: Task 'attempt_local90163926_0002_r_000000_0' done.
2018-04-14 12:56:42,042 INFO mapred.Task: Final Counters for attempt_local90163926_0002_r_000000_0: Counters: 24
File System Counters
FILE: Number of bytes read=948733
FILE: Number of bytes written=1577481
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
Map-Reduce Framework
Combine input records=0
Combine output records=0
Reduce input groups=1
Reduce shuffle bytes=32
Reduce input records=1
Reduce output records=1
Spilled Records=1
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=0
Total committed heap usage (bytes)=330301440
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Output Format Counters
Bytes Written=32
2018-04-14 12:56:42,072 INFO mapred.LocalJobRunner: Finishing task: attempt_local90163926_0002_r_000000_0
2018-04-14 12:56:42,073 INFO mapred.LocalJobRunner: reduce task executor complete.
2018-04-14 12:56:42,419 INFO mapreduce.Job: Job job_local90163926_0002 running in uber mode : false
2018-04-14 12:56:42,421 INFO mapreduce.Job: map 100% reduce 100%
2018-04-14 12:56:42,426 INFO mapreduce.Job: Job job_local90163926_0002 completed successfully
2018-04-14 12:56:42,448 INFO mapreduce.Job: Counters: 30
File System Counters
FILE: Number of bytes read=1897370
FILE: Number of bytes written=3154898
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0

Step 3 cdairlines_output

Step 4 ls –lrt

Step 5 cat part-r-00000



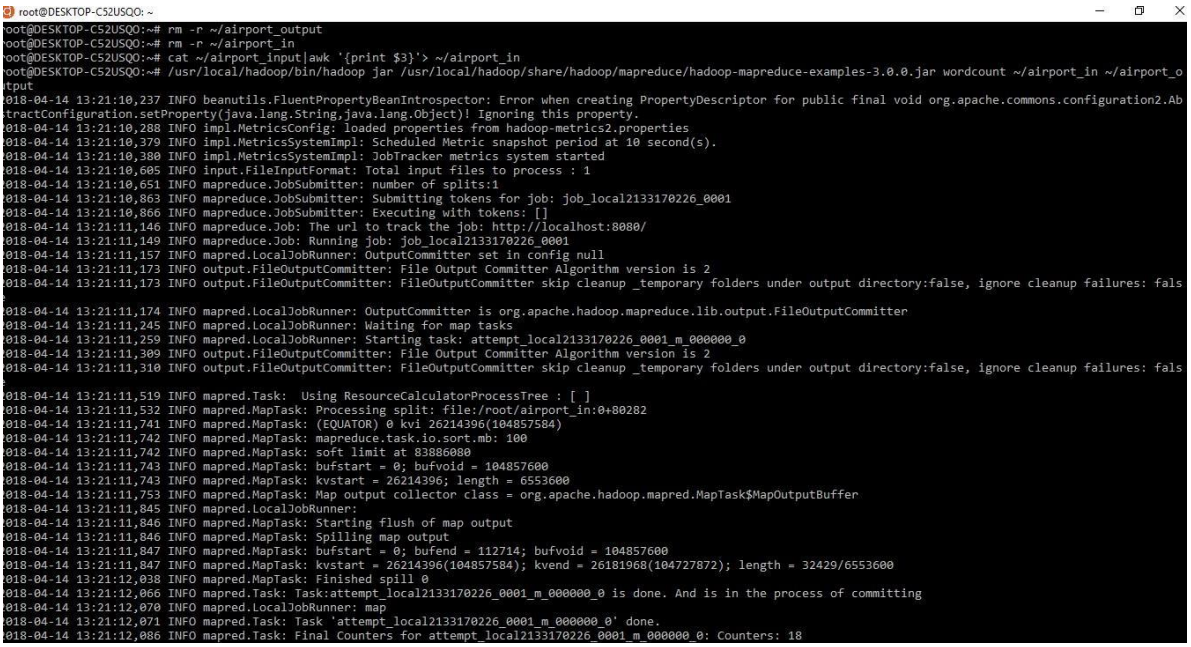
Q3. Which country (or) territory having highest Airports?

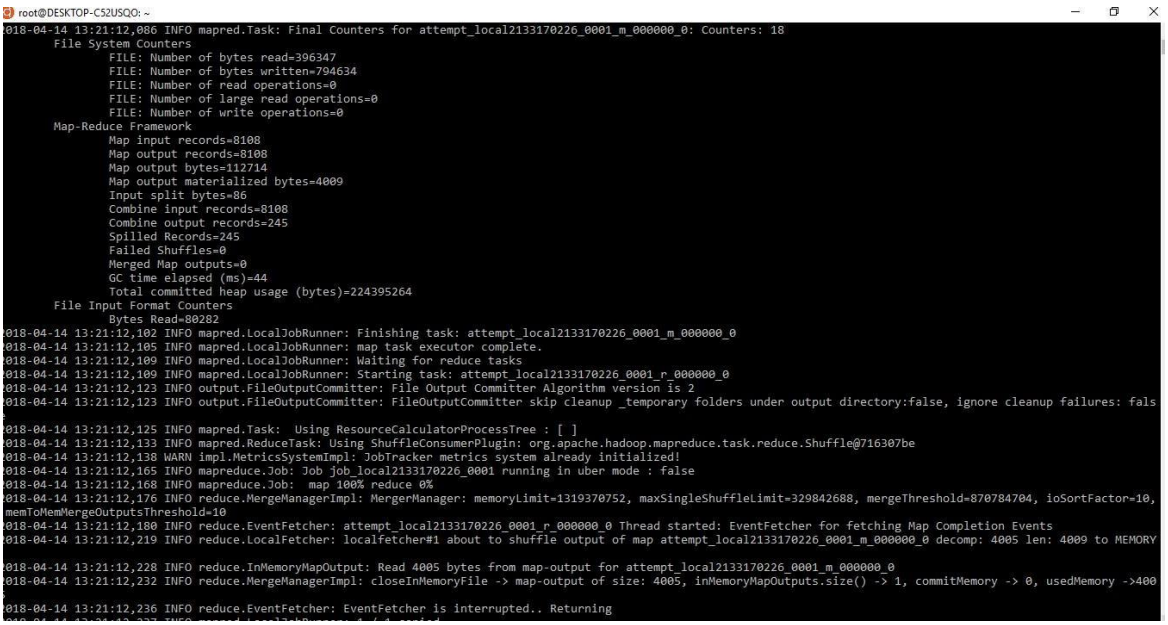
Step-1: rm -r ~/airport_output

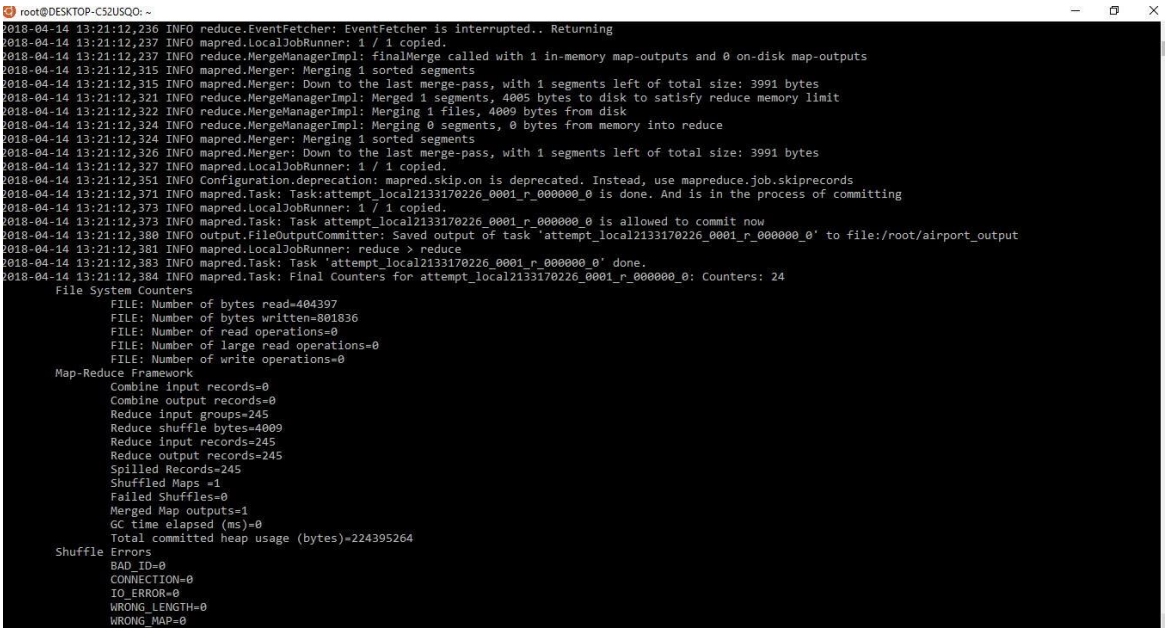
Step-2: rm -r ~/airport_in

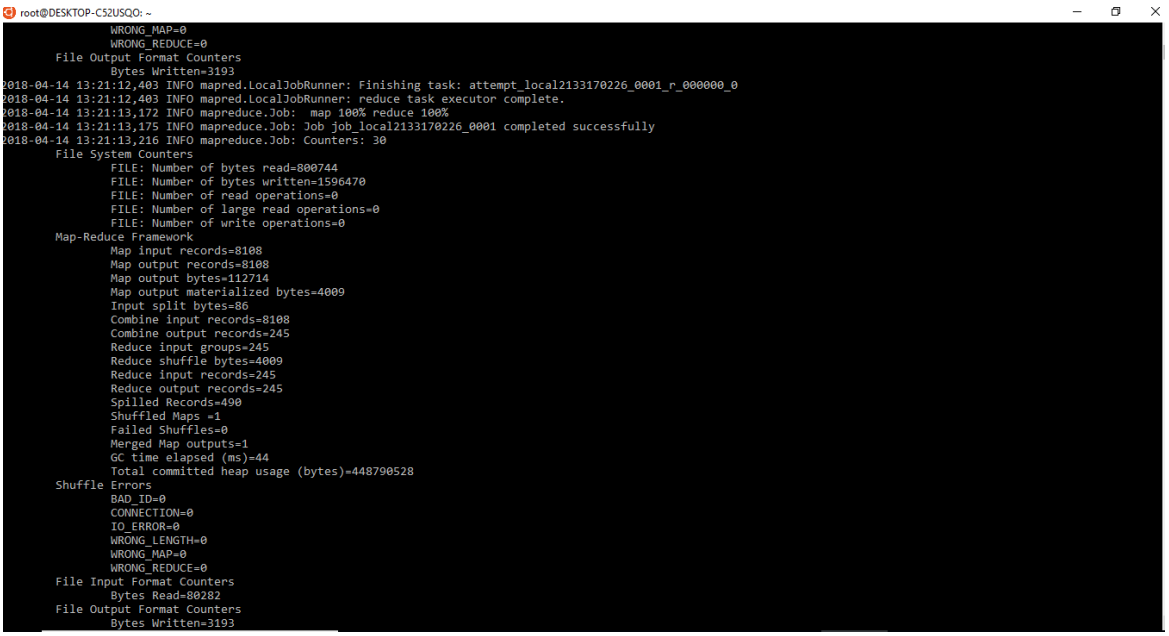
Step-3: cat ~/airport_input|awk '{print $3}'> ~/airport_in

Step-4:/usr/local/hadoop/bin/hadoop jar/usr/local/hadoop/share/hadoop/mapre-duce/hadoop-mapreduce-examples-3.0.0.jar wordcount ~/airport_in ~/airport_output

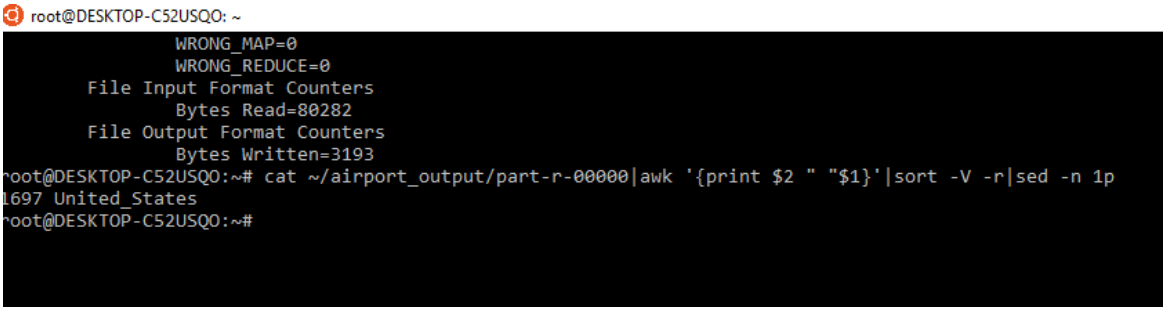





Step-5: cat ~/airport_output/part-r-00000|awk '{print $2 " "$1}'|sort -V -r|sed -n 1p

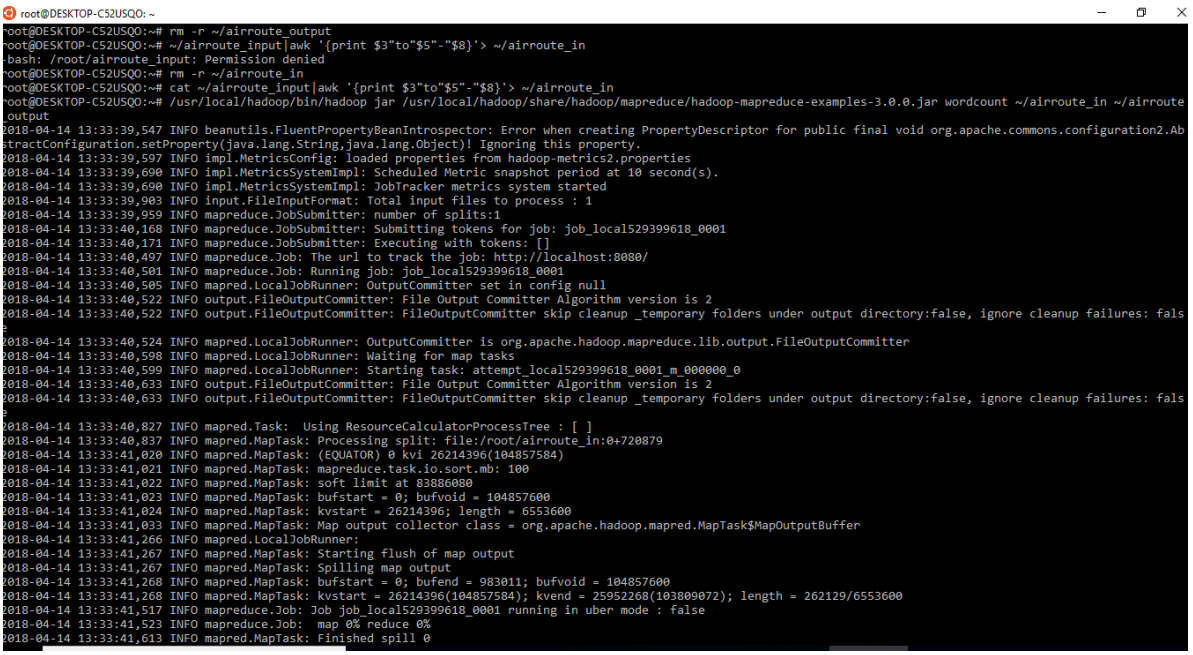


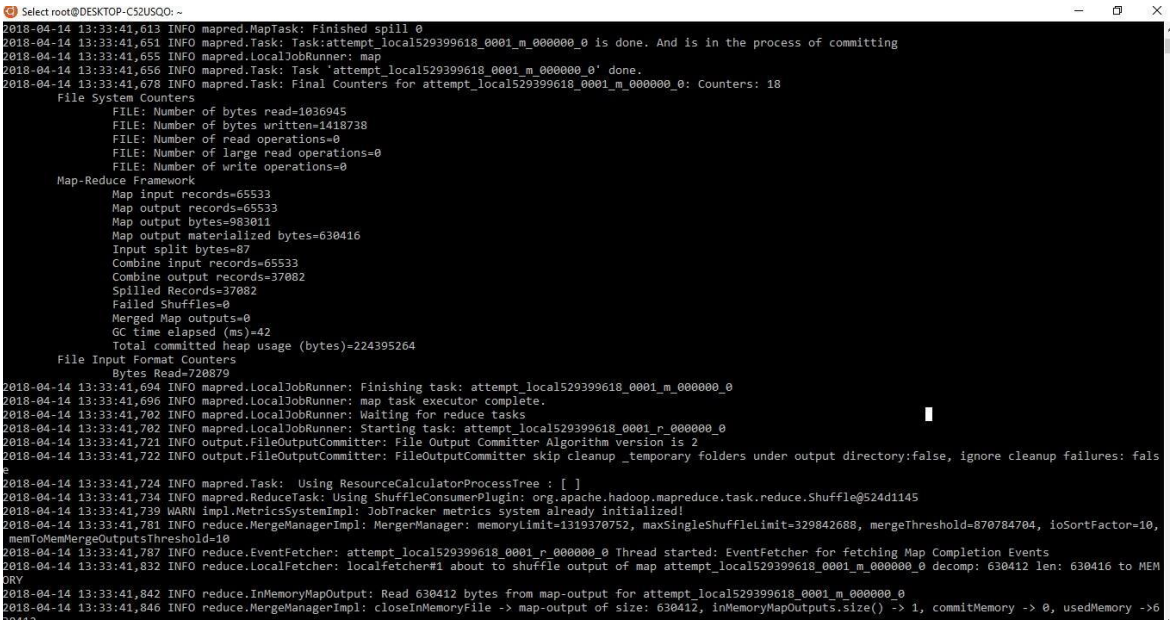
Q4 How many flight from YRT to YEK having zero(0) ,1 stops ?

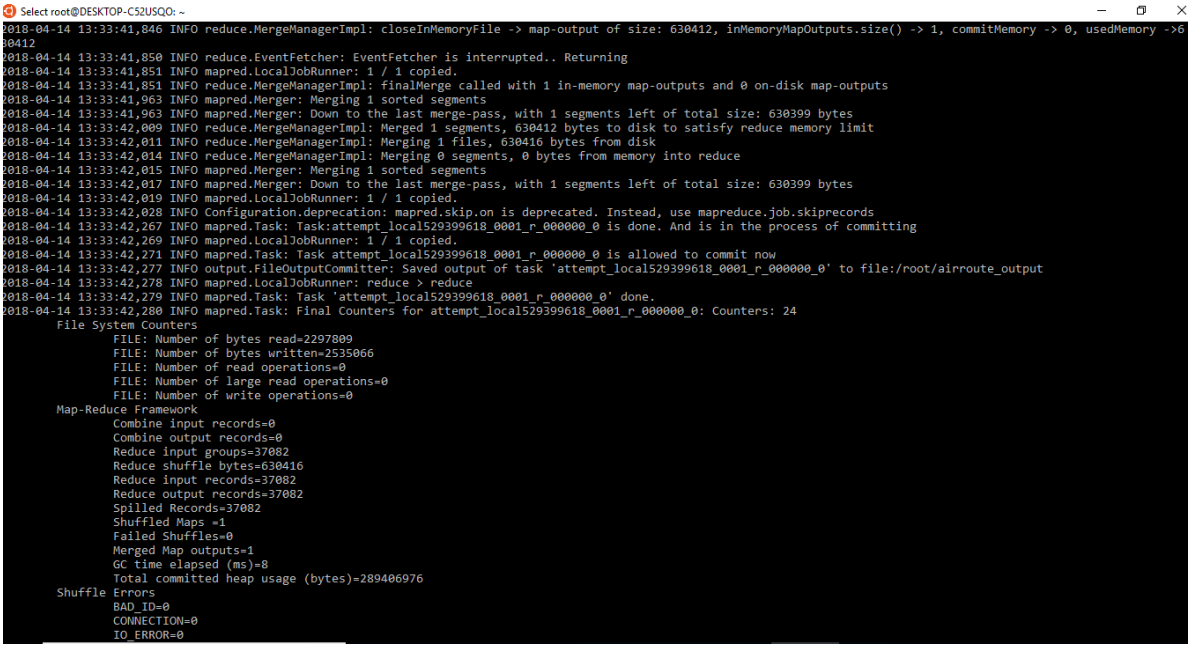
Step 1 rm -r ~/airroute_output

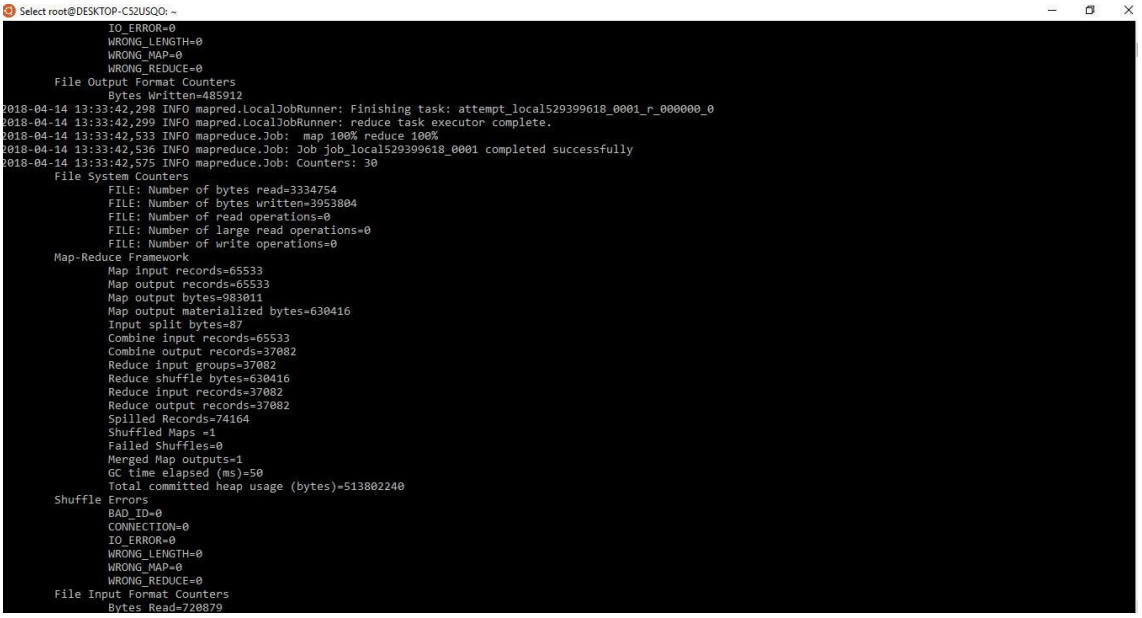
Step 2 cat ~/airroute_input|awk '{print $3"to"$5"-"$8}'> ~/airroute_in

Step3 /usr/local/hadoop/bin/hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.0.jar wordcount ~/airroute_in ~/airroute_output



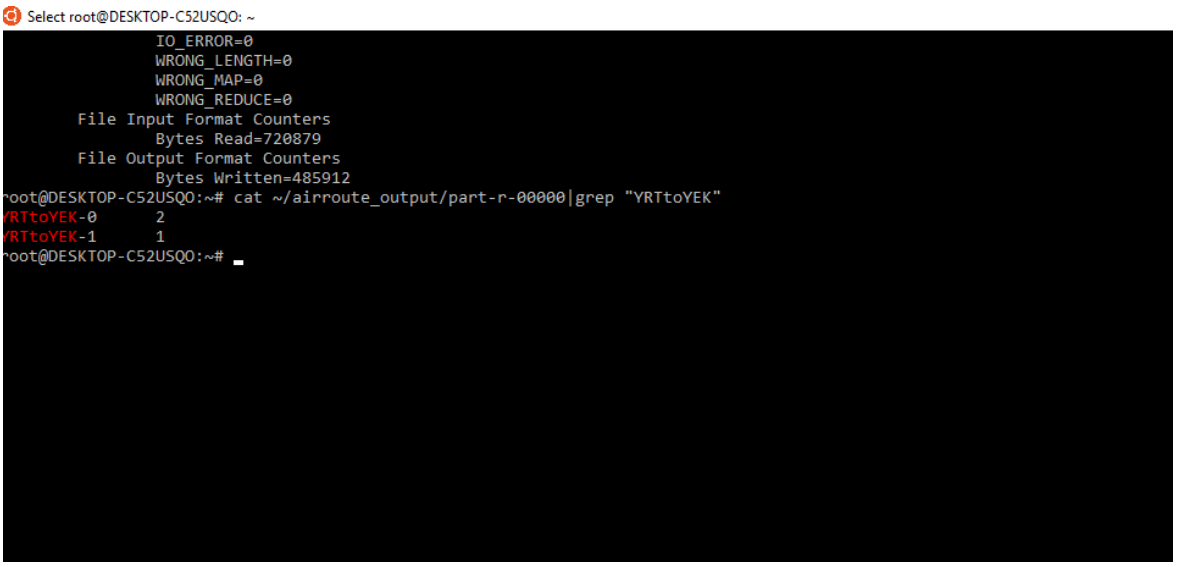




Select root@DESKTOP-C52USQO: ~
2018-04-14 13:33:41,846 INFO reduce.MergeManagerImpl: closeInMemoryFile -> map-output of size: 630412, inMemoryMapOutputs.size() -> 1, commitMemory -> 0, usedMemory ->6
30412
2018-04-14 13:33:41,850 INFO reduce.EventFetcher: EventFetcher is interrupted.. Returning
2018-04-14 13:33:41,851 INFO mapred.LocalJobRunner: 1 / 1 copied.
2018-04-14 13:33:41,851 INFO reduce.MergeManagerImpl: finalMerge called with 1 in-memory map-outputs and 0 on-disk map-outputs
2018-04-14 13:33:41,963 INFO mapred.Merger: Merging 1 sorted segments
2018-04-14 13:33:41,963 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 630399 bytes
2018-04-14 13:33:42,009 INFO reduce.MergeManagerImpl: Merged 1 segments, 630412 bytes to disk to satisfy reduce memory limit
2018-04-14 13:33:42,011 INFO reduce.MergeManagerImpl: Merging 1 files, 630416 bytes from disk
2018-04-14 13:33:42,014 INFO reduce.MergeManagerImpl: Merging 0 segments, 0 bytes from memory into reduce
2018-04-14 13:33:42,015 INFO mapred.Merger: Merging 1 sorted segments
2018-04-14 13:33:42,017 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 630399 bytes
2018-04-14 13:33:42,019 INFO mapred.LocalJobRunner: 1 / 1 copied.
2018-04-14 13:33:42,028 INFO Configuration.deprecation: mapred.skip.on is deprecated. Instead, use mapreduce.job.skiprecords
2018-04-14 13:33:42,267 INFO mapred.Task: Task:attempt_local529399618_0001_r_000000_0 is done. And is in the process of committing
2018-04-14 13:33:42,269 INFO mapred.LocalJobRunner: 1 / 1 copied.
2018-04-14 13:33:42,271 INFO mapred.Task: Task attempt_local529399618_0001_r_000000_0 is allowed to commit now
2018-04-14 13:33:42,277 INFO output.FileOutputCommitter: Saved output of task 'attempt_local529399618_0001_r_000000_0' to file:/root/airroute_output
2018-04-14 13:33:42,278 INFO mapred.LocalJobRunner: reduce > reduce
2018-04-14 13:33:42,279 INFO mapred.Task: Task 'attempt_local529399618_0001_r_000000_0' done.
2018-04-14 13:33:42,280 INFO mapred.Task: Final Counters for attempt_local529399618_0001_r_000000_0: Counters: 24
File System Counters
FILE: Number of bytes read=2297809
FILE: Number of bytes written=2535066
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
Map-Reduce Framework
Combine input records=0
Combine output records=0
Reduce input groups=37082
Reduce shuffle bytes=630416
Reduce input records=37082
Reduce output records=37082
Spilled Records=37082
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=8
Total committed heap usage (bytes)=289406976
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0


Select root@DESKTOP-C52USQO: ~
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Output Format Counters
Bytes Written=485912
2018-04-14 13:33:42,298 INFO mapred.LocalJobRunner: Finishing task: attempt_local529399618_0001_r_000000_0
2018-04-14 13:33:42,299 INFO mapred.LocalJobRunner: reduce task executor complete.
2018-04-14 13:33:42,533 INFO mapreduce.Job: map 100% reduce 100%
2018-04-14 13:33:42,536 INFO mapreduce.Job: Job job_local529399618_0001 completed successfully
2018-04-14 13:33:42,575 INFO mapreduce.Job: Counters: 30
File System Counters
FILE: Number of bytes read=3334754
FILE: Number of bytes written=3953804
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
Map-Reduce Framework
Map input records=65533
Map output records=65533
Map output bytes=983011
Map output materialized bytes=630416
Input split bytes=87
Combine input records=65533
Combine output records=37082
Reduce input groups=37082
Reduce shuffle bytes=630416
Reduce input records=37082
Reduce output records=37082
Spilled Records=74164
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=50
Total committed heap usage (bytes)=513802240
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=720879

Step 4 cat ~/airroute_output/part-r-00000|grep "YRTtoYEK"

Step 5 rm -r ~/airroute_in



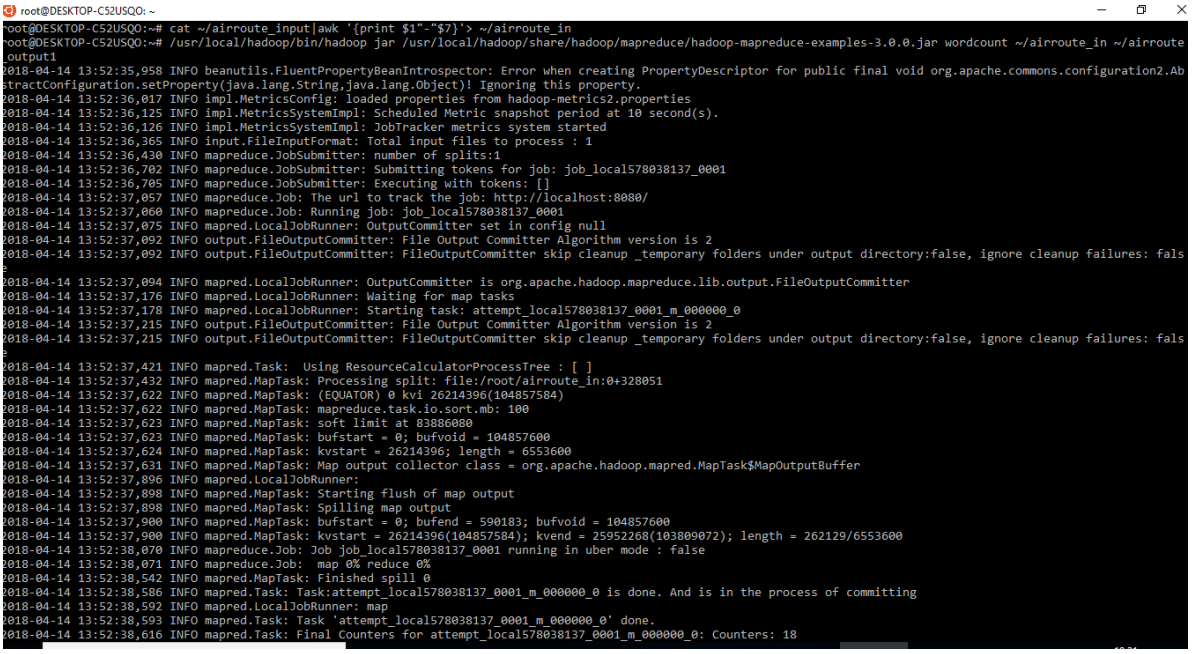
Q5.How many flights having same air code for flight which uses code share

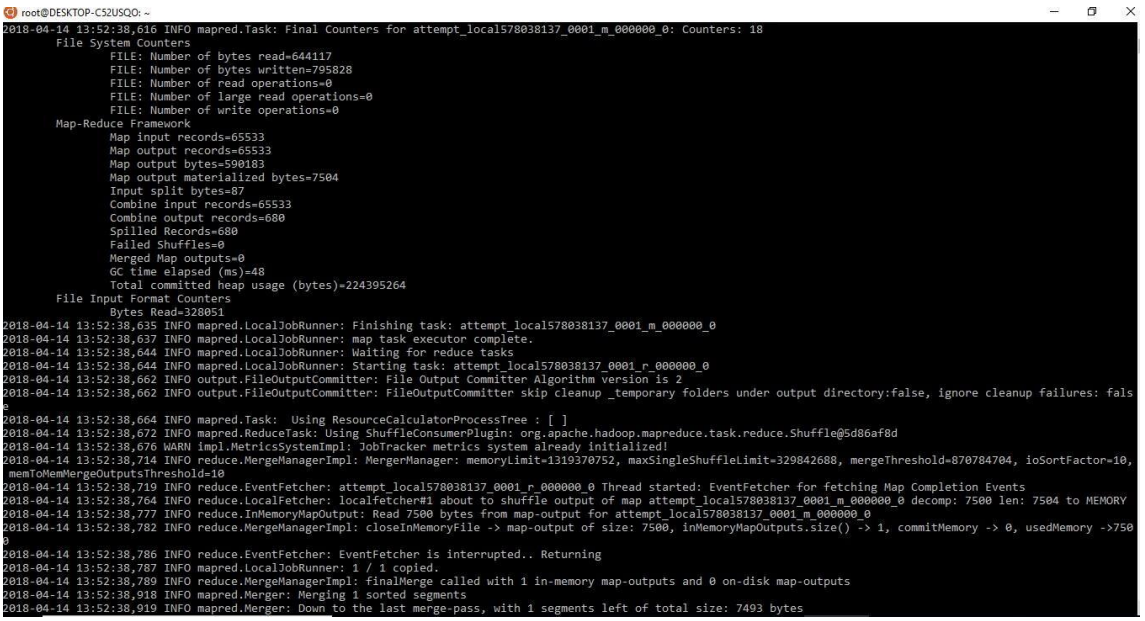
Step1: cd ~

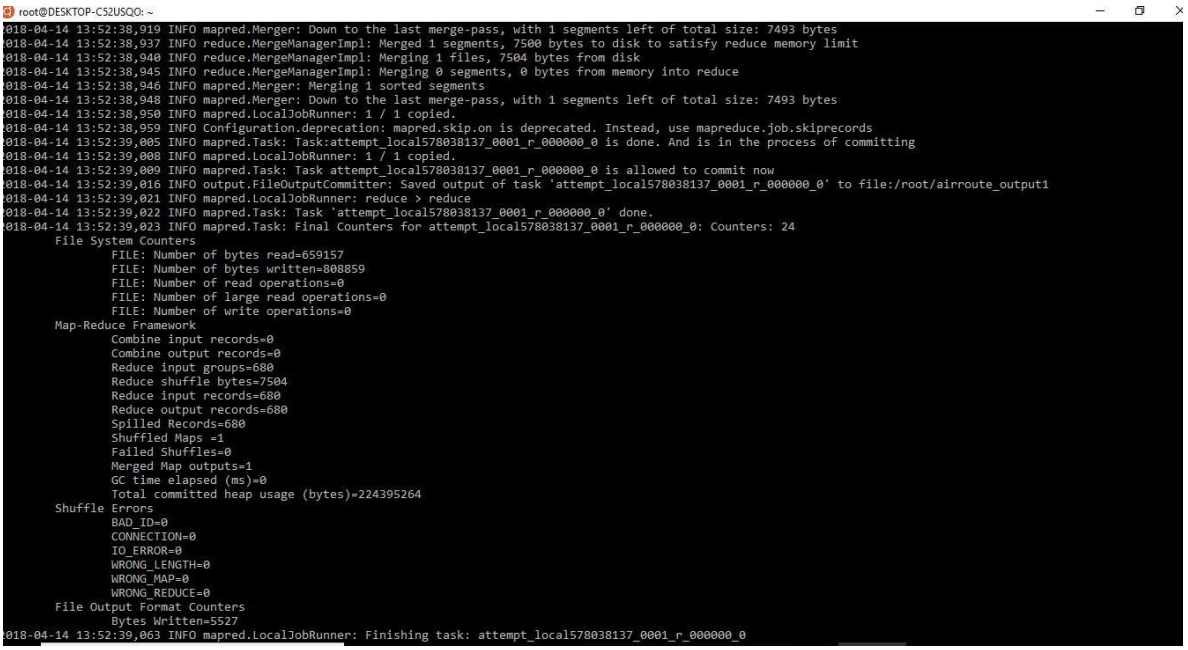
Step2: rm -r ~/airroute_output1

Step3: cat ~/airroute_input|awk '{print $1"-"$7}'> ~/airroute_in

Step 4:/usr/local/hadoop/bin/hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.0.jar wordcount ~/airroute_in ~/airroute_output1

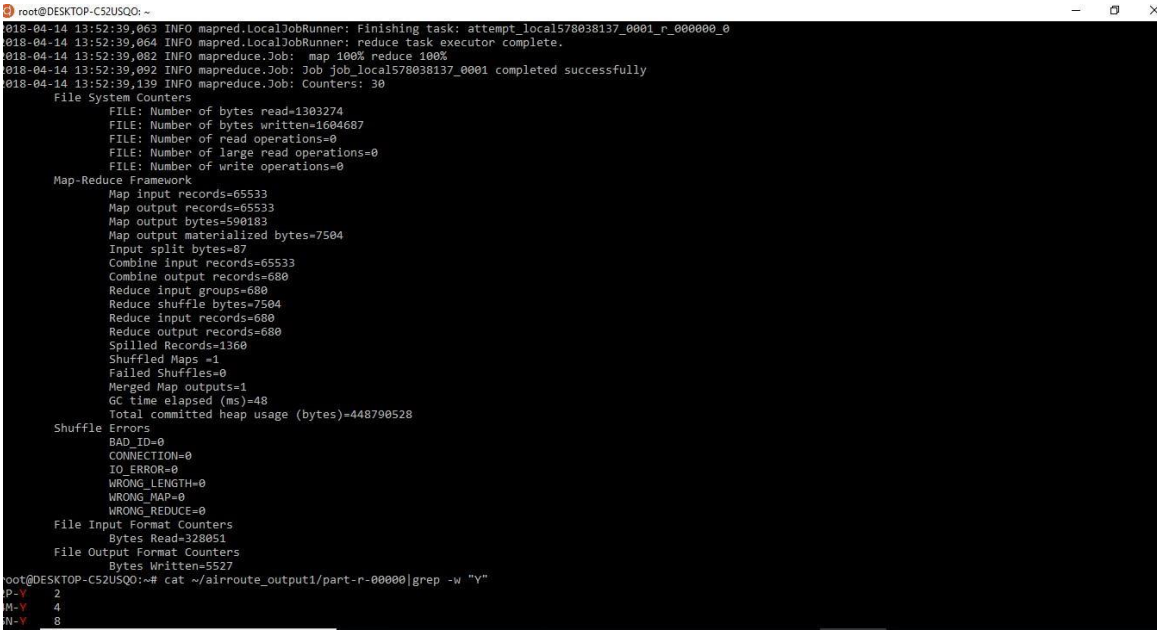


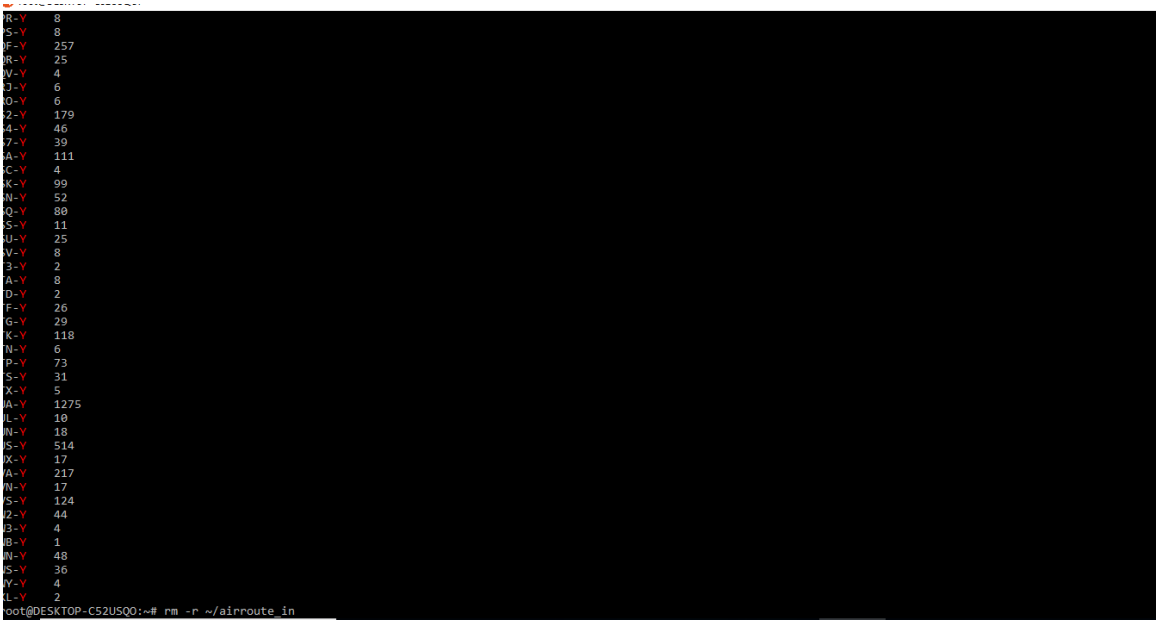



Step 5: cat ~/airroute_output1/part-r-00000|grep -w "Y"

Step 6: rm -r ~/airroute_in




root@DESKTOP-C52USQO: ~
GN-Y 8
GT-Y 21
7F-Y 2
7H-Y 200
9K-Y 2
9W-Y 131
A3-Y 9
AA-Y 1089
AB-Y 116
AC-Y 384
AD-Y 2
AE-Y 20
AF-Y 619
AI-Y 61
AM-Y 270
AP-Y 12
AR-Y 50
AS-Y 292
AT-Y 4
AV-Y 43
AY-Y 154
AZ-Y 550
B2-Y 2
B7-Y 16
BA-Y 177
BE-Y 55
BR-Y 16
BT-Y 4
CA-Y 474
CI-Y 16
CM-Y 35
CU-Y 4
CX-Y 8
CY-Y 2
CZ-Y 214
DC-Y 2
DE-Y 23
DL-Y 835
DY-Y 24
EI-Y 54
EK-Y 18
ET-Y 115
EY-Y 52
F7-Y 4


FR-Y 8
FS-Y 8
GF-Y 257
GR-Y 25
GV-Y 4
HJ-Y 6
HO-Y 6
I2-Y 179
I4-Y 46
I7-Y 39
IA-Y 111
IC-Y 4
IK-Y 99
IN-Y 52
IQ-Y 80
IS-Y 11
IU-Y 25
IV-Y 8
J3-Y 2
JA-Y 8
JD-Y 2
JF-Y 26
JG-Y 29
JK-Y 118
JN-Y 6
JP-Y 73
JS-Y 31
JX-Y 5
KA-Y 1275
KL-Y 10
KN-Y 18
KS-Y 514
KX-Y 17
LA-Y 217
LN-Y 17
LS-Y 124
M2-Y 44
M3-Y 4
MB-Y 1
MN-Y 48
MS-Y 36
MY-Y 4
NL-Y 2
root@DESKTOP-C52USQO:~# rm -r ~/airroute_in

# TESTING[8]

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or

other defects), and verifying that the software product is fit for use.

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test

- ✓ Meets the requirements that guided its design and development,
- ✓ Responds correctly to all kinds of inputs,
- ✓ Performs its functions within an acceptable time,
- ✓ Is sufficiently usable,
- ✓ Can be installed and run in its intended environments, and
- ✓ Achieves the general result its stakeholder's desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects). The job of testing is an iterative process as when one bug is fixed, it can illuminate other, deeper bugs, or can even create new ones.

Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors.[1]

Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted.

## Types of Testing

Functional Testing

In software development, functional testing relates to the testing of a system's functionality. Typically the testing of each function is performed independently. Functional testing is generally performed against a specific requirement, providing a check as to whether it works as it should e.g. does the system do x when y is pressed = yes/no.

Regression Testing

Regression testing looks at whether software that has previously worked as it should continues to do so after an update or change has been made. Changes can vary from an update to the actual software e.g. a new version or patch that has been released, or it can be used when an integrated application is added or updated. Regression testing is typically a form of functional testing but it is specifically focused on looking for new issues and risks in existing functions that have previously worked.

Compatibility Testing

Compatibility testing is a non-functional type of testing which looks at how software performs across a range of devices, operating systems and browsers. To be effective it is recommended to always perform compatibility testing on real environments rather than using emulators. With the increasing focus on digital transformation initiatives, compatibility testing is growing in importance, particularly when considering user experience and customer satisfaction.

Automated Testing

Automated testing refers to a type of testing that uses independant software to test the system being tested. Automated testing can be used to perform other types of testing such as functional or performance testing. Automated testing lends itself well to testing which is repetitive in nature and can be time-consuming if performed manually e.g. functional

regression testing. The pre-scripted nature of automated testing can enable increased test coverage to be achieved.

Smoke / Sanity Testing

Smoke testing checks whether fundamental functionality in a piece of software is working. Smoke testing is typically used at an early stage in the software development lifecycle to determine if the system is stable enough to begin more extensive testing or whether there are any basic issues that would prevent testing or waste time.

Acceptance Testing

Acceptance testing is focused on users' requirements from a system and checks whether these are satisfied by the system. To perform acceptance testing a set of acceptance criteria is normally specified to test against, with automated tests often being used alongside un-scripted exploratory testing to better-represent a user's approach to using the software.

Performance Testing

Performance testing is a type of non-functional testing (a test level). It can look at the stability, responsiveness and speed of a system amongst other things. Generally performance testing is carried out in a representative test environment replicating the numbers of users – often in the hundreds or thousands – anticipated to be using the system concurrently. There are a number of sub-categories to performance testing such as stress testing, peak/load testing, and soak testing.

Accessibility Testing

Accessibility testing is a form of usability testing. In the UK accessibility testing is used to check websites and software are accessible for people with disabilities including those with disabilities relating to hearing, sight, cognitive understanding and old age. Those with disabilities often make use of assistive technology such as screen readers so accessibility testing checks that the various elements of a page are tagged properly to be read by these technologies.

Usability testing

Usability testing checks how intuitive and 'usable' software is for the end-users. It is generally conducted by real users (rather than emulators) and is objective-led e.g. find a

red jacket on the site and add to your shopping basket, rather than giving a user specific steps to follow to complete a task. Checklists can also be used to test against recognised usability principles. This type of testing is used to understand just how user-friendly a piece of software is.

Security Testing

Security testing is a category of testing, performed to identify vulnerabilities in a system and related infrastructure, in order to protect customer and organisation data, as well as intellectual property. There are a number of different sub-categories to security testing such as penetration testing, which aims to identify vulnerabilities which an attacker could exploit from external or internal access.

## TEST CASES

Pre-Conditions: An Airline.csv file should be created.

**Test Case** : SA_101

**System** : Airline Analysis **Test Case Name**:Airline.csv

**Designed by**: Yukti

**Executed by**: Yukti **Design Date**: 01/05/2020

**Short Description** :Check the Airline.csv file **Execution Date**: 04/05/2020

| STEP | ACTION | EXPECTED SYSTEM RESPONSE | ACTUAL RESULT | PASS/ FAIL | COMMENT |
|---|---|---|---|---|---|
| 1. | Changing Source of the file from airline to abcd. | Should show an error. | Error Shown | Pass | <Executed> |

| | | | | | |
|---|---|---|---|---|---|
| 2. | Leaving any keywords | Should show an error. | Error Shown | Pass | <Executed> |
| 3. | Changing any data | Should show an error. | Error Shown | Pass | <Executed> |

Post-Conditions: All the conditions done normal or default after test

**Test Case** : SA_102

**System** : Airline Analysis **Test Case Name**: Checking

**Designed by**: Yukti

**Executed by**: Yukti **Design Date**: 01/05/2020

**Short Description** :Check the Airline.csv file **Execution Date**: 04/05/2020

Pre-Conditions:An Airroute.csv file should be created

| STEP | ACTION | EXPECTED SYSTEM RESPONSE | ACTUAL RESULT | PASS/ FAIL | COMMENT |
|---|---|---|---|---|---|
| 1. | Changing Source of the file from air route to abcd. | Should show an error. | Error Shown | Pass | <Executed> |

| | | | | | |
|---|---|---|---|---|---|
| 2. | Leaving any keywords | Should show an error. | Error Shown | Pass | <Executed> |
| 3. | Changing any data | Should show an error. | Error Shown | Pass | <Executed> |

Post-Conditions: All the conditions done normal or default after test

**Test Case** : SA_103

**System** : Airline Analysis **Test Case Name**: Checking

**Designed by**: Yukti

**Executed by**: Yukti **Design Date**: 01/05/2020

**Short Description** :Check the Airline.csv file **Execution Date**: 04/05/2020

Pre-Conditions: Airport.csv file should be created

| .STEP | ACTION | EXPECTED SYSTEM RESPONSE | ACTUAL RESULT | PASS/ FAIL | COMMENT |
|---|---|---|---|---|---|
| 1. | Changing Source of the file from airport to abcd. | Should show an error. | Error Shown | Pass | <Executed> |
| 2. | Leaving any keywords | Should show an error. | Error Shown | Pass | <Executed> |

| | | | | | |
|---|---|---|---|---|---|
| 3. | Changing any data | Should show an error. | Error Shown | Pass | <Executed> |

Post-Conditions: All the conditions done normal or default after test.

**Test Case** : SA_104

**System** : Airline Analysis **Test Case Name**: Checking

**Designed by**: Yukti

**Executed by**: Yukti **Design Date**: 01/05/2020

**Short Description** :Check the Airline.csv file **Execution Date**: 04/05/2020

Pre-Conditions:

- ✓ Knowing about the logic.
- ✓ Creating JAR file Wordcount.jar.

| STEP | ACTION | EXPECTED SYSTEM RESPONSE | ACTUAL RESULT | PASS/ FAIL | COMMENT |
|---|---|---|---|---|---|
| 1. | Checking whether the data received is a valid JSON form. | Valid JSON format | Valid JSON format. | Pass | <Executed> |
| 2. | The whole data must be broken into different fields | Whole JSON must be broken accordingly into different fields. | Got the expected result. | Pass | <Executed> |

Post-Conditions: Getting the result in structured form

**Test Case** : SA_105

**System** : Airline Analysis **Test Case Name**: Checking

**Designed by**: Yukti

**Executed by**: Yukti **Design Date**: 01/05/2020

**Short Description** :Check the result of Query **Execution Date**: 04/05/2020

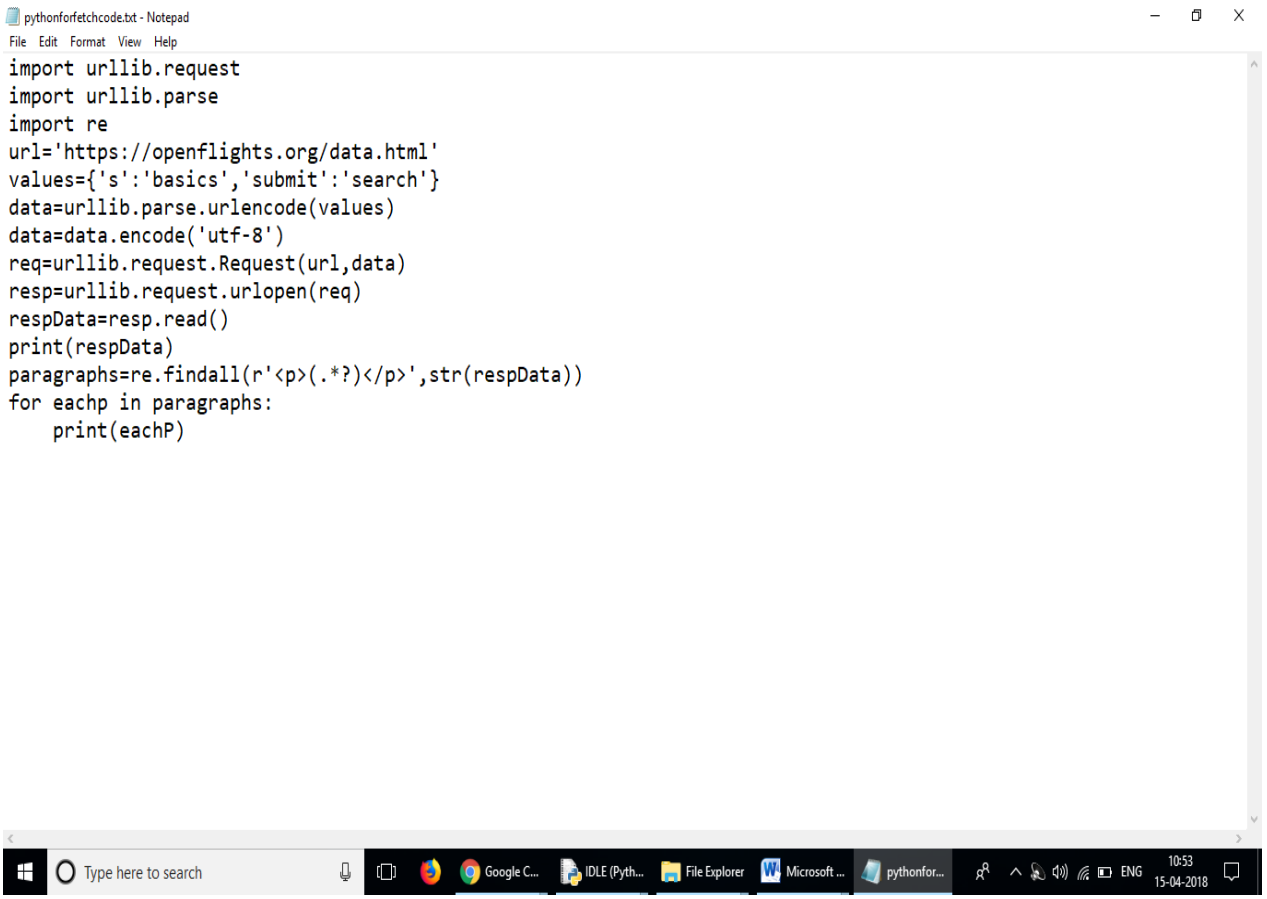
Pre-Conditions: The data should be in the structured form.

| STEP | ACTION | EXPECTED SYSTEM RESPONSE | ACTUAL RESULT | PASS/ FAIL | COMMENT |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| 1. | Use of required command | It should give the correct result according to the queries | Output in form of data | Pass | <Executed> |
| 2. | Aggregation of the result. | Should Give a valid aggregation. | Got a valid aggregation of the result. | Pass | <Executed> |

## APPENDIX A: SCREESHOTS OF PROGRAM

TASK –I: Download the airline data (data set) from website relating to a particular event using python language.



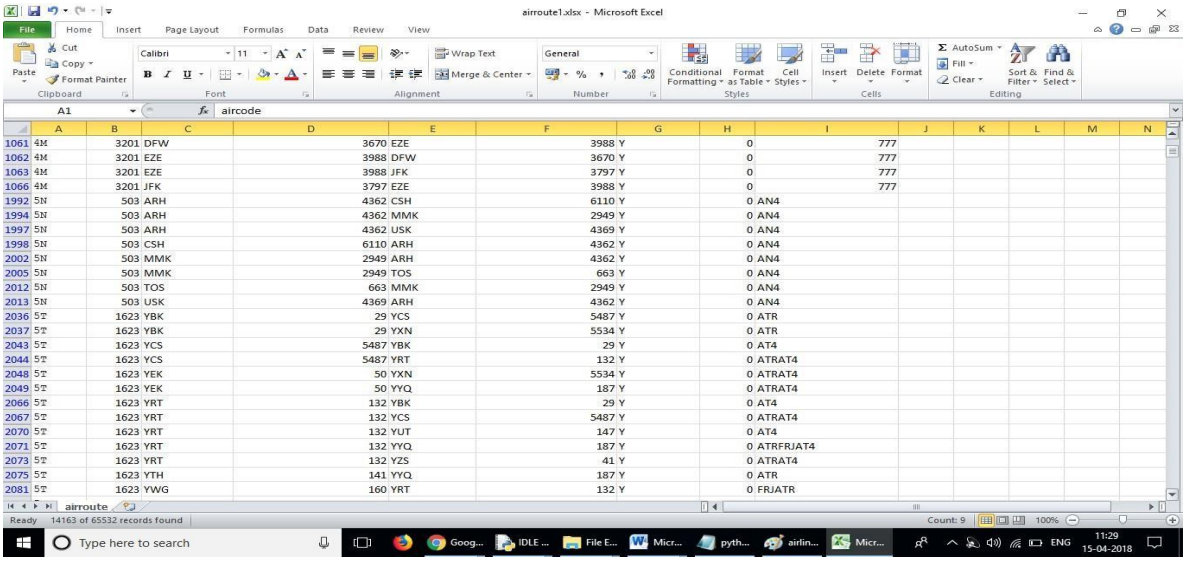
# APPENDIX B: SOURCE CODE TASK –II

MAPPER CODE SCREENSHOTS

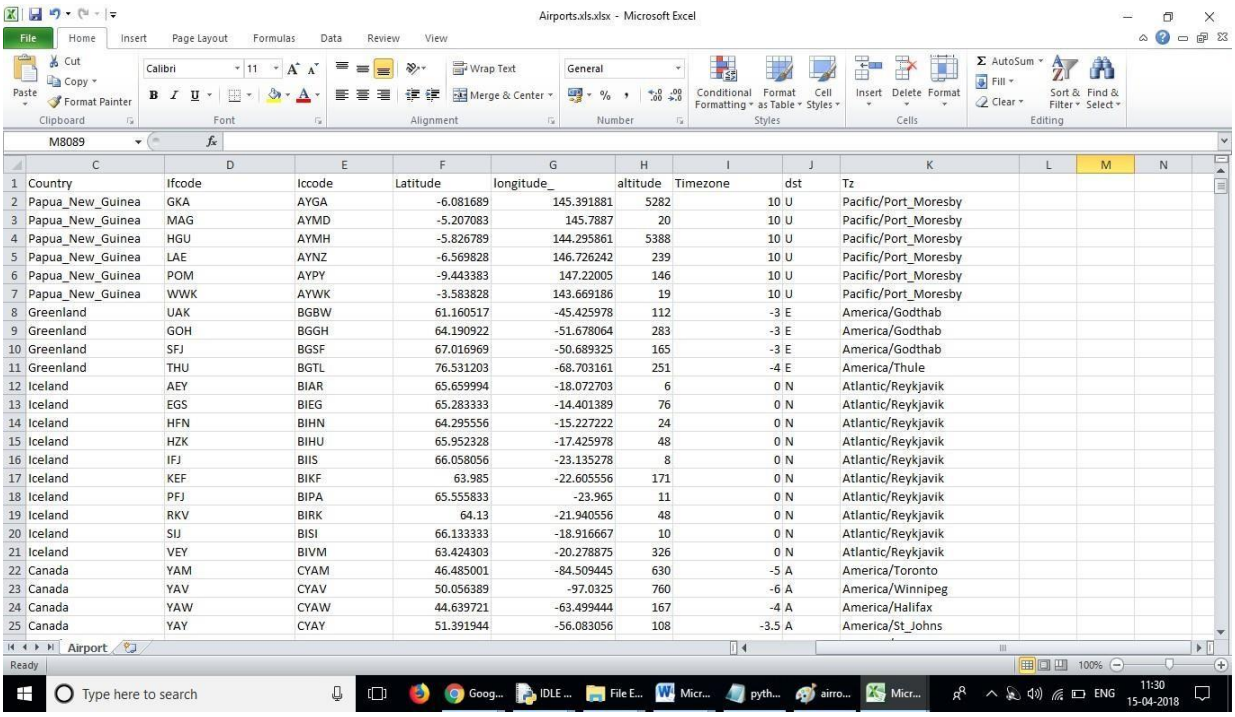
```java
import java.io.IOException;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

// Calculate occurrences of a character
public class AlphaReducer extends Reducer<Text, LongWritable, Text, LongWritable> {
    private LongWritable result = new LongWritable();

    public void reduce(Text key, Iterable<LongWritable> values, Context context)
            throws IOException, InterruptedException {
        long sum = 0;
        for (LongWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}
```

STRUCTURED DATA

**1.** Airline data set



## 2. Airports data set



Airports.xls.xlsx - Microsoft Excel

| | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Country | Ifcode | Iccode | Latitude | longitude_ | altitude | Timezone | dst | Tz |
| 2 | Papua_New_Guinea | GKA | AYGA | -6.081689 | 145.391881 | 5282 | 10 | U | Pacific/Port_Moresby |
| 3 | Papua_New_Guinea | MAG | AYMD | -5.207083 | 145.7887 | 20 | 10 | U | Pacific/Port_Moresby |
| 4 | Papua_New_Guinea | HGU | AYMH | -5.826789 | 144.295861 | 5388 | 10 | U | Pacific/Port_Moresby |
| 5 | Papua_New_Guinea | LAE | AYNZ | -6.569828 | 146.726242 | 239 | 10 | U | Pacific/Port_Moresby |
| 6 | Papua_New_Guinea | POM | AYPY | -9.443383 | 147.22005 | 146 | 10 | U | Pacific/Port_Moresby |
| 7 | Papua_New_Guinea | WWK | AYWK | -3.583828 | 143.669186 | 19 | 10 | U | Pacific/Port_Moresby |
| 8 | Greenland | UAK | BGBW | 61.160517 | -45.425978 | 112 | -3 | E | America/Godthab |
| 9 | Greenland | GOH | BGGH | 64.190922 | -51.678064 | 283 | -3 | E | America/Godthab |
| 10 | Greenland | SFJ | BGSF | 67.016969 | -50.689325 | 165 | -3 | E | America/Godthab |
| 11 | Greenland | THU | BGTL | 76.531203 | -68.703161 | 251 | -4 | E | America/Thule |
| 12 | Iceland | AEY | BIAR | 65.659994 | -18.072703 | 6 | 0 | N | Atlantic/Reykjavik |
| 13 | Iceland | EGS | BIEG | 65.283333 | -14.401389 | 76 | 0 | N | Atlantic/Reykjavik |
| 14 | Iceland | HFN | BIHN | 64.295556 | -15.227222 | 24 | 0 | N | Atlantic/Reykjavik |
| 15 | Iceland | HZK | BIHU | 65.952328 | -17.425978 | 48 | 0 | N | Atlantic/Reykjavik |
| 16 | Iceland | IFJ | BIIS | 66.058056 | -23.135278 | 8 | 0 | N | Atlantic/Reykjavik |
| 17 | Iceland | KEF | BIKF | 63.985 | -22.605556 | 171 | 0 | N | Atlantic/Reykjavik |
| 18 | Iceland | PFJ | BIPA | 65.555833 | -23.965 | 11 | 0 | N | Atlantic/Reykjavik |
| 19 | Iceland | RKV | BIRK | 64.13 | -21.940556 | 48 | 0 | N | Atlantic/Reykjavik |
| 20 | Iceland | SIJ | BISI | 66.133333 | -18.916667 | 10 | 0 | N | Atlantic/Reykjavik |
| 21 | Iceland | VEY | BIVM | 63.424303 | -20.278875 | 326 | 0 | N | Atlantic/Reykjavik |
| 22 | Canada | YAM | CYAM | 46.485001 | -84.509445 | 630 | -5 | A | America/Toronto |
| 23 | Canada | YAV | CYAV | 50.056389 | -97.0325 | 760 | -6 | A | America/Winnipeg |
| 24 | Canada | YAW | CYAW | 44.639721 | -63.499444 | 167 | -4 | A | America/Halifax |
| 25 | Canada | YAY | CYAY | 51.391944 | -56.083056 | 108 | -3.5 | A | America/St_Johns |

## 3.0 Word Count Program for Map reduce

WordCount.java

```
1  import java.io.IOException;
2  import java.util.*;
3
4  import org.apache.hadoop.fs.Path;
5  import org.apache.hadoop.conf.*;
6  import org.apache.hadoop.io.*;
7  import org.apache.hadoop.mapreduce.*;
8  import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
9  import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
10 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
11 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
12
13 public class WordCount {
14
15  public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
16     private final static IntWritable one = new IntWritable(1);
17     private Text word = new Text();
18
19     public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
20         String line = value.toString();
21         StringTokenizer tokenizer = new StringTokenizer(line);
22         while (tokenizer.hasMoreTokens()) {
23             word.set(tokenizer.nextToken());
24             context.write(word, one);
25         }
26     }
27  }
```

## References


[1] http://cra.org/ccc/wpcontent/uploads/sites/2/2015/05 /bigdatawhitepaper.pdf

[2] www.ijcsmc.com/docs/papers/June2017/V6I6201764.pdf

[3] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[4] https://www.uml-diagrams.org/index-examples.html.

[5] https://www.researchgate.net/figure/The-MapReduce-architecture-MapReduce-

[6] https://flume.apache.org/

[7] https://www.digitalocean.com/community/tutorials/how-to-install-hadoop-in-stand-alone-mode-on-ubuntu-16-04

[8] https://www.tutorialspoint.com/sdlc/sdlc_v_model.htm

[9] https://www.ten10.com/types-testing-introduction-different-types-software-testing