1.Introduction

1.1 .Problem Definition

This project aimed to develop a scalable Internet traffic analysis system using Hadoop that can manage packets on HDFS.

1.2 .Literature survey

We have read some IEEE papers related to hadoop based traffic analysis system. We got more help from Yeonhee Lee and Youngseok Lee's paper on "Towards scalable internet traffic analysis using hadoop". Also we read paper by Dipti J. Suryawanshi, Prof. Mr. U. A. Mande on "Traffic Measurement and Analysis with Hadoop".This paper also account in gaining knowledge about big data analytics. Detailed study of these and other literature is described in next chapter.

2.Literature Survey and Study

D. Venkatesh, M. Prasanna, N. Madhava Reddy[1] discussed about use of the traffic monitoring system which performs packet analysis on massive Internet traffic in a scalable manner. Hadoop based traffic monitoring system provides scalable data processing with many storage services based on a distributed computing system which can be best utilized for traffic measurements and analysis.

YuanjunCai, Bin Wu, XinweiZhang[2], Min Luo and Jinzhao Su presented a Hadoop based system for huge network traffic data which accepts input from multiple packet traces, performs flow identification, characteristics mining and flow clustering, system output provides guideline in resource allocation and flow scheduling.

Yeonhee Lee and Youngseok Lee[3] discussed the about hadoop based traffic observing system that performs packet analysis of multi- terabyte of data set in scalable manner.They also explained the performance issues related with traffic analysis MapReduce approach.

S.J Shafer, S. Rixner, and Alan L[4]. Cox discuss about performance of Hadoop based system. Hadoop is most accepted distributed computing framework for managing massive data in distributed environment. Hadoop makes use of file system in distributedmanner. The HDFS (Hadoop Distributed File System) is independent across both hardware and software platforms. In this paper a deep performance analysis of HDFS was done and it shows a lot of performance issues. Initially the issue was on architectural bottleneck cause the inefficient usage of HDFS. The another limitation was based on portability limitations which limited the java implementation. This paper tries to find solution for the bottleneck and portability problems in the Hadoop based system.

J. Erman, M. Arlitt, and A. Mahanti[6] explain the value of the clustering environment to identify traffic that are similar using only transport layer statistics. They have considered two

clustering algorithms one is K-Means and another is DBSCAN, that have not been used previously. The results showed that although DBSCAN has limitation compared to K-Means.

A lot of research has been done in the analysis of internet traffic using hadoop, Ranganathan T.G and Narayanan H.M[7] discussed the model for the same system using virtual network created by the Mininet. Then the meta data about the traffic flow in the designed virtual network captured in the form of log file. Hadoop cluster is used for the flow analysis and defining the flow control algorithm.

Hadoop has become a popular infrastructure for massive data analytics because it facilitates scalable data processing and storage services on a distributed computing system consisting of commodity hardware. Yuanjun Cai, Bin Wu, Xinwei Zhang present a Hadoop-based traffic analysis system, which accepts input from multiple data traces, performs flow identification, characteristics mining and flow clustering, output of the system provides guidance in resource allocation, flow scheduling and some other tasks. Experiment on a dataset about 8G size from university datacenter network shows that the system is able to finish flow characteristics mining on a four node cluster within 23minutes.

Yeonhee Lee and Youngseok Lee present a Hadoop-based traffic monitoring system that performs IP, TCP, HTTP, and NetFlow analysis of multi-terabytes of Internet traffic in a scalable manner. From experiments with a 200-node test bed, we achieved 14 Gbps throughput for 5 TB files with IP and HTTP-layer analysis MapReduce jobs. We also explain the performance issues related with traffic analysis MapReduce jobs.

Parekh Nilaykumar B and ,Modasa, Aravalli[19] proposed the network traffic monitoring system that will provide a new approach to measuring and analysis Internet traffic which is based on MapReduce scenario. This new approach in hadoop will try to improve the computational time, more fault tolerance and flow control of system and will handle or deal with Big data during internet traffic measurement and monitoring.

Jeffrey Dean and Sanjay Ghemawat gave in short introduction about mapreduce programing scenario. They state that hadoop as it is developed in java, it can be deployed on any commodity hardware as well as it is platform independent. Mapreduce approach in hadoop gives a distributed parallel processing environment which will reduce computing time for analysis and increase efficiency of system. He also explains that many programmers prefer the MapReduce approach to solve the big data problems. Big data store centres like google, yahoo, facebook are also using this approach to solve day to day problems.

3. Software Requirement Specification

3.1.Introduction

3.1.1.Document Purpose

After analysis of number of ongoing research we came across that the tools such as wireshark, netflowetc are not that much efficient for the large data set of internet traffic. So we decide the topic "**Internet Traffic analysis using Hadoop**".

Big data is needed where volume of data is beyond capacity of regularly used softwares to manage data. Big data might be data in petabytes(1024 terabytes) or exabytes(1024 petabytes) consist million records of millions of peoples from different sources. The data is typically loosely structured data. Big Data as high volume, velocity and variety information assets that may be in the form of text file or some other source of storage. Big data is used for storage purpose.

The primary purpose of this document is to specify the requirements for the given project and to make familiar user or actor with the project.

3.1.2.Product Scope

The main scope of this project will be as following:

a) To design and implement a Traffic flow identification System using hadoop.
b) The analysis of the internet traffic on the network which is large in size.
c) Generate analysis such that it will be useful for network administrator to monitor faults and also to plan for the upfront risks using previous patterns.
d) To deploy the hadoop cluster for the proposed project.
e) To increase quality of service provided by internet service provider.

This project will predict the future changes which need to be done in the network. It will also help the network administrator to consider the new network policies. Along with the project, deployment of Hadoop cluster will help upcoming research in the field of Big Data and Data analytics for the students and researchers. This project presents a Hadoop-based tool that offers not only Internet traffic measurement utilities to network operators but also various analysis capabilities on a huge amount of packet data to network researchers and analysts. This work establishes a guideline for researchers on how to write network measurement applications with MapReducealgorithm and adopt high-level data flow language.

3.1.3.Intended Audience and Document Overview

This work establishes a guideline for researchers on how to write network measurement applications with MapReduce algorithm and adopt high-level data flow language. Along with the project, deployment of Hadoop cluster will help upcoming research in the field of Big Data and Data analytics for the students and researchers.

This work also helpful for network administrator and ISP(Internet Service Provider) for future plan and services of Network. It also helps for NSP(Network Service Provider) to monitor huge data of network.

3.2.Overall Description

3.2.1.PRODUCT PERSPECTIVE

Looking back in the year 2014-15,the network of Rajarambapu Institute of Technology have many problems like they were not robust and there was savior cases of network failure, there may be different issues but network admin was unable to provide the quality of service.

Hence we decided to do such analysis for the future planning and better quality of services (Qos).

3.2.2.PRODUCT FUNCTIONALITY

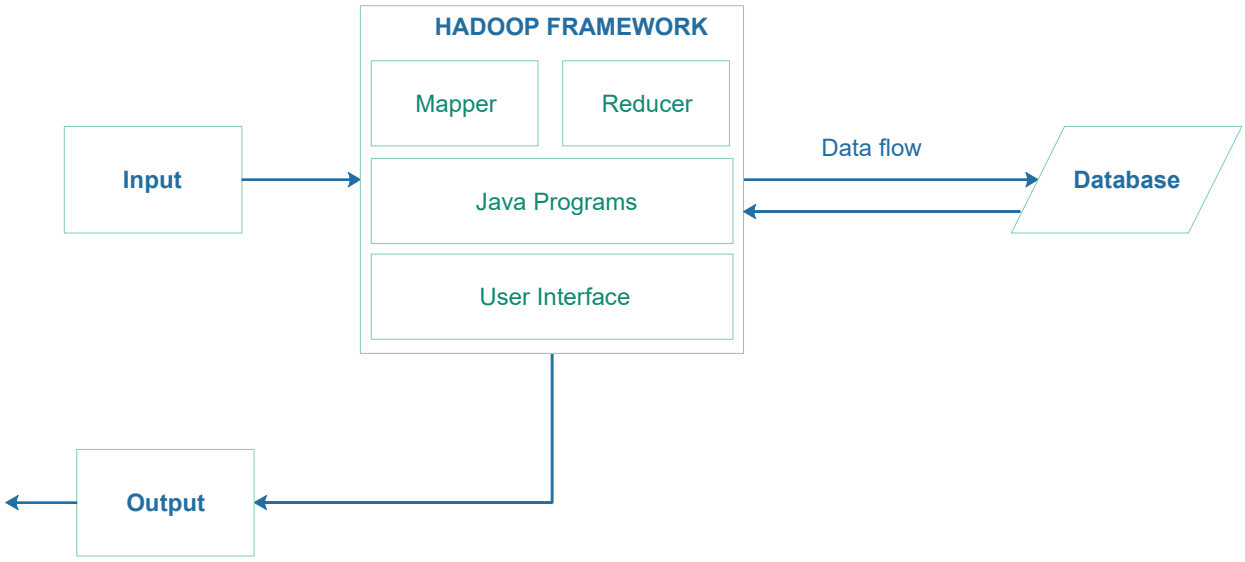

Fig 1. Product Functionality

3.2.2.1. Front End functionality

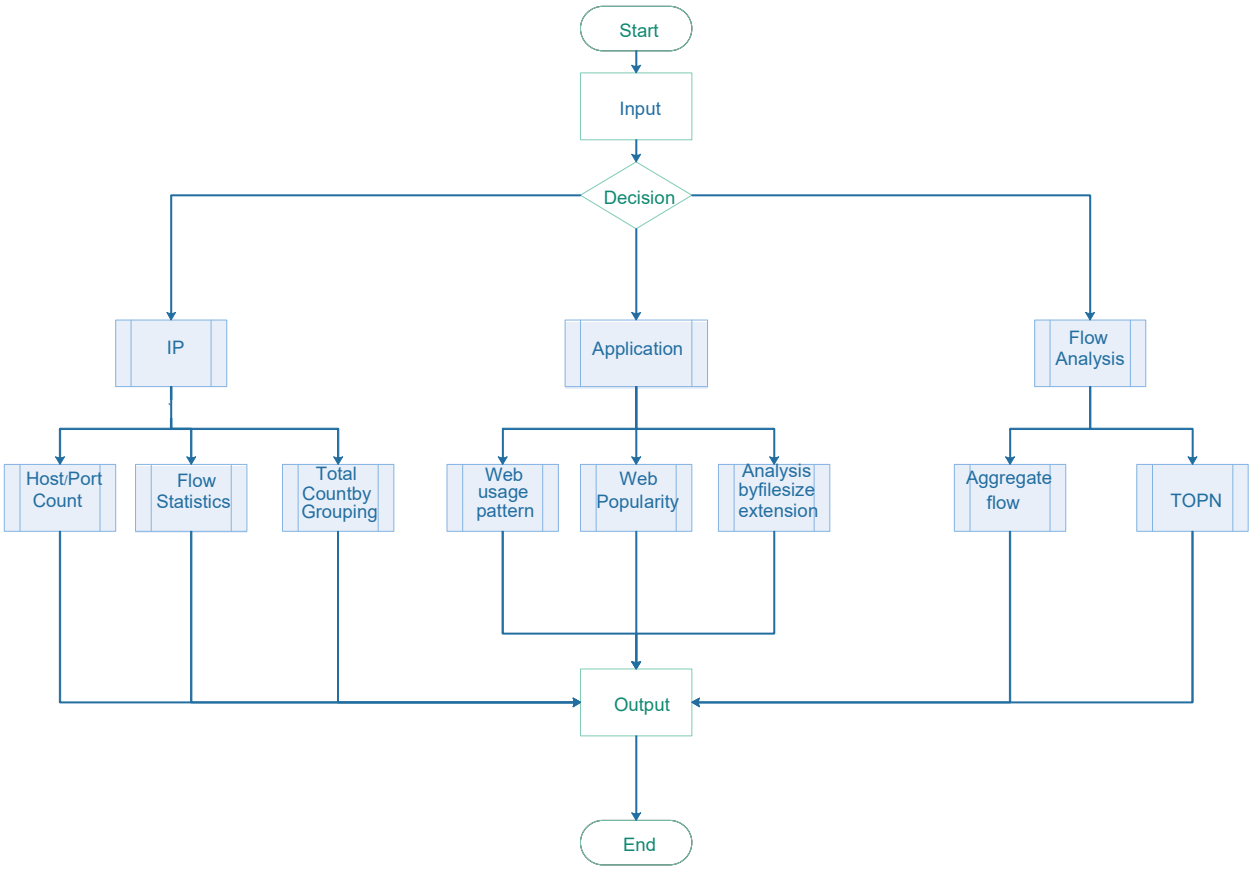

Fig 2. Front End Functionality

The application of given project will be real time internet traffic analysis in high speed data networks. In Hadoop, text files are common as an input format. In Front End Functionality diagram we shown functionality. By using ip we can counts per host, total counting by grouping and flow statics in which we can finds most popular statics such as TCP flow.

Application analysis in which we can sort web URLs for user by using web usage pattern, web popularity in which we calculate the web popularity per web-site and we can identity whether data is image or video by using file size extension . Flow analysis in which we calculate most visited website Top N and also we aggregate flow to human readable format.

3.2.3.Users and Characteristics

Network Administrator: To monitor fault. To characterize network user behaviour and network usage.
ISP: This product help to increase the Quality of Service of Network.

3.2.4.Operating Environment
Software Requirements

i. To collect and trace packet data we require packet tracing tool which can monitor flow on high speed networks.

ii. To deploy Hadoop cluster we require hadoop stack software bundle.

iii. For MapReduce programs we require Eclipse IDE for java or Python 3.2 if we decide to do using python.

Hardware Requirements

There is no single hardware requirement set for installing Hadoop. but we will be using

1. Dual quad core CPUs
2. 8GB Memory
3. 80 GB Hard disk.

Operating Systems Requirements

The following operating systems are supported:

1. Red Hat Enterprise Linux (RHEL) v5.x or 6.x (64-bit)

2. CentOS v5.x or 6.x (64-bit)

3. Ubuntu 14.04 (64-bit)

Browser Requirements

You must have a machine capable of running a graphical browser to use this tool. The supported browsers are:

1. Windows (Vista, 7)

2. Internet Explorer 9.0 and higher

3. Firefox latest stable release

4. Safari latest stable release

5. Google Chrome latest stable release

3.2.5.DESIGN AND IMPLEMENTATION CONSTRAINTS

1. Requires RAM of all nodes in cluster should be in well condition.

2. Requires Multi-node cluster setup along with its maintenance.

3. It will not be applicable for very large volume of data.

4. Within 8 month of duration we have to complete this product.

5. MapReduce programs can be implemented using java or Python. We will be using Java as working team is comfortable with it, but at certain point if we feel to switch programing language we are going to do so.

Hadoop

Hadoop is an open-source framework that allows to store and process big data in a distributed environment 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.

Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes. In short, Hadoop framework is capabale enough to develop applications capable of running on clusters of computers and they could perform complete statistical analysis for a huge amounts of data.

Hadoop Architecture

Hadoop framework includes following four modules:

- **Hadoop Common:** These are Java libraries and utilities required by other Hadoop modules. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.
- **Hadoop YARN:** This is a framework for job scheduling and cluster resource management.
- **Hadoop Distributed File System (HDFS™):** A distributed file system that provides high-throughput access to application data.
- **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.

Installation of Hadoop

Installing Java

sudo apt-get update

sudo apt-get install default-jdk

Adding a dedicated Hadoop user

sudo addgroup hadoop

sudo adduser –ingroup hadoop hduser

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.

   sudo apt-get install ssh

Install Hadoop

tar xvzf hadoop-2.6.0.tar.gz

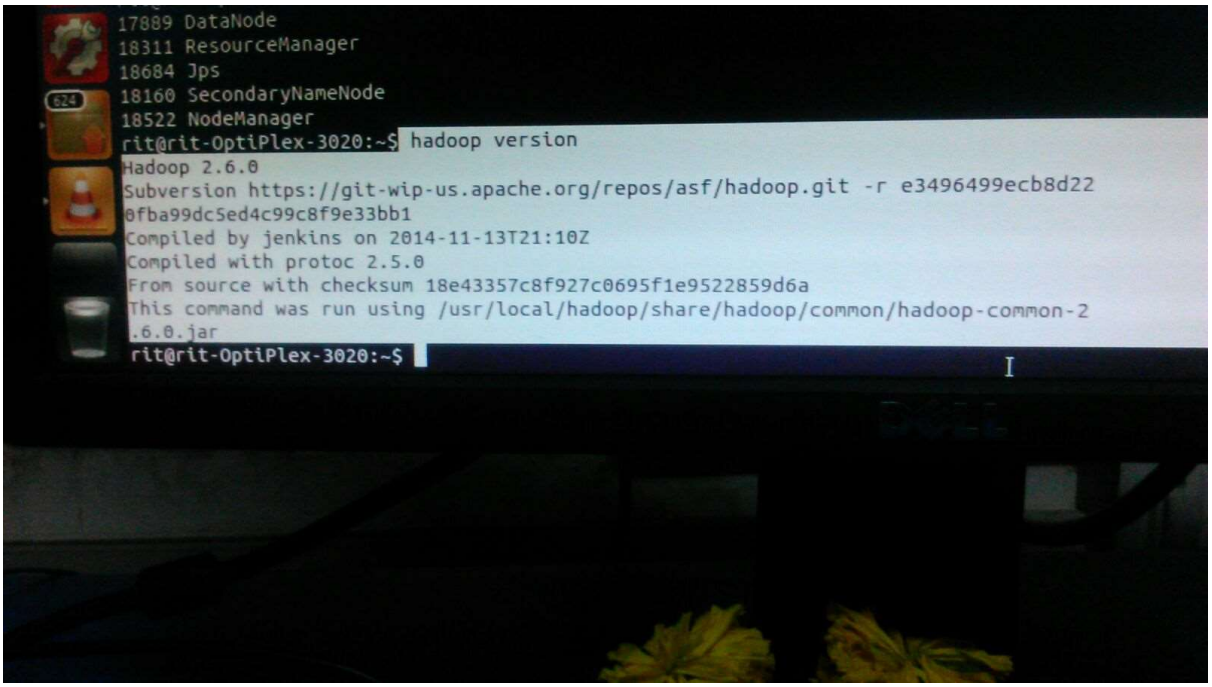

Fig 3 Extraction of hadoop-2.6.0.tar.gz.

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

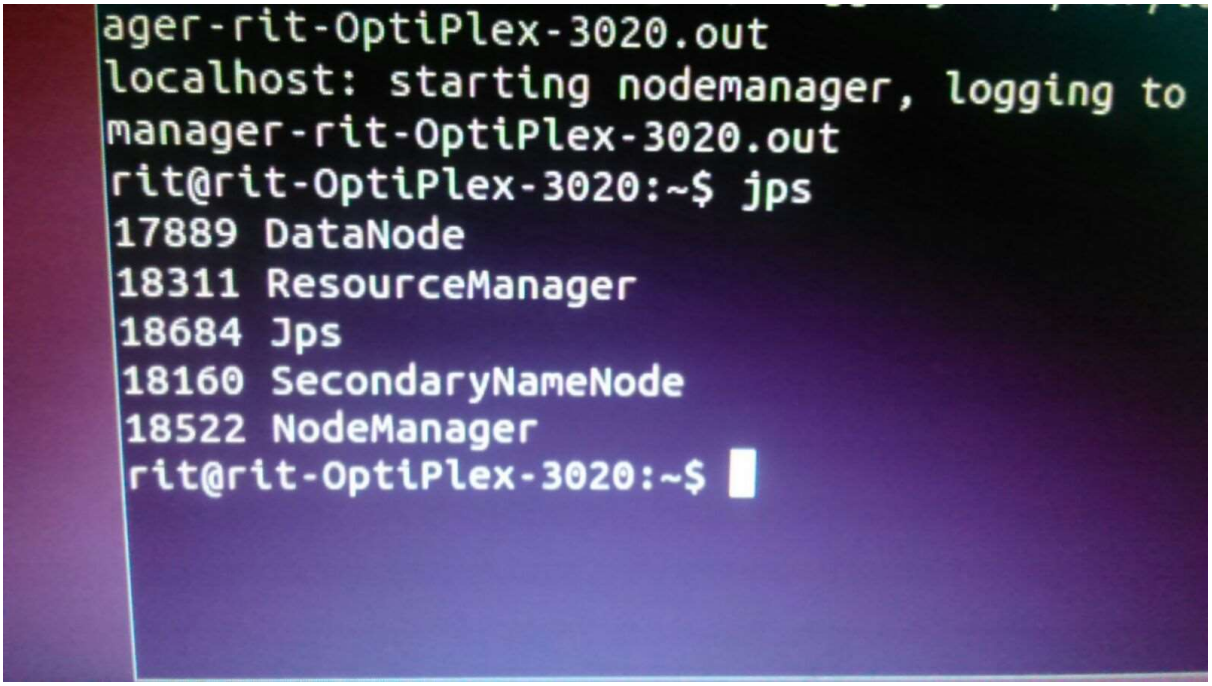

Fig 4 Installation of hadoop

MapReduce

Hadoop **MapReduce** is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

The term MapReduce actually refers to the following two different tasks that Hadoop programs perform:

- **The Map Task:** This is the first task, which takes input data and converts it into a set of data, where individual elements are broken down into tuples (key/value pairs).
- **The Reduce Task:** This task takes the output from a map task as input and combines those data tuples into a smaller set of tuples. The reduce task is always performed after the map task.

Typically both the input and the output are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

The MapReduce framework consists of a single master **JobTracker** and one slave **TaskTracker** per cluster-node. The master is responsible for resource management, tracking resource consumption/availability and scheduling the jobs component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves **TaskTracker** execute

the tasks as directed by the master and provide task-status information to the master periodically.

The **JobTracker** is a single point of failure for the Hadoop MapReduce service which means if JobTracker goes down, all running jobs are halted.

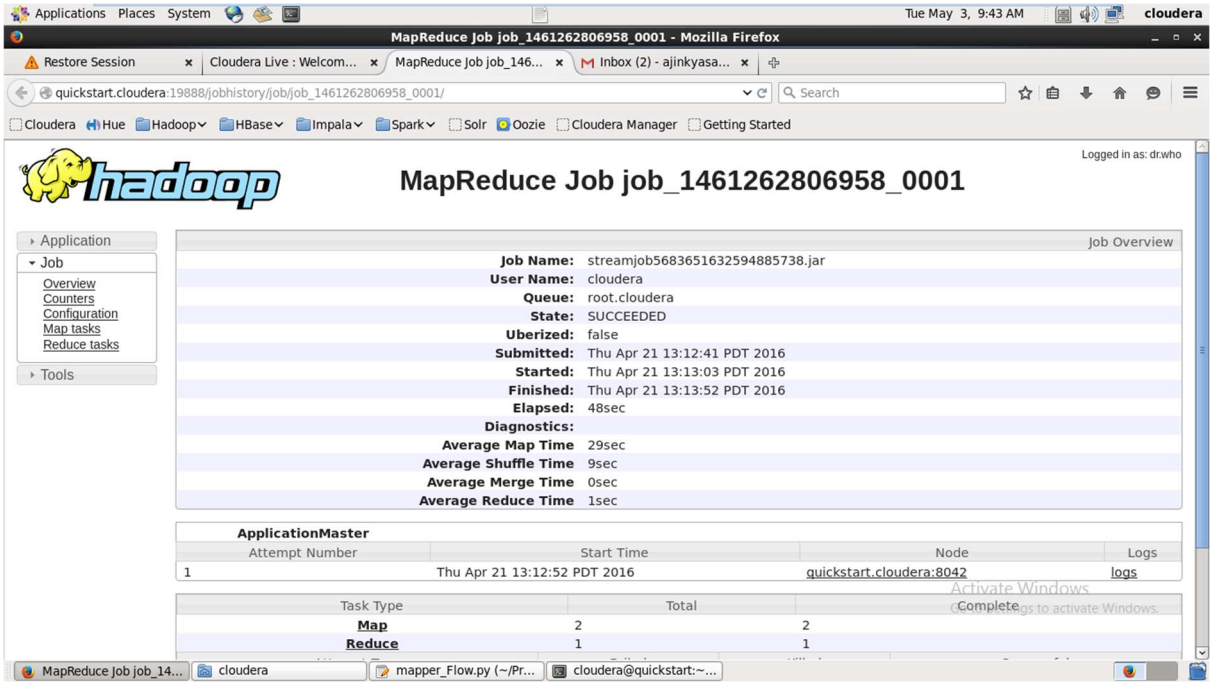

Fig 5 Mapreduce Job

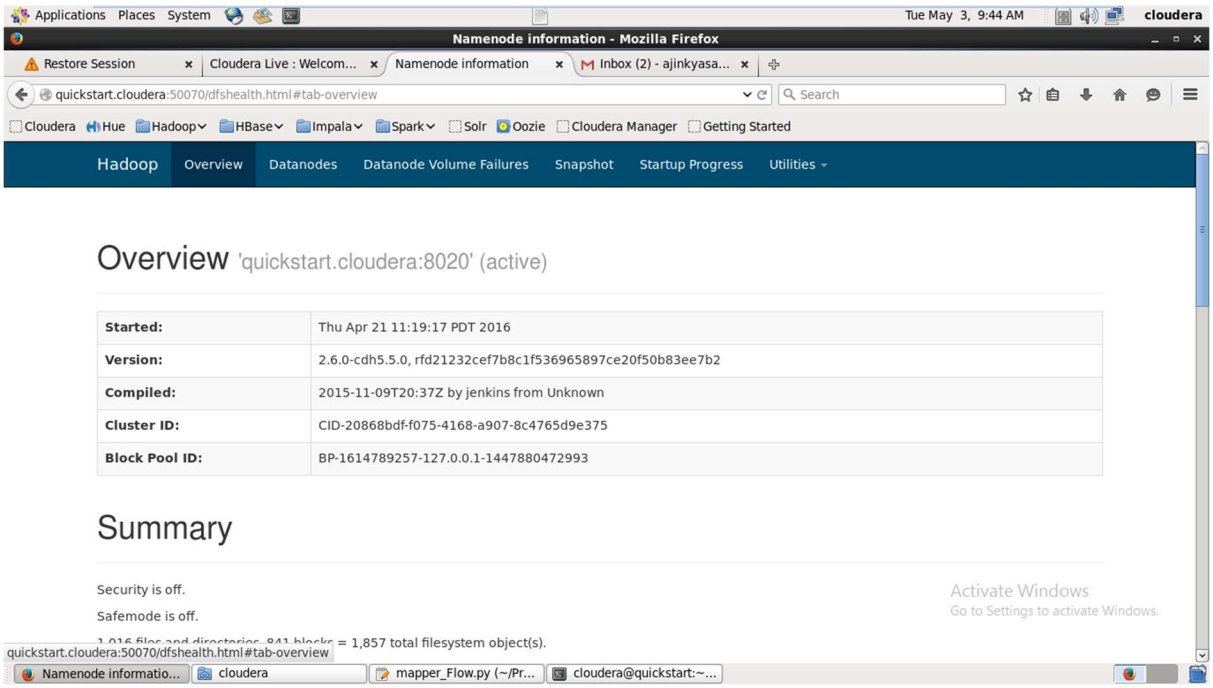

Fig 6 Namenode

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 large clusters (thousands of computers) of small computer machines in a reliable, fault-tolerant manner.

HDFS uses a master/slave architecture where master consists of a singleNameNode that manages the file system metadata and one or more slaveDataNodes that store the actual data.

A file in an HDFS namespace is split into several blocks and those blocks are stored in a set of DataNodes. The NameNode determines the mapping of blocks to the DataNodes. The DataNodes takes care of read and write operation with the file system. They also take care of block creation, deletion and replication based on instruction given by NameNode.

HDFS provides a shell like any other file system and a list of commands are available to interact with the file system.

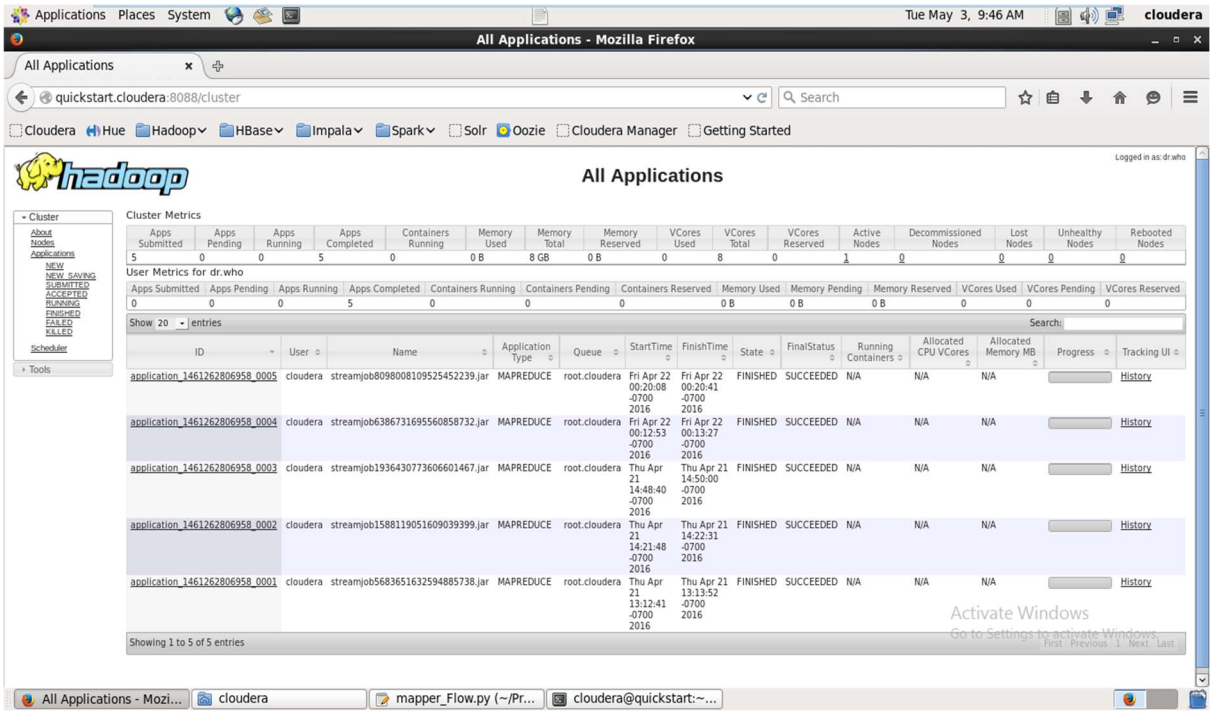

Fig 7 All application of hadoop

3.2.6. USER DOCUMENTATION

This is a simple analysis project to be used by network admin. However, there are various tasks to be conducted within this project. The system is divided in various modules,

which will perform various different functions. To understand the complexity of design, the users will be provided with a user manual, which will help to locate the different functions in the project.

3.2.7.ASSUMPTIONS AND DEPENDENCIES

Throughout of this project we are mainly focusing on effective analysis of the acquired data sets and statically presentation of the results.

3.3.Specific Requirement

3.3.1. External Interface Requirements

3.3.1.1. User Interfaces

- Help and Tooltips are available for easy understanding.
- Graphical interface is available for ease and convenience of the user.
- Analysis representation using pi chart.
- Tools strip menu is available for faster access of menus.

3.3.1.2. Hardware Interfaces

While it is possible to deploy all of Hadoop on a single host, this is appropriate only for initial evaluation. In general you should use at least three hosts: one master host and two slaves. This project's workloads are CPU bound and are characterized by the need of large number of CPUs and large amounts of memory to store in-process data. (This usage pattern is typical for natural language processing or HPCC workloads.)

3.3.1.3. Software Interfaces

Hadoop framework is based on the Java so for installing hadoop stack we need firstly install java on given machine. Hadoop is only supported by linux distributions so we will be using Ubuntu. If time permits we will try to interface wireshark or any packet snatcher with our project so as to automate fetching data for analysis purpose.

3.3.1.4. Communications Interfaces

Communication between nodes can be easily achieved by connecting all of your nodes to a Gigabit Ethernet switch. In order to use the spare socket for adding more CPUs in future, you can also consider using either a six or an eight core CPU. Hadoop uses SSH encryption for data and result sharing of the MapReduce jobs.

3.3.2. Functional Requirements

This project is divided into three major modules

1. Hadoop Cluster Implementation

2. Implementing Mapreduce programs for given properties i.e data used,IP specific analysis

3. Administrator UI generation along with report generation functionality.

3.3.3.Behaviour Requirements

3.3.3.1. Use Case View

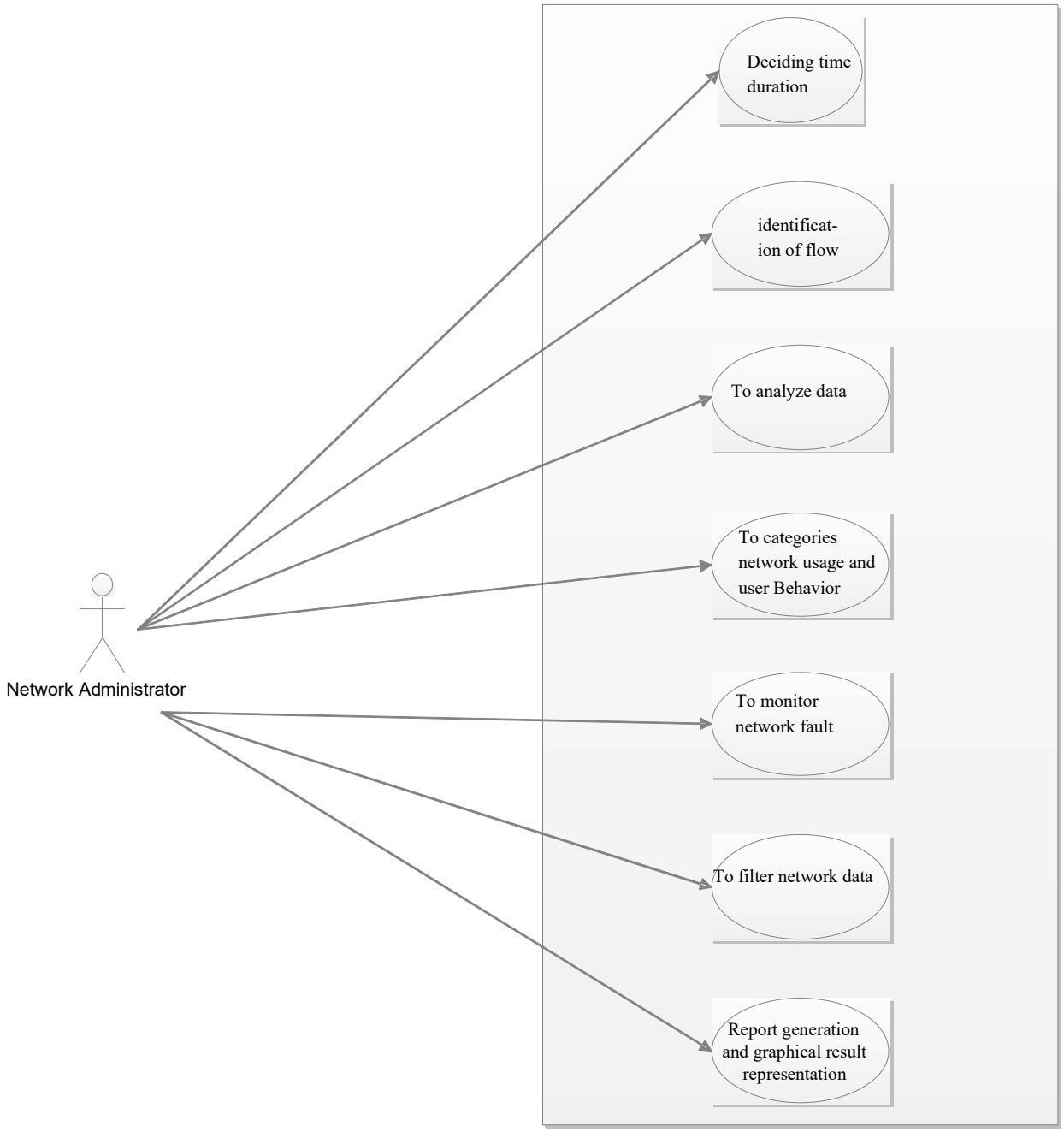

Fig 8. Use Case View

4. System Design& implementation details

4.1. System Architecture

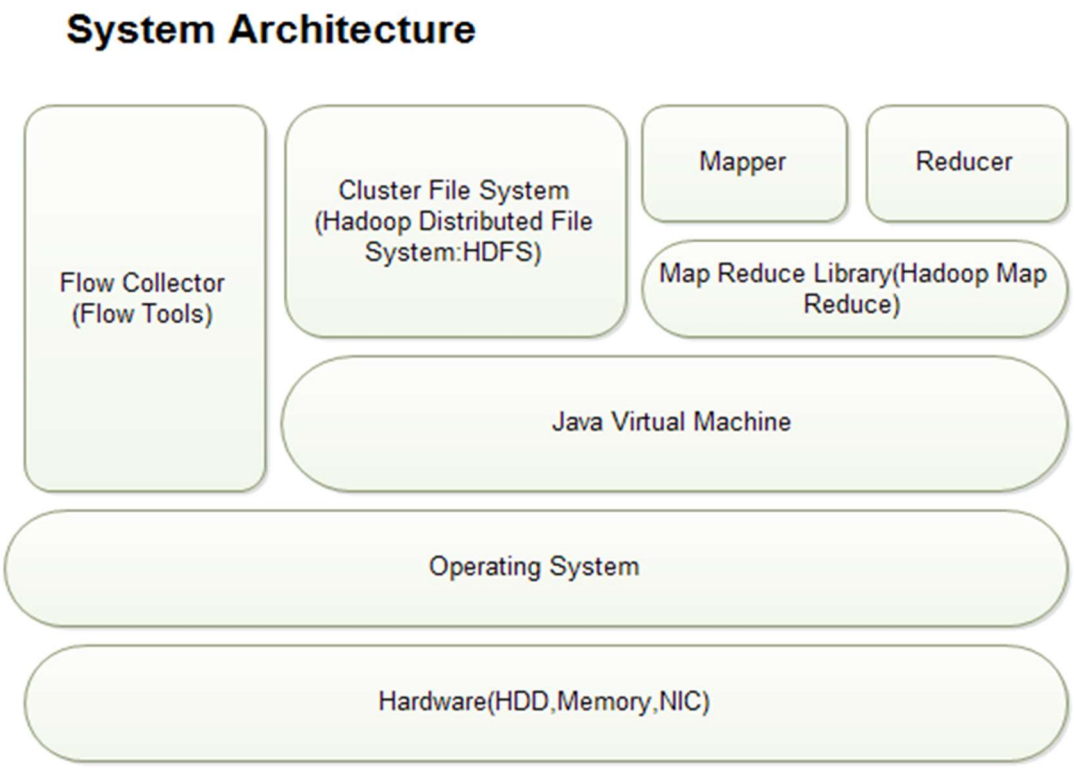

Fig 9. System Architecture[2]

4.2. Data Flow Diagram

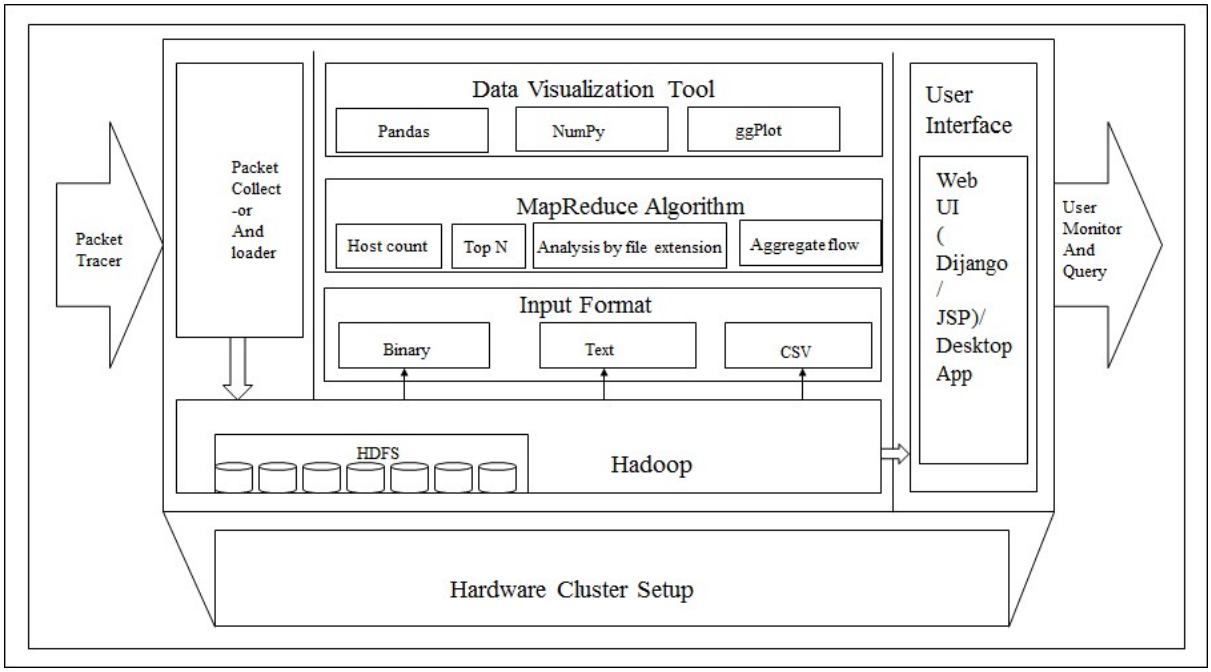

Fig 10 Data Flow Diagram[3]

In diagram we shown overview of system. We collect dataset by using packet tracing tool .We store dataset in HDFS (Hadoop Distributed File system). In Hadoop, text files are common as an input format. We use Mapreduce algorithm for analysis of data. In Mapreduce algorithm we use mapper class for maps and sorts data. The output of Mapper class is used as input by Reducer class, which searches matching pairs and reduces them.

The application of given project will be real time internet traffic analysis in high speed data networks. By using Mapreduce algorithm we finds Top N most visited website, counts per host, flow statics, Analysis by file extension and total counting by grouping. After analysis of data we use data visualization tool to represent resulted record.

Data visualization tool such as Pandas library, Numpy and ggplot. We use Pandas library for data analysis. Numpy use for computing data with python .We use ggplot for plotting graph . By using these library we represent Resulted record .We provide User friendly interface so user can easily interact with our application.

5. Coding /Code Templates

We have developed simple mapper and reducer programs in python. In mapper program, input file content is splitted into lines and line is splitted into words. Then each word is compared with pattern of regular expression. If match found count is set to 1.

In reducer program, it find out the matching words and add up count for similar words. The re package is used for regular expression and sys package is also imported.

5.1 Sample code of GUI:

```
from Tkinter import *
import Tkinter as tk
import tkFont
import tkMessageBox
import Tkinter,tkFileDialog
import subprocess as sub
import os
win = Tk()



name2 = """IP Analysis"""
wip = Label(win,padx = 10,text=name2,state=DISABLED,font="bold")
wip.grid(row=6, column=0)


Bhostcounttxt = Button(win,text="      Host Count      ",state=DISABLED,command=lambda:
sub.call('/home/a/job.sh'))
Bhostcounttxt.grid(row=7, column=0)
Bflowtxt = Button(win,text=" Flow Statistics ",state=DISABLED)
Bflowtxt.grid(row=7, column=2)
Bgrpbytxt = Button(win,text="     Count By Grouping     ",state=DISABLED)
Bgrpbytxt.grid(row=7, column=7)


#application analysis
name3 = """Application Analysis"""
wapp = Label(win,padx = 10,text=name3,state=DISABLED,font="bold")
wapp.grid(row=8, column=0)
```

```
Bwebusetxt = Button(win,text="Web Usage Pattern",state=DISABLED)
Bwebusetxt.grid(row=9, column=0)
Bwebpoptxt = Button(win,text="Web Popularity",state=DISABLED)
Bwebpoptxt.grid(row=9, column=2)
Bfileextxt = Button(win,text="Analysis By File Extension",state=DISABLED)
Bfileextxt.grid(row=9, column=7)

#flow analysis
name4 = """Flow Analysis"""
wflow = Label(win,padx = 10,text=name4,state=DISABLED,font="bold")
wflow.grid(row=10, column=0)

BAgrflowtxt = Button(win,text="   Aggregate Flow   ",state=DISABLED)
BAgrflowtxt.grid(row=11, column=0)
Btopntxt = Button(win,text="       TOP N       ",state=DISABLED)
Btopntxt.grid(row=11, column=2)

#LABLE
name = """Internet Traffic Analysis"""
w1 = Label(win,padx = 10,text=name)
w1.grid(row=0, column=0)
w1.grid(row=0,column=0,columnspan=2,pady=5,padx=5)

#ENTRY
name1 = """Input"""
w2 = Label(win,padx = 10,text=name1)
```

```
w2.grid(row=1, column=0)
w2.grid(row=1,column=0,columnspan=2,pady=5,padx=5)


v = StringVar()
e = Entry(win,textvariable=v)
e.grid(row=2, column=0)
e.grid(row=2,column=0,columnspan=2,pady=5,padx=5)


#File Browser
def file_browser():
        file = tkFileDialog.askopenfile(parent=win,initialdir='/home/          a',title='Choose a
file')
        path=file.name
        print(path)
        v.set(path)
        e.configure(state='disable')
#BROWZ BUTTON
b3 = Button(win,text="BROWZ",command=file_browser)
b3.grid(row=2, column=2)




#Second PART


#DDLIST
#name1 = """ITA"""
```

```
#var = tk.StringVar(win)
#var.set('IP')
#lst1 = ['IP','Application','Flow analysis']
#w3 =OptionMenu(win, var, *lst1)
#w3.grid(row=3, column=0)



def onselect(evt):
        print lb1.curselection()
        w = evt.widget
        index = int(w.curselection()[0])
        value = w.get(index)
        print 'You selected item %d: "%s"' % (index, value)
    if value == 'IP Analysis':
                Bhostcounttxt['state']='normal'
                Bflowtxt['state']='normal'
                Bgrpbytxt['state']='normal'
                wip['state']='normal'

                Bwebusetxt['state']='disabled'
                Bwebpoptxt['state']='disabled'
                Bfileextxt['state']='disabled'
                BAgrflowtxt['state']='disabled'
                Btopntxt['state']='disabled'
                wapp['state']='disabled'
                wflow['state']='disabled'
```

```
if value == 'Application Analysis':
        Bwebusetxt['state']='normal'
        Bwebpoptxt['state']='normal'
        Bfileextxt['state']='normal'
        wapp['state']='normal'
        Bhostcounttxt['state']='disabled'
        Bflowtxt['state']='disabled'
        Bgrpbytxt['state']='disabled'
        BAgrflowtxt['state']='disabled'
        Btopntxt['state']='disabled'
        wflow['state']='disabled'
        wip['state']='disabled'
        if value == 'Flow Analysis':
        BAgrflowtxt['state']='normal'
        Btopntxt['state']='normal'
        wflow['state']='normal'

        Bhostcounttxt['state']='disabled'
        Bflowtxt['state']='disabled'
        Bgrpbytxt['state']='disabled'
        Bwebusetxt['state']='disabled'
        Bwebpoptxt['state']='disabled'
        Bfileextxt['state']='disabled'
        wip['state']='disabled'
        wapp['state']='disabled'
        #BAgrflowtx = Button(win,text="Aggregate Flow",state=NORMAL)
```

```
#Btopntxt = Button(win,text="TOP N",state=NORMAL)


#lb1= tk.Listbox(height=3)
lb1=Listbox(win, height=6);
lb1.insert(1,"IP Analysis")
lb1.insert(2,"")
lb1.insert(3,"Application Analysis")
lb1.insert(4,"")
lb1.insert(5,"Flow Analysis")
lb1.insert(6,"")
lb1.grid(row=3, column=0)
lb1.bind('<<ListboxSelect>>', onselect)
#lb1.pack();
#OK BUTTON
#b3 = Button(win,text="OK")
#b3.grid(row=3, column=1)

#EXECUTE BUTTON
#b4 = Button(win, text="Execute!",)
#b4.grid(row=0,column=4,rowspan=3,sticky=NSEW)

win.mainloop()
```

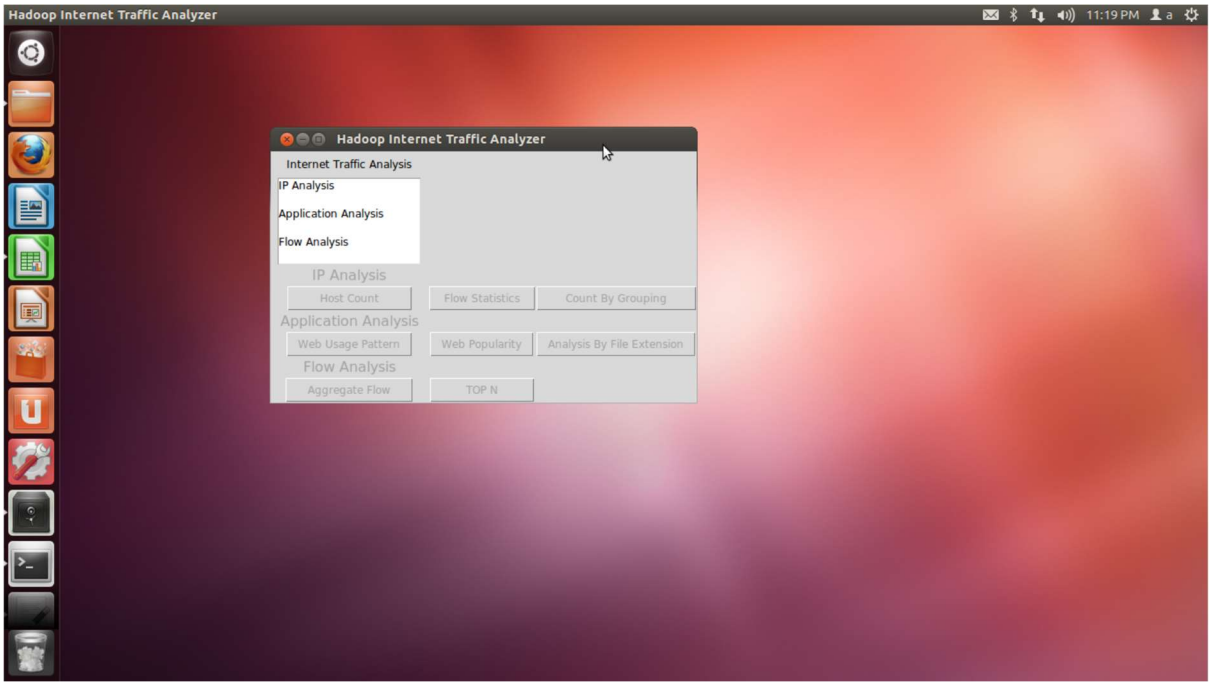

Fig. 11. Main Graphical User Interface

In figure we shown our main graphical user interface. In that figure we given three option to user such as IP analysis, Application analysis and flow analysis. In IP analysis we get host count, flow statics and count by grouping. In application analysis we get Web usages pattern , web popularity and analysis by file extension. And in Flow analysis we get Aggregate flow and TOP N of web.

5.2 sample Code of bar graph:

```
import sys

from matplotlib import pyplot as plt

data=[]

p=""

c=0

count=[]

protocol=[]

value=[]



for line in sys.stdin:

    line = line.strip()
```

```
    line=line.strip("\n")
    print line
    # remove leading and trailing whitespace
    words = line.split()
    print words
    for x in words:
        data.append(x)
print data
for x in data:
        if (data.index(x))%2==0:
                protocol.append(data[data.index(x)])
                c+=10
                count.append(c)
        else:
                value.append(int(data[data.index(x)]))

print count
print"\nEnd"
print protocol
print"\nEnd"
print value

plt.bar(count,value,align='center')
plt.xticks(count,protocol)
plt.title('Prorocol Analysis')
plt.ylabel('Usage')
plt.xlabel('Protocols')
plt.show()
```

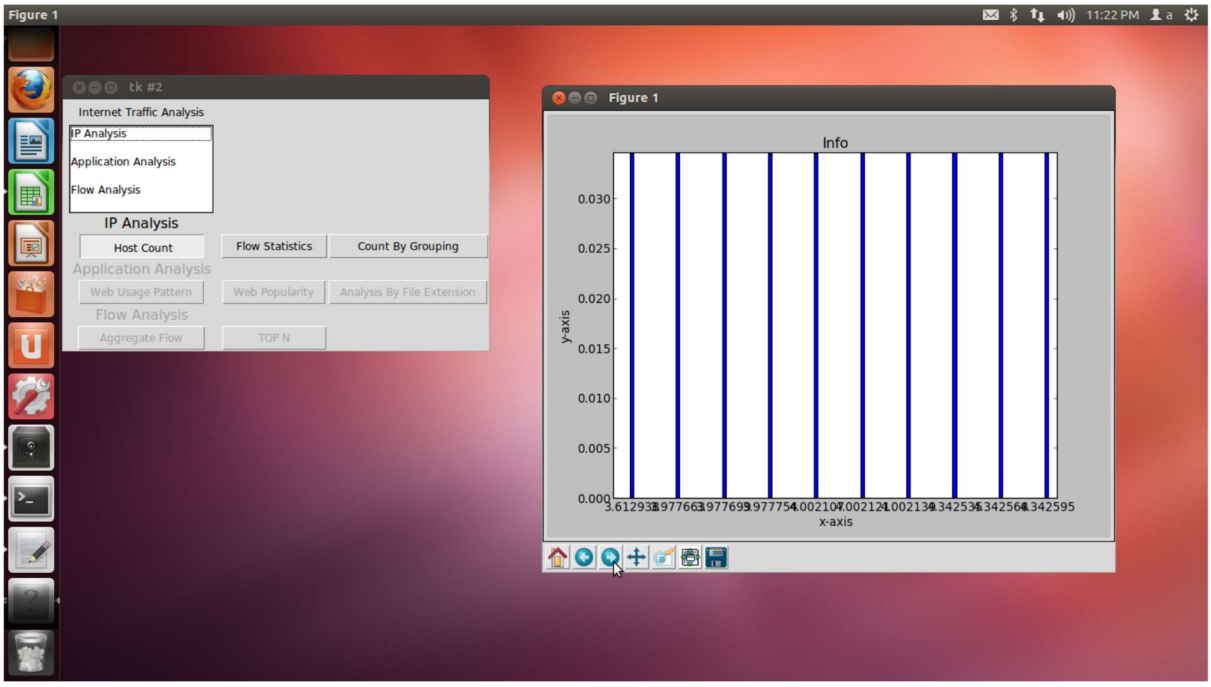

Fig. 12. Bar graph of Host Count

In this bar graph many functionalities are provided such as zooming, save image as, print, forward and backward .In zooming we can zoom in and zoom out pie chart for better understanding. We can also save this pie chart with extension such as .pdf, .png and .jpg which help in future.

5.3 Sample code of mapper flow

```
#!/usr/bin/env python

import sys

p=0

src_ip=[]

dest_ip=[]

# input comes from STDIN (standard input)

for line in sys.stdin:

    # remove leading and trailing whitespace

    line = line.strip()

    # split the line into words

    words = line.split()

    # increase counters
```

```
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
            #print words
            try:
                    ip_addr=words[words.index(word)+3]
                    address=words[words.index(word)+1]
                    if word=="Source" and address=="Address":
                            src_ip.append(ip_addr)
                    if word=="Destination" and address=="Address":
                            dest_ip.append(ip_addr)

            except:
                    p+1
for x,y in zip(src_ip,dest_ip):
            p=x+"->"+y
            print '%s\t%s' % (p , 1)
```

5.4 Sample code of Reducer of flow

```
#!/usr/bin/env python
#reducer For Flow Analysis
from operator import itemgetter
import sys
wordcount={}
total_entries=0
```

```
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    word, count = line.split('\t', 1)
    if word not in wordcount:
        wordcount[word] = 1
            total_entries+=1
    else:
        wordcount[word] += 1
            total_entries+=1


for dummy_word,dummy_count in wordcount.items():
    print dummy_word+"\t"+str(dummy_count)+"\t"
#print "Total Packets: "total_entries
```

5.5 Sample Mapper code of protocol

```
#!/usr/bin/env python
import sys
p=0
# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
```

```
# write the results to STDOUT (standard output);
# what we output here will be the input for the
# Reduce step, i.e. the input for reducer.py
#
# tab-delimited; the trivial word count is 1
   #print words
   try:
            protocol_no=words[words.index(word)+2]
            if word=="Protocol" and protocol_no=="6":
                     print '%s\t%s' % ("TCP" , 1)
            if word=="Protocol" and protocol_no=="17":
                     print '%s\t%s' % ("UDP" , 1)
            if word=="Protocol" and protocol_no=="1":
                     print '%s\t%s' % ("ICMP" , 1)
   except:
            p+1
```

5.6 Sample code of Protocol:

```
import socket, sys
from struct import *

fo = open("foo.txt", "wb")

#Convert a string of 6 characters of ethernet address into a dash separated hex string
def eth_addr (a) :
  b = "%.2x:%.2x:%.2x:%.2x:%.2x:%.2x" % (ord(a[0]) , ord(a[1]) , ord(a[2]), ord(a[3]), ord(a[4]) , ord(a[5]))
  return b
```

```
#create a AF_PACKET type raw socket (thats basically packet level)
#define ETH_P_ALL    0x0003          /* Every packet (be careful!!!) */
try:
    s = socket.socket( socket.AF_PACKET , socket.SOCK_RAW , socket.ntohs(0x0003))
except socket.error , msg:
    print 'Socket could not be created. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]
    sys.exit()

# receive a packet
while True:

    packet = s.recvfrom(65565)

    #packet string from tuple
    packet = packet[0]

    #parse ethernet header
    eth_length = 14

    eth_header = packet[:eth_length]
    eth = unpack('!6s6sH' , eth_header)
    eth_protocol = socket.ntohs(eth[2])
    every_packet_data='\n\n\nDestination MAC : ' + eth_addr(packet[0:6]) + ' Source MAC : '
+ eth_addr(packet[6:12]) + ' Protocol : ' + str(eth_protocol)+"\n"
    print every_packet_data
    fo.write(every_packet_data)
```

```
    #Parse IP packets, IP Protocol number = 8
    if eth_protocol == 8 :
        #Parse IP header
        #take first 20 characters for the ip header
        ip_header = packet[eth_length:20+eth_length]

        #now unpack them :)
        iph = unpack('!BBHHHBBH4s4s' , ip_header)

        version_ihl = iph[0]
        version = version_ihl >> 4
        ihl = version_ihl & 0xF

        iph_length = ihl * 4

        ttl = iph[5]
        protocol = iph[6]
        s_addr = socket.inet_ntoa(iph[8]);
        d_addr = socket.inet_ntoa(iph[9]);
            eth_packet_data='Version : ' + str(version) + ' IP Header Length : ' + str(ihl) + ' TTL :
' + str(ttl) + ' Protocol : ' + str(protocol) + ' Source Address : ' + str(s_addr) + ' Destination
Address : ' + str(d_addr)+"\n"
        print eth_packet_data
            fo.write(eth_packet_data)
        #TCP protocol
        if protocol == 6 :
            t = iph_length + eth_length
            tcp_header = packet[t:t+20]
```

```
        #now unpack them :)
        tcph = unpack('!HHLLBBHHH' , tcp_header)

        source_port = tcph[0]
        dest_port = tcph[1]
        sequence = tcph[2]
        acknowledgement = tcph[3]
        doff_reserved = tcph[4]
        tcph_length = doff_reserved >> 4
        tcp_data='Source Port : ' + str(source_port) + ' Dest Port : ' + str(dest_port) + '
Sequence Number : ' + str(sequence) + ' Acknowledgement : ' + str(acknowledgement) + '
TCP header length : ' + str(tcph_length)+"\n"
        print tcp_data
        fo.write(tcp_data)
        h_size = eth_length + iph_length + tcph_length * 4
        data_size = len(packet) - h_size

        #get data from the packet
        data = packet[h_size:]

        print 'Data : ' + data
            fo.write(data)
    #ICMP Packets
    elif protocol == 1 :
        u = iph_length + eth_length
        icmph_length = 4
        icmp_header = packet[u:u+4]
```

```
            #now unpack them :)
            icmph = unpack('!BBH' , icmp_header)

            icmp_type = icmph[0]
            code = icmph[1]
            checksum = icmph[2]
            icmp_data='Type : ' + str(icmp_type) + ' Code : ' + str(code) + ' Checksum : ' +
str(checksum)+"\n"
            print icmp_data
            fo.write(icmp_data)
            h_size = eth_length + iph_length + icmph_length
            data_size = len(packet) - h_size

            #get data from the packet
            data = packet[h_size:]
            fo.write(data)
            print 'Data : ' + data

        #UDP packets
        elif protocol == 17 :
            u = iph_length + eth_length
            udph_length = 8
            udp_header = packet[u:u+8]

            #now unpack them :)
            udph = unpack('!HHHH' , udp_header)
```

```
            source_port = udph[0]

            dest_port = udph[1]

            length = udph[2]

            checksum = udph[3]

            udp_data='Source Port : ' + str(source_port) + ' Dest Port : ' + str(dest_port) + ' Length
: ' + str(length) + ' Checksum : ' + str(checksum)+"\n"

            print udp_data

            fo.write(udp_data)

            h_size = eth_length + iph_length + udph_length

            data_size = len(packet) - h_size


            #get data from the packet

            data = packet[h_size:]


            print 'Data : ' + data

                fo.write(data)

        #some other IP packet like IGMP

        else :

            print 'Protocol other than TCP/UDP/ICMP'


        print

if __name__ == "__main__":

  main(sys.argv)
```

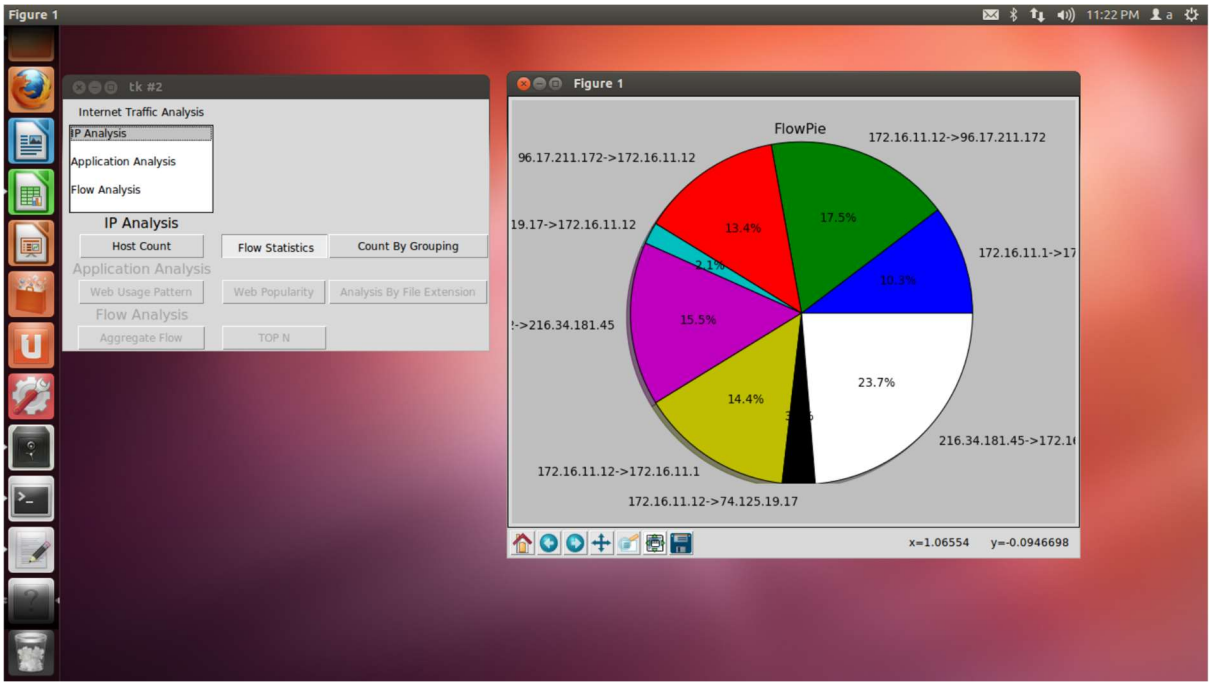

Fig. 13. Flow Analysis

In this figure we shown pie chart to represent host count. In which we used Numpy and pandas library for pie chart. Numpy library is used to calculate percentage of each ip. By using these library we represent user friendly pie chart with different color for different ip with label. Also we had given pie chart title , scale of X axis and Y axis.

In this pie chart many functionalities are provided such as zooming, save image as, print, forward and backward .In zooming we can zoom in and zoom out pie chart for better understanding. We can also save this pie chart with extension such as .pdf, .png and .jpg.

6. Applications

The application of given project will be real time internet traffic analysis in high speed data networks.

This kind of analysis will be milestone for ISP to improve their quality of services and detect faults.

This could be very useful for upcoming technologies such as reliance GeoNet and 4G, 5G,6G.

7. Conclusion

This project is based on analysis of the captured packet data which will ultimately suggest the major precautions to be taken by the network administrator for preventing the network failures, using Hadoop. This project is mainly focussed on analysis of large volume of data

using mapreduce approach in hadoop which reduces time for analysis. User can see the internet traffic data analysis based on different parameters to know about network usage and user behaviour in the network.
The Main focus of this project will be on the analysis of the extracted data and predicting the major precautions to the network administrator about the Internet traffic failure using **Hadoop**. We will propose necessary precautions to meet the Quality of Services of the internet in the campus. This project will also give a chance to researcher students to explore the Big Data Analytics using proposed Hadoop cluster in the Institute campus.

8. Future Work

In future we can extend scope by performing real-time traffic analysis in high-speed networks. Also traffic analysis will accept multiple file formats. It will also be possible to estimate the average packet retransmission because of TCP. Future purposed work is to make this application more responsive. Along with it our scope is to design and develop algorithms for future predictions using probabilistic algorithm such as Bayesian algorithm. We would like to use other hadoop based tools such as Hive, sqoop, flume etc. to improve system efficiency. There will be scope to perform real-time traffic analysis in high-speed networks. Also in future multiple file formats would be accepted as an input for traffic analysis. It will also be possible to calculate the average packet retransmission which is caused by TCP.

9.Bibliography / References

[1]"Hadoop Based Network Traffic Measurement with Scalable Data Processing In Fine Grained Networks" by D. Venkatesh, M. Prasanna, N. Madhava Reddy in May 2015

[2]YuanjunCai, Bin Wu, XinweiZhang, Min Luo and Jinzhao Su, "Flow identification and characteristics mining from internet traffic with hadoop" in 978-1-4799-4383-8/1 at IEEE 2014

[3]"Traffic Measurement and Analysis with Hadoop" by Dipti J. Suryawanshi, Prof. Mr. U. A. Mande published in international Journal of Engineering Research & Technology (October - 2013)

[4]"A Study on Scalable Internet Traffic Measurement and Analysis with Hadoop" by Yogesh V. Kadam, Prof. VaibhavDhore publisher International Journal Of Engineering And Computer Science ISSN:2319-7242 Volume 2 Issue 11,Nov 2013

[5]"An Internet traffic analysis method with MapReduce" by Youngseok Lee Chungnam Nat. Univ., Daejeon, South Korea ; Wonchul Kang ; Hyeongu Son published in Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP dated 19-23 April 2010

[6]J. Erman, M. Arlitt, and A. Mahanti, “Traffic classification usingclustering algorithms,” in Proceedings of the 2006 SIGCOMMworkshop on Mining network data. ACM, 2006, pp. 281–286.

[7]HANDLING INTERNET TRAFFIC USING BIG DATA ANALYTICS Ranganatha T.G1, Narayana H.M2, International Journal of Research In Science & Engineering e-ISSN: 2394-8299 Volume: 1 Special Issue: 2

[8]“MapReduce: Simplified Data Processing on Large Clusters” Jeffrey Dean and Sanjay Ghemawat ,Google, Inc.

[9] Mapreduce: Simplified Data Processing On Large Clusters Jeffrey Dean And Sanjay Ghemawat ,Google, Inc

[10] Hadoop, HTTP://HADOOP.APACHE.ORG

[11] Wireshark, HTTP://WWW.WIRESHARK.ORG

[12] Y. Lee And Y. Lee, A Hadoop-Based Packet Processing Tool, Tma, April 2011.

[13] cisco netflow, HTTP://WWW.CISCO.COM/WEB/GO/NETFLOW.

[14] Network Traffic Measurement: http://en.wikipedia.org/wiki/networktrafficmeasurement

[15] MapReduce framework: HTTP://D2I.INDIANA.EDU/HMR

[16] BigData : HTTP://VELVETCHAINSAW.COM/2012/07/20/THREEVS-OF-BIG-DATA-AS-APPLIED-CONFERENCES

[17] A Survey on Internet Traffic Measurement and Analysis Parekh Nilaykumar B., Prof. Alka J. Patel Department of CSE, Governmernt Engineering Collage,Modasa, Aravalli,Gujarat,India

[18] Cloudera:www.cloudera.com/

[19] Hadoop streaming: https://hadoop.apache.org/docs/r1.2.1/streaming.html

[20] Matplotlib: matplotlib.org

[21] ggplot:yhat.org

[22] tkinter: wiki.python.org/tkinter

10. Appendix A

Big Data: Big data is needed where volume of data is beyond capacity of regularly used soft wares to manage data.
Hadoop: an open-source distributed computing framework implemented in Java.
Hive: It is data warehousing tool for providing agile and elastic traffic analysis framework.
Internet Traffic analysis: To satisfy demands for the deep analysis of ever-growing Internet traffic data
MapReduce: It is a divide and conquer approach to solve the analysis of big data provided by the Hadoop framework