
BIG
DATA
ANALYTICS
SOFTWARE
SIZE
INTERNET
TECHNOLOGIES
MANAGEMENT
LARGE
EVERY
STORAGE
COMPLEX
DATABASES
SETS
SHARED
INCLUDE
SYSTEMS
NETWORKS
INFORMATION
SEARCH

TABLE OF CONTENTS

BIG DATA

1.1 What is Big Data?

Big data is a term that describes the large volume of data – both structured and unstructured – that inundates 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. Big data can be analyzed for insights that lead to better decisions and strategic business moves. Big data is changing the way people within organizations work together. It is creating a culture in which business and IT leaders must join forces to realize value from all data. Insights from big data can enable all employees to make better decisions—deepening customer engagement, optimizing operations, preventing threats and fraud, and capitalizing on new sources of revenue. But escalating demand for insights requires a fundamentally new approach to architecture, tools and practices.

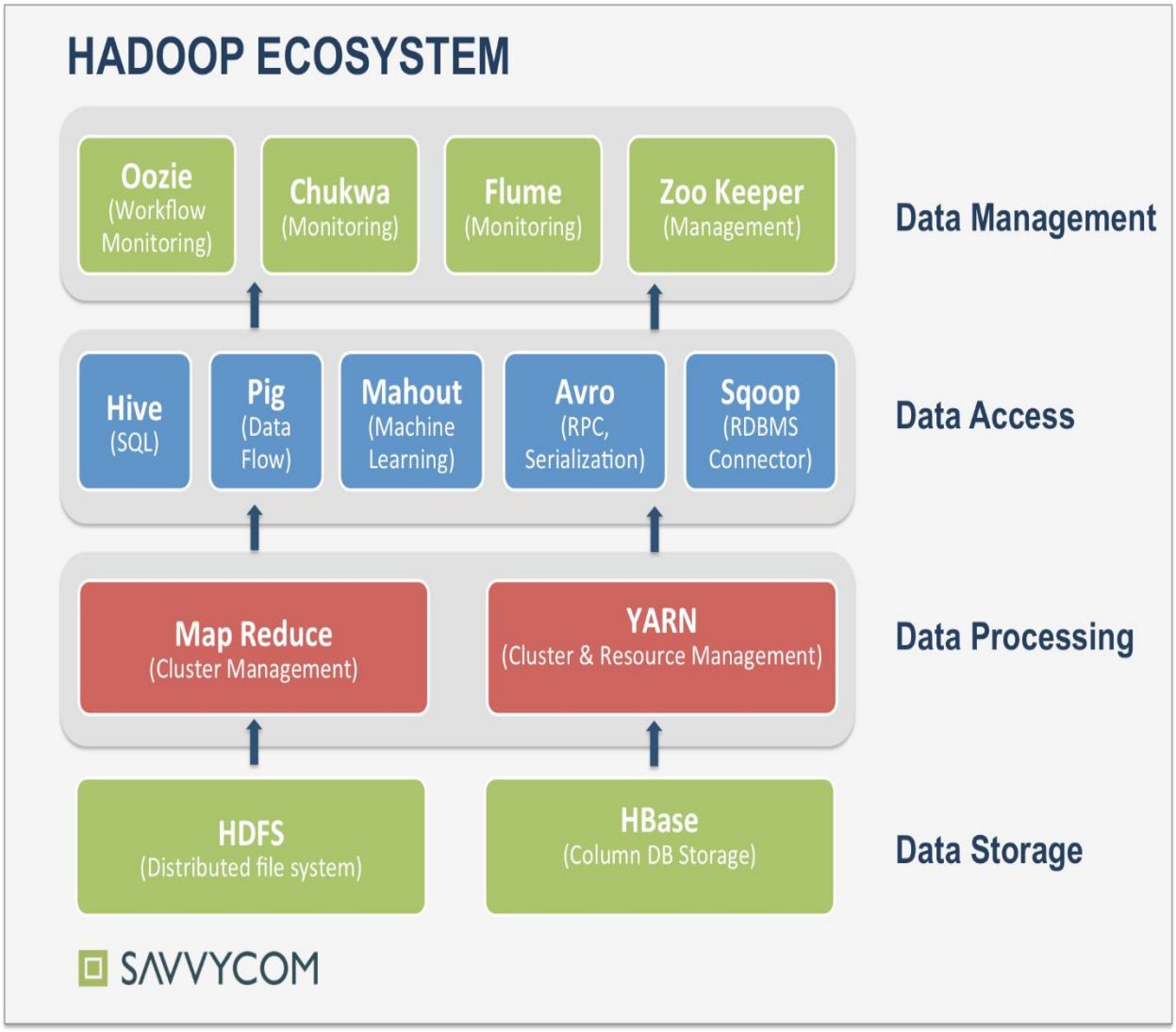
1.2 Background

We uses various scenarios in Big Data to provide desired outcomes.

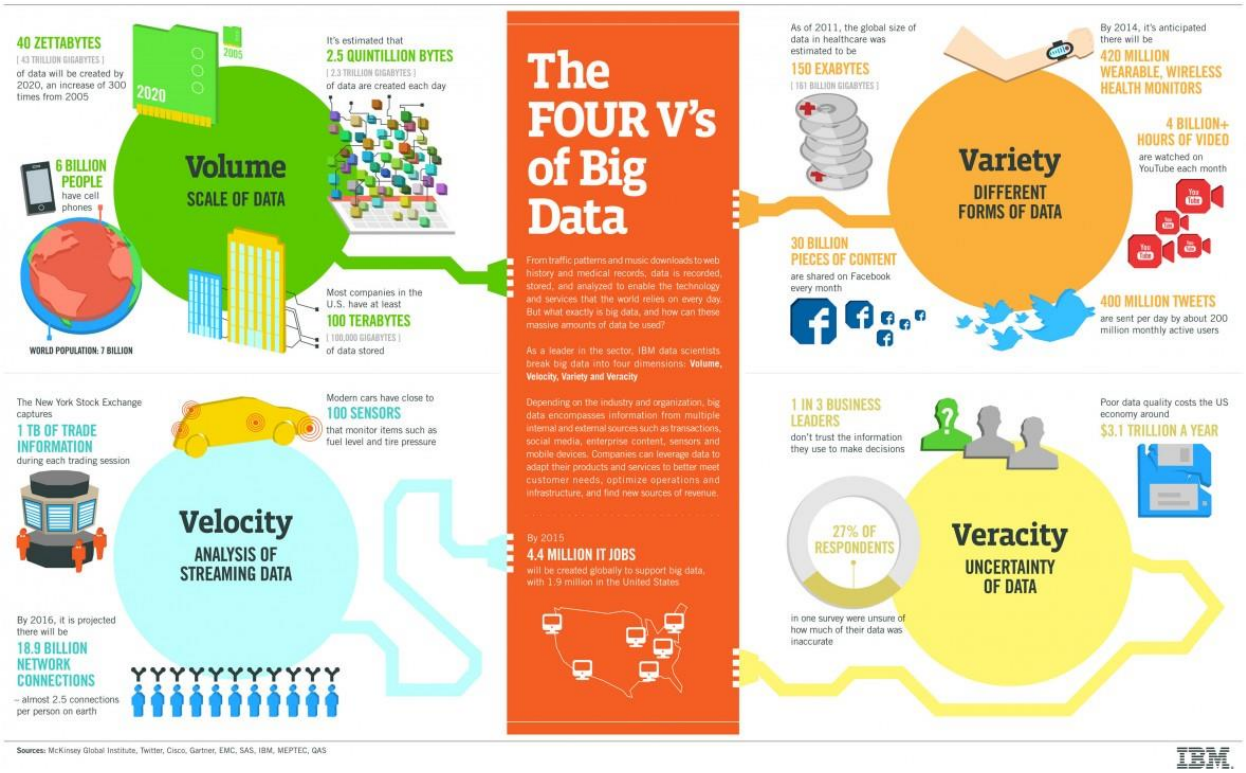
1.2.1 Big Data Challenges:

Challenges include capture, storage, analysis, data curation, search, sharing, transfer, visualization, querying, updating and information privacy.

1.3 Hadoop Ecosystem



1.4 Characteristics of Big Data



Big data can be described by the following characteristics:

Volume

The quantity of generated and stored data. The size of the data determines the value and potential insight- and whether it can actually be considered big data or not.

Variety

The type and nature of the data. This helps people who analyze it to effectively use the resulting insight.

Velocity

Velocity is the frequency of incoming data that needs to be processed. Ex, I n YouTube a movie teaser gets released within 5 minutes millions of likes and comments are posted.

Veracity

Veracity refers to the trustworthiness of the data.

1.5 Applications

Big data has increased the demand of information management specialists so much so that SoftwareAG, OracleCorporation, IBM, Microsoft, SAP, EMC, HP and Dell have spent more than $15 billion on software firms specializing in data management and analytics. In 2010, this industry was worth more than $100 billion and was growing at almost 10 percent a year: about twice as fast as the software business as a whole.

1.6 Advantages of Hadoop

1.6.1 Scalable

Hadoop is a highly scalable storage platform, because it can store and distribute very large data sets across hundreds of inexpensive servers that operate in parallel. Unlike traditional relational database systems (RDBMS) that can't scale to process large amounts of data, Hadoop enables businesses to run applications on thousands of nodes involving thousands of terabytes of data.

1.6.2 Cost Effective

Hadoop also offers a cost effective storage solution for businesses' exploding data sets. The problem with traditional relational database management systems is that it is extremely cost prohibitive to scale to such a degree in order to process such massive volumes of data. Hadoop, on the other hand, is designed as a scale-out architecture that can affordably store all of a company's data for later use. The cost savings are staggering: instead of costing thousands to tens

of thousands of pounds per terabyte, Hadoop offers computing and storage capabilities for hundreds of pounds per terabyte.

1.6.3 Flexible

Hadoop enables businesses to easily access new data sources and tap into different types of data (both structured and unstructured) to generate value from that data. This means businesses can use Hadoop to derive valuable business insights from data sources such as social media, email conversations or clickstream data. In addition, Hadoop can be used for a wide variety of purposes, such as log processing, recommendation systems, data warehousing, and market campaign analysis and fraud detection.

1.6.4 Fast

Hadoop's unique storage method is based on a distributed file system that basically 'maps' data wherever it is located on a cluster. The tools for data processing are often on the same servers

where the data is located, resulting in much faster data processing. If you're dealing with large volumes of unstructured data, Hadoop is able to efficiently process terabytes of data in just minutes, and petabytes in hours.

1.6.5 Resilient To Failure

A key advantage of using Hadoop is its fault tolerance. When data is sent to an individual node, that data is also replicated to other nodes in the cluster, which means that in the event of failure, there is another copy available for use.
The MapR distribution goes beyond that by eliminating the NameNode and replacing it with a distributed No NameNode architecture that provides true high availability. Our architecture provides protection from both single and multiple failures.

TOOLS

2.1 Hadoop Frame Work

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

The project includes these modules:

Hadoop Common: The common utilities that support the other Hadoop modules.

Hadoop Distributed File System (HDFS™): A distributed file system that provides high-throughput access to application data.

Hadoop YARN: A framework for job scheduling and cluster resource management.

Hadoop Map Reduce: A YARN-based system for parallel processing of large data sets.

Benefits

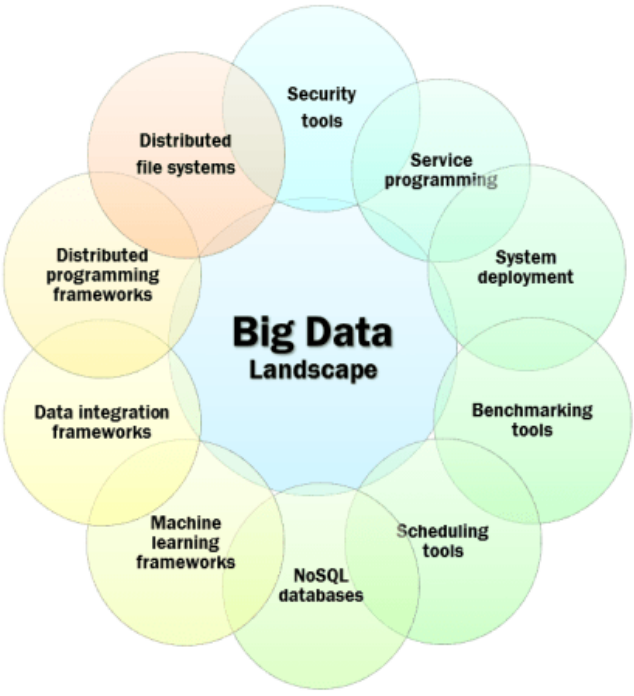
Some of the reasons organizations use Hadoop is its' ability to store, manage and analyse vast amounts of structured and unstructured data quickly, reliably, flexibly and at low-cost.

Scalability and Performance – distributed processing of data local to each node in a cluster enables Hadoop to store, manage, process and analyse data at petabyte scale.

Reliability – large computing clusters are prone to failure of individual nodes in the cluster. Hadoop is fundamentally resilient – when a node fails processing is re-directed to the remaining nodes in the cluster and data is automatically re-replicated in preparation for future node failures.

Flexibility – unlike traditional relational database management systems, you don't have to create structured schemas before storing data. You can store data in any format, including semi-structured or unstructured formats, and then parse and apply schema to the data when read.

Low Cost – unlike proprietary software, Hadoop is open source and runs on low-cost commodity hardware.



2.2 Hive

The Apache Hive data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. Structure can be projected onto data already in storage. A command line tool and JDBC driver are provided to connect users to Hive.

Benefits

Time-It take very less time to write Hive Query compared to Map Reduce code. For example, the word count problem which takes around 50 lines of code can be written in 5 lines in Hive. So, you save time.

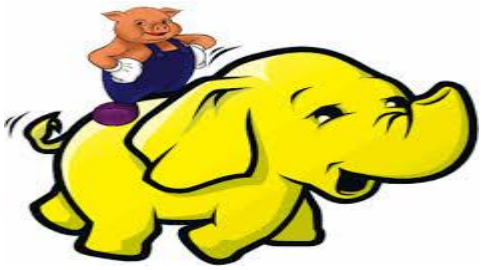
Easy-It is very easy to write query involving joins (if there are few joins) in Hive.

Maintenance-It has very low maintenance and is very simple to learn & use (low learning curve).

2.3 Pig

Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable

to substantial parallelization, which in turns enables them to handle very large data sets.



Benefits

Ease of programming. It is trivial to achieve parallel execution of simple, "embarrassingly parallel" data analysis tasks

Optimization opportunities. The way in which tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.

Extensibility. Users can create their own functions to do special-purpose processing.

PROJECT DEFINITION

3.1 Project Objective

Here our current project fully based on H1B visa employer data. This the most common visa status applied for and held by international students once they complete college/higher studies and those who are work in full-time or part-time. Our objective is to perform analysis on the H1B visa applicants between the years 2011-2016.

3.2 Project Scope

Our H1B visa applicant's scope is to provide clear outcomes for the clients as per their requirements.

3.3 Output for Project

Our project outputs comes with clear scenarios like different use cases, conditions and filtration that has applied on each phases. So it should be clear vision about what client expected.

3.4 Project Budget

Since we use commodity hardware for the analysis of data it is more cost effective, scalable, flexible and fast.

3.5 Project Duration

Since Hadoop is more faster to execute ,it will execute the entire large dataset in seconds .Time required to complete the project is very less.

3.6 Project Fields

The dataset description is as follows:
The columns in the dataset include:

- **CASE_STATUS**: Status associated with the last significant event or decision. Valid values include "Certified," "Certified-Withdrawn," Denied," and "Withdrawn".

  Certified: Employer filed the LCA, which was approved by DOL

Certified Withdrawn: LCA was approved but later withdrawn by employer

Withdrawn: LCA was withdrawn by employer before approval

Denied: LCA was denied by DOL

- **EMPLOYER_NAME:** Name of employer submitting labor condition application.
- **SOC_NAME**: the Occupational name associated with the SOC_CODE. SOC_CODE is the occupational code associated with the job being requested for temporary labour condition, as classified by the Standard Occupational Classification (SOC) System.

- **JOB_TITLE**: Title of the job
- **FULL_TIME_POSITION**: Y = Full Time Position; N = Part Time Position
- **PREVAILING_WAGE:** Prevailing Wage for the job being requested for temporary labor condition. The wage is listed at annual scale in USD. 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.
- **YEAR:** Year in which the H1B visa petition was filed
- **WORKSITE:** City and State information of the foreign worker's intended area of employment
- **lon:** longitude of the Worksite
- **lat:** latitude of the Worksite

3.7 Project Use Cases

We will be performing analysis on the H1B visa applicants between the years 2011-2015. After analyzing the data, we can derive the following facts.

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.

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.

3)Which industry has the most number of Data Scientist positions?

4)Which top 5 employers file the most petitions each year?

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

6) 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.

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

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

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

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

11) Export result for question no 10 to MySql database.

SUCCESS RATE % = (Certified + Certified Withdrawn)/Total x 100

The dataset has nearly 3 million records.

PROJECT THROUGH MAPREDUCE

CASE 1:

6) 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.

EXECUTION CODE:

```
package case_status_count;

import java.io.IOException;
import java.util.Map.Entry;
import java.util.Spliterator;
import java.util.function.Consumer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
```

```
public  class Case_status extends Configured implements Tool {

    public static class Map extends Mapper<LongWritable,Text,Text,Text>
    {
        public void map(LongWritable k,Text v,Context c)throws
IOException,InterruptedException
        {
            String s[]=v.toString().split("\t");
            String case_status=s[1];
            String year=s[7];
            String data=case_status+";"+year;
            c.write(new Text(case_status),new Text(data));

        }
    }

    public static class Part extends Partitioner<Text,Text>
    {
        @Override
        public int getPartition(Text key, Text value, int numReduceTask) {

            String s[]=value.toString().split(";");
            String year=s[1];
            if(year == "2016")
            {
                return 0;
            }
            else if(year == "2015")
            {
                return 1;
            }
            else if(year == "2014")
            {
                return 2;
            }
            else if(year == "2013")
            {
                return 3;
            }
            return 0;
```

```
        }

    }

    public static class Red extends Reducer<Text,Text, Text,Text>
    {
        public void reduce(Text key,Iterable<Text> value,Context c)throws
IOException,InterruptedException
        {
            String str[]=value.toString().split(";");
            String year=str[1];
            int count=0;String data="";
            int counta=0,countb=0,countc=0,countd=0;
            double pera=0.0,perb=0.0,perc=0.0,perd=0.0;
            for(String case_st:str)
            {

                count=count+1;
                if(case_st == "CERTIFIED")
                {
                    counta=counta+1;
                    pera=(counta/count)*100;
                     data=(year+";"+counta+";"+pera).toString();
                }
                else if(case_st == "CERTIFIED-WITHDRAWN")
                {
                    countb=countb+1;
                    perb=(countb/count)*100;
                     data=(year+";"+countb+";"+perb).toString();
                }
                else if(case_st == "WITHDRAWN")
                {
                    countc=countc+1;
                    perc=(countc/count)*100;
                     data=(year+";"+countc+";"+perc).toString();
                }
                else if(case_st == "DENIED")
                {
                    countd=countd+1;
                    perd=(countd/count)*100;
```

```
                        data=(year+";"+countd+";"+perd).toString();
                    }

                }
                c.write(new Text(key),new Text(data) );
            }
        }


        public static void main(String[] args)throws Exception {
            int res=ToolRunner.run(new Configuration(),new Case_status(), args);
            System.exit(0);
        }

        @Override
        public int run(String[] args) throws Exception {
            Configuration conf =new Configuration();
            Job j= new Job(conf,"percentage and count of each case_status/each year");
            j.setJarByClass(Case_status.class);
            j.setMapperClass(Map.class);
            j.setPartitionerClass(Part.class);
            j.setNumReduceTasks(4);
            j.setReducerClass(Red.class);
            j.setOutputKeyClass(Text.class);
            j.setOutputValueClass(Text.class);

            FileInputFormat.addInputPath(j,new Path(args[0]));
            FileOutputFormat.setOutputPath(j,new Path(args[1]));
            System.exit(j.waitForCompletion(true)?0:1);
            return 0;
        }

    }
```

CASE 2:

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

EXECUTION CODE:

```
package wages;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class Prevailing_wages extends Configured implements Tool {

    public static class Map extends Mapper<LongWritable,Text,Text,Text>
    {
            public void map(LongWritable k,Text v,Context c)throws
IOException,InterruptedException
            {
                    String s[]=v.toString().split("\t");
                    String job =s[4];
                    String time=s[5];
                    String wages=s[6];
                    String year=s[7];
                    String data =time+";"+wages+";"+year;
                    c.write(new Text(job), new Text(data));
            }
    }

    public static class Part extends Partitioner<Text, Text>
```

```
{
        @Override
        public int getPartition(Text key, Text value, int numReduceTask) {

                String str[]=value.toString().split(";");
                String year=str[2];
                if (year == "2016")
                {
                        return 0;

                }
                else if(year == "2015")
                {
                        return 1;
                }
                else if(year == "2014")
                {
                        return 2;
                }
                else if(year == "2013")
                {
                        return 3;
                }
                return 0;
        }
}

public static class Red extends Reducer<Text,Text,Text,Text>
{
        public void reduce(Text key,Iterable<Text> value,Context c)throws
IOException,InterruptedException
        {
                String s[]=value.toString().split(";");
        int sumy=0,county=0;
        int sumn=0,countn=0;
        double avgy=0.0,avgn=0.0;String data="";
        int wages = Integer.parseInt(s[1]);
                for(String amt:s)
                {
```

```
                if(s[0] == "y")
                {
                        sumy=sumy+wages;
                        county=county+1;
                        avgy=sumy/county;
                }
                else
                {
                        sumn=sumn+wages;
                        countn=countn+1;
                        avgn=sumn/countn;
                }

                }
                data =  (avgy +";"+ avgn).toString();
                c.write(new Text(key),new Text(data));

        }
}

public int run(String [] args)throws Exception
{
        Configuration conf = new Configuration();
        Job j = new Job(conf,"avg prevailing_wages for eachyear");
        j.setJarByClass(Prevailing_wages.class);
        j.setMapperClass(Map.class);
        j.setPartitionerClass(Part.class);
        j.setNumReduceTasks(4);
        j.setReducerClass(Red.class);
        j.setOutputKeyClass(Text.class);
        j.setOutputValueClass(Text.class);
        FileInputFormat.addInputPath(j,new Path(args[0]));
        FileOutputFormat.setOutputPath(j,new Path(args[1]));


        System.exit(j.waitForCompletion(true)?0:1);
        return 0;


}
```

```
    public static void main(String ar[]) throws Exception
      {
        int res = ToolRunner.run(new Configuration(), new
Prevailing_wages(),ar);
        System.exit(0);
      }

}
```

CASE 3:

9) Which are top ten employers who have the highest success rate in petitions?
EXECUTION CODE:

```
package sucess;

import java.io.IOException;
import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Sucess_rate {


   public static void main(String[] args) throws
IOException,InterruptedException,ClassNotFoundException{

   Configuration conf =new Configuration();
   Job j=new Job(conf,"top 10 highest sucess rate in petition");
   j.setJarByClass(Sucess_rate.class);
```

```
    j.setMapperClass(Map.class);
    j.setNumReduceTasks(1);
    j.setReducerClass(Red.class);
    j.setOutputKeyClass(Text.class);
    j.setOutputValueClass(DoubleWritable.class);

    FileInputFormat.addInputPath(j,new Path(args[0]));
    FileOutputFormat.setOutputPath(j,new Path (args[1]));

    System.exit(j.waitForCompletion(true)?0:1);

     }


     public class Map extends Mapper<LongWritable,Text,Text,Text>
     {
          public void map(LongWritable k,Text v,Context c)throws
IOException,InterruptedException
          {
               String s[]=v.toString().split("\t");
               String case_status=s[1];
               String emp_name=s[2];
               String data = case_status+";"+emp_name;
               c.write(new Text(emp_name),new Text(data) );
          }
     }

     public class Red extends Reducer<Text,Text,Text,DoubleWritable>
     {
          private TreeMap<Text,DoubleWritable> obj=new
TreeMap<Text,DoubleWritable>();
          public void reduce(Text k,Text v,Context c)throws
IOException,InterruptedException
          {
               String s[]=v.toString().split(";");
               String case_status=s[0];
               int count=0,countcer=0,countcw=0;
               double sucess_rate=0.0;
               for(String status:s)
               {
```

```
                    count =count+1;

                    if(case_status == "CERTIFIED")
                    {
                        countcer=countcer+1;
                    }
                    else if(case_status== "CERTIFIED-WITHDRAWN")
                    {
                        countcw=countcw+1;
                    }
                    sucess_rate=((countcer+countcw)/count)*100;

                }
                c.write(new Text(k),new DoubleWritable(sucess_rate));

                if(obj.size()>5)
                {
                    obj.remove(obj.firstKey());
                }

            }

            protected void cleanup(Context c)throws
    IOException,InterruptedException
            {
                for(DoubleWritable t:obj.descendingMap().values())
                {
                    c.write(new Text(), t);
                }
            }

            }
        }
```

CASE 4:

10) Which are the top 10 job positions which have the highest success rate in petitions?
EXECTION CODE:

```
package Sucess;

import java.io.IOException;
import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Sucess_rate {


    public static void main(String[] args) throws
IOException,InterruptedException,ClassNotFoundException{

    Configuration conf =new Configuration();
    Job j=new Job(conf,"top 10 highest sucess rate in petition based on jobtitle");
    j.setJarByClass(Sucess_rate.class);
    j.setMapperClass(Map.class);
    j.setNumReduceTasks(1);
    j.setReducerClass(Red.class);
    j.setOutputKeyClass(Text.class);
    j.setOutputValueClass(DoubleWritable.class);

    FileInputFormat.addInputPath(j,new Path(args[0]));
    FileOutputFormat.setOutputPath(j,new Path (args[1]));

    System.exit(j.waitForCompletion(true)?0:1);

     }


     public class Map extends Mapper<LongWritable,Text,Text,Text>
```

```
    {
        public void map(LongWritable k,Text v,Context c)throws IOException,InterruptedException
        {
            String s[]=v.toString().split("\t");
            String case_status=s[1];
            String job=s[4];
            String data = case_status+";"+job;
            c.write(new Text(job),new Text(data) );
        }
    }

    public class Red extends Reducer<Text,Text,Text,DoubleWritable>
    {
        private TreeMap<Text,DoubleWritable> obj=new TreeMap<Text,DoubleWritable>();
        public void reduce(Text k,Text v,Context c)throws IOException,InterruptedException
        {
            String s[]=v.toString().split(";");
            String case_status=s[0];
            int count=0,countcer=0,countcw=0;
            double sucess_rate=0.0;
            for(String status:s)
            {
                count =count+1;

                if(case_status == "CERTIFIED")
                {
                    countcer=countcer+1;
                }
                else if(case_status== "CERTIFIED-WITHDRAWN")
                {
                    countcw=countcw+1;
                }
                sucess_rate=((countcer+countcw)/count)*100;

            }
            c.write(new Text(k),new DoubleWritable(sucess_rate));
```

```
                if(obj.size()>5)
                {
                        obj.remove(obj.firstKey());
                }

        }

        protected void cleanup(Context c)throws
IOException,InterruptedException
        {
                for(DoubleWritable t:obj.descendingMap().values())
                {
                        c.write(new Text(), t);
                }
        }

        }
}
```

CASE 5:

11) Export result for question no 10 to MySql database.

```
Sqoop export  --connecct jdbc:mysql://localhost/cloudera
              --username root
              --password cloudera
              --table  success_rate
              --update –mode allowinsert
              --update-key id
              --export-dir /project_out/10q_output
              --input-fields-terminated-by ‘;’;
```

PROJECT THROUGH HIVE

CASE 1:

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

EXECUTION QUERY:

CASE 2:

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

CASE 3:

2.A)Which part of the US has the most Data Engineer jobs for each year?
EXCEUTION QUERY:
select worksite,count(job_title) as cnt,year from h1b_final where job_title == "DATA ENGINEER" and year == "2016" group by worksite,year order by cnt desc limit 5;

output:
MENLO PARK, CALIFORNIA 35 2016
NEW YORK, NEW YORK 34 2016
SAN FRANCISCO, CALIFORNIA 33 2016
SAN MATEO, CALIFORNIA 9 2016
CHICAGO, ILLINOIS 6 2016

-->select worksite,count(job_title) as cnt,year from h1b_final where job_title == "DATA ENGINEER" and year == "2015" group by worksite,year order by cnt desc limit 5;

SAN FRANCISCO, CALIFORNIA 33 2015
NEW YORK, NEW YORK 25 2015

```
MENLO PARK, CALIFORNIA          19      2015
SAN MATEO, CALIFORNIA   12      2015
SEATTLE, WASHINGTON     6       2015

-->select worksite,count(job_title) as cnt,year from h1b_final where job_title
== "DATA ENGINEER" and year == "2014" group by worksite,year order
by cnt desc limit 5;

MENLO PARK, CALIFORNIA          13      2014
SAN FRANCISCO, CALIFORNIA       12      2014
NEW YORK, NEW YORK      9       2014
MOUNTAIN VIEW, CALIFORNIA       5       2014
OMAHA, NEBRASKA   4       2014

-->select worksite,count(job_title) as cnt,year from h1b_final where job_title
== "DATA ENGINEER" and year == "2013" group by worksite,year order
by cnt desc limit 5;

MENLO PARK, CALIFORNIA          10      2013
SAN FRANCISCO, CALIFORNIA       5       2013
NEW YORK, NEW YORK      4       2013
DUBLIN, OHIO    2       2013
DETROIT, MICHIGAN       2       2013

-->select worksite,count(job_title) as cnt,year from h1b_final where job_title
== "DATA ENGINEER" and year == "2012" group by worksite,year order
by cnt desc limit 5;

SAN FRANCISCO, CALIFORNIA       7       2012
PONTIAC, MICHIGAN       3       2012
LOS ANGELES, CALIFORNIA         2       2012
LEXINGTON, MASSACHUSETTS        2       2012
SEATTLE, WASHINGTON     2       2012

-->select worksite,count(job_title) as cnt,year from h1b_final where job_title
== "DATA ENGINEER" and year == "2011" group by worksite,year order
by cnt desc limit 5;
```

SAN FRANCISCO, CALIFORNIA 3 2011
PLANO, TEXAS 2 2011
FARMINGTON HILLS, MICHIGAN 2 2011
ATLANTA, GEORGIA 1 2011
NORWALK, CONNECTICUT 1 2011

CASE 4:

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

EXCEUTION QUERY:

select worksite,count(case_status)as cnt,year from h1b_final where case_status == "CERTIFIED" and year == "2016" group by worksite,year order by cnt desc limit 10;

NEW YORK, NEW YORK 34639 2016
SAN FRANCISCO, CALIFORNIA 13836 2016
HOUSTON, TEXAS 13655 2016
ATLANTA, GEORGIA 11678 2016
CHICAGO, ILLINOIS 11064 2016
SAN JOSE, CALIFORNIA 9642 2016
IRVING, TEXAS 7286 2016
SUNNYVALE, CALIFORNIA 7227 2016
CHARLOTTE, NORTH CAROLINA 6954 2016
DALLAS, TEXAS 6501 2016

--> select worksite,count(case_status)as cnt,year from h1b_final where case_status == "CERTIFIED" and year == "2015" group by worksite,year order by cnt desc limit 10;

NEW YORK, NEW YORK	31266	2015
HOUSTON, TEXAS	15242	2015
SAN FRANCISCO, CALIFORNIA	12594	2015
ATLANTA, GEORGIA	10500	2015
SAN JOSE, CALIFORNIA	9589	2015
CHICAGO, ILLINOIS	9239	2015
SUNNYVALE, CALIFORNIA	7502	2015
CHARLOTTE, NORTH CAROLINA	6509	2015
IRVING, TEXAS	6120	2015
DALLAS, TEXAS	6039	2015

PROJECT THROUGH PIG

CASE 1:

3)Which industry has the most number of Data Scientist positions?

```
data = load '/home/cloudera/Desktop/hadoop_project/hadoop_data' using PigStorage('\t') AS
(id:int,case_status:chararray,employer_name:chararray,soc_name:chararray,job_title:chararray,full_time_position:chararray,prevailing_wages:int,year:chararray,worksite:chararray,longitude:double,latitude:double);
new = foreach data generate case_status,job_title,employer_name;
filterdata = filter new by case_status == 'CERTIFIED' AND job_title == 'DATA SCIENTIST';
groupdata = group filterdata by employer_name;
getcount = foreach groupdata generate group,COUNT(filterdata.case_status) as cnt;
ordercount = order getcount by cnt desc;
limitdata = limit ordercount 10;
dump limitdata;
```

output:

```
(MICROSOFT CORPORATION,135)
(FACEBOOK, INC.,90)
(UBER TECHNOLOGIES, INC.,46)
(TWITTER, INC.,30)
(AIRBNB, INC.,23)
(AGILONE, INC.,19)
(LINKEDIN CORPORATION,16)
(WAL-MART ASSOCIATES, INC.,15)
(IBM CORPORATION,15)
(GROUPON, INC.,14)
```

CASE 2:

4)Which top 5 employers file the most petitions each year?

EXCEUTION SCRIPT:

data = load '/home/cloudera/Desktop/hadoop_project/hadoop_data' using PigStorage('\t') AS

(id:int,case_status:chararray,employer_name:chararray,soc_name:chararray,job_title:chararray,full_time_position:chararray,prevailing_wages:int,year:chararray,worksite:chararray,longitude:double,latitude:double);

new = foreach data generate case_status,employer_name,year;

groupdata = group new by (employer_name,year);

getcount = foreach groupdata generate FLATTEN(group),COUNT(new.case_status) as cnt;

orderdata = order getcount by cnt desc;

limitdata = limit orderdata 5;

dump limitdata;

(INFOSYS LIMITED,2015,33245)

(INFOSYS LIMITED,2013,32223)

(INFOSYS LIMITED,2016,25352)

(INFOSYS LIMITED,2014,23759)

(CAPGEMINI AMERICA INC,2016,16725)

--->data = load '/home/cloudera/Desktop/hadoop_project/hadoop_data' using PigStorage('\t') AS

(id:int,case_status:chararray,employer_name:chararray,soc_name:chararray,job_title:chararray,full_time_position:chararray,prevailing_wages:int,year:chararray,worksite:chararray,longitude:double,latitude:double);

```
filterdata = filter data by year == '2016';
new = foreach filterdata generate case_status,employer_name,year;
groupdata = group new by employer_name;
getcount = foreach groupdata generate group,COUNT(new.case_status) as
cnt;
orderdata = order getcount by cnt desc;
limitdata = limit orderdata 5;
dump limitdata;
```

output for 2016:

(INFOSYS LIMITED,25352)
(CAPGEMINI AMERICA INC,16725)
(TATA CONSULTANCY SERVICES LIMITED,13134)
(WIPRO LIMITED,10607)
(IBM INDIA PRIVATE LIMITED,9787)

ouput for 2015:
(INFOSYS LIMITED,33245)
(TATA CONSULTANCY SERVICES LIMITED,16553)
(WIPRO LIMITED,12201)
(IBM INDIA PRIVATE LIMITED,10693)
(ACCENTURE LLP,9605)

output for 2014:

(INFOSYS LIMITED,23759)

(TATA CONSULTANCY SERVICES LIMITED,14098)

(WIPRO LIMITED,8365)

(DELOITTE CONSULTING LLP,7017)

(ACCENTURE LLP,5498)

CASE 3:

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

EXCEUTION SCRIPT:

```
data = load '/home/cloudera/Desktop/hadoop_project/hadoop_data' using PigStorage('\t') AS

(id:int,case_status:chararray,employer_name:chararray,soc_name:chararray,job_title:chararray,full_time_position:chararray,prevailing_wages:int,year:chararray,worksite:chararray,longitude:double,latitude:double);

new = foreach data generate case_status,job_title,year;

groupdata = group new by (job_title,year);

getcount = foreach groupdata generate FLATTEN(group),COUNT(new.case_status) as cnt;

orderdata = order getcount by cnt desc;

limitdata = limit orderdata 10;

dump limitdata;
```

(PROGRAMMER ANALYST,2016,53743)

(PROGRAMMER ANALYST,2015,53436)

(PROGRAMMER ANALYST,2014,43114)

(PROGRAMMER ANALYST,2013,33880)

(PROGRAMMER ANALYST,2012,33066)

(PROGRAMMER ANALYST,2011,31799)

(SOFTWARE ENGINEER,2016,30668)

(SOFTWARE ENGINEER,2015,27259)

(SOFTWARE ENGINEER,2014,20500)

(SOFTWARE ENGINEER,2013,15680)

-->data = load '/home/cloudera/Desktop/hadoop_project/hadoop_data' using PigStorage('\t') AS

(id:int,case_status:chararray,employer_name:chararray,soc_name:chararray,job_title:chararray,full_time_position:chararray,prevailing_wages:int,year:chararray,worksite:chararray,longitude:double,latitude:double);

filterdata = filter data by year == '2016';

new = foreach filterdata generate case_status,job_title,year;

groupdata = group new by job_title;

getcount = foreach groupdata generate group,COUNT(new.case_status) as cnt;

orderdata = order getcount by cnt desc;

limitdata = limit orderdata 10;

dump limitdata;

output for 2016:

(PROGRAMMER ANALYST,53743)
(SOFTWARE ENGINEER,30668)
(SOFTWARE DEVELOPER,14041)
(SYSTEMS ANALYST,12314)
(COMPUTER PROGRAMMER,11668)
(BUSINESS ANALYST,9167)
(COMPUTER SYSTEMS ANALYST,6900)
(SENIOR SOFTWARE ENGINEER,6439)
(DEVELOPER,6084)
(TECHNOLOGY LEAD - US,5410)

output for 2015:

(PROGRAMMER ANALYST,53436)
(SOFTWARE ENGINEER,27259)
(COMPUTER PROGRAMMER,14054)
(SYSTEMS ANALYST,12803)
(SOFTWARE DEVELOPER,10441)
(BUSINESS ANALYST,8853)
(TECHNOLOGY LEAD - US,8242)
(COMPUTER SYSTEMS ANALYST,7918)

(TECHNOLOGY ANALYST - US,7014)

(SENIOR SOFTWARE ENGINEER,6013)

CASE 4:

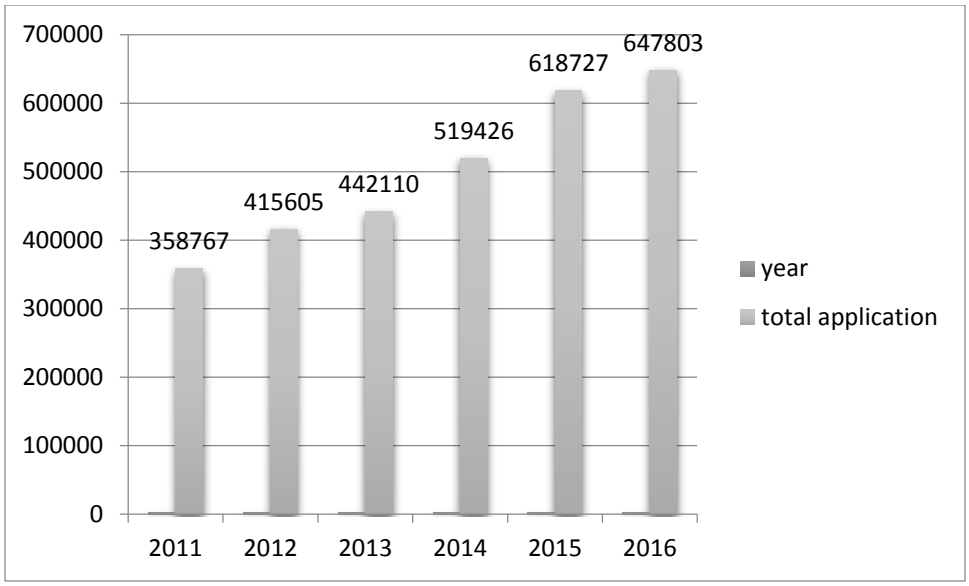
7) Create a bar graph to depict the number of applications for each year

EXECUTION SCRIPT:

```
data = load '/home/cloudera/Desktop/hadoop_project/hadoop_data' using PigStorage('\t') AS
(id:int,case_status:chararray,employer_name:chararray,soc_name:chararray,job_title:chararray,full_time_position:chararray,prevailing_wages:int,year:chararray,worksite:chararray,longitude:double,latitude:double);
new = foreach data generate case_status,year;
groupdata = group new by year;
getcount = foreach groupdata generate group,COUNT(new.case_status);
dump getcount;
```

RESULT:

```
(2011,358767)
(2012,415605)
(2013,442110)
(2014,519426)
(2015,618727)
(2016,647803)
```


700000
600000
500000
400000
300000
200000
100000
0
358767
415605
442110
519426
618727
647803
2011
2012
2013
2014
2015
2016
year
total application

SOFTWARE AND HARDWARE REQUIREMENT

- **Operating System** : Windows 7,8,10 and Mac.
- **Supporting software's:** Ubuntu, putty, Oracle VM Virtual Box, WinSCP, Cloudera 5.10.
- **RAM** : Minimum 8GB.(For better performance)
- **Installation steps:**

STEPS FOR UBUNTU:

To see the distribution/version you are using, you can try:

```
lsb_release -a
```

to find out version of ubuntu
cat /etc/lsb-release

Installing Java

Hadoop framework is written in Java!!

```
user1@localhost:~$ cd ~
pwd

# Update the source list
user@laptop(local directory):~$ sudo apt-get update

# The OpenJDK project is the default version of Java
```

```
# that is provided from a supported Ubuntu repository.
user@laptop:~$ sudo apt-get install default-jdk


user@laptop:~$ java -version
java version "1.7.0_65"
OpenJDK Runtime Environment (IcedTea 2.5.3) (7u71-2.5.3-0ubuntu0.14.04.1)
OpenJDK 64-Bit Server VM (build 24.65-b04, mixed mode)
```

Adding a dedicated Hadoop user

```
user@laptop:~$ sudo addgroup hadoop
Adding group `hadoop' (GID 1002) ...
Done.


user@laptop:~$ sudo adduser --ingroup hadoop hduser
Adding user `hduser' ...
Adding new user `hduser' (1001) with group `hadoop' ...
Creating home directory `/home/hduser' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hduser
Enter the new value, or press ENTER for the default
       Full Name []:
       Room Number []:
       Work Phone []:
       Home Phone []:
       Other []:
```

```
Is the information correct? [Y/n] Y
```

Installing SSH

ssh has two main components:

1. **ssh** : The command we use to connect to remote machines - the client.
2. **sshd** : The daemon that is running on the server and allows clients to connect to the server.

The **ssh** is pre-enabled on Linux, but in order to start **sshd** daemon, we need to install**ssh** first. Use this command to do that :

```
user@laptop:~$ sudo apt-get install ssh
```

This will install ssh on our machine. If we get something similar to the following, we can think it is setup properly:

```
user@laptop:~$ which ssh
/usr/bin/ssh

user@laptop:~$ which sshd
/usr/sbin/sshd
```

Create and Setup SSH Certificates

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus our local machine. For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost.

So, we need to have SSH up and running on our machine and configured it to allow SSH public key authentication.

Hadoop uses SSH (to access its nodes) which would normally require the user to enter a password. However, this requirement can be eliminated by creating and setting up SSH certificates using the following commands. If asked for a filename just leave it blank and press the enter key to continue.

```
user@laptop:~$ su hduser
Password:
hduser@laptop:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
50:6b:f3:fc:0f:32:bf:30:79:c2:41:71:26:cc:7d:e3 hduser@laptop
The key's randomart image is:
+--[ RSA 2048]----+
|        .oo.o    |
|       . .o=. o  |
|      . + .  o . |
|       o =    E  |
|        S +      |
|         . +     |
|          O +    |
|           O o   |
|            o..  |
+-----------------+


hduser@laptop:/home/k$ cat $HOME/.ssh/id_rsa.pub >>
$HOME/.ssh/authorized_keys
```

The second command adds the newly created key to the list of authorized keys so that Hadoop can use ssh without prompting for a password.

We can check if ssh works:

```
hduser@laptop:/home/k$ ssh localhost

The authenticity of host 'localhost (127.0.0.1)' can't be established.

ECDSA key fingerprint is e1:8b:a0:a5:75:ef:f4:b4:5e:a9:ed:be:64:be:5c:2f.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.

Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-40-generic x86_64)

...
```

Install Hadoop

```
hduser@laptop:~$ su hduser

wget http://mirror.fibergrid.in/apache/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz

OR

wget http://redrockdigimark.com/apachemirror/hadoop/common/hadoop-2.6.0/ hadoop-2.6.0.tar.gz
```

```
hduser@laptop:~$ tar xvzf hadoop-2.6.0.tar.gz
a folder would be created by the name of hadoop-2.6.0
rename this folder to hadoop
mv hadoop-2.6.0 hadoop
```

We want to move the Hadoop installation to the **/usr/local/hadoop** directory using the following command:

```
hduser@laptop:~$ sudo mv hadoop /usr/local/
[sudo] password for hduser:
hduser is not in the sudoers file.  This incident will be reported.
```

Oops!... We got:

```
"hduser is not in the sudoers file. This incident will be reported."
```

This error can be resolved by logging in as a root user, and then add **hduser** to **sudo**:

```
hduser@laptop:~/hadoop-2.6.0$ su <old user> (earlier user)
Password:

k@laptop:/home/hduser$ sudo adduser hduser sudo
[sudo] password for k:
Adding user `hduser' to group `sudo' ...
Adding user hduser to group sudo
Done.
```

Now, the **hduser** has root priviledge, we can move the Hadoop installation to the**/usr/local/hadoop** directory without any problem:

```
k@laptop:/home/hduser$ sudo su hduser
```

```
cd ~
hduser@laptop:~/$ sudo mv hadoop /usr/local/
hduser@laptop:~/$ sudo chown -R hduser:hadoop /usr/local/hadoop
```

----------standalone done till here----------

Setup Configuration Files

The following files will have to be modified to complete the Hadoop setup:

1. ~/.bashrc
2. /usr/local/hadoop/etc/hadoop/hadoop-env.sh
3. /usr/local/hadoop/etc/hadoop/core-site.xml
4. /usr/local/hadoop/etc/hadoop/mapred-site.xml
5. /usr/local/hadoop/etc/hadoop/hdfs-site.xml

1. ~/.bashrc:

Before editing the **.bashrc** file in our home directory, we need to find the path where Java has been installed to set the **JAVA_HOME** environment variable using the following command:

```
hduser@laptop update-alternatives --config java
There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java
Nothing to configure.
```

Now we can append the following to the end of **~/.bashrc**:

```
hduser@laptop:~$ nano ~/.bashrc


#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
```

```
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END

hduser@laptop:~$ source ~/.bashrc
```

note that the JAVA_HOME should be set as the path just before the '.../bin/':

```
hduser@ubuntu-VirtualBox:~$ javac -version
javac 1.7.0_75

hduser@ubuntu-VirtualBox:~$ which javac
/usr/bin/javac
s
hduser@ubuntu-VirtualBox:~$ readlink -f /usr/bin/javac
/usr/lib/jvm/java-7-openjdk-amd64/bin/javac
```

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

We need to set **JAVA_HOME** by modifying **hadoop-env.sh** file.

```
hduser@laptop:~$ nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh


export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

Adding the above statement in the **hadoop-env.sh** file ensures that the value of JAVA_HOME variable will be available to Hadoop whenever it is started up.

3. /usr/local/hadoop/etc/hadoop/core-site.xml:

The **/usr/local/hadoop/etc/hadoop/core-site.xml** file contains configuration properties that Hadoop uses when starting up.
This file can be used to override the default settings that Hadoop starts with.

```
hduser@laptop:~$ sudo mkdir -p /app/hadoop/tmp
hduser@laptop:~$ sudo chown hduser:hadoop /app/hadoop/tmp
```

Open the file and enter the following in between the <configuration></configuration> tag:

```
hduser@laptop:~$ nano /usr/local/hadoop/etc/hadoop/core-site.xml


<configuration>
 <property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
 </property>

 <property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
  <description>The name of the default file system.  A URI whose
  scheme and authority determine the FileSystem implementation. The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
```

```
  the FileSystem implementation class.  The uri's authority is used
to
  determine the host, port, etc. for a filesystem.</description>
 </property>
</configuration>
```

4. /usr/local/hadoop/etc/hadoop/mapred-site.xml

By default, the **/usr/local/hadoop/etc/hadoop/** folder contains **/usr/local/hadoop/etc/hadoop/mapred-site.xml.template** file which has to be renamed/copied with the name **mapred-site.xml**:

```
hduser@laptop:~$ cp /usr/local/hadoop/etc/hadoop/mapred-
site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml
```

The **mapred-site.xml** file is used to specify which framework is being used for MapReduce.
We need to enter the following content in between the <configuration></configuration> tag:

```
<configuration>
 <property>
  <name>mapred.job.tracker</name>
  <value>localhost:54311</value>
  <description>The host and port that the MapReduce job tracker
runs
  at.  If "local", then jobs are run in-process as a single map
  and reduce task.
  </description>
 </property>
</configuration>
```

5. /usr/local/hadoop/etc/hadoop/hdfs-site.xml

The **/usr/local/hadoop/etc/hadoop/hdfs-site.xml** file needs to be configured for each host in the cluster that is being used.
It is used to specify the directories which will be used as the **namenode** and the**datanode** on that host.

Before editing this file, we need to create two directories which will contain the namenode and the datanode for this Hadoop installation.
This can be done using the following commands:

```
hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode

hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode

hduser@laptop:~$ sudo chown -R hduser:hadoop /usr/local/hadoop_store
```

Open the file and enter the following content in between the <configuration></configuration> tag:

```
hduser@laptop:~$ nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml


<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
```

```
  </description>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
 </property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
 </property>
</configuration>
```

Format the New Hadoop Filesystem

Now, the Hadoop file system needs to be formatted so that we can start to use it. The format command should be issued with write permission since it creates **current** directory under **/usr/local/hadoop_store/hdfs/namenode** folder:

```
hduser@laptop:~$ hadoop namenode -format
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

15/04/18 14:43:03 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = laptop/192.168.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.6.0
```

```
STARTUP_MSG:   classpath = /usr/local/hadoop/etc/hadoop

...

STARTUP_MSG:   java = 1.7.0_65

************************************************************/

15/04/18 14:43:03 INFO namenode.NameNode: registered UNIX signal handlers for [TERM, HUP, INT]

15/04/18 14:43:03 INFO namenode.NameNode: createNameNode [-format]

15/04/18 14:43:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Formatting using clusterid: CID-e2f515ac-33da-45bc-8466-5b1100a2bf7f

15/04/18 14:43:09 INFO namenode.FSNamesystem: No KeyProvider found.

15/04/18 14:43:09 INFO namenode.FSNamesystem: fsLock is fair:true

15/04/18 14:43:10 INFO blockmanagement.DatanodeManager: dfs.block.invalidate.limit=1000

15/04/18 14:43:10 INFO blockmanagement.DatanodeManager: dfs.namenode.datanode.registration.ip-hostname-check=true

15/04/18 14:43:10 INFO blockmanagement.BlockManager: dfs.namenode.startup.delay.block.deletion.sec is set to 000:00:00:00.000

15/04/18 14:43:10 INFO blockmanagement.BlockManager: The block deletion will start around 2015 Apr 18 14:43:10

15/04/18 14:43:10 INFO util.GSet: Computing capacity for map BlocksMap

15/04/18 14:43:10 INFO util.GSet: VM type       = 64-bit

15/04/18 14:43:10 INFO util.GSet: 2.0% max memory 889 MB = 17.8 MB

15/04/18 14:43:10 INFO util.GSet: capacity       = 2^21 = 2097152 entries

15/04/18 14:43:10 INFO blockmanagement.BlockManager: dfs.block.access.token.enable=false

15/04/18 14:43:10 INFO blockmanagement.BlockManager: defaultReplication         = 1

15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxReplication = 512

15/04/18 14:43:10 INFO blockmanagement.BlockManager: minReplication = 1
```

```
15/04/18      14:43:10      INFO      blockmanagement.BlockManager:
maxReplicationStreams      = 2

15/04/18      14:43:10      INFO      blockmanagement.BlockManager:
shouldCheckForEnoughRacks  = false

15/04/18      14:43:10      INFO      blockmanagement.BlockManager:
replicationRecheckInterval = 3000

15/04/18      14:43:10      INFO      blockmanagement.BlockManager:
encryptDataTransfer        = false

15/04/18      14:43:10      INFO      blockmanagement.BlockManager:
maxNumBlocksToLog          = 1000

15/04/18 14:43:10 INFO namenode.FSNamesystem: fsOwner             =
hduser (auth:SIMPLE)

15/04/18 14:43:10 INFO namenode.FSNamesystem: supergroup          =
supergroup

15/04/18 14:43:10 INFO namenode.FSNamesystem: isPermissionEnabled =
true

15/04/18 14:43:10 INFO namenode.FSNamesystem: HA Enabled: false

15/04/18 14:43:10 INFO namenode.FSNamesystem: Append Enabled: true

15/04/18  14:43:11  INFO  util.GSet:  Computing  capacity  for  map
INodeMap

15/04/18 14:43:11 INFO util.GSet: VM type       = 64-bit

15/04/18 14:43:11 INFO util.GSet: 1.0% max memory 889 MB = 8.9 MB

15/04/18 14:43:11 INFO util.GSet: capacity       = 2^20 = 1048576
entries

15/04/18  14:43:11  INFO  namenode.NameNode:  Caching  file  names
occuring more than 10 times

15/04/18  14:43:11  INFO  util.GSet:  Computing  capacity  for  map
cachedBlocks

15/04/18 14:43:11 INFO util.GSet: VM type       = 64-bit

15/04/18 14:43:11 INFO util.GSet: 0.25% max memory 889 MB = 2.2 MB

15/04/18 14:43:11 INFO util.GSet: capacity       = 2^18 = 262144
entries

15/04/18        14:43:11        INFO        namenode.FSNamesystem:
dfs.namenode.safemode.threshold-pct = 0.9990000128746033

15/04/18        14:43:11        INFO        namenode.FSNamesystem:
dfs.namenode.safemode.min.datanodes = 0
```

```
15/04/18 14:43:11 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension = 30000

15/04/18 14:43:11 INFO namenode.FSNamesystem: Retry cache on namenode is enabled

15/04/18 14:43:11 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 millis

15/04/18 14:43:11 INFO util.GSet: Computing capacity for map NameNodeRetryCache

15/04/18 14:43:11 INFO util.GSet: VM type = 64-bit

15/04/18 14:43:11 INFO util.GSet: 0.029999999329447746% max memory 889 MB = 273.1 KB

15/04/18 14:43:11 INFO util.GSet: capacity = 2^15 = 32768 entries

15/04/18 14:43:11 INFO namenode.NNConf: ACLs enabled? false

15/04/18 14:43:11 INFO namenode.NNConf: XAttrs enabled? true

15/04/18 14:43:11 INFO namenode.NNConf: Maximum size of an xattr: 16384

15/04/18 14:43:12 INFO namenode.FSImage: Allocated new BlockPoolId: BP-130729900-192.168.1.1-1429393391595

15/04/18 14:43:12 INFO common.Storage: Storage directory /usr/local/hadoop_store/hdfs/namenode has been successfully formatted.

15/04/18 14:43:12 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0

15/04/18 14:43:12 INFO util.ExitUtil: Exiting with status 0

15/04/18 14:43:12 INFO namenode.NameNode: SHUTDOWN_MSG:

/************************************************************

SHUTDOWN_MSG: Shutting down NameNode at laptop/192.168.1.1

************************************************************/
```

Note that **hadoop namenode -format** command should be executed once before we start using Hadoop.
If this command is executed again after Hadoop has been used, it'll destroy all the data on the Hadoop file system.

Starting Hadoop

Now it's time to start the newly installed single node cluster.
We can use **start-all.sh** or (**start-dfs.sh** and **start-yarn.sh**)

```
k@laptop:~$ cd /usr/local/hadoop/sbin

k@laptop:/usr/local/hadoop/sbin$ ls
distribute-exclude.sh    start-all.cmd        stop-balancer.sh
hadoop-daemon.sh         start-all.sh         stop-dfs.cmd
hadoop-daemons.sh        start-balancer.sh    stop-dfs.sh
hdfs-config.cmd          start-dfs.cmd        stop-secure-dns.sh
hdfs-config.sh           start-dfs.sh         stop-yarn.cmd
httpfs.sh                start-secure-dns.sh  stop-yarn.sh
kms.sh                   start-yarn.cmd       yarn-daemon.sh
mr-jobhistory-daemon.sh  start-yarn.sh        yarn-daemons.sh
refresh-namenodes.sh     stop-all.cmd
slaves.sh                stop-all.sh

k@laptop:/usr/local/hadoop/sbin$ sudo su hduser

hduser@laptop:/usr/local/hadoop/sbin$ start-all.sh
hduser@laptop:~$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
15/04/18 16:43:13 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
```

```
localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-namenode-laptop.out

localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-laptop.out

Starting secondary namenodes [0.0.0.0]

0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-laptop.out

15/04/18 16:43:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

starting yarn daemons

starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-laptop.out

localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-laptop.out
```

We can check if it's really up and running:

```
hduser@laptop:/usr/local/hadoop/sbin$ jps

9026 NodeManager

7348 NameNode

9766 Jps

8887 ResourceManager

7507 DataNode

7350 Secondary Namenode
```

The output means that we now have a functional instance of Hadoop running on our VPS (Virtual private server).

Another way to check is using **netstat**:

```
hduser@laptop:~$ netstat -plten | grep java

(Not all processes could be identified, non-owned process info

 will not be shown, you would have to be root to see it all.)

tcp             0          0  0.0.0.0:50020                 0.0.0.0:*
LISTEN      1001       1843372     10605/java
```

```
tcp              0          0  127.0.0.1:54310                 0.0.0.0:*
LISTEN      1001       1841277     10447/java

tcp              0          0  0.0.0.0:50090                   0.0.0.0:*
LISTEN      1001       1841130     10895/java

tcp              0          0  0.0.0.0:50070                   0.0.0.0:*
LISTEN      1001       1840196     10447/java

tcp              0          0  0.0.0.0:50010                   0.0.0.0:*
LISTEN      1001       1841320     10605/java

tcp              0          0  0.0.0.0:50075                   0.0.0.0:*
LISTEN      1001       1841646     10605/java

tcp6             0          0  :::8040                              :::*
LISTEN      1001       1845543     11383/java

tcp6             0          0  :::8042                              :::*
LISTEN      1001       1845551     11383/java

tcp6             0          0  :::8088                              :::*
LISTEN      1001       1842110     11252/java

tcp6             0          0  :::49630                             :::*
LISTEN      1001       1845534     11383/java

tcp6             0          0  :::8030                              :::*
LISTEN      1001       1842036     11252/java

tcp6             0          0  :::8031                              :::*
LISTEN      1001       1842005     11252/java

tcp6             0          0  :::8032                              :::*
LISTEN      1001       1842100     11252/java

tcp6             0          0  :::8033                              :::*
LISTEN      1001       1842162     11252/java
```

Stopping Hadoop

```
$ pwd

/usr/local/hadoop/sbin


$ ls

distribute-exclude.sh   httpfs.sh                        start-all.sh
start-yarn.cmd    stop-dfs.cmd        yarn-daemon.sh
```

```
hadoop-daemon.sh         mr-jobhistory-daemon.sh   start-balancer.sh
start-yarn.sh     stop-dfs.sh         yarn-daemons.sh

hadoop-daemons.sh          refresh-namenodes.sh        start-dfs.cmd
stop-all.cmd      stop-secure-dns.sh

hdfs-config.cmd          slaves.sh                    start-dfs.sh
stop-all.sh       stop-yarn.cmd

hdfs-config.sh         start-all.cmd            start-secure-dns.sh
stop-balancer.sh  stop-yarn.sh
```

We run **stop-all.sh** or (**stop-dfs.sh** and **stop-yarn.sh**) to stop all the daemons running on our machine:

```
hduser@laptop:/usr/local/hadoop/sbin$ pwd

/usr/local/hadoop/sbin

hduser@laptop:/usr/local/hadoop/sbin$ ls

distribute-exclude.sh   httpfs.sh                     start-all.cmd
start-secure-dns.sh  stop-balancer.sh    stop-yarn.sh

hadoop-daemon.sh         kms.sh                        start-all.sh
start-yarn.cmd       stop-dfs.cmd        yarn-daemon.sh

hadoop-daemons.sh        mr-jobhistory-daemon.sh   start-balancer.sh
start-yarn.sh        stop-dfs.sh         yarn-daemons.sh

hdfs-config.cmd            refresh-namenodes.sh        start-dfs.cmd
stop-all.cmd         stop-secure-dns.sh

hdfs-config.sh            slaves.sh                    start-dfs.sh
stop-all.sh          stop-yarn.cmd

hduser@laptop:/usr/local/hadoop/sbin$

hduser@laptop:/usr/local/hadoop/sbin$ stop-all.sh

This script is Deprecated. Instead use stop-dfs.sh and stop-yarn.sh

15/04/18  15:46:31  WARN  util.NativeCodeLoader:  Unable  to  load
native-hadoop  library  for  your  platform...  using  builtin-java
classes where applicable

Stopping namenodes on [localhost]

localhost: stopping namenode

localhost: stopping datanode

Stopping secondary namenodes [0.0.0.0]

0.0.0.0: no secondarynamenode to stop
```

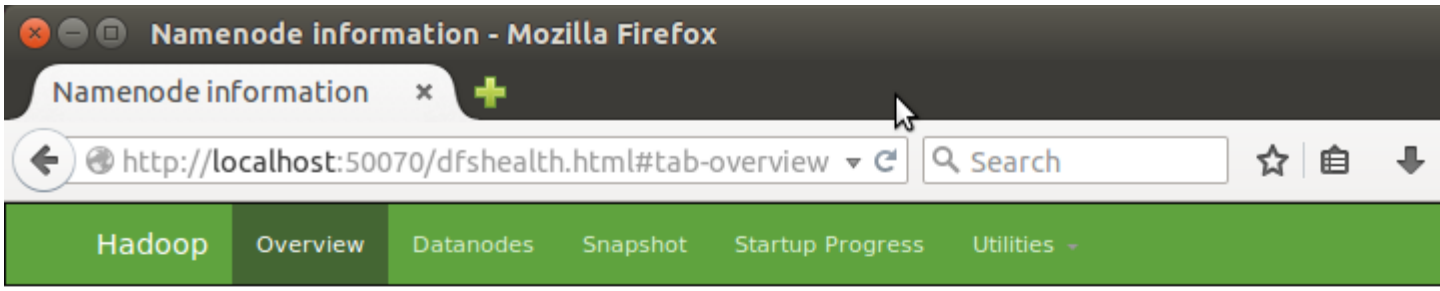
```
15/04/18 15:46:59 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
stopping yarn daemons
stopping resourcemanager
localhost: stopping nodemanager
no proxyserver to stop
```

Hadoop Web Interfaces

Let's start the Hadoop again and see its Web UI:

```
hduser@laptop:/usr/local/hadoop/sbin$ start-all.sh
```

http://127.0.0.1:50070/ - web UI of the NameNode daemon



Overview 'localhost:54310' (active)

Started:	Sat Apr 18 15:53:55 PDT 2015
Version:	2.6.0, re3496499ecb8d220fba99dc5ed4c99c8f9e33bb1
Compiled:	2014-11-13T21:10Z by jenkins from (detached from e349649)
Cluster ID:	CID-e2f515ac-33da-45bc-8466-5b1100a2bf7f
Block Pool ID:	BP-130729900-192.168.1.1-1429393391595

Summary

Security is off.

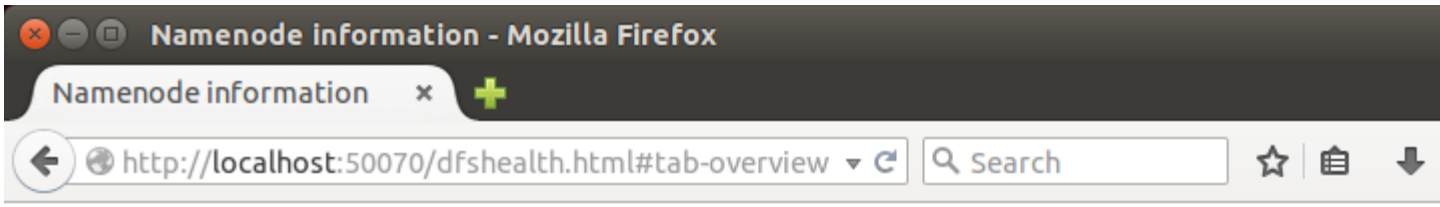
Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 58.41 MB of 167.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 28.34 MB of 29.94 MB Commited Non Heap Memory. Max Non Heap Memory is 214 MB.

http://localhost:50070/dfshealth.html#tab-startup-progress



Summary

Security is off.

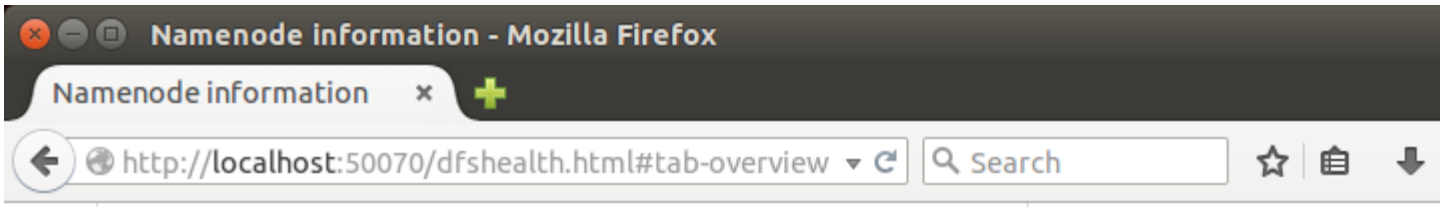
Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 58.41 MB of 167.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 28.34 MB of 29.94 MB Commited Non Heap Memory. Max Non Heap Memory is 214 MB.

Configured Capacity:	454.29 GB
DFS Used:	24 KB
Non DFS Used:	125.8 GB
DFS Remaining:	328.49 GB
DFS Used%:	0%
DFS Remaining%:	72.31%
Block Pool Used:	24 KB
Block Pool Used%:	0%
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	4/18/2015, 3:53:55 PM



DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	4/18/2015, 3:53:55 PM

NameNode Journal Status

Current transaction ID: 2

Journal Manager	State
FileJournalManager(root=/usr/local /hadoop_store/hdfs/namenode)	EditLogFileOutputStream(/usr/local/hadoop_store/hdfs/namenode/cu /edits_inprogress_0000000000000000002)

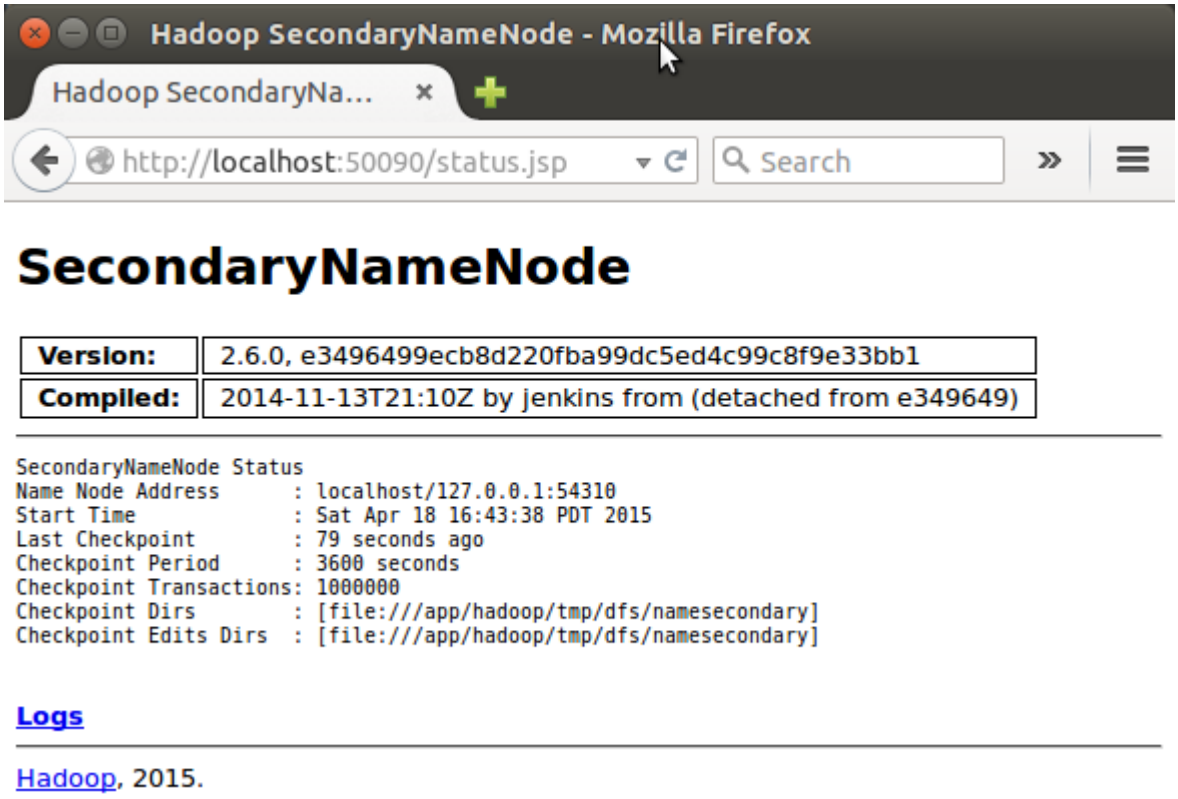
NameNode J

NameNode Storage

Storage Directory	Type	S
/usr/local/hadoop_store/hdfs/namenode	IMAGE_AND_EDITS	A

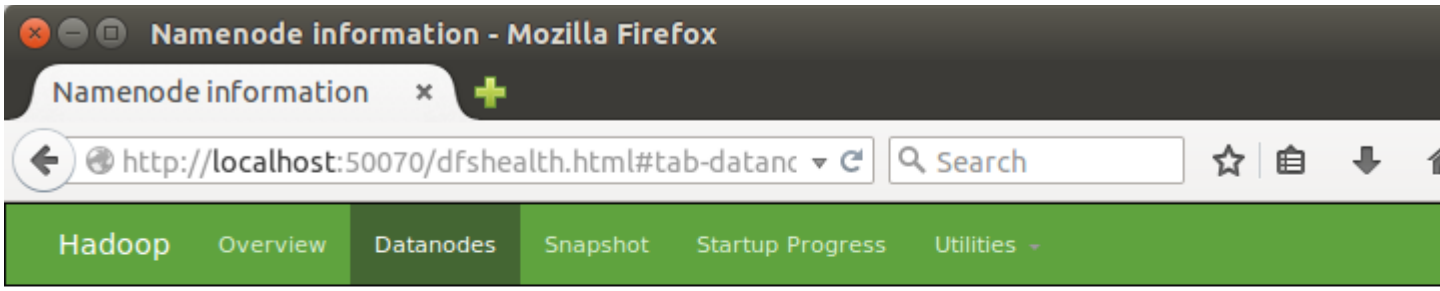
Hadoop, 2014.

SecondaryNameNode



(Note) I had to restart Hadoop to get this Secondary Namenode.

DataNode



Datanode Information

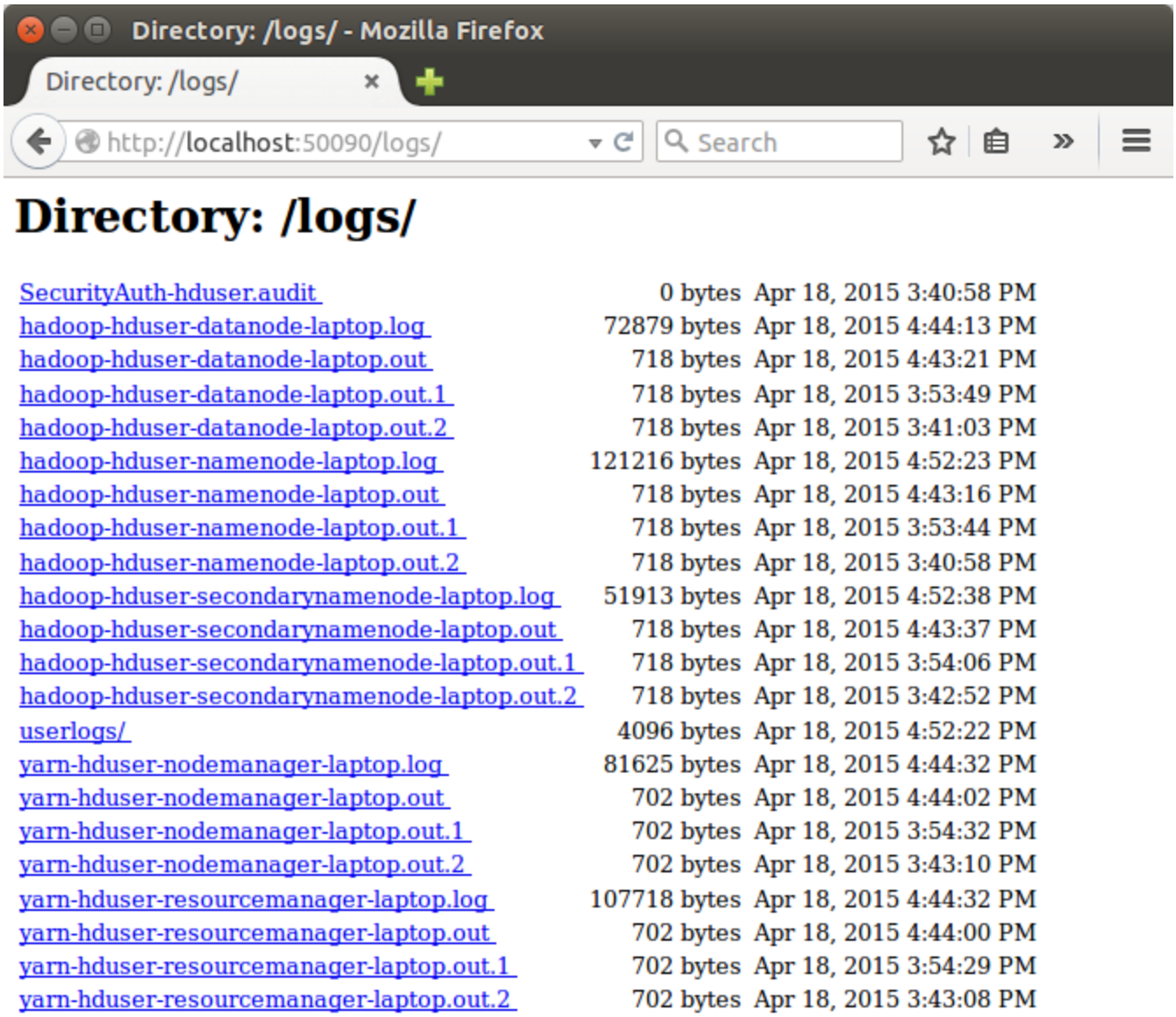
In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes
laptop (127.0.0.1:50010)	1	In Service	454.29 GB	28 KB	125.83 GB	328.47 GB	0	28 KB (0%)	0

Decomissioning

Node	Last contact	Under replicated blocks	Blocks with no live replicas	Under Replicated Blocks In files under constructio

Hadoop, 2014.



Install Eclipse on Ubuntu 10.04 Lucid Lynx

From Ubuntu Shell

1.Open terminal by hitting CTRL+ALT+T or from Applications->accessories->Terminal.

2.Type the command written below,you may have to enter your user account password to install the Eclipse.

$ su hduser
$ sudo apt-get install eclipse
3.Downloading and installation process will finish soon.
4.That's all Now,enjoy using eclipse.Access it from Applications->Programming->Eclipse.

HIVE INSTALLATION:

download hive from apache site in home directory

cd ~ [this command will take you to home dir]

pwd [/home/hduser]

wget http://archive.apache.org/dist/hive/hive-1.2.1/apache-hive-1.2.1-bin.tar.gz

OR

wget http://redrockdigimark.com/apachemirror/hive/hive-1.2.1/apache-hive-1.2.1-bin.tar.gz

tar -xvzf apache-hive-1.2.1-bin.tar.gz

ls

apache-hive-1.2.1-bin

mv apache-hive.1.2.1-bin **hive**

sudo mv **hive** /usr/local/

sudo chown -R hduser:hadoop /usr/local/hive

nano ~/.bashrc

export HIVE_HOME=/usr/local/hive

export PATH=$PATH:$HIVE_HOME/bin

source ~/.bashrc

hadoop fs -mkdir -p /tmp/hive

hadoop fs -mkdir -p **/user/hive/warehouse**

hadoop fs -chmod g+w /tmp/hive

hadoop fs -chmod g+w /user/hive/warehouse

sudo mkdir /usr/local/hive/iotmp

sudo chown -R hduser:hadoop /usr/local/hive/iotmp

sudo chmod 777 /usr/local/hive/iotmp

hadoop fs -chmod 777 /tmp/hive

Configuration files for Hive are under /usr/local/hive/conf

hive-default.xml.template contains the default values for various configuration variables that come prepackaged in a Hive distribution. In order to override any

of the values, create `hive-site.xml` instead and set the value in that file as shown above.

`hive-default.xml.template` is located in the `conf` directory in your installation root, and `hive-site.xml` should also be created in the same directory.

cd /usr/local/hive/conf

cp `hive-default.xml.template hive-site.xml`

gedit hive-site.xml or nano hive-site.xml

do not append. make changes in the values in hive-site.xml

```
<property>
    <name>hive.exec.scratchdir</name>
    <value>/tmp/hive</value>
    <description>HDFS root scratch dir for
Hive jobs which gets created with write all
(733) permission. For each connecting user,
an HDFS scratch dir:
${hive.exec.scratchdir}/&lt;username&gt; is
created, with
${hive.scratch.dir.permission}.</descriptio
n>
  </property>
  <property>
    <name>hive.exec.local.scratchdir</name>
    <value>/usr/local/hive/iotmp</value>
```

```
    <description>Local scratch space for
Hive jobs</description>
  </property>
  <property>

<name>hive.downloaded.resources.dir</name>
    <value>/usr/local/hive/iotmp</value>
    <description>Temporary local directory
for added resources in the remote file
system.</description>
  </property>
<property>
   <name>hive.querylog.location</name>
    <value>/usr/local/hive/iotmp</value>
    <description>Location of Hive run time
structured log file</description>
 </property>
```

Hive meta store settings in hive-site.xml

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:derby:;databaseName=/usr/local/hive/metasto
re_db;create=true</value>
  <description>JDBC connect string for a
JDBC metastore</description>
 </property>
```

make changes by adding the below command in hive-env.sh

if the file is not available then copy file from default file

cp hive-env.sh.template hive-env.sh

add the following line in the hive-env.sh

export HADOOP_USER_CLASSPATH_FIRST=true

$ hive

hive>

PIG INSTALLATION:

Pig Installation and execution of a query

1. Download pig from pig.apache.org

2. copy it to the hadoop user /home/hduser/

```
wget http://archive.apache.org/dist/pig/pig-0.13.0/pig-0.13.0.tar.gz
```

tar xvzf pig-0.13.0.tar.gz

mv pig-0.13.0 pig

sudo mv /home/hduser/pig /usr/local/

```
sudo chown -R hduser:hadoop /usr/local/pig
```

```
nano ~/.bashrc
```

export PATH=$PATH:/usr/local/pig/bin

source ~/.bashrc

for (map reduce mode)

$ pig

or

for (local mode)

$ pig -x local

grunt>

----installation ends here ---

3. extract , make sure the extracted directory has the same permission (hduser:hadoop), insert path to $PATH
4. Start the hadoop server (./start-all.sh), check with jps
5. Copy the data to HDFS,
>hadoop dfs -copyFromLocal /user/hduser/<file-name>
6. run script using pig
>pig <script-name>
7. check if output is generated
>hadoop dfs -ls /user/hduser/<file-name>/
8. check output file
> hadoop dfs -cat /user/hduser/<output-dir>/part-r-00000

Conclusion

With these different scenarios we can give different dimensional solution in accurate with a huge dataset. This type of data solution will help the companies to maintain their records up-to-date in more effective manner. Hadoop will help us to perform analysis part and execution part very faster and easier.