**Project Documentation**

**What is Big Data?**

**Big Data** is a collection of large datasets that cannot be processed usingtraditional computing techniques. It is not a single technique or a tool, rather it involves many areas of business and technology.

**What Comes Under Big Data?**

Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data.

* **Black Box Data:** It is a component of helicopter, airplanes, and jets, etc.It captures voices of the flight crew, recordings of microphones and earphones, and the performance information of the aircraft.
* **Social Media Data:** Social media such as Facebook and Twitter holdinformation and the views posted by millions of people across the globe.
* **Stock Exchange Data:** The stock exchange data holds information aboutthe ‘buy’ and ‘sell’ decisions made on a share of different companies made by the customers.
* **Power Grid Data:** The power grid data holds information consumed by aparticular node with respect to a **base** station.
* **Transport Data:** Transport data includes model, capacity, distance andavailability of a vehicle.
* **Search Engine Data:** Search engines retrieve lots of data from differentdatabases.

Thus Big Data includes huge volume, high velocity, and extensible variety of data. The data in it will be of three types.

* **Structured data**: Relational data.
* **Semi Structured data**: XML data.
* **Unstructured data**: Word, PDF, Text, Media Logs.

**Benefits of Big Data**

* Using the information kept in the social network like Facebook, the marketing agencies are learning about the response for their campaigns, promotions, and other advertising mediums.
* Using the information in the social media like preferences and product perception of their consumers, product companies and retail organizations are planning their production.
* Using the data regarding the previous medical history of patients, hospitals are providing better and quick service.

**Data Technologies**

Big data technologies are important in providing more accurate analysis, which may lead to more concrete decision-making resulting in greater operational efficiencies, cost reductions, and reduced risks for the business.

To harness the power of big data, you would require an infrastructure that can manage and process huge volumes of structured and unstructured data in real-time and can protect data privacy and security.

There are various technologies in the market from different vendors including Amazon, IBM, Microsoft, etc., to handle big data. While looking into the technologies that handle big data, we examine the following two classes of technology:

**Operational Big Data**

These include systems like MongoDB that provide operational capabilities for real-time, interactive workloads where data is primarily captured and stored.

NoSQL Big Data systems are designed to take advantage of new cloud computing architectures that have emerged over the past decade to allow massive computations to be run inexpensively and efficiently. This makes operational big data workloads much easier to manage, cheaper, and faster to implement.

Some NoSQL systems can provide insights into patterns and trends based on real-time data with minimal coding and without the need for data scientists and additional infrastructure.

**Analytical Big Data**

These includes systems like Massively Parallel Processing (MPP) database systems and MapReduce that provide analytical capabilities for retrospective and complex analysis that may touch most or all of the data.

MapReduce provides a new method of analyzing data that is complementary to the capabilities provided by SQL, and a system based on MapReduce that can be scaled up from single servers to thousands of high and low end machines.

**Big Data Challenges**

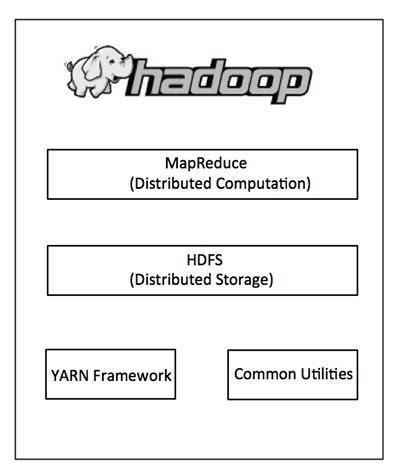
The major challenges associated with big data are as follows:

* Capturing data
* Analysis
* Storage
* Searching
* Sharing

**Hadoop Architecture**

At its core, Hadoop has two major layers namely:

1. Processing/Computation layer (MapReduce), and
2. Storage layer (Hadoop Distributed File System).



**MapReduce**

MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner.

**What is MapReduce?**

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, *reduce* task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into *mappers* and *reducers* is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

* MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

o **Map stage**: The map or mapper’s job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

o **Reduce stage**: This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

**Hadoop Distributed File System**

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets.

Apart from the above-mentioned two core components, Hadoop framework also includes the following two modules:

* **Hadoop Common:** These are Java libraries and utilities required by otherHadoop modules.
* **Hadoop YARN:** This is a framework for job scheduling and cluster resourcemanagement.

**How Does Hadoop Work?**

It is quite expensive to build bigger servers with heavy configurations that handle large scale processing, but as an alternative, you can tie together many commodity computers with single-CPU, as a single functional distributed system and practically, the clustered machines can read the dataset in parallel and provide a much higher throughput. Moreover, it is cheaper than one high-end server. So this is the first motivational factor behind using Hadoop that it runs across clustered and low-cost machines.

Hadoop runs code across a cluster of computers. This process includes the following core tasks that Hadoop performs:

* Data is initially divided into directories and files. Files are divided into uniform sized blocks of 128M and 64M (preferably 128M).
* These files are then distributed across various cluster nodes for further processing.
* HDFS, being on top of the local file system, supervises the processing.
* Blocks are replicated for handling hardware failure.
* Checking that the code was executed successfully.
* Performing the sort that takes place between the map and reduce stages.
* Sending the sorted data to a certain computer.
* Writing the debugging logs for each job.

**Advantages of Hadoop**

* Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatic distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.
* Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.
* Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
* Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

**What is Hive?**

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

**Hive is not**

 A relational database

 A design for OnLine Transaction Processing (OLTP)

 A language for real-time queries and row-level updates

**Features of Hive**

Here are the features of Hive:

 It stores schema in a database and processed data into HDFS.

 It is designed for OLAP.

 It provides SQL type language for querying called HiveQL or HQL.

 It is familiar, fast, scalable, and extensible.

**What is Apache Pig?**

Apache Pig is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows. Pig is generally used with Hadoop; we can perform all the data manipulation operations in Hadoop using Apache Pig.

To write data analysis programs, Pig provides a high-level language known as Pig Latin. This language provides various operators using which programmers can develop their own functions for reading, writing, and processing data.

To analyze data using Apache Pig, programmers need to write scripts using Pig Latin language. All these scripts are internally converted to Map and Reduce tasks. Apache Pig has a component known as Pig Engine that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs.

**Features of Pig Apache**

 **Rich set of operators:** It provides many operators to perform operations like join, sort, filer, etc.

** Ease of programming**: Pig Latin is similar to SQL and it is easy to write a Pig script if you are good at SQL.

**Optimization opportunities:** The tasks in Apache Pig optimize their execution automatically, so the programmers need to focus only on semantics of the language.

 **Extensibility:** Using the existing operators, users can develop their own functions to read, process, and write data.

** UDF’s**: Pig provides the facility to create User-defined Functions in other programming languages such as Java and invoke or embed them in Pig Scripts.

 **Handles all kinds of data**: Apache Pig analyzes all kinds of data, both structured as well as unstructured. It stores the results in HDFS.

**Project Overview**

**Assumptions:**

1. Hadoop Cluster is running.

2. Hadoop Ecosystem tools are installed.

3. H1B VISA data available in HDFS.

**Tools and Framework Used:**

1. Java

2. MySQL

3. Hadoop Distributed File Systems (HDFS).

4. Yet Another Recourse Negotiator (YARN) architecture.

5. Apache Pig.

6. Apache Hive.

7. Apache Sqoop.

**Project Implementation**

**Uploading The Data**

**HIVE**

1. **Creating Table In Hive Using The Delimiters**

CREATE TABLE h1b(s\_no int,case\_status string,employer\_name string, soc\_name string, job\_title string, full\_time\_position string,prevailing\_wage int,year string, worksite string, longitute double, latitute double )

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES

("separatorChar" = ",",

"quoteChar" = "\"")

STORED AS TEXTFILE

TBLPROPERTIES('skip.header.line.count' = '1');-------**For removing the header**

1. **Loading the Data**

LOAD DATA LOCAL INPATH '<Path of the file>' OVERWRITE INTO TABLE <Table name>;

**Uploading the Data Into HDFS**

1. **Creating another table in Hive and loading the data from main table to second table using the command**

CREATE TABLE h1b1(s\_no int, case\_status string,employer\_name string, soc\_name string, job\_title string, full\_time\_position string,prevailing\_wage int,year string, worksite string, longitute double, latitute double )

ROW FORMAT DELIMITED FIELDS terminated by ‘\t’

NOTE:- ‘\t’ represents TAB SPACE

1. **Changing the data to TAB SPACE separation**

Command – insert overwrite table <secondtablename> select \* from <firsttablename>

1. **Saving the Tab Space separated data into local**

Insert overwrite local directory <pathwherethedatashouldbesaved > row format delimited fields terminated by ‘\t’ select \* from <secondtablename>

1. **Uploading the data in HDFS**

Command :-Hadoop fs –put <pathofthefileinthelocalsystem> <HDFSpath>

**PIG**

H1b = load ‘path’ using PigStorage(‘\t’) as (s\_no,casestatus,employer\_name,soc\_name, job\_title, full\_time\_position,prevailing\_wage,yearg,worksite, longitute, latitute)

**UseCase 1a:**

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

**Hive:**

Select count(\*) as kill,year from h1b where job\_title="DATA ENGINEER" group by job\_title,year order by kill ;

**PIG**

filt = filter h1\_b by job\_title == 'DATA ENGINEER';

dis = foreach filt generate soc\_name,year;

grupby = group dis by $1;

cunt = foreach grupby generate $0,COUNT(dis.soc\_name);

**MapReduce**

hadoop jar Que1a.jar Que1a /project/h1\_b /outputTask1a

hadoop fs –cat /outputQue1a/\*

**SAMPLE OUTPUT**

|  |  |  |
| --- | --- | --- |
| **HIVE** | **MapReduce** | **PIG** |
| 18 2011  32 2012  41 2013  89 2014  160 2015  251 2016 | 2011 18  2012 32  2013 41  2014 89  2015 160  2016 251 | (2011,18)  (2012,32)  (2013,41)  (2014,89)  (2015,160)  (2016,251) |

**Usecase 1b:-**

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

**Hive**

Select count(\*) as count1,job\_title from h1b group by job\_title order by count1 desc limit 5;

**Pig**

first = foreach h1b generate job\_title;

second = group first by $0;

third = foreach second generate $0,COUNT(first.$0);

fourth = order third by $1 desc;

fifth = limit fourth 5;

**Usecase2a:-**

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

**Hive**

select war4,m.kill,m.year from(select max(thop) as war4,b.year as req from (Select count(\*) as thop,year,worksite as kill from h1b where job\_title="DATA ENGINEER"group by year,worksite) b group by b.year)n join (select count(\*) as thop1,year,worksite as kill from h1b where job\_title="DATA ENGINEER" group by year,worksite)m on n.req=m.year and n.war4=m.thop1 order by year;

**Pig**

dis =foreach h1b generate worksite,job\_title,year;

fil2011 = FILTER dis by job\_title == 'DATA ENGINEER';

grp2011 = group fil2011 by (worksite,year);

cunt2011 = foreach grp2011 generate group,COUNT(fil2011.$0) as mostData;

odr2011 = ORDER cunt2011 by $1 desc;

lim2011 = LIMIT odr2011 1;

dis =foreach h1b generate worksite,job\_title,year;

fil2012 = FILTER dis by job\_title == 'DATA ENGINEER';

grp2012 = group fil2012 by (worksite,year);

cunt2012 = foreach grp2012 generate group,COUNT(fil2012.$0) as mostData;

odr2012 = ORDER cunt2012 by $1 desc;

lim2012 = LIMIT odr2012 1;

dis =foreach h1b generate worksite,job\_title,year;

fil2013 = FILTER dis by job\_title == 'DATA ENGINEER';

grp2013 = group fil2013 by (worksite,year);

cunt2013 = foreach grp2013 generate group,COUNT(fil2013.$0) as mostData;

odr2013 = ORDER cunt2013 by $1 desc;

lim2013 = LIMIT odr2013 1;

dis =foreach h1b generate worksite,job\_title,year;

fil2014 = FILTER dis by job\_title == 'DATA ENGINEER';

grp2014 = group fil2014 by (worksite,year);

cunt2014 = foreach grp2014 generate group,COUNT(fil2014.$0) as mostData;

odr2014 = ORDER cunt2014 by $1 desc;

lim2014 = LIMIT odr2014 1;

dis =foreach h1b generate worksite,job\_title,year;

fil2015 = FILTER dis by job\_title == 'DATA ENGINEER';

grp2015 = group fil2015 by (worksite,year);

cunt2015 = foreach grp2015 generate group,COUNT(fil2015.$0) as mostData;

odr2015 = ORDER cunt2015 by $1 desc;

lim2015 = LIMIT odr2015 1;

dis =foreach h1b generate worksite,job\_title,year;

fil2016 = FILTER dis by job\_title == 'DATA ENGINEER';

grp2016 = group fil2016 by (worksite,year);

cunt2016 = foreach grp2016 generate group,COUNT(fil2016.$0) as mostData;

odr2016 = ORDER cunt2016 by $1 desc;

lim2016 = LIMIT odr2016 1;

unionALL = union lim2011,lim2012,lim2013,lim2014,lim2015,lim2016;

**Usecase2b**

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

**Hive**

Select \* from( Select rank() over (partition by year order by kill desc) as thop,worksite,year from (Select count(\*) as kill,worksite,year from h1b where case\_status="CERTIFIED" or case\_status="CERTIFIED-WITHDRAWN" group by worksite,year ) a ) n where thop<6;

**Pig**

dis =foreach h1b generate worksite,case\_status,year;

fil2011 = FILTER dis by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN';

grp2011 = group fil2011 by (worksite,year);

cunt2011 = foreach grp2011 generate group,COUNT(fil2011.$0) as mostData;

odr2011 = ORDER cunt2011 by $1 desc;

lim2011 = LIMIT odr2011 1;

dis =foreach h1b generate worksite,case\_status,year;

fil2012 = FILTER dis by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN';

grp2012 = group fil2012 by (worksite,year);

cunt2012 = foreach grp2012 generate group,COUNT(fil2012.$0) as mostData;

odr2012 = ORDER cunt2012 by $1 desc;

lim2012 = LIMIT odr2012 5;

dis =foreach h1b generate worksite,case\_status,year;

fil2013 = FILTER dis by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN';

grp2013 = group fil2013 by (worksite,year);

cunt2013 = foreach grp2013 generate group,COUNT(fil2013.$0) as mostData;

odr2013 = ORDER cunt2013 by $1 desc;

lim2013 = LIMIT odr2013 5;

dis =foreach h1b generate worksite,case\_status,year;

fil2014 = FILTER dis by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN';

grp2014 = group fil2014 by (worksite,year);

cunt2014 = foreach grp2014 generate group,COUNT(fil2014.$0) as mostData;

odr2014 = ORDER cunt2014 by $1 desc;

lim2014 = LIMIT odr2014 5;

dis =foreach h1b generate worksite,case\_status,year;

fil2015 = FILTER dis by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN';

grp2015 = group fil2015 by (worksite,year);

cunt2015 = foreach grp2015 generate group,COUNT(fil2015.$0) as mostData;

odr2015 = ORDER cunt2015 by $1 desc;

lim2015 = LIMIT odr2015 5;

dis =foreach h1b generate worksite,case\_status,year;

fil2016 = FILTER dis by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN';

grp2016 = group fil2016 by (worksite,year);

cunt2016 = foreach grp2016 generate group,COUNT(fil2016.$0) as mostData;

odr2016 = ORDER cunt2016 by $1 desc;

lim2016 = LIMIT odr2016 5;

**UseCase3:**

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

**Hive**

Select soc\_name,count(\*) as war1 from h1b where job\_title="DATA SCIENTIST" group by soc\_name order by war1 desc limit 1 ;

**Pig**

filterds = Filter h1b by job\_title == 'DATA SCIENTIST';

neededdata = FOREACH filterds GENERATE $3,job\_title;

finaldata = group neededdata by $0;

countofscient = FOREACH finaldata GENERATE $0,COUNT(neededdata.job\_title);

odr = order countofscient by $1 desc;

dump odr;

**UseCase4**

**Which top 5 employers file the most petitions each year?**

**Hive**

select thop,employer\_name,year from(select rank() over (partition by year order by war1 desc) as thop,employer\_name,year from(Select count(\*) as war1,employer\_name,year from h1b group by employer\_name,year) a)b where thop<6 and year is not null;

**Pig**

dis =foreach h1\_b generate employer\_name,year;

fil2011 = FILTER dis by year == '2011';

grp2011 = group fil2011 by (employer\_name,year);

cunt2011 = foreach grp2011 generate group,COUNT(fil2011.$0) as petin;

odr2011 = ORDER cunt2011 by $1 desc;

lim2011 = LIMIT odr2011 5;

dis =foreach h1\_b generate employer\_name,year;

fil2012 = FILTER dis by year == '2012';

grp2012 = group fil2012 by (employer\_name,year);

cunt2012 = foreach grp2012 generate group,COUNT(fil2012.$0) as petin;

odr2012 = ORDER cunt2012 by $1 desc;

lim2012 = LIMIT odr2012 5;

dis =foreach h1\_b generate employer\_name,year;

fil2013 = FILTER dis by year == '2013';

grp2013 = group fil2013 by (employer\_name,year);

cunt2013 = foreach grp2013 generate group,COUNT(fil2013.$0) as petin;

odr2013 = ORDER cunt2013 by $1 desc;

lim2013 = LIMIT odr2013 5;

dis =foreach h1\_b generate employer\_name,year;

fil2014 = FILTER dis by year == '2014';

grp2014 = group fil2014 by (employer\_name,year);

cunt2014 = foreach grp2014 generate group,COUNT(fil2014.$0) as petin;

odr2014 = ORDER cunt2014 by $1 desc;

lim2014 = LIMIT odr2014 5;

dis =foreach h1\_b generate employer\_name,year;

fil2015 = FILTER dis by year == '2015';

grp2015 = group fil2015 by (employer\_name,year);

cunt2015 = foreach grp2015 generate group,COUNT(fil2015.$0) as petin;

odr2015 = ORDER cunt2015 by $1 desc;

lim2015 = LIMIT odr2015 5;

dis =foreach h1\_b generate employer\_name,year;

fil2016 = FILTER dis by year == '2016';

grp2016 = group fil2016 by (employer\_name,year);

cunt2016 = foreach grp2016 generate group,COUNT(fil2016.$0) as petin;

odr2016 = ORDER cunt2016 by $1 desc;

lim2016 = LIMIT odr2016 5;

unionall = union lim2011,lim2012,lim2013,lim2014,lim2015,lim2016;

dump unionall;

**UseCase5**

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

**Hive**

select \* from(Select rank() over (partition by year order by war desc) as thop,job\_title,year from (Select count(\*) as war,year,job\_title from h1b group by job\_title,year ) a)b where b.thop<11 and year is not null;

**Pig**

dis =foreach h1b generate job\_title,year;

fil2011 = FILTER dis by year == '2011';

grp2011 = group fil2011 by (job\_title,year);

cunt2011 = foreach grp2011 generate group,COUNT(fil2011.$0) as petin;

odr2011 = ORDER cunt2011 by $1 desc;

lim2011 = LIMIT odr2011 10;

fil2012 = FILTER dis by year == '2012';

grp2012 = group fil2012 by (job\_title,year);

cunt2012 = foreach grp2012 generate group,COUNT(fil2012.$0) as petin;

odr2012 = ORDER cunt2012 by $1 desc;

lim2012 = LIMIT odr2012 10;

fil2013 = FILTER dis by year == '2013';

grp2013 = group fil2013 by (job\_title,year);

cunt2013 = foreach grp2013 generate group,COUNT(fil2013.$0) as petin;

odr2013 = ORDER cunt2013 by $1 desc;

lim2013 = LIMIT odr2013 10;

fil2014 = FILTER dis by year == '2014';

grp2014 = group fil2014 by (job\_title,year);

cunt2014 = foreach grp2014 generate group,COUNT(fil2014.$0) as petin;

odr2014 = ORDER cunt2014 by $1 desc;

lim2014 = LIMIT odr2014 10;

fil2015 = FILTER dis by year == '2015';

grp2015 = group fil2015 by (job\_title,year);

cunt2015 = foreach grp2015 generate group,COUNT(fil2015.$0) as petin;

odr2015 = ORDER cunt2015 by $1 desc;

lim2015 = LIMIT odr2015 10;

fil2016 = FILTER dis by year == '2016';

grp2016 = group fil2016 by (job\_title,year);

cunt2016 = foreach grp2016 generate group,COUNT(fil2016.$0) as petin;

odr2016 = ORDER cunt2016 by $1 desc;

lim2016 = LIMIT odr2016 10;

unionall = union lim2011,lim2012,lim2013,lim2014,lim2015,lim2016;

dump unionall;

**UseCase6**

**Find the percentage and the count of each case status on total applications for each year. Create a graph depicting the pattern of All the cases over the period of time.**

**Hive**

Select case\_status,year,(maharshi\*100)/a.thop from (Select count(\*) as maharshi ,case\_status,year from h1b group by case\_status,year) b,(select count(\*) as thop from h1b) a order by case\_status;

**Pig**

totalrecgrup = group h1b ALL;

totalrec = foreach totalrecgrup generate COUNT(h1\_b.soc\_name)as totalApplications;

--------------------------------------------------------------------------------

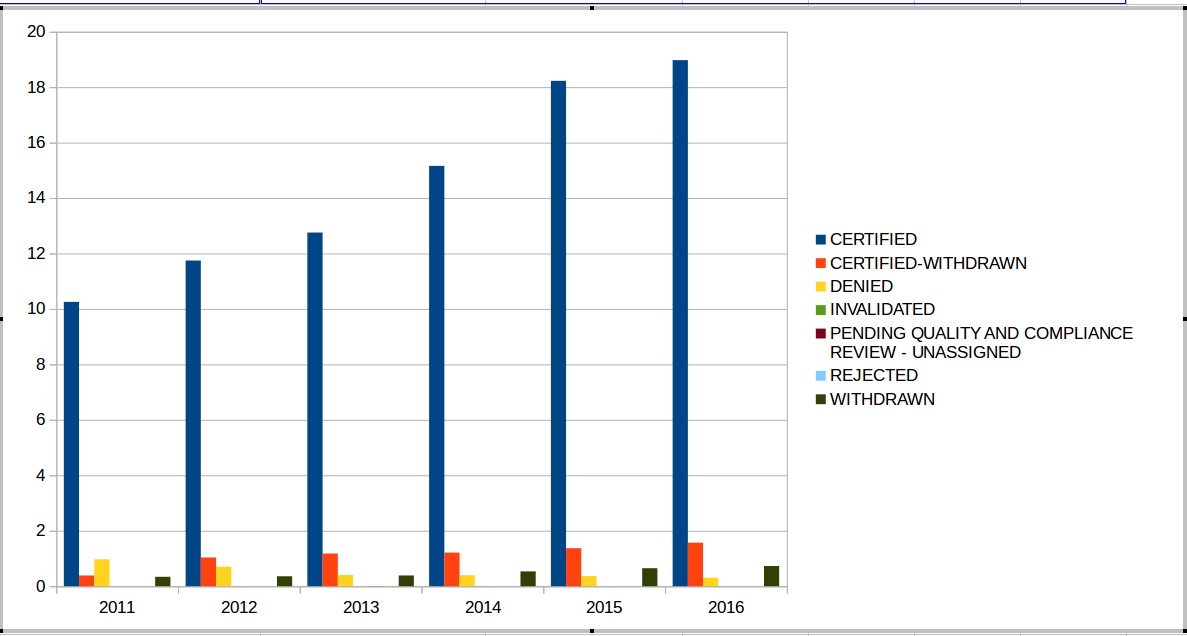
dis = foreach h1b generate year,case\_status;

grp = GROUP dis by year;

cunt = foreach grp generate $0,COUNT($1) as case\_application;

perc = foreach cunt generate $0,ROUND\_TO(((case\_application/(double)totalrec.totalApplications)\*100),2);

odr = order perc by $1 desc;



**UseCase7**

**Create a bar graph to depict the number of applications for each year**

**Hive**

select count(\*),year from h1b group by year order by year;

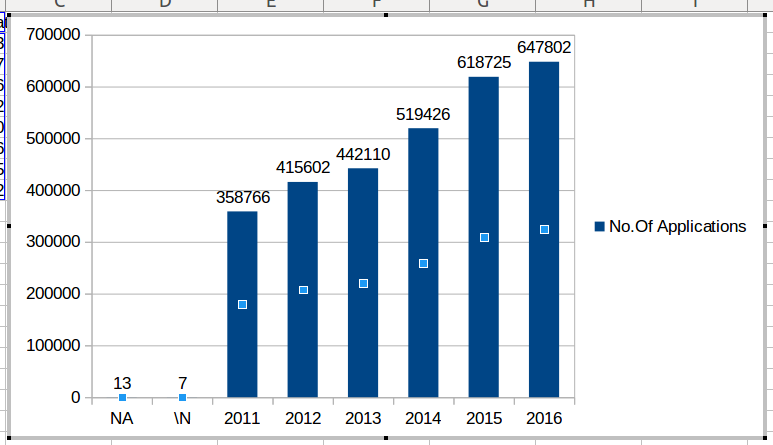
**Pig**

dis = foreach h1b generate year,soc\_name;

groupby = group dis by year;

cunt = foreach groupby generate $0,COUNT(dis.soc\_name);

dump cunt;



**UseCase8**

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

**Hive**

select avg(prevailing\_wage),job\_title,year from h1b group by year,job\_title,FULL\_TIME\_POSITION order by job\_title;

**Pig**

fil = filter h1b by full\_time\_position == 'Y';

tme\_Y = group fil by (job\_title,year);

sum\_Y = foreach tme\_Y generate group,AVG(fil.prevailing\_wage) as AVG\_Y;

odr\_Y = order sum\_Y by $0 desc;

lim1 = limit sum\_Y 1;

fil = filter h1b by full\_time\_position == 'N';

tme\_Y1 = group fil by (job\_title,year);

sum\_N = foreach tme\_N generate group,AVG(fil.prevailing\_wage) as AVG\_N;

odr\_N = order sum\_N by $0 desc;

lim2 = limit odr\_N 1;

**UseCase9**

**Which are top ten employers who have the highest success rate in petitions?**

**Hive**

Select employer\_name,(kill\*100)/kill1 as top10 from (select count(\*) as kill1,1 as tojoin from h1b)b join (Select count(\*) as kill,1 as tojoin,employer\_name from h1b where case\_status="CERTIFIED" or case\_status="CERTIFIED WITHDRAWN" group by employer\_name) a on a.tojoin=b.tojoin order by top10 limit 10;

**Pig**

totalrecgrup = group h1b ALL;

totalrec = foreach totalrecgrup generate COUNT(h1\_b.soc\_name)as totalApplications;

------------------------------------

rec = foreach h1\_b generate employer\_name,case\_status;

fil\_suc = FILTER rec by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN';

rec1 = foreach fil\_suc generate employer\_name,case\_status;

groupby = group rec1 by employer\_name;

coutofEmplye = foreach groupby generate $0,COUNT(rec1.case\_status) as cerftifiedApplications;

perc = foreach coutofEmplye generate $0,(cerftifiedApplications/(double)totalrec.totalApplications)\*100;

odr = order perc by $1 desc;

lim = LIMIT odr 10;

**UseCase10**

**Which are the top 10 job positions which have the highest success rate in petitions?**

**Hive**

Select job\_title,(thop\*100)/kill1 as top10 from (Select count(\*) as thop,job\_title from h1b where case\_status=” CERTIFIED “ or case\_status=”CERTIFIED WITHDRAWN” group by job\_title )b,(Select count(\*) as kill1 from h1b) a order by top10 limit 10;

**Pig**

for total applications

-----------------------

totalrecgrup = group h1\_b ALL;

totalrec = foreach totalrecgrup generate COUNT(h1\_b.soc\_name)as totalApplications;

------------------------------------

rec = foreach h1\_b generate job\_title,case\_status;

fil\_suc = FILTER rec by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN';

rec1 = foreach fil\_suc generate job\_title,case\_status;

groupby = group rec1 by job\_title;

coutofEmplye = foreach groupby generate $0,COUNT(rec1.case\_status) as cerftifiedApplications;

d = foreach coutofEmplye generate $0,(cerftifiedApplications/(double)totalrec.totalApplications)\*100;

odr = order d by $1 desc;

lim = LIMIT odr 10;

**UseCase11**

**Export Command**

sqoop export -m 1 --connect jdbc:mysql://localhost/h1\_b --username root --password '' --table H1\_B1 --export-dir /user/hive/warehouse/project.db/h1\_b\_cpy2/\*

**Import command**

sqoop import --connect jdbc:mysql://localhost/adventureworks --username root --password '' --table vendor --hive-import --hive-table niit.sam\_vendor1 -m 1;