TRACKING COVID-19 IN INDIA

A
Project Report
Submitted in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

P.Suraj 1602-17-737-052

Y.Abhiram 1602-17-737-004

H.Sai Prakash 1602-17-737-039

Under the guidance of

Mr.M Vishnu Chaitanya

Assistant Professor

Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2019-20

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

DECLARATION BY THE CANDIDATE

We, **P.Suraj, Y.Abhiram, H.Sai Prakash** bearing hall ticket numbers, **1602-17-737-052**,**1602-17-737-004**,**1602-17-737-039** respectively hereby declare that the project report entitled **"Tracking COVID-19 in India"** under the guidance of **Mr.M.VishnuChaitanya,** Assistant Professor, Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfilment of the requirement of THEME BASED PROJECT of VI Semester of **Bachelor of Engineering** in **Information Technology.**

This is a record of bonafide work carried out by me and the results embodied in this project.

P.Suraj
1602-17-737-052

Y.Abhiram
1602-17-737-004

H.Sai Prakash
1602-17-737-039

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

BONAFIDE CERTIFICATE

This is to certify that the project entitled **"TRACKING COVID-19 IN INDIA"** being submitted by **P.Suraj, Y.Abhiram, H.Sai Prakash** bearing **H.T.NO:1602-17-737-052 ,1602-17-737-004 ,1602-17-737-039** in partial fulfilment of the requirements for the completion of THEME BASED PROJECT of Bachelor of Engineering, VI Semester, in Information Technology is a record of bonafide work carried out by him/her under my guidance.

Mr.Vishnu Chaitanya — **Assistant Professor,IT** — **Internal Guide**

Dr.RamMohanRao — **HOD,IT**

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them. We would like to take the opportunity to express our humble gratitude to **Mr.M Vishnu Chaitanya**, our internal guide, under whose guidance we have finished this project successfully. Her constant guidance and willingness to share her vast knowledge made us understand this project and its manifestations in great depths and helped us to complete the work. We would like to thank all faculty members and staff of the Department of Information Technology, Vasavi College of Engineering for their generous help in various ways for the completion of this project. Finally, yet importantly, We would like to express our heartfelt thanks to our respected **Dr.K.Ram Mohan Rao(HOD-IT)** who was very helpful in providing resources for the project and also our project co-ordinators **Mrs.Leela Pallava(Asst.Prof-IT)** and **Mr.Srinivasa Chakravarthy(Asst.Prof-IT)** whose efforts were constant in monitoring our project.

P.Suraj

1602-17-737-052

Y.Abhiram

1602-17-737-004

H.Sai Prakash

1602-17-737-039

ABSTRACT

The first case of the COVID-19 pandemic in India was reported on 30 January 2020, originating from China. As of 10 May 2020, the Ministry of Health and Family Welfare have confirmed a total of 62,939 cases, 19,358 recoveries (including 1 migration) and 2,109 deaths in the country.The infection rate of COVID-19 in India is reported to be 1.7, significantly lower than in the worst affected countries.

The outbreak has been declared an epidemic in more than a dozen states and union territories, where provisions of the Epidemic Diseases Act, 1897 have been invoked, and educational institutions and many commercial establishments have been shut down.For almost a month no new cases were reported in India, however, on 8th March, 5 new cases were reported again since then cases have been rising affecting 14 states till April and later the whole country.

This Project uses Hadoop Environment to analyze COVID-19 in India. The analysis is done mainly using Hadoop MapReduce and Hadoop Distributed File Systems (HDFS). Using the built in libraries and jar files to analyze the datasets available on the spread of COVID-19 in India. We will be working on use cases like getting the most affected states in India date wise and overall. We will be visualizing the results using pie charts and graphs. We will also be using the MapReduce and Python libraries to visualize the outputs.

We will be writing algorithms which can be applicable to many similar types of diseases and hence these algorithms may be applicable to other pandemic like Ebola etc., Hence building a robust Health Care Intelligence.

Turning the whole of Health Care departments intelligent by removing all the dependencies in the system and making it robust. HealthCare Intelligence is our aim and future scope of this project.

TABLE OF CONTENTS

List Of Figures

3.0 System Design

5.5 Graphical Representation

Chapter 1

Introduction

The world's technological per-capita capacity to store information has roughly doubled every 40 months since the 1980s; as of 2012, every day 2.5 Exabytes(2.5×260 bytes) of data are generated. Based on a report prediction, the global data volume will grow exponentially from 4.4 ZB to 44 zettabytes between 2013 and 2020. By 2025, IDC predicts there will be 163 zettabytes of data.

Increase in data over the internet made it extremely difficult for developers to properly operate that data and this is where Big Data came into picture in 2012. Big data technologies make it extremely simpler to work with large amounts of data and made life simpler afterwards. Big Data technologies differ from traditional DBMS in many ways which makes it possible unlike DBMS and RDBMS. Hadoop ecosystem, and other cloud technologies are a way to the new internet world with no problematic issues as before. Collaboration of these technologies with domains like MapR, Spark, Pig through Java, Python and other programming languages reduces developers' effort through converting Big Data variables to traditional variable types through these programming languages. Hadoop ecosystem consists of various components like HDFS, Hadoop, Spark, Pig etc., are part of a process to work with real types of data on the internet like social media etc., and understanding this entire ecosystem enables us to use the data we want in any way like traditional methods. Using large amounts of data like from Google, Facebook, etc., efficiently is made possible due to Big Data technologies today. Not only big tech giants nowadays, even small corporations like Hospitals, Industries etc., also use big data for ease of use and convenience. Big Data not only makes working with large amounts of data easier but also simpler than to work with smaller amounts of data sets where traditional DBMS methods used became comparatively difficult to that of Big Data technologies.

Apache Hadoop is a collection of open-source software utilities that

facilitate using a network of many computers to solve problems involving massive amounts of data and computation where Apache is the company that currently owns Hadoop software. It provides a software framework for distributed storage and processing of big data using the MapReduce programming model. Hadoop data is stored in different files and the files are stored in a cluster with regard to a collection of files. HDFS abbreviated as Hadoop Distributed File System comes with Hadoop installation is collection of files which are stored in Hadoop clusters and it can be installed with Hadoop in linux of windows enables us to work with clusters where we deployed our data. The data we store in Hadoop is deployed into different files in 1 or more clusters depending on the convenience of the software and when we require to work with the data HDFS commands are useful to configure the program near the clusters and obtain the result.

For analyzing data sets in a Hadoop environment various methods are used. Some of them are MapReduce, Spark, Pig etc., many more. Variables in a Hadoop environment are different to that of the regular system variables like Java, C, Python. For example, variable in int is referred to as int writable in java, long int as long writable and string as text writable and so on and so forth. This project uses MapReduce. MapReduce is written in the Java programming language. It consists of 3 parts.

1.Mapper

2.Shuffler

3.Reducer

The Mapper parts maps the name or identity in the data set with the variable we want or to analyze and the name of the variable we are referring to is known as Key and the value associated with it is known as Value

Shuffler parts apply the program to all the files in the cluster or across the clusters or it can be said that the program can be generalized and also possible to apply to different data sets.

The Reducer part is the main crux of the program where it can only output 1 variable per program i.e., a single valued output. The reducer part reduces all the variables associated with the key into a single value it can be by any arithmetic or logical operation like multiplying all values etc., It is always advisable to apply one Reducer program for a particular Key-Value pair.

The entire MapReduce program runs on the concept of Key-Value pair and it exists in every phase of the program

1.1. Overview

"Tracking of COVID-19 in India" aims to analyze the available data related to the spread of COVID-19 in India. This project runs on Ubuntu on a Hadoop environment in Virtual box. Available data sets are deployed into the OS and Map Reduce program is written for analysis of the data sets through case studies. Later when the program runs it actually runs on the Hadoop cluster as follows.

First the program is brought near the datasets or the way in which computation is faster in the HDFS. Later the MapReduce program is applied on the data sets that are mentioned on the Hadoop commands where for executing a java file we need to first execute mapper as so on. The input file and output file are to be mentioned on the command line as a part of the compilation command. The data is stored on the given files as per the commands given on the reducer part.

1.2 Purpose

The first case of the COVID-19 pandemic in India was reported on 30 January 2020, originating from China. As of 10 May 2020, the Ministry of Health and Family Welfare have confirmed a total of 62,939 cases, 19,358 recoveries (including 1 migration) and 2,109 deaths in the country. Infection rate of COVID-19 in India is reported to be 1.7, significantly lower than in the worst affected countries.

This Project uses Hadoop Environment to analyze COVID-19 in India. The analysis is done mainly using Hadoop MapReduce and Hadoop

Distributed File Systems (HDFS). Using the built-in libraries and jar files we analyzed the datasets available on the spread of COVID-19 in India. We will be working on use cases like getting the most affected states in India date wise and overall. We will be visualizing the results using pie charts and graphs. We will also be using the MapReduce and Python libraries to visualize the outputs.

1.3 Scope

Applying the same on a larger scale on all the pandemic data available in the world and creating a separate database for the pandemics. These databases may be useful for analyzing the future and the spread of pandemics may decrease. The project can be generalized so that any similar case studies can be analyzed and extrapolated for the better extent without a new MapReduce program for each individual case study.

The whole health care sector can be turned intelligent using such technologies. Health Care Intelligence will be the future scope of this project as well as the exemplar analysis of pandemic outbreak and its repercussion in data science fields like Kaggle etc.,

1.4 Technologies Used

- **Hadoop**

Hadoop is a framework that allows you to first store Big Data in a distributed environment, so that you can process it in parallel. There are basically two components in Hadoop: HDFS and YARN.

- **MapReduce**

MapReduce is the data processing layer. It processes the huge amount of structured and unstructured data stored in HDFS. MapReduce processes data in parallel by dividing the job into a set of independent tasks. So, parallel processing improves speed and reliability.

- **HDFS**

HDFS is a distributed file system allowing multiple files to be

stored and retrieved at the same time at an unprecedented speed. It is one of the basic components of the Hadoop framework.

- **Python**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

1.5 Software and Hardware Requirements

1.5.1 Hardware Requirements

Single node running- Pentium 4 processor, 8 GB RAM, 80GB HDD or 20GB SSD or 100 (hybrid), DVD optical drive, and Broadband internet connection.

1.5.2 Software Requirements

Hadoop runs well on Linux. The operating systems of choice are:

RedHat Enterprise Linux (RHEL) 32/64 bit (64 preferable) or

Debian Linux (GNU) 32/64 bit (64 preferable)

This is a tested Linux distro that is geared for Enterprise. Comes with RedHat support

CentOS

Source compatible distro with RHEL. Free. Very popular for running Hadoop. Use a later version (version 6.x).

Ubuntu/Linux

The Server edition of Ubuntu is a good fit not the Desktop edition. Long Term Support (LTS) releases are recommended, because they continue to be updated for at least 2 years.

1.5.3 Recommended configuration

2-100 nodes running- Intel Pentium processor, 512 MB RAM, 40 GB HDD, high speed network connectivity (optic fiber recommended) & uninterrupted power supply.

1.6 Objective

Through this project i.e., through the analysis on the case studies which generated outputs in graph or text format we try to make the easy for inhabitants to perceive and comprehend the repercussions of the disease in India which in turn increases the awareness and exposure among the people also makes it easy for professional to understand the outbreak in a better way which may help in improvement of the work in any beneficial purpose. It also serves as a good example of future analysis of the disease by various people to take this project and its case studies into account as it may add up to their experience.

1.6.1 Technical Objective

The technical aspect of the project will be implemented using Hadoop MapReduce. The system must be able to access time series i.e., date wise cases of all the states affected or not affected by the corona virus and use these to calculate the overall totals based on the data.

1.6.2 Experimental Objective

This project will be implemented using a Hadoop MapReduce technology to visualize the data and analyze the data available on the spread of COVID-19 in India.

Chapter 2

System Analysis

2.1 Previous System

There were many existing technologies to analyse data,one among them was Apache Hadoop.The old existing technologies are

- Microsoft HDInsight
- NoSQL
- Hive
- Sqoop
- PolyBase
- Big data in EXCEL
- Presto

and many more...

When it comes to the previous systems that were analysing the COVID-19 data of India are covid19india a website(https://www.covid19india.org/),COVID-19 rest API for india (an API https://api.rootnet.in/),The Ministry Of Health and Family Welfare(https://www.mohfw.gov.in/),World Health Organization(https://www.who.int/).The mentioned websites and sources either use the data to analyse the further spread of COVID-19 or to showcase the current situation to the World/India.Many of the mentioned are either government officials or crowdsourced websites,so we can get the importance of the project and the level of the impact the project would be making.

2.2 Drawbacks or Cons of the Existing System

As the data sets size increases the consistency, fault tolerance etc., becomes a problem though with few thousands of cases it may not pose a problem but when the case studies are taken into consideration for an entire continent or globe significant problems arises so this cannot be disregarded as the problem seems too subtle if the data sets size is significant.These are some of the drawbacks of the existing methods.

Different systems have different drawbacks when compared amongst themselves like UI,transparency of the analysed data etc.,But when it comes to our project,we have more drawbacks than them as we are trying to replicate them.But we are more robust when compared and our system has a better future scope than theirs,as we are aiming to revolutionize the Health care system and we are trying to do the same analysis with all the pandemics.These studies of pandemics will surely predict from occurence of another pandemic in the world.The future generation will be having existing solutions to deal with the pandemics and a robust health care system,which can adapt itself to the situation.

2.3 Feasibility Study

As far as the feasibility of those various systems goes it becomes very difficult with increase in the complexity of the data sets i.e., increase in number of parameters also non-linear data sets of large quantity.

Whereas in regard to the feasibility of using Hadoop is considered firstly, there is no problem of data sets size as Hadoop clusters are deployed and data is stored in those clusters. Later it may become difficult with varied range and types of case studies that are required for the analysis of pandemic outbreak reasons and its study but it is still better compared to the traditional methods and MapReduce using Java can solve lot of problems of wide range of case studies but only for single output for each parameter with similar query problems but Apache Spark can help to solve more complex queries for diversified types of queries.

Our system is highly feasible when compared to the existing solutions

because of the following reasons:

Economically Feasibility:

This system is economically feasible as the softwares involved in the implementation Ubuntu,Apache Hadoop,Python and Java are free and open sourced to use.And the processing speed of hadoop is way higher and much preferred than other existing softwares.It is easily Scalable as well.

Technical Feasibility:

This system is highly feasible technically as there involves very little hardware requirements and they are open sourced as well.

Behavioral Feasibility:

This system is easy to use,there involves no complexity at all as all the hdfs commands are somewhat similar to the linux commands and they are easily adaptable to work on the system.This System has totally been built keeping the user's point of view in mind.

2.4 Features of this system

It shows case studies such as Top 5 heavily affected states, deceased people per district in each state, new cases registered on daily basis in data format or file format and more visual easily perceivable graphical format which consists of histograms, pie charts bar graphs etc., respectively.

2.5 Overview of Hadoop Ecosystem

Apache Hadoop is a collection of open-source software utilities providing a software framework for distributed storage and processing of big data using the MapReduce programming model. Stable possible version of Hadoop (at present) for Ubuntu or Linux is used i.e., (2.9.3). It is always advisable to download not the latest version but the previous version or its predecessor version of it for ease of use for the licensed Apache

website.

HDFS (Hadoop Distributed File System): HDFS has the most important job to perform in the Hadoop framework. It distributes the data and stores it on each node present in a cluster, simultaneously. This process reduces the total time to store data onto the disk.

MapReduce (Read/Write Large Datasets into/from Hadoop using MR): Hadoop MapReduce is another important part of the system that processes the huge volumes of data stored in a cluster. It allows parallel processing of all the data stored by HDFS. Moreover, it resolves the issue of high cost of processing through the massive scalability in a cluster.

Apache Pig (Pig is a kind of ETL for the Hadoop ecosystem): It is the high-level scripting language to write the data analysis programmes for huge data sets in the Hadoop cluster. Pig enables developers to generate query execution routines for analysis of large data sets. The scripting language is known as Pig Latin, which one key part of Pig, and the second key part is a compiler.

Apache HBase (OLTP/NoSQL) sources: It is a column-oriented database that supports the working of HDFS on a real-time basis. It is enabled to process large database tables, i.e. a file containing millions of rows and columns. An important use of HBase is the efficient use of master nodes for managing region servers.

Apache Hive (Hive is a SQL engine on Hadoop): With a SQL-like interface, Hive allows for the squaring of data from HDFS. The Hive version of SQL language is called HiveQL.

Apache Sqoop (Data Import/Export from RDBMS [SQL sources] into Hadoop): It is an application that helps with the import and export of data from Hadoop to other relational database management systems. It can transfer the bulk of your data. Sqoop is based on connector architecture that backs the plugins for establishing connectivity to new external systems.

Apache Flume(Data Import from Unstructured(Social Media sites)/Structured into Hadoop) : It is an application that allows the

storage of streaming data into a Hadoop cluster, such as data being written to log files is a good example of streaming data.

Apache Zookeeper (coordination tool used in a clustered environment): Its role is to manage the coordination between the above-mentioned applications for their efficient functioning in the Hadoop ecosystem.

2.6 Features of Hadoop

- **Open Source**:Hadoop is an open-source project, which means its source code is available free of cost for inspection, modification, and analyses that allows enterprises to modify the code as per their requirements.
- **Highly Scalable**:Hadoop cluster is scalable means we can add any number of nodes (**horizontal scalable**) or increase the hardware capacity of nodes (**vertical scalable**) to achieve high computation power. This provides horizontal as well as vertical scalability to the Hadoop framework.
- **Fault Tolerance**:Fault tolerance is the most important feature of Hadoop.HDFS in Hadoop 2 uses a **replication mechanism to** provide fault tolerance.It creates a replica of each block on the different machines depending on the replication factor (by default, it is 3). So if any machine in a cluster goes down, data can be accessed from the other machines containing a replica of the same data. Hadoop 3 has replaced this replication mechanism by **erasure coding**. Erasure coding provides the same level of fault tolerance with less space. With Erasure coding, the storage overhead is not more than 50%.
- **High Availability**:This feature of Hadoop ensures the high availability of the data, even in unfavorable conditions.Due to the fault tolerance feature of Hadoop , if any of the DataNodes goes down, the data is available to the user from different DataNodes containing a copy of the same data.\Also, the high availability Hadoop cluster consists of 2 or more running NameNodes (active and passive) in a hot standby configuration. The active node is the

NameNode, which is active. Passive node is the standby node that reads edit logs modification of active NameNode and applies them to its own namespace.

- **Cost-Effective**:Since the Hadoop cluster consists of nodes of commodity hardware that are inexpensive, thus provides a cost-effective solution for storing and processing big data. Being an open-source product, Hadoop doesn't need any license.
- **Faster in Data Processing**:Hadoop stores data in a distributed fashion, which allows data to be processed distributedly on a cluster of nodes. Thus it provides lightning-fast processing capability to the Hadoop framework.
- **Data Locality concept**:Hadoop is popularly known for its data locality feature means moving computation logic to the data, rather than moving data to the computation logic. This feature of Hadoop reduces the bandwidth utilization in a system.
- **Feasibility**:Unlike the traditional system, Hadoop can process unstructured data. Thus provide feasibility to the users to analyze data of any formats and size.
- **Easy to use**:Hadoop is easy to use as the clients don't have to worry about distributing computing. The processing is handled by the framework itself.
- **Data Reliability**:In Hadoop due to the replication of data in the cluster, data is stored reliably on the cluster machines despite machine failures.The framework itself provides a mechanism to ensure data reliability by Block Scanner, Volume Scanner, Disk Checker, and Directory Scanner. If your machine goes down or data gets corrupted, then also your data is stored reliably in the cluster and is accessible from the other machine containing a copy of data.

Chapter 3

System Design

3.1 Use Case Diagram

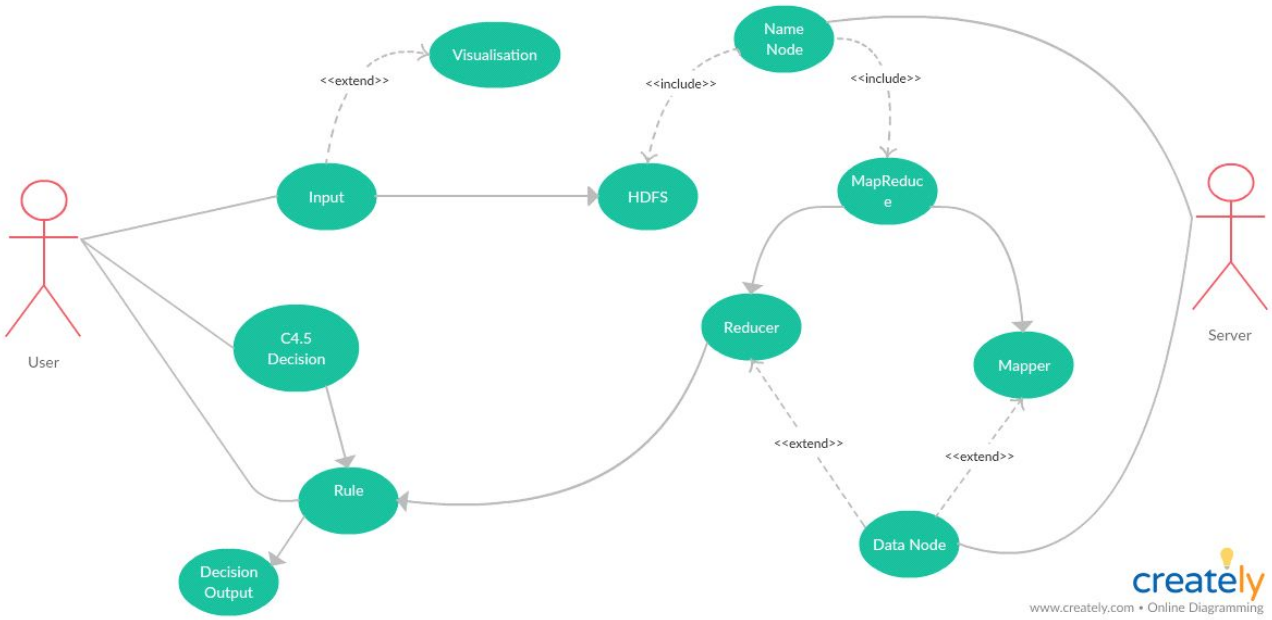

Fig 3.1 Use Case Diagram

The above use case diagram shows the relationship of the various actors with the system.We have represented the process of adding the input files to the HDFS and applying the algorithm and getting them at the HDFS end pictographically using a use case diagram.

3.2 Sequence Diagram

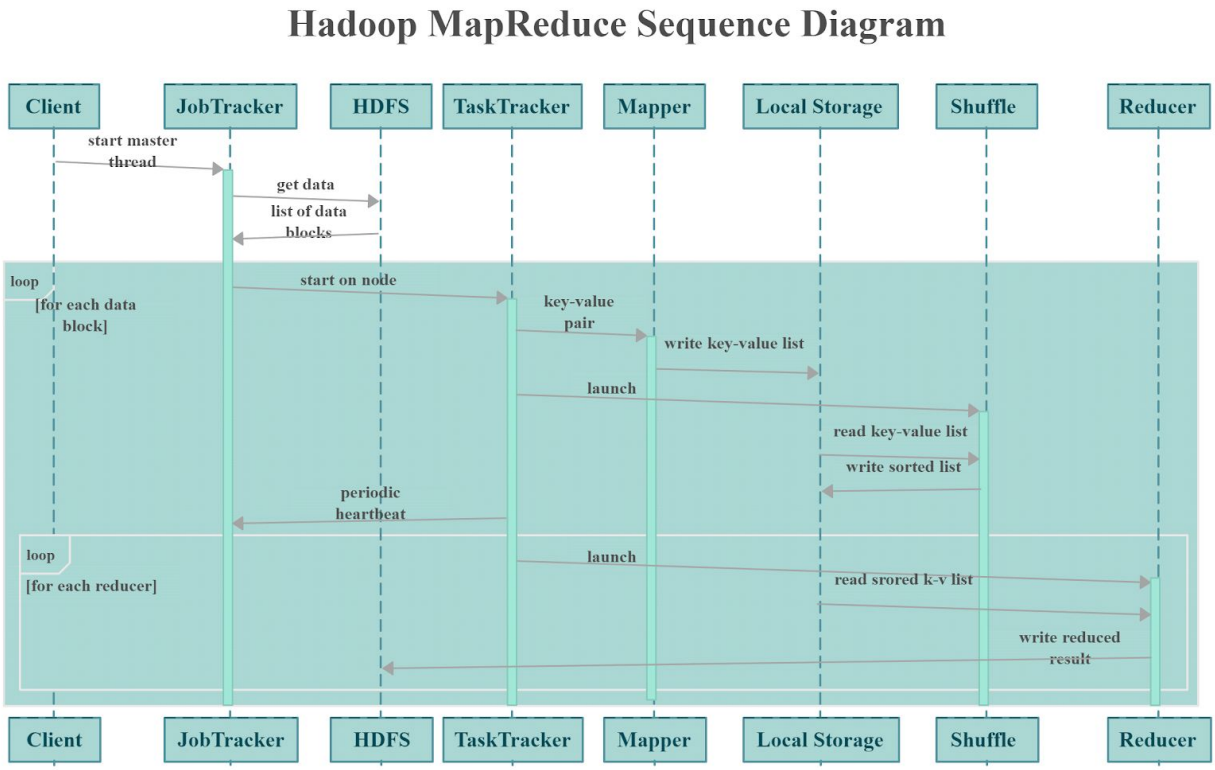

Fig 3.2 Sequence Diagram

In the above sequence diagram the working of sequencing of activities has been represented.

3.3 Data Flow Diagram

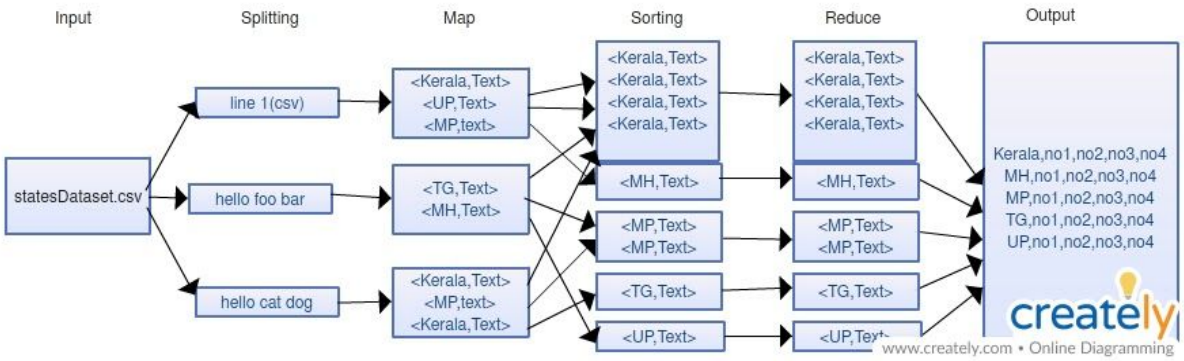

Fig 3.3 Data Flow Diagram

In the above figure the general flow diagram of any Hadoop map reduce

system has been given. Our program consists of 9 Mappers and 4 reducers. We also have used CSVReader.jar to separate out the .csv files precisely.

Chapter 4

Implementation

4.1 Description of Main Modules and Code

All the Jobs present in the program:

```
Job job1 = Job.getInstance(getConf(), "CountTotalCases");

Job job2 = Job.getInstance(getConf(), "sortByConfirmed");

Job job3 = Job.getInstance(getConf(), "sortByRecovered");

Job job4 = Job.getInstance(getConf(), "sortByDeaths");

Job job5 = Job.getInstance(getConf(), "genderStats");

Job job6 = Job.getInstance(getConf(), "TotalStats");

Job job7 = Job.getInstance(getConf(), "sortByActive");

Job job8 = Job.getInstance(getConf(), "ageStats");

Job job9 = Job.getInstance(getConf(), "nationStats");
```

Job Description:

```
job1.setJarByClass(CovidStats.class);

job1.setMapperClass(MapProc.class);

job1.setReducerClass(ReduceProc.class);

job1.setMapOutputKeyClass(Text.class);

job1.setMapOutputValueClass(Text.class);

FileInputFormat.addInputPath(job1, new Path(args[0]+"newdatatset.csv"));

FileOutputFormat.setOutputPath(job1, new Path(args[1]+"main"));
```

```
job1.waitForCompletion(true);
end = new Date().getTime();
System.out.println("\nJob 1 took " + (end-start) + "milliseconds\n");
```

MapProc Mapper:

```
public static class MapProc extends Mapper < LongWritable, Text, Text, Text > {
   private Text word = new Text();
   private long numRecords = 0;

   public void map(LongWritable offset, Text lineText, Context context)
   throws IOException,
   InterruptedException {
      String line = lineText.toString();
      Text currentWord = new Text(), currentVal;
      if (line.length() > 0) {
         CSVReader reader = new CSVReader(new StringReader(line));
         String[] nextLine;
         try {
            if ((nextLine = reader.readNext()) != null) {
               if (nextLine.length == 7) {
                  String stName = nextLine[1].trim();
                  currentWord = new Text(stName);
                  Long Confcount = Long.parseLong(nextLine[3].trim());
                  Long Reccount = Long.parseLong(nextLine[4].trim());
                  Long Deathcount = Long.parseLong(nextLine[5].trim());
                  Long Activecount = Long.parseLong(nextLine[6].trim());
```

```
                    currentVal = new Text(stName + "\t" + Confcount + "\t" + Reccount +
"\t" + Deathcount + "\t" + Activecount);
                    context.write(currentWord, currentVal);
                } else {
                    throw new Exception();
                }

            }
        } catch (NoClassDefFoundError e) {} catch (Exception e) {} finally {
            reader.close();
        }
    }
}
```

ReduceProc Reducer:

```
public static class ReduceProc extends Reducer < Text, Text, Text, Text > {
   @Override
   public void reduce(Text word, Iterable < Text > counts, Context context)
   throws IOException,
   InterruptedException {
      int sum = 0, sum1 = 0, sum2 = 0, sum3 = 0;
      int total = 0, total1 = 0, total2 = 0, total3 = 0;
      String cVal;
      Text newVal = new Text();
      for (Text count: counts) {
```

```
            cVal = count.toString();
            CSVReader reader = new CSVReader(new StringReader(cVal), '\t');
            String[] nextLine = reader.readNext();
            total = Integer.parseInt(nextLine[1].trim());
            total1 = Integer.parseInt(nextLine[2].trim());
            total2 = Integer.parseInt(nextLine[3].trim());
            total3 = Integer.parseInt(nextLine[4].trim());
            sum += total;
            sum1 += total1;
            sum2 += total2;
            sum3 += total3;
            newVal = new Text(sum + "\t" + sum1 + "\t" + sum2 + "\t" + sum3);
        }
        context.write(word, newVal);
    }
}
```

MapConf Mapper:

```
public static class MapConf extends Mapper < LongWritable, Text, LongWritable, Text > {
    private Text word = new Text();
    private long numRecords = 0;

    public void map(LongWritable offset, Text lineText, Context context)
    throws IOException,
    InterruptedException {
```

```
String line = lineText.toString();
Text currentWord = new Text(), currentVal;
LongWritable currentKey = new LongWritable();
if (line.length() > 0) {
   CSVReader reader = new CSVReader(new StringReader(line), '\t', '"', 0);
   String[] nextLine;
   try {
      if ((nextLine = reader.readNext()) != null) {
         if (nextLine.length == 5) {
            String stName = nextLine[0].trim();
            String ConfTot = nextLine[1].trim();
            String RecTot = nextLine[2].trim();
            String DeathTot = nextLine[3].trim();
            String ActiveTot = nextLine[4].trim();
            currentKey = new LongWritable(Long.parseLong(ConfTot));
                currentVal = new Text(stName + "\t" + RecTot + "\t" + DeathTot +
"\t" + ActiveTot);
            context.write(currentKey, currentVal);
         } else {
            throw new Exception();
         }

      }
   } catch (NoClassDefFoundError e) {} catch (Exception e) {} finally {
      reader.close();
```

```
        }
      }
   }
}
```

ReduceSort Reducer:

```
public static class ReduceSort extends Reducer < LongWritable, Text, LongWritable, Text > {
   public void reduce(LongWritable word, Iterable < Text > vals, Context context)
   throws IOException,
   InterruptedException {
      long max = word.get();
      word = new LongWritable(max);
      for (Text val: vals)
         context.write(word, val);

   }
}
```

MapGender Mapper:

```
public static class MapGender extends Mapper < LongWritable, Text, Text, IntWritable > {
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();
   private long numRecords = 0;

   public void map(LongWritable offset, Text lineText, Context context)
```

```
    throws IOException,
    InterruptedException {
        String line = lineText.toString();
        Text currentWord = new Text();
        if (line.length() > 0) {
            CSVReader reader = new CSVReader(new StringReader(line));
            String[] nextLine;
            try {
                if ((nextLine = reader.readNext()) != null) {
                    if (nextLine.length == 20) {
                        String gender = nextLine[5].trim();
                        currentWord = new Text(gender);
                        context.write(currentWord, one);
                    } else {
                        throw new Exception();
                    }

                }
            } catch (NoClassDefFoundError e) {} catch (Exception e) {}
        }
    }
}
```

ReduceCounts Reducer:

```
  public static class ReduceCounts extends Reducer < Text, IntWritable, Text, IntWritable > {
```

```
    @Override
    public void reduce(Text word, Iterable < IntWritable > counts, Context context)
    throws IOException,
    InterruptedException {
        int sum = 0;
        for (IntWritable count: counts) {
            sum += count.get();
        }
        context.write(word, new IntWritable(sum));
    }
}
```

and many more.,

Chapter 5

Results

5.1 State Wise Output:

In this output the name of the state along with the total confirmed cases, recovered cases, deceased, active cases are given resp.,

```
Andaman and Nicobar Islands 33   32   0    1
Andhra Pradesh1650 524  33   1093
Arunachal Pradesh  1    1    0    0
Assam     43   33   1    9
Bihar     528  129  4    395
Chandigarh     116  21   1    94
Chhattisgarh   44   36   0    8
Dadra and Nagar Haveli  0    0    0    0
Daman and Diu 0    0    0    0
Delhi     4898 1431 64   3403
```

```
Goa  7    7    0    0
Gujarat   5804 1195 319  4290
Haryana   517  253  6    258
Himachal Pradesh   41   35   2    4
Jammu and Kashmir  711  303  8    400
Jharkhand 115  27   3    85
Karnataka 651  321  27   303
Kerala    500  462  4    34
Ladakh    42   17   0    25
Lakshadweep    0    0    0    0
Madhya Pradesh2942 856  166  1920
Maharashtra    14541     2465 582  11494
Manipur   2    2    0    0
Meghalaya 12   10   1    1
Mizoram   1    0    0    1
Nagaland  0    0    0    0
Odisha    169  60   1    108
Puducherry     12   6    0    6
Punjab    1232 128  23   1081
Rajasthan 3061 1438 77   1546
Sikkim    0    0    0    0
Tamil Nadu     3550 1409 31   2110
Telangana 1085 585  29   471
Tripura   29   2    0    27
Uttar Pradesh  2766 802  50   1914
Uttarakhand    60   39   1    20
West Bengal    1259 218  133  908
```

5.2 Confirmed Cases Sort:

In this we are sorting all the states as per the confirmed cases they have.

```
14541 Maharashtra  2465 582  11494
5804 Gujarat   1195 319  4290
4898 Delhi     1431 64   3403
3550 Tamil Nadu    1409 31   2110
```

```
3061 Rajasthan 1438 77   1546
2942 Madhya Pradesh 856  166  1920
2766 Uttar Pradesh  802  50   1914
1650 Andhra Pradesh 524  33   1093
1259 West Bengal    218  133  908
1232 Punjab    128  23   1081
1085 Telangana 585  29   471
711  Jammu and Kashmir  303  8    400
651  Karnataka 321  27   303
528  Bihar     129  4    395
517  Haryana   253  6    258
500  Kerala    462  4    34
169  Odisha    60   1    108
116  Chandigarh     21   1    94
115  Jharkhand 27   3    85
60   Uttarakhand    39   1    20
44   Chhattisgarh   36   0    8
43   Assam     33   1    9
42   Ladakh    17   0    25
41   Himachal Pradesh    35   2    4
33   Andaman and Nicobar Islands 32   0    1
29   Tripura   2    0    27
12   Meghalaya 10   1    1
12   Puducherry     6    0    6
7    Goa  7    0    0
2    Manipur   2    0    0
1    Arunachal Pradesh   1    0    0
1    Mizoram   0    0    1
0    Daman and Diu  0    0    0
0    Dadra and Nagar Haveli  0    0    0
0    Lakshadweep    0    0    0
0    Nagaland  0    0    0
0    Sikkim    0    0    0
```

Similarly, we have sorted the on the basis of the recovered cases, deceased and active cases.

5.3 Nationality based:

In this we have found out from which country the affected person is from (from all the available data).

```
Canada    1
India     2658
Indonesia 15
Italy     18
Malaysia  1
Myanmar   1
Nationality    1
Philippines    2
Thailand  2
Tibet     1
United Kingdom7
United States of America     1
```

5.4 Gender based:

We have calculated the total number of males and females affected by the corona virus in India from all the available data.

```
M    3547
F    1765
```

5.5 Graphical Representations:

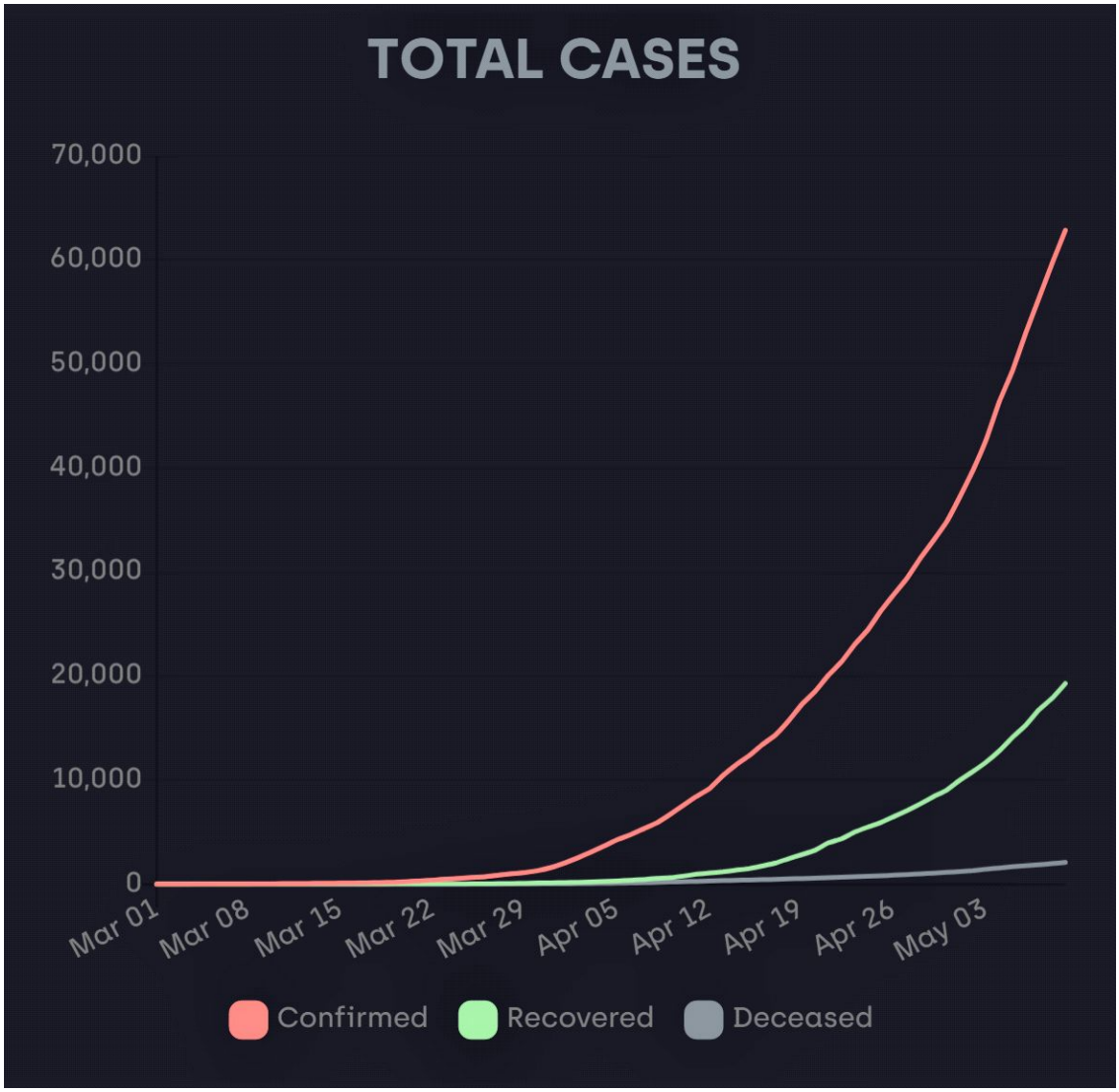

Fig 5.5.1 Total Cases

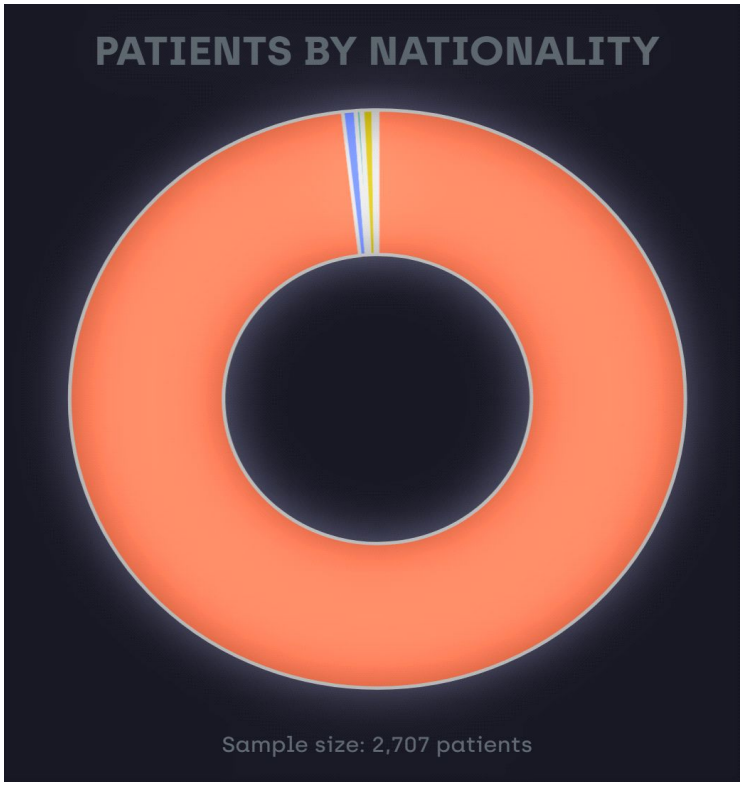

Fig 5.5.2 Patient By Nationality

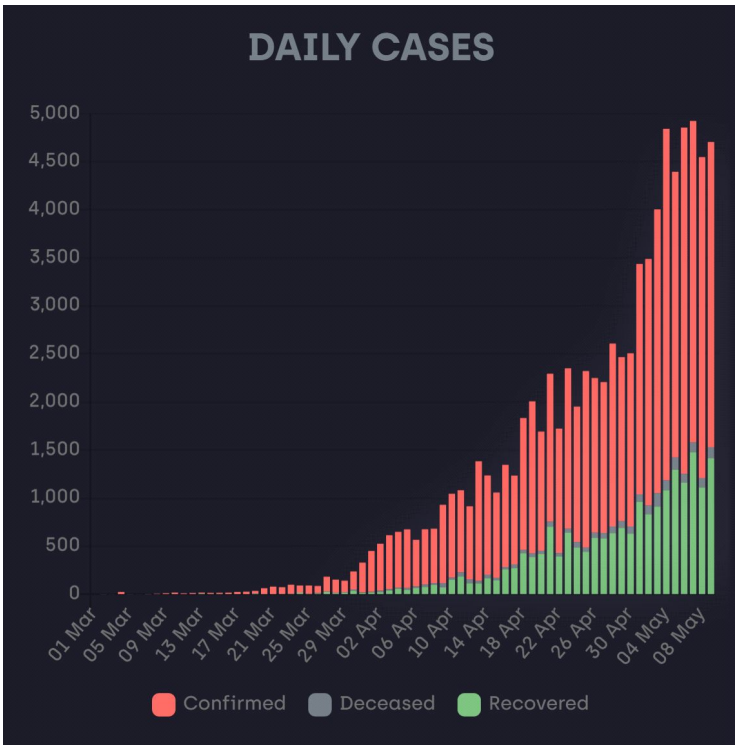

Fig 5.5.3 Daily Cases

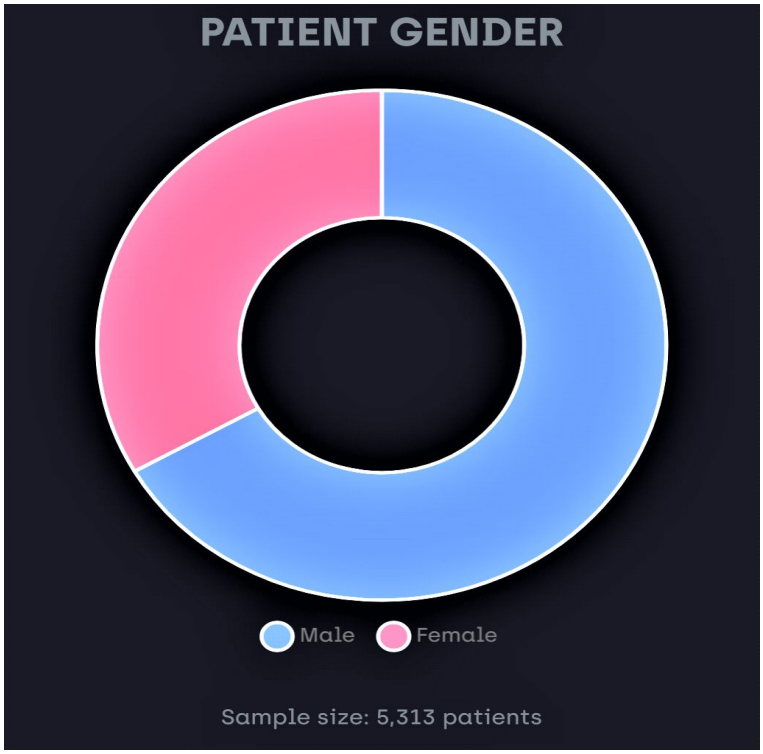

Fig 5.5.4 Patient Gender

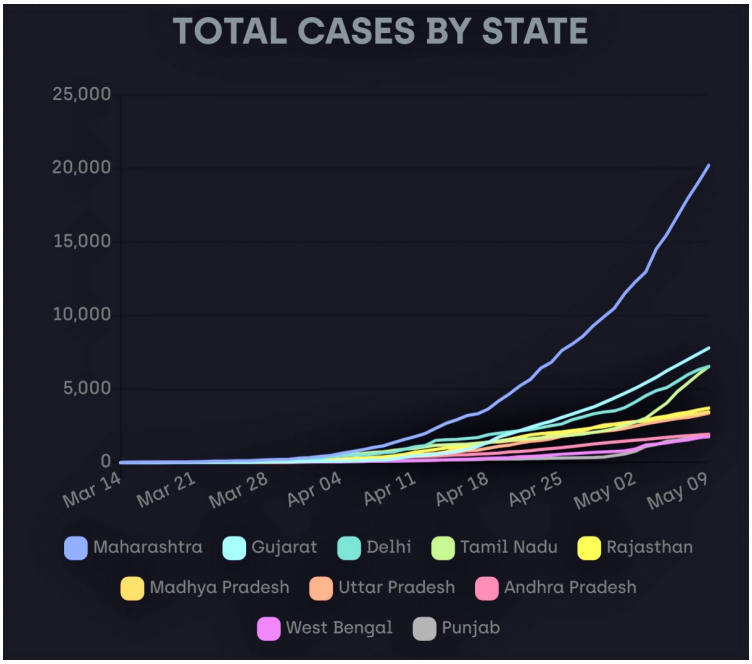

Fig 5.5.5 Total Cases By State

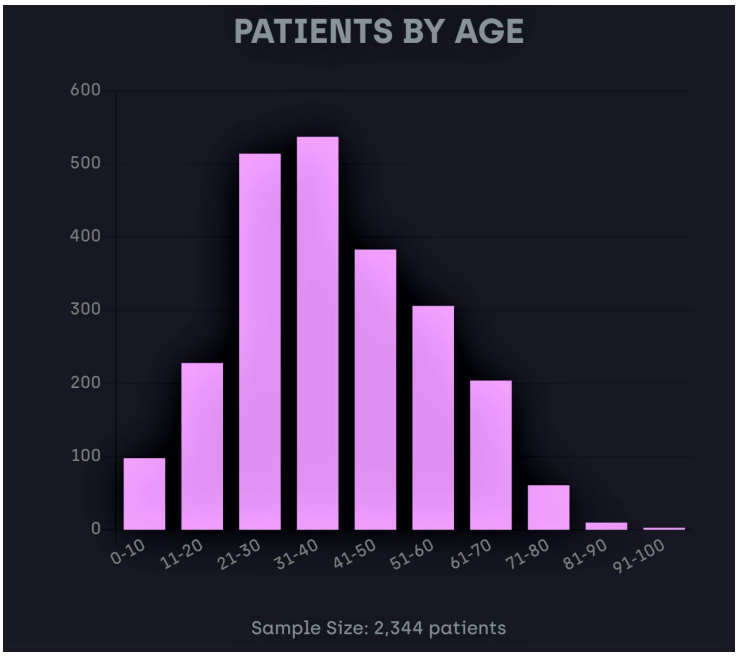

Fig 5.5.6 Patient By Age

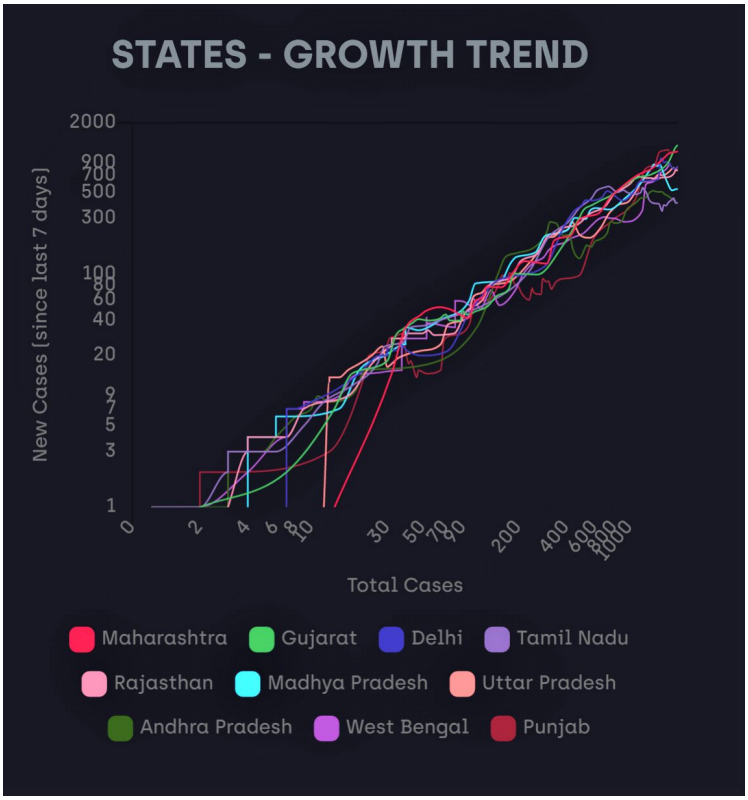

Fig 5.5.7 States-Growth Trend

Chapter 6

Conclusion & Future Scope

6.1 Conclusion

In this project we have analyzed the spread of COVID-19 in India, by considering all the states and the way they are affected by the pandemic. We have seen the analysis based on the confirmed cases, active cases, recovered cases, deaths, age group, gender of the person affected. We have visualized all the outputs and verified whether the curve has become flat or not, as the flattening of the curve is important. This project has applied the Hadoop MapReduce technique to deduce all the outputs in a quick and robust manner.

6.2 Future Scope

Applying the same on a larger scale on all the pandemic data available in the world and creating a separate database for the pandemics. These databases may be useful in future the spread of then pandemics may decrease.

The whole health care sector can be turned intelligent using such technologies. HealthCare Intelligence will be the future scope of this project.

Chapter 7

References

- YahooDeveloperNetwork: https://developer.yahoo.com/hadoop/tutorial/module4.html
- Apache Hadoop Documentation:

https://hadoop.apache.org/docs/stable/

- Python Documentation:

  https://docs.python.org/3/

- Java Documentation:

  https://docs.oracle.com/en/java/

- StackOverflow:

  https://stackoverflow.com/

- Edureka:

  https://www.edureka.co/

  https://www.youtube.com/user/edurekaIN

- Installation Guide:

  https://woir.in/steps-to-install-vm-for-workshop

- HDFS Commands:

  https://data-flair.training/blogs/hadoop-hdfs-commands/

- WordCount Tutorial:

  https://woir.in/day-1-assignments-tutorial

- Other MapReduce Projects used for reference:

  https://github.com/topics/mapreduce-java

- Datasets and Important articles:

  https://api.rootnet.in/

  https://api.covid19india.org/

  https://www.kaggle.com/

  https://towardsdatascience.com/5-datasets-about-covid-19-you-can-use-right-now-46307b1406a