**H1B-DATA ANALYSIS USING HADOOP**

INTERNSHIP PROJECT REPORT

*Submitted by*

1. **Baby Shanthi-15311A1201**

Under the esteemed Supervision of

Ms.Nagamani

In partial fulfillment of the requirement for the award of the degree of

**B.TECH**

in

Information Technology

**SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(An Autonomous Institution)**

****

Yamnampet,Ghatkesar,R.R District,Hyderabad-501301

(Affiliated to JNT University Hyderabad,Hyderabad and Approved by AICTE-New Delhi)

**Department of Information Technology**

**Sreenidhi Institute of Science and Technology**

(An autonomous Institution)



**Department of Information Technology**

**DECLARATION BY THE CANDIDATE**

I here by declare that the Internship project Report entitled **H1-B Visa Data Analysis using Hadoop** submitted by **A.Baby Shanthi-15311A1201** to **Jawaharlal Nehru Technological University, Hyderabad,** in partial fulfillment of the requirement for the award of the degree of **B.TECH** in **Information Technology** is a record of bonafide work carried out by them under the supervision of  **Ms.Nagamani** (YoungMinds Technology Solutions PVt.LTD.; )We further declare that the work reported has not been submitted and will not be submitted ,either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Place: Hyderabad Signature of the candidates

Date: August 2018 **(A.BABY SHANTHI)**

**Sreenidhi Institute of Science and Technology**

(An autonomous Institution)



**Department of Information Technology**

**BONAFIDE CERTIFICATE**

This is to certify that the Internship project Report entitled **H1-B Visa Data Analysis using Hadoop** submitted by **A.Baby Shanthi-15311A1201** to **Jawaharlal Nehru Technological University,Hyderabad,**in partial fulfillment of the requirement for the award of the degree of **B.TECH** in **Information Technology** is a record of bonafide work carried out by them under our guidance.The report fulfills the requirements as per the regulations of this University and in our opinion it meets the necessary standards for submission.The contents of this report have not been submitted and will not be submitted either in part or in full,for the award of any other degree or diploma and the same is certified.

**HOD Dr.Balaram SNSIT-IT**

**ACKNOWLEDGEMENT**

Foremost,We would like to express our deepest appreciation to the **Sreenidhi Institue of Science and Technology,Hyderabad** management who provided us the possibility to complete this report.

We would like to express our sincere gratitude to **Dr.T.CH.SHIVA REDDY,Principal of Sreenidhi Institute of Science and Technlogy,Hyderabad** for the continuos support in completion of my report.

We own the great gratitude to **Dr.Balaram Head of the Department of Information Technology ,Sreenidhi Institute ofScience and Technology,Hyderabad f**or approving the work with great interest and encouragement during the completion of work in the institution.

We would like to express ourspecial appreciation and thanks to our coordinator **Ms.Nagamani of YoungMinds Technology Solutions PVt.LTD.;** who has been a tremendous mentor for us.We would like to thank him for encuraging our work and for allowing us to grow as a graduate engineer.Your advice on both work as well as on our career has been priceless.

We also thank all the faculty members of **Sreenidhi Institute of Science and Technology,Hyderabad** for giving me the courage and strength we needed to complete our goals.This acknowledgement would be incomplete without expressing our whole hearted thanks to my family and friends who motivated me during the course of work.

**ABSTRACT**

The H1B is an employment-based, non-immigrant visa category for temporary foreign workers in the United States. For a foreign national to apply for H1B visa, an US employer must offer a job and petition for H1B visa with the US immigration department. This is the most common visa status applied for and held by international students once they complete college/ higher education (Masters, Ph.D.) and work in a full-time position. We will be performing analysis on the H1B visa applicants between the years 2011-2016.

The main objective of this project is to demonstrate by using Hadoop concepts, how data generated from datasets can be mined and utilized to make targeted, real time and informed decisions.

**CONTENTS**

**1.**Introduction to BigData

1.1.what is big data?

**1.**2.what is hadoop?

1.3.Hadoop eco System

1.4.Sketch of hdfs architecture

1.4.1 Hdfs

1.4.2 YARN

1.4.3.Hadoop frame work

2.Analysis using hadoop

2.1.Hadoop Data Analysis Technologies used in this project

2.1.1 Mapreduce

3.Hadoop SetUp

3.1. installation

3.2 mapreduce running commands

4.Coding

5.Conclusion

6.Bibliography

**INTRODUCTION**

**1.1** **What is Big Data?**

Big Data - "a collection of data sets so large and complex that it becomes difficult to process using the available database management tools. The challenges include how to capture, curate, store, search, share, analyze and visualize Big Data” . In today's environment, we have access to more types of data. These data sources include online transactions, social networking activities, mobile device services, internet gaming etc.

Big Data is a collection of data sets that are large and complex in nature. They constitute both structured and unstructured data that grow large so fast that they are not manageable by traditional relational database systems or conventional statistical tools. Big Data is defined as any kind of data source that has at least three shared characteristics:

Extremely large Volumes of data Extremely high Velocity of data Extremely wide Variety of data

According to Big Data: Concepts, Methodologies, Tools, and Applications,

Volume I by Information Resources Management Association (IRMA), "organizations today are at the tipping point in terms of managing data. Data sources are ever expanding. Data from Facebook, Twitter, YouTube, Google etc., are to grow 50X in the next 10

years. Over 2.5 exabytes of data is generated every day. Some of the sources of huge volume of data are:

1. A typical large stock exchange captures more than 1 TB of data every day.
2. There are over 5 billion mobile phones in the world which are producing enormous amount of data on daily basis.
3. YouTube users upload more than 48 hours of video every minute.
4. Large social networks such as Twitter and Facebook capture more than 10 TB of data daily.
5. There are more than 30 million networked sensors in the world which further produces TBs of data every day. "

Structured and semi-structured formats have some limitations with respect to

handling large quantities of data. Hence, in order to manage the data in the Big Data world, new emerging approaches are required, including document, graph, columnar, and geospatial database architectures. Collectively, these are referred to as NoSQL, or not only SQL, databases. In essence the data architectures need to be mapped to the types of transactions. Doing so will help to ensure the right data is available when you need it.

**1.2** **What is Hadoop?**

As organizations are getting flooded with massive amount of raw data, the challenge here is that traditional tools are poorly equipped to deal with the scale and complexity of such kind of data. That's where Hadoop comes in. Hadoop is well suited to

meet many Big Data challenges, especially with high volumes of data and data with a variety of structures.

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

Hadoop consists of two main components

1. A distributed processing framework named MapReduce (which is now supported by a component called YARN(Yet Another Resource Negotiator) and
2. A distributed file system known as the Hadoop Distributed File System, or HDFS. In Hadoop you can do any kind any kind of aggregation of data whether it is one-

month old data or one-year-old data. Hadoop provides a mechanism called MapReduce model to do distributed processing of large data which internally takes care of data even if one machine goes down.

**1.3** **Hadoop Ecosystem**

Hadoop is a shared nothing system where each node acts independently throughout the system. A framework where a piece of work is divided among several parallel MapReduce task. Each task operated independently on cheap commodity servers. This enables businesses to generate values from data that was previously considered too

expensive to be stored and processed in a traditional data warehouse or OLTP (Online Transaction Processing) environment. In the old paradigm, companies would use a traditional enterprise data warehouse system and would buy the biggest data warehouse they could afford and store the data on a single machine. However, with the increasing amount of data, this approach is no longer affordable nor practical.

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

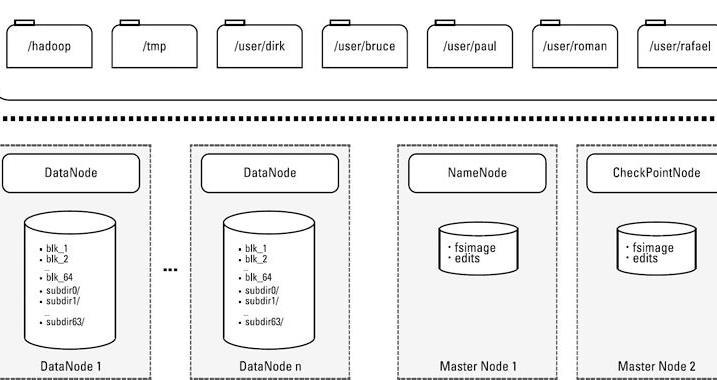
**1.4** **Sketching Out the HDFS Architecture**

1.4.1 Storage Component: Hadoop Distributed File System (HDFS)

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

more power and capacity. Google File System (GFS) was developed based on this architectural model. HDFS is designed based on similar concept.

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



-

Figure HDFS as a user-space-level file system

The above figure shows that a Hadoop cluster is made up of two classes of servers: slave nodes, where the data is stored and processed and master nodes, which

govern the management of the Hadoop cluster. On each of the master nodes and slave nodes, HDFS runs special services and stores raw data to capture the state of the file system. In the case of the slave nodes, the raw data consists of the blocks stored on the node, and with the master nodes, the raw data consists of metadata that maps data blocks to the files stored in HDFS.

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

How many times the data file has been replicated? When does the NameNode start?

How many DataNodes constitute a NameNode, capacity of the NameNode and space utilization?

The DataNode comprises of data processing, all the processing data that is stored on the DataNode and deployed on each machine.

The actual storage of the files being processed and serving read and write request for the clients

In the earlier versions of Hadoop there was only one NameNode attached to the DataNode which was a single point of failure. Hadoop version 2.x provides multiple NameNode where secondary NameNode can take over in the event of a primary

NameNode failure. Secondary NameNode is responsible for performing periodic check points in the event of a primary NameNode failure. You can start secondary NameNode by providing checkpoints that provide high availability within HDFS.

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

But what happens if one of these four machines fails? HDFS creates a self-healing architecture by replicating the same data across multiple nodes. So it can process the data in a high availability environment. For example, if we have three DataNodes and one NameNode, the data is transferred from the client environment into HDFS DataNode. The replication factor defines the number of times a data block is replicated in a clustered environment. Let’s say we have a file that is split into two data blocks across three DataNodes. If we are processing these files to a three DataNode cluster and we set the replication factor to three. If one of the nodes fails, the data from the failed nodes is redistributed among the remaining active nodes and the other nodes will complete the processing function

**1.4.2 Processing Component: Yet Another Resource Negotiator (YARN)**

YARN (Yet Another Resource Negotiator) is a resource manager that identifies on which machine a particular task is going to be executed. The actual processing of the task or program will be done by Node Manager. In Hadoop 2.2, YARN augments the MapReduce platform and serves as the Hadoop operating system. Hadoop 2.2 separates the resource management function from data processing allowing greater flexibility. This way MapReduce only performs data processing while resource management is isolated in YARN. Being the primary resource manager in HDFS, YARN enables enterprises to store data in a single place and interact with it in multiple ways with consistent levels of service. In Hadoop 1.0 the NameNode used job tracker and the DataNode used task tracker to manage resources. In Hadoop 2.x, YARN splits up into two major functionalities of the job tracker - the resource management and job scheduling. The client reports to the resource manager and the resource manager allocates resources to jobs using the resource container, Node Manager and app master. The resource container splits memory, CPU, network bandwidth among other hardware constraints into a single unit. The Node Manager receives updates from the resource containers which communicate with the app master. The Node Manager is the framework for containers, resource monitoring and for reporting data to the resource manager and scheduler.

**1.4.3 Hadoop Framework**

Hadoop Framework comprises of Hadoop Distributed File System and the

MapReduce framework. The Hadoop framework divides the data into smaller chunks and

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

In order to provide high availability what Hadoop does is replicate each part of the data onto other machines that are present within the cluster. The number of copies it will replicate depends on the replication factor. By default the replication factor is 3, in such a case there will be 3 copies of each part of the data on three different machines. In order to reduce the bandwidth and latency time, it will store two copies on the same rack and third copy on a different rack. For example, in the above example, NODE 1 and NODE 2 are on rack one and NODE 3 and NODE 4 are on rack two. Then the first two copies of part 1 will be stored on NODE 1 and third copy will be stored either on NODE 3 or NODE 4. Similar process is followed in storing remaining parts of the data. The HDFS takes care of the networking required by these nodes in order to communicate.

**2.H1-B VISA DATA ANALYSIS USING HADOOP**

H1B Visa  is a nonimmigrant Visa which is designed to allow U.S. employers to employ foreign nationals in specialty occupations in the United States of America for a specified period. People from outside U.S. who are looking for employment need to have a H1-B Visa under the Immigration and Nationality Act.

The main objective of this project is to analyze the dataset and gain insights from it like increase of applications every year from 2011 to 2016 , certified members for every year ,salaries for different occupation,success criteria for particular job or company etc.

Since many students after their graduation apply for this visa , it will be useful to know how the intake is and about the companies over their and most preferred and popular jobs,several insights like this.

**2.1** **Hadoop Data Analysis Technologies used in this project**

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

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

**2.1.1 MapReduce**

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

For this project,dataset is around 300000 records which is extracted from kaggle website. The columns are:

|  |  |  |  |
| --- | --- | --- | --- |
| Case Status | Status associated with the last significant event or decision. Valid values include “Certified,” “Certified-Withdrawn,” Denied,” and “Withdrawn”. This feature will help us analyze what share of the H-1B visa is taken by different employers/ job positions. | |  |
|  |  | |  |
|  |  | |  |
| Employer | Name of employer submitting the H1-B application. Used in comparing salaries and number of applications of various employers. | |  |
| Name |  | |  |
|  |  | |  |
| Soc Name | Standard Occupational Classification (SOC) system is a [federal statistical standard](https://www.whitehouse.gov/omb/information-regulatory-affairs/statistical-programs-standards/) used by federal agencies to classify workers into occupational categories for the purpose of collecting, calculating, or disseminating data. All workers are classified into one of 867 detailed occupations according to their occupational definition | |  |
|  |  | |  |
| Job Title | Title of the job using which we can filter specific job positions for e.g., Data Scientist, Data Engineer etc. | |  |
|  |  | |  |
|  |  | |  |
| Full Time | This contains whether the job is full time or part time | |  |
|  |  | |  |
|  |  | |  |
| Prevailing Wage | The prevailing wage for a job position is defined as the average wage paid to similarly employed workers in the requested occupation in the area of intended employment. The prevailing wage is based on the employer’s minimum requirements for the position. (Source). This column will be one of the key metrics of the data analysis. | |  |
|  |  | |  |
|  |  | |  |
| Year | This column contains the data in the year thevisa has applied | |  |
|  |  | |  |
|  |  | |  |
|  |  | |  |
| Worksite | The foreign worker’s intended area of employment. We will explore the relationship between prevailing wage for Data Scientist position across different locations. | |  |
|  |  | |  |
| Lon ,lat | Contains longitude and latitude of a particular city,to get clarity of weather conditions of the work site | |  |
|  |  | |  |
|  |  | |  |
|  |  |  |  |

**3.HADOOP SETUP**

The following section explains a step by step approach of completing Hadoop

setup on local machine.

**3.1** **Installing a Virtual Machine**

**Step 1** Create a virtual box on your operating system using the link below

<http://www.oracle.com/technetwork/serverstorage/virtualbox/downloads/index.html>

**Step 2** Setup Hadoop on your virtual box using the link below

<http://share.edureka.co/pydio/data/public/hadoop>

**Step 3** Import the file downloaded from “STEP 2” on your virtual machine

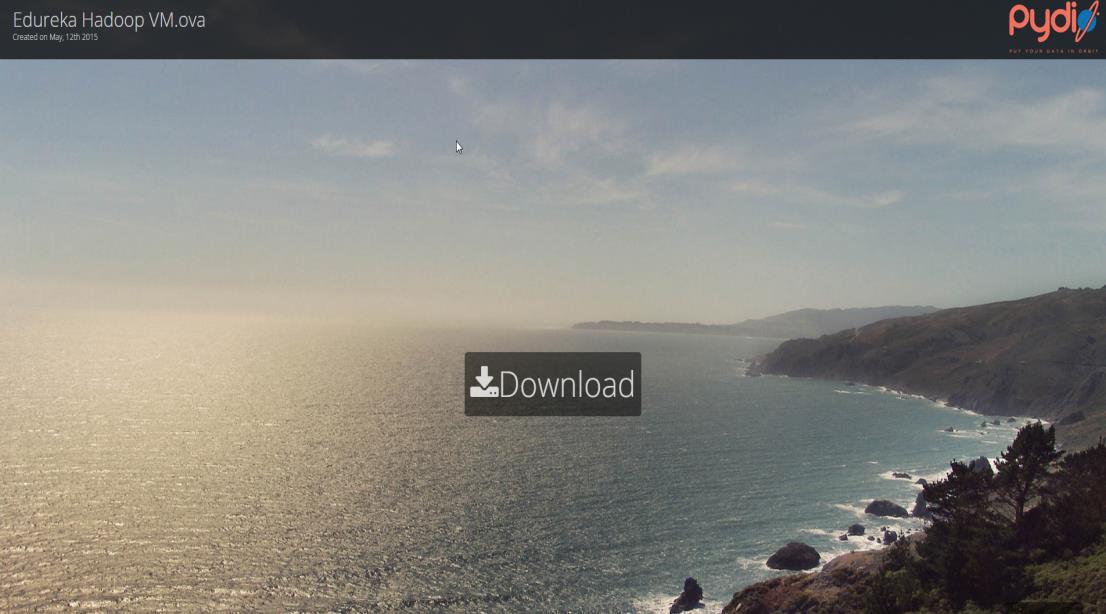


Figure Hadoop VM.

**3.2** **Running MapReduce**

**Step 1**

Type command ‘start-all.sh, to start all the demons

Type command ‘sudo jps’ to check all the demons

NameNode

ResourceManager

DataNode

Jps

NodeManger

**Step 2** To compile and execute a program in Hadoop use the following command.

This command is used to create a directory to store the compiled java classes.

**$ mkdir <dir name>**

**Step 3** Copy H1-B data set Input from local machine into HDFS using

**$ hadoop fs –put <srcpth> <des path>**

**Step 4** Now run your jar file in Hadoop environment run command

**$ hadoop jar <jarfile> <packagename. mainclass> <input file path>**

**4 .Retrieving wanted insights using mapreduce**

1. **Calculating certified people for a particular job and percentage of certified members**

**Code for map job:**

package pack;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class mapp extends Mapper<LongWritable, Text, Text, Text> {

public void map(LongWritable key, Text value, Context con)

throws IOException, InterruptedException {

String line[]=value.toString().split(",");

String status =line[26];

String job =line[10];

con.write(new Text(job) ,new Text(status));

}

}

**Code for red job:**

package pack;

import java.io.IOException;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class redd extends Reducer<Text, Text, Text, Text> {

public void reduce(Text key, Iterable<Text> values, Context con)

throws IOException, InterruptedException {

float total=0,certified=0,per=0;

for (Text val : values) {

total++;

if(val.toString().contains("CERTIFIED"))

certified++;

}

per=((certified\*100)/total);

con.write(new Text(key),new Text(total +"\t"+certified+"\t"+per));

}

}

**MapReduceDriver code:**

package pack;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class com {

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "JobName");

job.setJarByClass(pack.com.class);

job.setMapperClass(pack.mapp.class);

job.setReducerClass(pack.redd.class);

// TODO: specify output types

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

// TODO: specify input and output DIRECTORIES (not files)

FileInputFormat.setInputPaths(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

if (!job.waitForCompletion(true))

return;

}

}

**Output:**

**It consists of 336 records some of them are**

**Job name applied certified percent**

ASSOCIATES, INC." 138.0 128.0 92.753624

CHEMICALS, INC." 345.0 334.0 96.81159

CO." 631.0 609.0 96.51347

PREVIOUSLY KNOWN AS GCI" 553.0 540.0 97.649185

TOUCHE LLP" 694.0 688.0 99.135445

YOUNG U.S. LLP" 768.0 758.0 98.697914

ACCOUNTANTS 132.0 121.0 91.666664

ACCOUNTANTS AND AUDITORS 21471.0 20234.0 94.23874

ACTUARIES 1377.0 1331.0 96.6594

ADULT BASIC AND SECONDARY EDUCATION AND LITERACY 116.0 109.0 93.965515

AEROSPACE ENGINEERS 627.0 585.0 93.30144

AGRICULTURAL ENGINEERS 208.0 192.0 92.30769

AGRICULTURAL SCIENCES TEACHERS, POSTSECONDARY 202.0 192.0 95.04951

ANESTHESIOLOGISTS 193.0 176.0 91.19171

ANIMAL SCIENTISTS 263.0 243.0 92.39544

ARCHITECTURAL AND CIVIL DRAFTERS 1700.0 1530.0 90.0

ARCHITECTURAL AND ENGINEERING MANAGERS 2679.0 2521.0 94.10228

ARCHITECTURE TEACHERS, POSTSECONDARY 223.0 211.0 94.618835

AREA, ETHNIC, AND CULTURAL STUDIES TEACHERS, 160.0 155.0 96.875

ART DIRECTORS 887.0 810.0 91.31905

ART, DRAMA, AND MUSIC TEACHERS, POSTSECONDARY 633.0 573.0 90.521324

ASTRONOMERS 208.0 195.0 93.75

ATHLETIC TRAINERS 146.0 132.0 90.41096

ATMOSPHERIC AND SPACE SCIENTISTS 338.0 324.0 95.85799

ATMOSPHERIC, EARTH, MARINE, AND SPACE SCIENCES 174.0 163.0 93.67816

Actuaries 2093.0 1931.0 92.25991

Aerospace Engineers 1331.0 1239.0 93.087906

Agricultural Sciences Teachers, Postsecondary 240.0 229.0 95.416664

Animal Breeders 283.0 269.0 95.053

Anthropology and Archeology Teachers, Postsecondar 194.0 180.0 92.78351

Architectural and Engineering Managers 4403.0 4046.0 91.89189

Architecture Teachers, Postsecondary 426.0 395.0 92.72301

Area, Ethnic, and Cultural Studies Teachers, Posts 320.0 298.0 93.125

**2.Finding no of applicants for data engineer job every year**

**Map code:**

package de;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class map5 extends Mapper<LongWritable, Text, Text, Text> {

public void map(LongWritable key, Text value, Context con)

throws IOException, InterruptedException {

String line[]=value.toString().split(",");

String job=line[4];

if(job.contains("DATA ENGINEER"))

{String year=line[7];

String status=line[1];

con.write(new Text(year),new Text(status));

}}

}

**Reduce code:**

package de;

import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class red5 extends Reducer<Text, Text, Text, Text> {

public void reduce(Text key, Iterable<Text> values, Context con)

throws IOException, InterruptedException {

// process values

int count=0,count1=0;

float per;

for (Text val : values) {

count++;

if(val.toString().contains("CERTIFIED"))

count1++;

}

per=(count1\*100)/count;

con.write(new Text(key),new Text(count+"\t"+count1+"\t"+per));

}

}

**MapReduceDriver code:**

package de;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class cob {

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "JobName");

job.setJarByClass(de.cob.class);

job.setMapperClass(de.map5.class);

job.setReducerClass(de.red5.class);

// TODO: specify output types

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

// TODO: specify input and output DIRECTORIES (not files)

FileInputFormat.setInputPaths(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

if (!job.waitForCompletion(true))

return;

}

}

**Output:**

**Year applied certified percent**

2011 60 55 91.0

2012 81 78 96.0

2013 151 141 93.0

2014 249 242 97.0

2015 392 380 96.0

1. 785 755 96.0

**3.Minimum and maximum wages for a job**

**Map code:**

package wage;

import java.io.IOException;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class map3 extends Mapper<LongWritable, Text, Text, FloatWritable> {

public void map(LongWritable key, Text value, Context con)

throws IOException, InterruptedException {

String line[]=value.toString().split("\t");

if(line.length>=5){

String a=line[5];

System.out.println(a);

if(a.equals("Y")) {

if(!line[6].equals("NA")){

float sal =Float.parseFloat(line[6]);

// String state=line[24]+line[10];

String job =line[4];

con.write(new Text(job), new FloatWritable(sal));

}

}

}}

}

**Red code:**

package wage;

import java.io.IOException;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class red3 extends Reducer<Text, FloatWritable, Text, Text> {

public void reduce(Text \_key, Iterable<FloatWritable> values, Context con)

throws IOException, InterruptedException {

float max=0,min=Float.MAX\_VALUE;String out="";

for (FloatWritable val : values) {

max=Math.max(val.get(), max);

min=Math.min(val.get(), min);

out=min+"-"+max;

}

con.write(new Text(\_key),new Text(out));

}

}

**MapReduce driver code:**

package wage;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class minmax {

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "JobName");

job.setJarByClass(wage.minmax.class);

job.setMapperClass(wage.map3.class);

job.setReducerClass(wage.red3.class);

// TODO: specify output types

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(FloatWritable.class);

// TODO: specify input and output DIRECTORIES (not files)

FileInputFormat.setInputPaths(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

if (!job.waitForCompletion(true))

return;

}

}

**Output:**

Consists of 1927 records

Some of them are:

**Jobname min-max wage**

ACCOUNTANTS 38501.0-1.1919232E8

ACCOUNTANTS AND AUDITORS 12584.0-1.8123248E8

ACTORS 42723.0-48672.0

ACTUARIES 46925.0-172037.0

ADMINISTRATIVE SERVICES MANAGERS 29286.0-149500.0

ADULT BASIC AND SECONDARY EDUCATION AND LITERACY 23171.0-80621.0

ADULT BASIC AND SECONDARY EDUCATION AND LITERACY T 32240.0-42203.0

ADVERTISING AND PROMORTIONS MANAGERS 56410.0-56410.0

ADVERTISING AND PROMOTIONS MANAGER 97552.0-97552.0

ADVERTISING AND PROMOTIONS MANAGERS 16848.0-240000.0

ADVERTISING SALES AGENTS 52.0-179774.0

AEROSPACE ENGINEERING AND OPERATIONS TECHNICIANS 30638.4-98000.0

AEROSPACE ENGINEERS 42162.0-156000.0

AGENTS AND BUSINESS MANAGERS OF ARTISTS, 24731.0-177174.0

AGRICULTURAL AND FOOD SCIENCE TECHNICIANS 22277.0-84822.0

AGRICULTURAL ENGINEERS 37440.0-106226.0

AGRICULTURAL INSPECTORS 24357.0-52853.0

AGRICULTURAL SCIENCES TEACHERS, POSTSECONDARY 19160.0-8.39488E7

AGRICULTURAL WORKERS, ALL OTHER 33945.6-33945.6

AIRCRAFT CARGO HANDLING SUPERVISSORS 61984.0-61984.0

AIRCRAFT MECHANICS AND SERVICE TECHNICIANS 21840.0-96720.0

AIRFIELD OPERATIONS SPECIALISTS 58718.0-58718.0

AIRLINE PILOTS, COPILOTS, AND FLIGHT ENGINEERS 33130.0-82680.0

AMUSEMENT AND RECREATION ATTENDANTS 17035.2-17035.2

ANESTHESIOLOGISTS 36858.0-414938.0

ANIMAL BREEDERS 16765.0-49483.0

ANIMAL SCIENTISTS 31408.0-132954.0

ANIMAL TRAINERS 20800.0-33342.4

ANTHROPOLOGIST AND ARCHEOLOGIST 54558.0-54558.0

ANTHROPOLOGISTS AND ARCHEOLOGISTS 32302.0-70866.0

ANTHROPOLOGY AND ARCHEOLOGY TEACHERS, 28970.0-115200.0

ANTHROPOLOGY AND ARCHEOLOGY TEACHERS, POSTSECONDAR 42250.0-52880.0

APPLICATION DEVELOPER 52354.0-52354.0

APPLICATION PROGRAMMER 71781.0-74714.0

APPRAISERS AND ASSESSORS OF REAL ESTATE 27082.0-93101.0

ARBITRATORS, MEDIATORS, AND CONCILIATORS 58739.0-94557.0

ARCHITECT, EXCEPT LANDSCAPE AND NAVAL 49254.0-71698.0

ARCHITECTS, EXCEPT LANDSCAPE AND NAVAL 25979.0-1.1499488E8

ARCHITECTURAL AND CIVIL DRAFTERS 17992.0-83202.0

ARCHITECTURAL AND ENGINEERING MANAGERS 35651.0-2.5179648E8

ARCHITECTURAL AND ENGINEERS MANAGERS 90626.0-90626.0

ARCHITECTURAL DRAFTERS 34549.0-49650.0

ARCHITECTURE TEACHERS, POSTSECONDARY 29360.0-341286.4

ARCHIVISTS 31616.0-64501.0

AREA, ETHNIC, AND CULTURAL STUDIES TEACHERS, 22430.0-128790.0

AREA, ETHNIC, AND CULTURAL STUDIES TEACHERS, POSTS 28130.0-28130.0

ART DIRECTORS 38106.0-163966.0

ART, DRAMA AND MUSIC TEACHERS, POSTSECONDARY 25180.0-58300.0

ART, DRAMA, AND MUSIC TEACHERS, POSTSECONDARY 25250.0-697800.0

ARTISTS AND RELATED WORKERS, ALL OTHER 26180.0-52603.2

ASSOCIATE DOCTOR OF CHIROPRACTIC 68890.0-68890.0

ASTRONOMERS 48401.6-133640.0

ATHLETIC TRAINERS 24623.0-320880.0

ATMOSPHERIC AND SPACE SCIENTISTS 38000.0-126776.0

ATMOSPHERIC, EARTH, MARINE, AND SPACE SCIENCES 37550.0-129960.0

ATMOSPHERIC, EARTH, MARINE, AND SPACE SCIENCES TEA 53557.0-53557.0

ATMOSPHERIC, EARTH, MARINE, SPACE SCIENCES TEACHER 47560.0-47560.0

ATTORNEY 79123.2-82472.0

AUDIO AND VIDEO EQUIPMENT TECHNICIANS 26558.0-73923.0

AUDIO-VISUAL AND MULTIMEDIA COLLECTIONS 21600.0-73174.0

AUDIOLOGISTS 47590.0-91915.0

AUDITORS 35672.0-96741.0

AUTOMOTIVE ENGINEER 66726.0-66726.0

AUTOMOTIVE ENGINEERS 57595.0-91748.8

AUTOMOTIVE SERVICE TECHNICIANS AND MECHANICS 15080.0-249995.0

**4.Top 5 popular jobs every year**

**Map code:**

package pop;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class map6 extends Mapper<LongWritable, Text, Text, Text> {

public void map(LongWritable key, Text value, Context con)

throws IOException, InterruptedException {

String line[]=value.toString().split("\t");

if(line.length>=7){

String job=line[3];

String year=line[7];

String join=year+"\t"+job;

if(!job.equals("NA") ){

if(!year.equals("NA")){

if(isNumeric(year) && year.length()==4){

con.write(new Text(join), new Text("1"));

}

}

}

}

}

public static boolean isNumeric(String str)

{

try

{

double d = Double.parseDouble(str);

}

catch(NumberFormatException nfe)

{

return false;

}

return true;

}

}

**Reducer code:**

package pop;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import java.util.TreeSet;

import java.util.\*;

import org.apache.commons.collections.map.HashedMap;

import org.apache.commons.math3.ode.JacobianMatrices.MismatchedEquations;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class red6 extends Reducer<Text, Text, Text, Text> {

public void reduce(Text key, Iterable<Text> values, Context con)

throws IOException, InterruptedException {

long count=0;

for (Text val : values) {

count=count+1;

}

con.write(new Text(key.toString().split("\t")[0]),new Text(count+"\t"+key.toString().split("\t")[1]));

}

}

**2nd Reducer code:**

package pop;

import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

import java.util.\*;

import java.util.Map.Entry;

public class red7 extends Reducer<Text, Text, Text, Text> {

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

HashMap hashMap=new HashMap<String, Double>();

Map<String, Double> sortedMapAsc;

for (Text val : values) {

String a[]=val.toString().split("\t");

hashMap.put(a[1],Double.parseDouble(a[0]));

}

sortedMapAsc = sortByComparator(hashMap, false);

Set s=sortedMapAsc.entrySet();

Iterator iterator=s.iterator();

int i=0;

while(iterator.hasNext()){

if(i<=5){

Entry e=(Entry) iterator.next();

context.write(new Text(key+","+e.getKey().toString()),new Text(e.getValue().toString()));

}else{

break;

}

i++;

}

}

private static Map<String, Double> sortByComparator(Map<String, Double> unsortMap, final boolean order)

{

List<Entry<String, Double>> list = new LinkedList<Entry<String, Double>>(unsortMap.entrySet());

// Sorting the list based on values

Collections.sort(list, new Comparator<Entry<String, Double>>()

{

public int compare(Entry<String, Double> o1,

Entry<String, Double> o2)

{

if (order)

{

return o1.getValue().compareTo(o2.getValue());

}

else

{

return o2.getValue().compareTo(o1.getValue());

}

}

});

// Maintaining insertion order with the help of LinkedList

Map<String, Double> sortedMap = new LinkedHashMap<String, Double>();

for (Entry<String, Double> entry : list)

{

sortedMap.put(entry.getKey(), entry.getValue());

}

return sortedMap;

}

}

**Mapreducer driver code:**

package pop;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class popjob {

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "JobName");

job.setJarByClass(pop.popjob.class);

job.setMapperClass(pop.map6.class);

job.setCombinerClass(pop.red6.class);

job.setReducerClass(pop.red7.class);

// TODO: specify output types

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

// TODO: specify input and output DIRECTORIES (not files)

FileInputFormat.setInputPaths(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

if (!job.waitForCompletion(true))

return;

}}

**Output:**

**Year soc name count**

2011,Computer Systems Analysts 3354.0

2011,Computer Programmers 3110.0

2011,Computer Software Engineers, Applications 2611.0

2011,Computer Software Engineers, Systems Software 1152.0

2011,Financial Analysts 574.0

2011,Management Analysts 522.0

2012,Computer Systems Analysts 67404.0

2012,Computer Programmers 53188.0

2012,Software Developers, Applications 47839.0

2012,Computer Occupations, All Other\* 22008.0

2012,Software Developers, Systems Software 12179.0

2012,Management Analysts 9485.0

2013,Computer Systems Analysts 84459.0

2013,Computer Programmers 57646.0

2013,Software Developers, Applications 56915.0

2013,Computer Occupations, All Other 29654.0

2013,Software Developers, Systems Software 13703.0

2013,Management Analysts 9825.0

2014,Computer Systems Analysts 92840.0

2014,Software Developers, Applications 77481.0

2014,Computer Programmers 72301.0

2014,Computer Occupations, All Other 39476.0

2014,Software Developers, Systems Software 15688.0

2014,Management Analysts 11587.0

2015,COMPUTER SYSTEMS ANALYSTS 51267.0

2015,COMPUTER PROGRAMMERS 37811.0

2015,SOFTWARE DEVELOPERS, APPLICATIONS 37743.0

2015,COMPUTER OCCUPATIONS, ALL OTHER 23214.0

2015,SOFTWARE DEVELOPERS, SYSTEMS SOFTWARE 7448.0

2015,MANAGEMENT ANALYSTS 5112.0

2016,SOFTWARE DEVELOPERS, APPLICATIONS 120546.0

2016,COMPUTER SYSTEMS ANALYSTS 98322.0

2016,COMPUTER PROGRAMMERS 81713.0

2016,COMPUTER OCCUPATIONS, ALL OTHER 53760.0

2016,SOFTWARE DEVELOPERS, SYSTEMS SOFTWARE 21789.0

2016,COMPUTER SYSTEMS ANALYST 16833.0

**5.Conclusion:**

Therefore, using mapreduce in hadoop framework few insights got extracted from large data set which would be useful for further analysis.

**6.Bibliography:**

**1.deRoos et al., Dirk. 2014.** "Chapter 4 - Storing Data in Hadoop—The HadoopDistributed File System". Hadoop for Dummies. John Wiley & Sons. <http://common.books24x7.com/toc.aspx?bookid=62641>(accessed August, 2016)

**2.Share.edureka.co. 2016.** Edureka Hadoop VM Download.[*http://share.edureka.co/*](http://share.edureka.co/) [Online]February 2016.<http://share.edureka.co/pydio/data/public/hadoop>

**3.** [**www.wikipedia.org**](http://www.wikipedia.org)