**Analysing H1B Visa**

**Using Hadoop Ecosystem**

Presented by:

Name: **Vishal Duratkar**

Student ID: **S171107500150**

Center: **Pune Deccan**

**ABSTACT**

**Big data**

Big data is a term that describes the large volume of data – both structured (Relational data), semi-structured (XML data) and unstructured (Word, PDF, Text, Media Logs.) data, that overrun a business on a day-to-day basis. But it’s not the amount of data that’s important. It’s what organizations do with the data that matters.

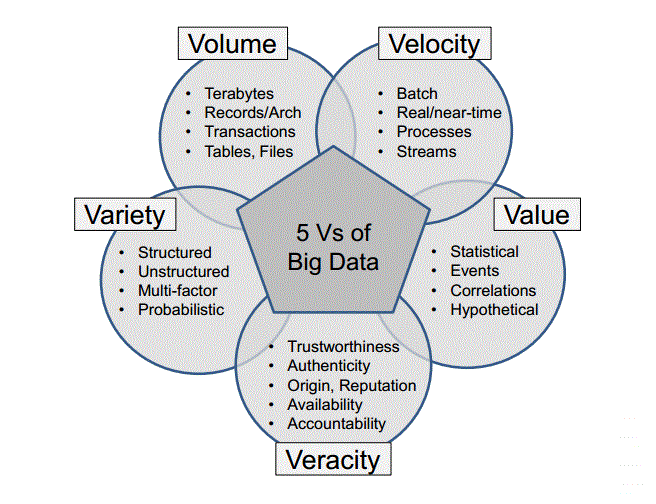


Figure 1: 5 V’s of Big data.

**Hadoop**

Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.

**Hadoop History**

As the World Wide Web grew in the late 1900s and early 2000s, search engines and indexes were created to help locate relevant information amid the text-based content. In the early years, search results were returned by humans. But as the web grew from dozens to millions of pages, automation was needed. Web crawlers were created, many as university-led research projects, and search engine start-ups took off (Yahoo, AltaVista, etc.).

One such project was an open-source web search engine called Nutch – the brainchild of Doug Cutting and Mike Cafarella. They wanted to return web search results faster by distributing data and calculations across different computers so multiple tasks could be accomplished simultaneously. During this time, another search engine project called Google was in progress. It was based on the same concept – storing and processing data in a distributed, automated way so that relevant web search results could be returned faster.

In 2006, Cutting joined Yahoo and took with him the Nutch project as well as ideas based on Google’s early work with automating distributed data storage and processing. The Nutch project was divided – the web crawler portion remained as Nutch and the distributed computing and processing portion became Hadoop (named after Cutting’s son’s toy elephant). In 2008, Yahoo released Hadoop as an open-source project. Today, Hadoop’s framework and ecosystem of technologies are managed and maintained by the non-profit Apache Software Foundation (ASF), a global community of software developers and contributors.

**Why hadoop is important?**

* **Ability to store and process huge amounts of any kind of data, quickly.** With data volumes and varieties constantly increasing, especially from social media and the Internet of Things (IoT), that's a key consideration.
* **Computing power.** Hadoop's distributed computing model processes big data fast. The more computing nodes you use, the more processing power you have.
* **Fault tolerance.** Data and application processing are protected against hardware failure. If a node goes down, jobs are automatically redirected to other nodes to make sure the distributed computing does not fail. Multiple copies of all data are stored automatically.
* **Flexibility.** Unlike traditional relational databases, you don’t have to pre-process data before storing it. You can store as much data as you want and decide how to use it later. That includes unstructured data like text, images and videos.
* **Low cost.** The open-source framework is free and uses commodity hardware to store large quantities of data.
* **Scalability.** You can easily grow your system to handle more data simply by adding nodes. Little administration is required.

**Hadoop Glossary**

**Hadoop common –** The libraries and utilities used by other Hadoop modules.

**Hadoop Distributed File System (HDFS) –** Java-based scalable system that stores data across multiple machines without prior organization.

**Yarn –** Yet another resource negotiator provides resource managements for the process running on Hadoop.

**MapReduce –** A parallel processing software framework. It is comprised of two steps. Map step is a master node that takes input and partitions them into smaller sub-problems and then distributes them to worker nodes. After the map step has taken place, the master node takes the answer to all of the sub-problems and combines them to produce output.

**Hadoop ecosystem components used in this project**

**Hive :** A data warehousing and SQL like query language that presents the data in the form of tables. Hive programming is similar to data warehousing.

**Pig :** A platform for manipulating data stored in HDFS and that includes a compiler for map reduce programs and high level language called Pig Latin.it provides a way to perform data extractions, transformation and loading and basic analysis without having to write MapReduce programs.

**Sqoop :** A connection and transfer mechanism that moves the data between Hadoop and relational databases.

**ZooKeeper :** An application that coordinates distributed processing.

**Acknowledgement**

I wish to thank our master trainer Mr. Sandeep Aggarwal and my tech mentor Ms. Jyoti Mittal for providing complete learning on big data and Hadoop and guiding me in accomplishing in the objectives of our project.

**Project Outline**

|  |  |
| --- | --- |
| Title | H1b Visa Data Analysis using Hadoop |
| Input | H1b Visa Data (2011-2016) |
| Data Elements | EMPLOYER\_NAME, CASE\_STATUS, SOC\_NAME, JOB\_TITLE, FULL\_TIME\_POSITION, PREVAILING\_WAGE, YEAR, WORKSITE, LONGITUDE, LATITUDE |
| purpose | Analyse various aspects like growth in application, Employer with highest success rate, successful job positions, areas for which most application got filed etc. |
| Methodology | Agile |

**Project implementation**

**Assumptions:**

1. Hadoop is running.

2. Ecosystem products (hive, pig) are installed

3. H1b visa data is available on HDFS in the text format.

Prerequisites:

The h1b visa data in csv format hence then needed to be converted in text file format in Hadoop file system.

**Job 1: Creating block size into 64 MB.**

Step 1. Cd /hadoop-2.7.1/etc/hadoop

step 2. /hadoop-2.7.1/etc/hadoop$ gedit hdfs-site.xml

step 3. set the blocksize(28311552 (27\*1024\*1024)) in hdfs-site.xml

<property>

<name>dfs.blocksize</name>

<value>28311552 </value>

</property>

step 4. Uploaded data into hdfs by using hadoop (hadoop fs -put gateway- /niit/ )command to check the block size . Before default block size was 128 MB and now it got changed into 27 MB.

**Job 2:**

**1 a) Is the number of petitions with Data Engineer job title increasing**

**over time?**

**b) Find top 5 job titles who are having highest growth in applications.**

1 a) Is the number of petitions with Data Engineer job title increasing over time?

**Implementation steps:**

**Technology used: MapReduce**

**MapReduce Code:**

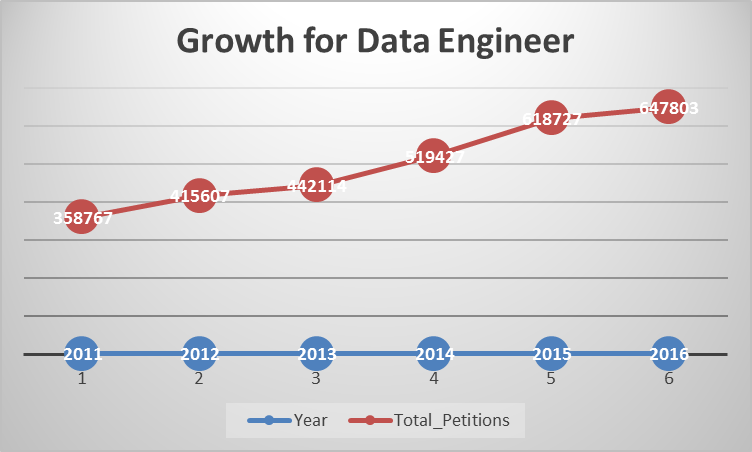
package sourceCode;  
import java.io.IOException;  
import java.util.Map;  
import java.util.TreeMap;  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.\*;  
import org.apache.hadoop.io.\*;  
import org.apache.hadoop.mapreduce.lib.input.\*;  
import org.apache.hadoop.mapreduce.lib.output.\*;  
import org.apache.hadoop.mapreduce.\*;  
  
public class Program1a   
{  
    public static class myMapper extends Mapper<LongWritable, Text, IntWritable, LongWritable>  
    {  
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException  
        {  
            String[] arr = value.toString().split("\t");  
            String jobTitle = arr[4];  
            int year = Integer.parseInt(arr[7]);  
            if(jobTitle.contains("DATA ENGINEER"));  
            context.write(new IntWritable(year),new LongWritable(1));  
        }  
    }  
      
    public static class myReducer extends Reducer<IntWritable, LongWritable, NullWritable, Text>   
    {  
        TreeMap<Integer, Long> tmap = new TreeMap<Integer, Long>();  
        static long temp = 0;  
        public void reduce(IntWritable key, Iterable<LongWritable> value, Context context) throws IOException, InterruptedException  
        {  
            long sum=0;  
            for (LongWritable val : value)   
            {  
                sum+= val.get();  
            }  
            //context.write(key, new IntWritable(sum));  
            tmap.put(key.get(), sum);  
        }  
          
        public void cleanup(Context context) throws IOException, InterruptedException   
        {  
            double avg = 0.0, finalAvg = 0.0;  
            int count = 0;  
              
            for(@SuppressWarnings("rawtypes") Map.Entry m:tmap.entrySet())  
            {  
                int mapKey = (int) m.getKey();  
                long mapValue = (long) m.getValue();  
                String tempResult = "";  
                  
                if(temp != 0)  
                {          
                    avg = (double)(mapValue - temp) \* 100 /temp;  
                    tempResult = String.format("%.2f", avg);  
                    temp = mapValue;  
                    finalAvg+=avg;                      
                    context.write(NullWritable.get(), new Text("" + mapKey + "\t" + mapValue + "\t" + tempResult + " %"));          
                }  
                else  
                {  
                    String tempString  = "Year" + "\t" + "Total\_P" + "\t" + "Growth(%)" + "\n" + mapKey + "\t" + mapValue;  
                    temp = mapValue;  
                    context.write(NullWritable.get(), new Text(tempString));          
                }  
                count++;  
            }  
            String tempResult = "\n\n Average Growth -> " + String.format("%.2f", finalAvg/count) + " %";  
            context.write(NullWritable.get(), new Text(tempResult));          
        }  
    }  
      
    public static void main(String[] args) throws ClassNotFoundException, IOException, InterruptedException  
    {  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf);  
          
        job.setJarByClass(Program1a.class);  
        job.setMapperClass(myMapper.class);  
        job.setReducerClass(myReducer.class);  
                  
        job.setMapOutputKeyClass(IntWritable.class);  
        job.setMapOutputValueClass(LongWritable.class);  
      
        job.setOutputKeyClass(NullWritable.class);  
        job.setOutputValueClass(Text.class);  
          
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
        System.exit(job.waitForCompletion(true)? 0: 1);  
          
    }  
}

**Sample output:**

Year   Total\_P   Growth(%)   
2011   358767

2012   415607   15.84%  
2013   442114    6.38%  
2014   519427   17.49%  
2015   618727   19.12%  
2016   647803    4.70 %

 Average Growth -> 10.59 %



Graph 1.A – Growth for Data Engineer

1 b) Find top 5 job titles who are having highest growth in applications.

**Technology used: MapReduce**

package sourceCode;  
import java.io.IOException;  
import java.util.Map;  
import java.util.TreeMap;  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.\*;  
import org.apache.hadoop.io.\*;  
import org.apache.hadoop.mapreduce.lib.input.\*;  
import org.apache.hadoop.mapreduce.lib.output.\*;  
import org.apache.hadoop.mapreduce.\*;

public class Program1b   
{  
    public static class myMapper extends Mapper<LongWritable, Text, Text, IntWritable>  
    {  
        public void map(LongWritable key, Text value,Context context) throws IOException, InterruptedException  
        {  
            String[] arr = value.toString().split("\t");  
            int year =  Integer.parseInt(arr[7]);  
            context.write(new Text(arr[4]), new IntWritable(year));  
        }  
    }   
    public static class myReducer extends Reducer<Text, IntWritable, NullWritable, Text>   
    {  
        TreeMap<Double, String> tmap = new TreeMap<Double, String>();  
          
        public void reduce(Text key, Iterable<IntWritable> value, Context context) throws IOException, InterruptedException  
        {  
            String rkey = key.toString();  
            long year11 = 0, year12 = 0, year13 = 0, year14 = 0, year15 =0, year16 =0;  
            double cyc1 = 0.0, cyc2 = 0.0, cyc3 = 0.0, cyc4 = 0.0, cyc5 = 0.0, finavg = 0.0;  
            for (IntWritable val : value)   
            {  
                if(val.get() == 2011)  
                    year11++;  
                else if(val.get() == 2012)  
                    year12++;   
                else if(val.get() == 2013)  
                    year13++;   
                else if(val.get() == 2014)  
                    year14++;   
                else if(val.get() == 2015)  
                    year15++;  
                else if(val.get() == 2016)  
                    year16++;   
            }  
              
            cyc1  = (year11!=0 ? ((year12 - year11) \* 100 /year11) : 0);  
            cyc2  = (year12!=0 ? ((year13 - year12) \* 100 /year12) : 0);  
            cyc3  = (year13!=0 ? ((year14 - year13) \* 100 /year13) : 0);  
            cyc4  = (year14!=0 ? ((year15 - year14) \* 100 /year14) : 0);  
            cyc5  = (year15!=0 ? ((year16 - year15) \* 100 /year15) : 0);  
              
            finavg = (cyc1 + cyc2 + cyc3 + cyc4 + cyc5)/5;  
            String temp = rkey + "\t" + "Cycle Growth -> ( " + cyc1 + "," + cyc2 + "," + cyc3 + "," + cyc4 + "," + cyc5 + ") \t" + "Average Growth -> " + finavg;  
            if((year11 + year12 + year13 + year14 + year15 + year16) > 1000)  
                tmap.put(finavg, temp);  
            /\*This condition is to exclude records like below  
             \* 2011    1,     2013    2, 2015    1, 2014    2, 2016    249,  
             \* BUSINESS ANALYST 2    Cycle Growth -> ( -100.0,0.0,0.0,-50.0,24800.0) Average Growth -> 4930.0  
            \*/  
              
            if(tmap.size() > 5)  
                tmap.remove(tmap.firstKey());  
                  
        }  
          
        public void cleanup(Context context) throws IOException, InterruptedException   
        {  
            for( String m:tmap.descendingMap().values())  
            {  
                context.write(NullWritable.get(), new Text(m));  
            }  
        }  
    }  
      
    public static void main(String[] args) throws ClassNotFoundException, IOException, InterruptedException  
    {  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf);  
          
        job.setJarByClass(Program1b.class);  
        job.setMapperClass(myMapper.class);  
        job.setReducerClass(myReducer.class);  
                  
        job.setMapOutputKeyClass(Text.class);  
        job.setMapOutputValueClass(IntWritable.class);  
      
        job.setOutputKeyClass(NullWritable.class);  
        job.setOutputValueClass(Text.class);  
          
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
        System.exit(job.waitForCompletion(true)? 0: 1);  
          
    }  
}

Sample output:

SENIOR SYSTEMS ANALYST JC60 Cycle Growth -> (21150.0,-7.0,31.0,17.0,86.0) Average Growth -> 4255.4

SYSTEMS ANALYST JC65 Cycle Growth -> (14850.0,-7.0, 23.0,-22.0,81.0) Average Growth -> 2985.0

MODULE LEAD Cycle Growth -> (14566.0,-96.0,98.0,51.0,-33.0) Average Growth -> 2917.2

SYSTEMS ANALYST - III Cycle Growth -> ( 0.0,14200.0,62.0,56.0,-27.0) Average Growth -> 2858.2

PROGRAMMER/DEVELOPER Cycle Growth -> ( -71.0,300.0,25.0,60.0,9381.0) Average Growth -> 1939.0

**Job 2:**

**a) Which part of the US has the most Data Engineer jobs for each year?**

**b) Find top 5 locations in the US who have got certified visa for each year.**

a) Which part of the US has the most Data Engineer jobs for each year?

**Technology used: Hive**

INSERT OVERWRITE DIRECTORY '/project/project2a' ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE  
select t.year, t.State, SUM(t.Total) as Count from

(

select year, trim(split(worksite,",")[1]) as State, count(\*)  as Total

from h1b\_final where job\_title like '%DATA ENGINEER%' group by year, worksite order by year, Total desc

) t group by year, t.State order by year, Count desc;

**Sample Output :**

2011    WASHINGTON    20 2014    CALIFORNIA    129   
2011    CALIFORNIA    13 2014    WASHINGTON    45

2011    NEW YORK    5 2014    NEW YORK    20

2012    WASHINGTON    32 2015    CALIFORNIA    197  
2012    CALIFORNIA    22 2015    WASHINGTON    66  
2012    MASSACHUSETTS    5 2015    NEW YORK    49

2013    CALIFORNIA    56 2016    CALIFORNIA    328  
2013    WASHINGTON    47 2016    WASHINGTON    140  
2013    GEORGIA    9 2016    NEW YORK    82

b) Find top 5 locations in the US who have got certified visa for each year.

INSERT OVERWRITE DIRECTORY '/project/project2b' ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE

select year,state, Count from

(

select s.year, s.state, s.Count, rank() over (partition by year order by Count desc) as rank from

(

select t.year, t.state, sum(t.total) as Count from

(

select year, trim(split(worksite,",")[1]) as state, count(\*) as total from partitionByYear where case\_status='CERTIFIED'group by year,worksite order by year, total desc

) t group by year,state order by year, Count desc

) s

) ranking where rank <=5;

**Sample output:**

2011    CALIFORNIA    56252 2014   CALIFORNIA   85164  
2011    NEW YORK    35244 2014   TEXAS    45091

2012    CALIFORNIA    64537 2015   CALIFORNIA   100710  
2012    NEW YORK    37086 2015   TEXAS    55066

2013    CALIFORNIA    72171 2016   CALIFORNIA   104070  
2013    NEW YORK    36460 2016   TEXAS    59694

**Job 3: Which industry has the most number of Data Scientist positions?**

**Technology used: Pig**

app\_details = load '/user/hive/warehouse/project.db/h1b\_final/\*' using PigStorage() as (s\_no:int, case\_status:chararray, employer\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_position:chararray, prevailing\_wage:int, year:int, worksite:chararray, longi:double, lati:double);

data\_extract = FOREACH app\_details generate $3 as soc\_name, $4 as job\_title;

filterByJob = FILTER data\_extract by job\_title matches '.\*DATA SCIENTIST\*.';

groupBySoc = GROUP filterByJob by soc\_name;

countBySoc = FOREACH groupBySoc generate group, COUNT(filterByJob);

orderByCount = order countBySoc by $1 desc;

Store orderByCount into '/project/project3' using PigStorage();

**Sample output:**

STATISTICIANS    564  
COMPUTER AND INFORMATION RESEARCH SCIENTISTS    466  
OPERATIONS RESEARCH ANALYSTS    350  
Computer and Information Research Scientists    205  
COMPUTER OCCUPATIONS, ALL OTHER    158  
Statisticians    135  
MATHEMATICIANS    131  
SOFTWARE DEVELOPERS, APPLICATIONS    127  
COMPUTER SYSTEMS ANALYSTS    111  
Operations Research Analysts    102

**Job 4: Which top 5 employers file the most petitions each year considering all case status?**

**Technology used: Hive**

INSERT OVERWRITE DIRECTORY '/project/project4' ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE  
select year,employer\_name,Total from   
(  
  select year,employer\_name, rank() over (partition by year order by Total desc) as rank, Total from

(

select year, employer\_name, count(\*) as Total from partitionByYear group by year, employer\_name order by year, Total Desc

) inner\_table  
) rankings where rankings.rank <=5;

**Sample output:**

2011   TATA CONSULTANCY SERVICES LIMITED    5416  
2011    MICROSOFT CORPORATION    4253

2012    INFOSYS LIMITED    15818  
2012    WIPRO LIMITED    7182

2013    INFOSYS LIMITED    32223  
2013    TATA CONSULTANCY SERVICES LIMITED    8790

2014    INFOSYS LIMITED    23759  
2014    TATA CONSULTANCY SERVICES LIMITED    14098

2015    INFOSYS LIMITED    33245  
2015    TATA CONSULTANCY SERVICES LIMITED    16553

2016    INFOSYS LIMITED    25352  
2016    CAPGEMINI AMERICA INC    16725  
2016    TATA CONSULTANCY SERVICES LIMITED    13134

**Job 5: Find the most popular top 10 job positions for H1B visa applications for each year?**

1. For all application

**Technology used: Hive**

INSERT OVERWRITE DIRECTORY '/project/project5a' ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE

Select year,job\_title, total from

(

select t.year, t.job\_title, t.Total, rank() over (partition by year order by total desc) as rank from

(

select year,job\_title, count(\*) as Total from partitionByYear group by year,job\_title order by year, Total desc

) t

) ranking where rank <=10;

**Sample output:**

2011    PROGRAMMER ANALYST    31799  
2011    SOFTWARE ENGINEER    12763

2012    PROGRAMMER ANALYST    33066  
2012    SOFTWARE ENGINEER    14437

2013    PROGRAMMER ANALYST    33880  
2013    SOFTWARE ENGINEER    15680

2014    PROGRAMMER ANALYST    43114  
2014    SOFTWARE ENGINEER    20500

2015    PROGRAMMER ANALYST    53436  
2015    SOFTWARE ENGINEER    27259

2016    PROGRAMMER ANALYST    53743  
2016    SOFTWARE ENGINEER    30668

1. For only certified application

INSERT OVERWRITE DIRECTORY '/project/project5b' ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE  
select year,job\_title, total from

(

select t.year, t.job\_title, t.Total, rank() over (partition by year order by total desc) as rank from

(

select year,job\_title, count(\*) as Total from partitionByYear where case\_status='CERTIFIED' group by year,job\_title order by year, Total desc

) t

) ranking where rank <=10;

**Sample output:**

2011    PROGRAMMER ANALYST    28806  
2011    SOFTWARE ENGINEER    11224

2012    PROGRAMMER ANALYST    29226  
2012    SOFTWARE ENGINEER    12273

2013    PROGRAMMER ANALYST    29906  
2013    SOFTWARE ENGINEER    12973

2014    PROGRAMMER ANALYST    38625  
2014    SOFTWARE ENGINEER    17278

2015    PROGRAMMER ANALYST    48203  
2015    SOFTWARE ENGINEER    23352

2016    PROGRAMMER ANALYST    47964  
2016    SOFTWARE ENGINEER    25890

**Job 6: Finding the percentage and count of each case status and also finding the number of applications then depicting this result in the chart**.

**Technology used: Pig**

app\_details = load '/user/hive/warehouse/project.db/h1b\_final/\*' using PigStorage() as (s\_no:int, case\_status:chararray, employer\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_position:chararray, prevailing\_wage:int, year:int, worksite:chararray, longi:double, lati:double);

data\_extract = FOREACH app\_details generate $1 as case\_status, $7 as Year;

groupByYear = GROUP data\_extract by $1;

--Total app count by year

countByYear = FOREACH groupByYear generate group, COUNT(data\_extract) as Total;

groupByCaseYear = GROUP data\_extract by (Year,case\_status);

countgroupByCaseYear = FOREACH groupByCaseYear generate group, COUNT(data\_extract);

sep\_countgroupByCaseYear = FOREACH countgroupByCaseYear generate group.Year as Year, group.case\_status as Case\_Status, $1 as Total;

join\_final = join sep\_countgroupByCaseYear by $0, countByYear by $0;

Final\_Output = FOREACH join\_final generate $0,$1,$2,$4, ROUND\_TO((double)($2 \* 100 )/$4,2);

Store Final\_Output into '/project/project6' using PigStorage('\t');

**Sample output:**

2011 DENIED 29130 358767 8.12

2011 CERTIFIED 307936 358767 85.83

2011 WITHDRAWN 10105 358767 2.82

2011 CERTIFIED-WITHDRAWN 11596 358767 3.23

2012 DENIED 21096 415607 5.08

2012 CERTIFIED 352668 415607 84.86

2012 WITHDRAWN 10725 15607 2.58

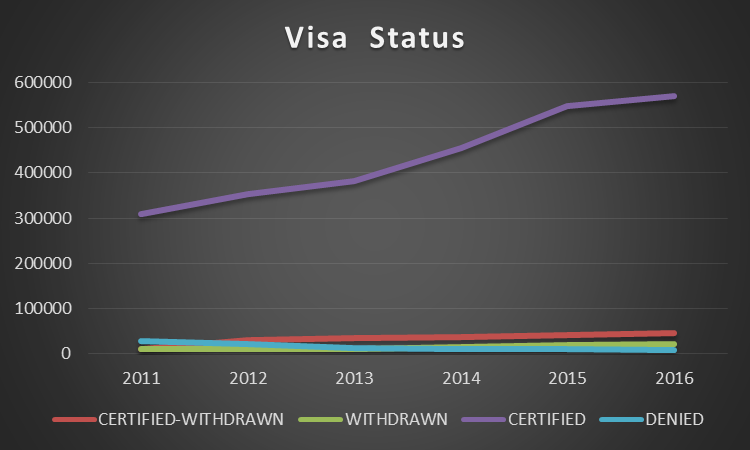
2012 CERTIFIED-WITHDRAWN 31118 415607 7.49

2013 CERTIFIED-WITHDRAWN 35432 442114 8.01

2013 WITHDRAWN 11590 442114 2.62

2013 CERTIFIED 382951 442114 86.62

2013 DENIED 12141 442114 2.75



Graph 6 – Growth for Visa case status

**Job 7: Create a bar graph to depict the number of applications for each year.**

**Technology used: Pig**

app\_details = load '/user/hive/warehouse/project.db/h1b\_final/\*' using PigStorage() as (s\_no:int, case\_status:chararray, employer\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_position:chararray, prevailing\_wage:int, year:int, worksite:chararray, longi:double, lati:double);  
  
data\_extract = FOREACH app\_details generate $1 as case\_status, $7 as Year;

groupByYear = GROUP data\_extract by $1;

countByYear = FOREACH groupByYear generate group, COUNT(data\_extract) as Total;

Store countByYear into '/project/project7' using PigStorage('\t');

**Sample output:**

2011    358767  
2012    415607  
2013    442114  
2014    519427  
2015    618727  
2016    647803



Graph 6 – Growth for Visa Application

**Job 8: Find the average Prevailing Wage for each Job for each Year (take part time and full time separate).**

**Technology used: Pig**

app\_details = load '/user/hive/warehouse/project.db/h1b\_final/\*' using PigStorage() as (s\_no:int, case\_status:chararray, employer\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_position:chararray, prevailing\_wage:int, year:int, worksite:chararray, longi:double, lati:double);

data\_extract = FOREACH app\_details generate $4 as job, $5 as shift, $6 as wage, $7 as year;

year11 = filter data\_extract by year == 2011 and shift=='Y';

groupByJob11 = GROUP year11 by job;

avg\_cal11 = Foreach groupByJob11 generate group, AVG(year11.wage) as Average;

fullTimeOrdered11 = Order avg\_cal11 by Average desc;

Store fullTimeOrdered11 into '/project/project8/2011/fullTime' using PigStorage();

app\_details = load '/user/hive/warehouse/project.db/h1b\_final/\*' using PigStorage() as (s\_no:int, case\_status:chararray, employer\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_position:chararray, prevailing\_wage:int, year:int, worksite:chararray, longi:double, lati:double);

data\_extract = FOREACH app\_details generate $4 as job, $5 as shift, $6 as wage, $7 as year;

year11 = filter data\_extract by year == 2011 and shift=='N';

groupByJob11 = GROUP year11 by job;

avg\_cal11 = Foreach groupByJob11 generate group, AVG(year11.wage) as Average;

partTimeOrdered11 = Order avg\_cal11 by Average desc;

Store partTimeOrdered11 into '/project/project8/2011/partTime' using PigStorage();

**Sample output:**

2011 Full Time

ASSOCIATE SQA ENGINEER    9853122.909090908  
TEACHER (MATHEMATICS)    9491354.0  
SENIOR AUDIT ASSOCIATE    9422870.636363637  
ELEMENTARY SCHOOL SPANISH TEACHER    9078950.0

2011 Part Time

INSTRUCTOR OF COMPUTER SCIENCE    189820.0  
HEAD, GLOBAL FINANCIAL SECTOR PRACTICE    189779.0  
SENIOR VICE PRESIDENT, SURVEY RESEARCH    189779.0  
DIRECTOR OF BRAND PARTNERSHIPS    189280.0

**Job 9: Which are top ten employers who have the highest success rate morethan 70% in petitions filed more than 1000?**

**Technology used: MapReduce**

package sourceCode;  
import java.io.IOException;  
import java.util.Comparator;  
import java.util.Map;  
import java.util.TreeMap;  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.\*;  
import org.apache.hadoop.io.\*;  
import org.apache.hadoop.mapreduce.lib.input.\*;  
import org.apache.hadoop.mapreduce.lib.output.\*;  
import org.apache.hadoop.mapreduce.\*;  
import sourceCode.Program1a.myMapper;  
import sourceCode.Program1a.myReducer;  
public class Program9   
{  
    public static class myMapper extends Mapper<LongWritable, Text, Text, Text>  
    {  
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException   
        {  
            String[] arr = value.toString().split("\t");  
            String employer\_name = arr[2].trim();  
            String case\_status = arr[1].trim();  
            context.write(new Text(employer\_name), new Text(case\_status));  
        }  
    }  
      
    public static class myReducer extends Reducer<Text, Text, NullWritable, Text>  
    {  
        Map<String, Double> tmap = new TreeMap<String, Double>();  
          
        public void reduce(Text key, Iterable<Text> value, Context context)  
        {  
            long totalCount = 0, certifiedCount = 0, certifiedWCount = 0;  
            for (Text val : value)   
            {  
                String temp = val.toString().toUpperCase();  
                if(temp.contains("CERTIFIED"))  
                    certifiedCount++;  
                else if (temp.contains("CERTIFIED-WITHDRAWN"))  
                    certifiedWCount++;  
                totalCount++;  
            }  
            if(totalCount > 1000)  
            {  
                double successRate = (double)((certifiedCount + certifiedWCount) \* 100)/totalCount;  
                if(successRate > 70)  
                    tmap.put(key.toString(), successRate);  
            }              
        }  
          
        class myClass implements Comparator<String>  
        {  
            Map<String, Double> tmap;                  
            public myClass(Map<String, Double> temp)  
            {    this.tmap = temp;    }  
              
            public int compare(String arg0, String arg1)   
            {  
                // TODO Auto-generated method stub  
                if(tmap.get(arg0) >= tmap.get(arg1))  
                    return -1;  
                else   
                    return 1;  
            }  
        }  
        public void cleanup(Context context) throws IOException, InterruptedException   
        {  
            myClass obj = new myClass(tmap);  
            Map<String, Double> tmap1 = new TreeMap<String, Double>(obj);  
            tmap1.putAll(tmap);  
            for (Map.Entry m : tmap1.entrySet())   
            {      
                String s\_rate = String.format("%.2f", (double)m.getValue());  
                context.write(NullWritable.get(), new Text("" + m.getKey().toString() + "\t" + s\_rate ));  
            }                                  
        }      
    }  
    public static void main(String[] args) throws ClassNotFoundException, IOException, InterruptedException  
    {  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf);  
          
        job.setJarByClass(Program9.class);  
        job.setMapperClass(myMapper.class);  
        job.setReducerClass(myReducer.class);  
                  
        job.setMapOutputKeyClass(Text.class);  
        job.setMapOutputValueClass(Text.class);  
      
        job.setOutputKeyClass(NullWritable.class);  
        job.setOutputValueClass(Text.class);  
          
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
        System.exit(job.waitForCompletion(true)? 0: 1);  
          
    }  
}

**Sample output:**

INFOSYS LIMITED 99.54

DIASPARK, INC. 99.51

ACCENTURE LLP 99.39

TECH MAHINDRA (AMERICAS),INC. 99.34

TATA CONSULTANCY SERVICES LIMITED 99.34

YASH TECHNOLOGIES, INC. 99.28

YASH & LUJAN CONSULTING, INC. 99.27

HCL AMERICA, INC. 99.27

RELIABLE SOFTWARE RESOURCES, INC. 99.15

NTT DATA, INC. 99.13

**Job 10: Which are the top 10 job positions which have the highest success rate morethan 70% in petitions filed more than 1000?**

**Technology used: MapReduce**

package sourceCode;  
import java.io.IOException;  
import java.util.Comparator;  
import java.util.Map;  
import java.util.TreeMap;  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.\*;  
import org.apache.hadoop.io.\*;  
import org.apache.hadoop.mapreduce.Mapper.Context;  
import org.apache.hadoop.mapreduce.lib.input.\*;  
import org.apache.hadoop.mapreduce.lib.output.\*;  
import org.apache.hadoop.mapreduce.\*;  
  
public class Program10   
{  
    public static class myMapper extends Mapper<LongWritable, Text, Text, Text>  
    {  
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException   
        {  
            String[] arr = value.toString().split("\t");  
            String job\_title = arr[4].trim();  
            String case\_status = arr[1].trim();  
            context.write(new Text(job\_title), new Text(case\_status));  
        }  
    }  
      
    public static class myReducer extends Reducer<Text, Text, NullWritable, Text>  
    {  
        Map<String, Double> tmap = new TreeMap<String, Double>();  
          
        public void reduce(Text key, Iterable<Text> value, Context context)  
        {  
            long totalCount = 0, certifiedCount = 0, certifiedWCount = 0;  
            for (Text val : value)   
            {  
                String temp = val.toString().toUpperCase();  
                if(temp.contains("CERTIFIED"))  
                    certifiedCount++;  
                else if (temp.contains("CERTIFIED-WITHDRAWN"))  
                    certifiedWCount++;  
                totalCount++;  
            }  
            if(totalCount > 1000)  
            {  
                double successRate = (double)((certifiedCount + certifiedWCount) \* 100)/totalCount;  
                if(successRate > 70)  
                    tmap.put(key.toString(), successRate);  
            }              
        }  
          
        class myClass implements Comparator<String>  
        {  
            Map<String, Double> tmap;                  
            public myClass(Map<String, Double> temp)  
            {    this.tmap = temp;    }  
              
            public int compare(String arg0, String arg1)   
            {  
                // TODO Auto-generated method stub  
                if(tmap.get(arg0) >= tmap.get(arg1))  
                    return -1;  
                else   
                    return 1;  
            }  
        }  
        public void cleanup(Context context) throws IOException, InterruptedException   
        {  
            myClass obj = new myClass(tmap);  
            Map<String, Double> tmap1 = new TreeMap<String, Double>(obj);  
            tmap1.putAll(tmap);  
            for (Map.Entry m : tmap1.entrySet())   
            {      
                String s\_rate = String.format("%.2f", (double)m.getValue());  
                context.write(NullWritable.get(), new Text("" + m.getKey().toString() + "\t" + s\_rate ));  
            }                                  
        }      
    }  
    public static void main(String[] args) throws ClassNotFoundException, IOException, InterruptedException  
    {  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf);  
          
        job.setJarByClass(Program10.class);  
        job.setMapperClass(myMapper.class);  
        job.setReducerClass(myReducer.class);  
                  
        job.setMapOutputKeyClass(Text.class);  
        job.setMapOutputValueClass(Text.class);  
      
        job.setOutputKeyClass(NullWritable.class);  
        job.setOutputValueClass(Text.class);  
          
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
        System.exit(job.waitForCompletion(true)? 0: 1);  
           
    }  
}

**Sample output:**

ASSOCIATE CONSULTANT - US 99.93

SYSTEMS ENGINEER - US 99.90

TEST ENGINEER - US 99.86

PRODUCTION SUPPORT ANALYST - US 99.86

TEST ANALYST - US 99.82

CONSULTANT - US 99.81

TECHNOLOGY LEAD - US 99.80

TECHNICAL TEST LEAD - US 99.80

SENIOR TECHNOLOGY ARCHITECT - US 99.79

**Job 11: Export result for question no 10 to MySql database.**

**Technology used: MyQql and Sqoop**

mysql> create database project;

mysql> use project;

mysql> create table problem10data ( job\_title varchar(40) not null, success\_rate double not null, primary key(job\_title));

sqoop export --connect jdbc:mysql://localhost/project --username root --password 'vishal' --table problem10data --export-dir /project/project10/part-r-00000 --input-fields-terminated-by '\t' -m 1;

**Sample output:**

.NET DEVELOPER | 95.04 |

| ACCOUNT MANAGER | 90.15 |

| ACCOUNTANT | 83.47 |

| ADVISORY MANAGER | 98.31 |

| ADVISORY SENIOR | 98.65 |

| ADVISORY SENIOR ASSOCIATE | 97.52 |

| ADVISORY STAFF | 98.01 |

| ANALYST | 95.19 |

| APPLICATION DEVELOPER | 91.46 |

| APPLICATION ENGINEER | 94.38

**Conclusion**

Following conclusion I can draw based on task performed by me.

1. MapReduce code in java makes the complex analysis quite easy. Code can be written to handle very complex join operations, apply different logic, which can be handled efficiently by using this approach.
2. Hive helps in the cleaning up of the data. Also allows us to use SQL on big data.
3. For normal group by join and filter based data retrieval, pig is very efficient
4. Sqoop is useful when we have data in sql tables that need to be imported in the hadoop file system.

**Reference**

* <http://www.student.niitcloudcampus.com/CloudCampus/Welcome.aspx>
* <https://www.sas.com/en_us/insights/big-data/hadoop.html>
* <https://www.google.co.in/>