ARUN VINUD SIVASUBRAMANIAN S
31326743 arunvinud@gmail.com

IMPLENTATION OF ID3 DECISION TREE IN HADOOP AND APACHE MAP REDUCE IN AMAZON AWS CLOUD SERVICE

Introduction:

Hadoop map reduce is used in the field of big data analysis. It basically consist of two parts a mapper and a reducer. The job of the mapper in simple words is to assign a integer to each of the input words and the job of the reducer is to remove the words which are not unique and add those integers and writes the output. The file system used by hadoop is called HDFS or Hadoop distributed file system. The Hadoop distributed file system (HDFS) is a distributed, scalable, and portable file-system written in Java for the Hadoop framework .HDFS stores large files (typically in the range of gigabytes to tera bytes) across multiple machines. It achieves reliability by replicating the data across multiple hosts.

Iterative Dichotomiser 3 Algorithm:

In decision tree learning,ID3 is an algorithm invented by Rous Quinlan used to generate a decision tree from a dataset
The ID3 algorithm begins with the original set S as the root node. On each iteration of the algorithm, it iterates through every unused attribute of the set S and calculates the entropy H(S) (or information gain IG(A) of that attribute. It then selects the attribute which has the smallest entropy (or largest information gain) value. The set S is then split by the selected attribute (e.g. age < 50, 50 <= age < 100, age >= 100) to produce subsets of the data. The algorithm continues to recurse on each subset, considering only attributes never selected before. Recursion on a subset may stop in one of these cases:

•every element in the subset belongs to the same class (+ or -), then the node is turned into a leaf and labelled with the class of the examples
•there are no more attributes to be selected, but the examples still do not belong to the same class (some are + and some are -), then the node is turned into a leaf and labeled with the most common class of the examples in the subset
•there are no examples in the subset, this happens when no example in the parent set was found to be matching a specific value of the selected attribute,

for example if there was no example with age >= 100. Then a leaf is created, and labelled with the most common class of the examples in the parent set.

Hadoop and Recursion:

ID3 can be implemented in a classic recursive fashion but the problem is hadoop doesn't support recursion. It doesn't support recursion to overcome the bottleneck of java heap size. Since all the recursive calls are stored in heaps and hadoop doesn't store those results recursion doesn't work in hadoop. In other words each map reduce job works independently.

Iterative Implementation of ID3 in Hadoop:

Since Hadoop doesn't support recursion we planned on implementing ID3 iteratively. The number of iteration is directly proportional to the number of the columns in the input dataset.

Each iteration calls a map reduce job and it writes the count of values matching a given criteria. These iteration continues till there are no columns to split or the selected column has only one decision like Yes or No.

Amazon EC2 Cluster Setup:

Before implementing ID3 we should setup a multi node hadoop cluster which runs in master-slave fashion. The job of the master node(system) is to allocate and initiate jobs in slave systems in other words it acts as a task tracker. It also merges the output records of the slave systems and produce the final output.

Configuration of master and slave systems:

Operating System:Ubuntu 64 bit

Hadoop version:1.2.1

Java version:Java 7 SDK

Web HDFS:Enabled

Memory Limit:5GB(Amazon Free Tier)

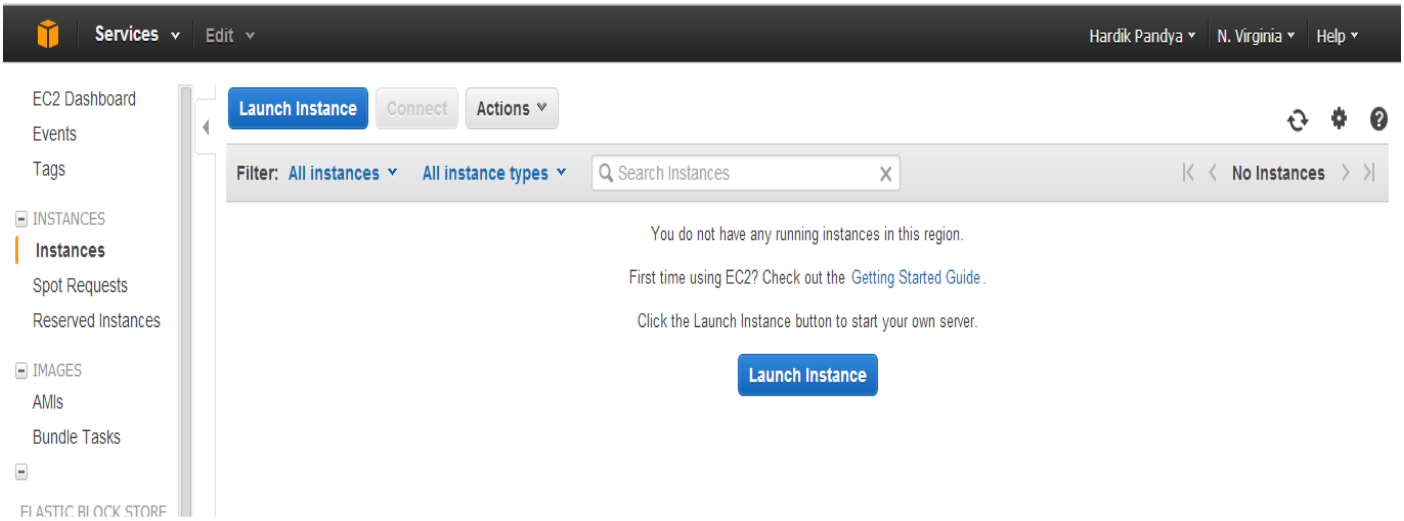
Total Online time:750 hrs

Authentication:SHA-1 Encryption

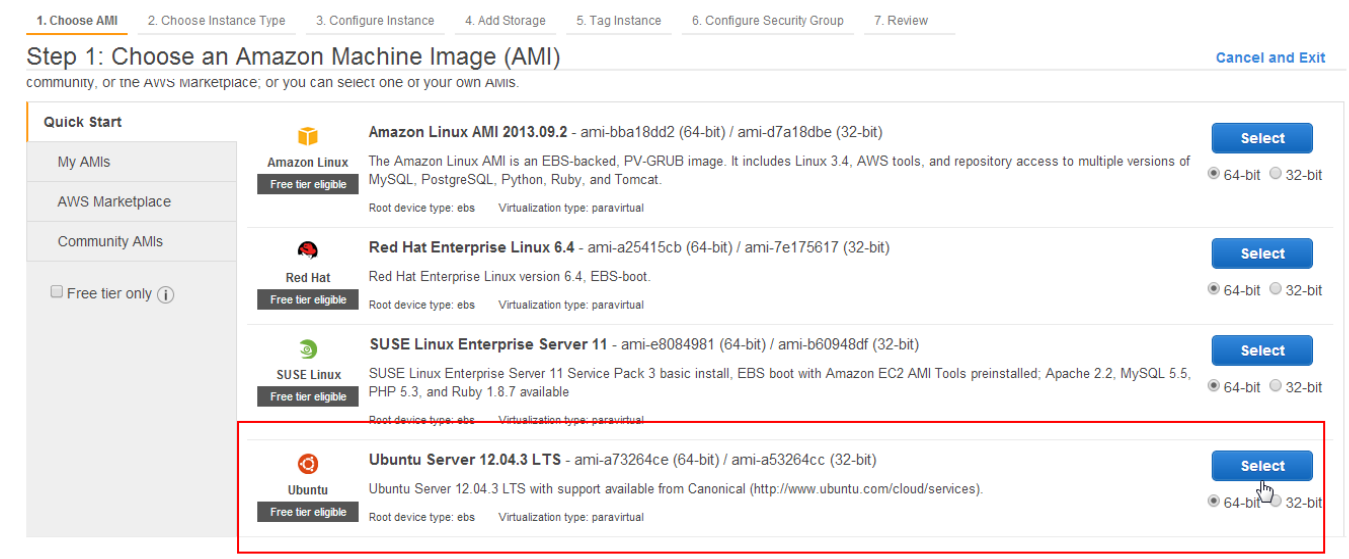
Some Screen shots on creation of Instances in Amazon EC2 interface:

Get Amazon AWS Account

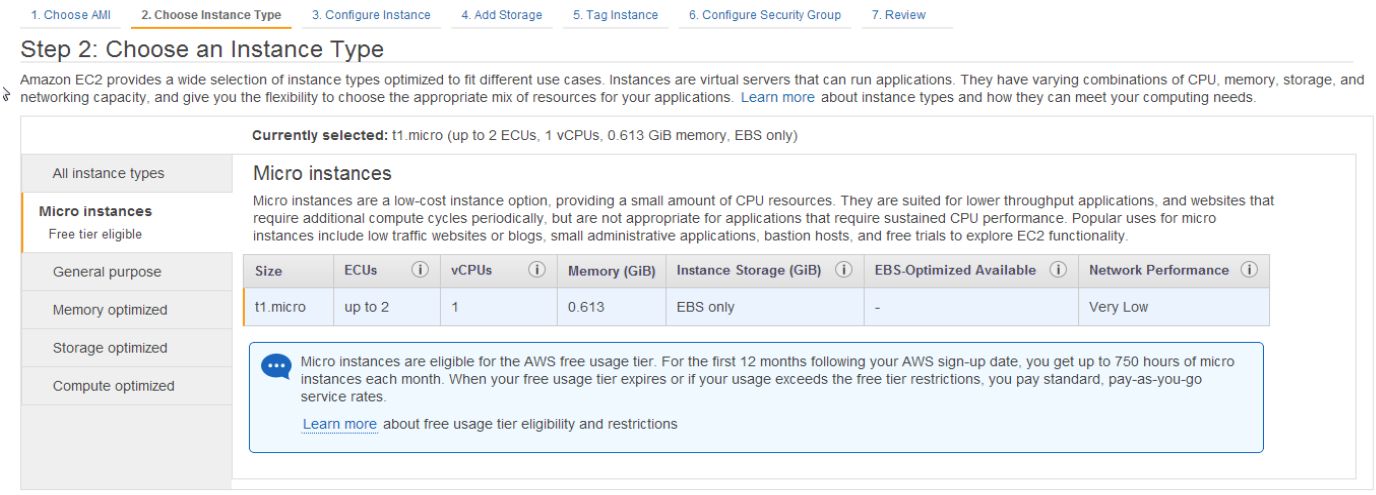
Launch Instance



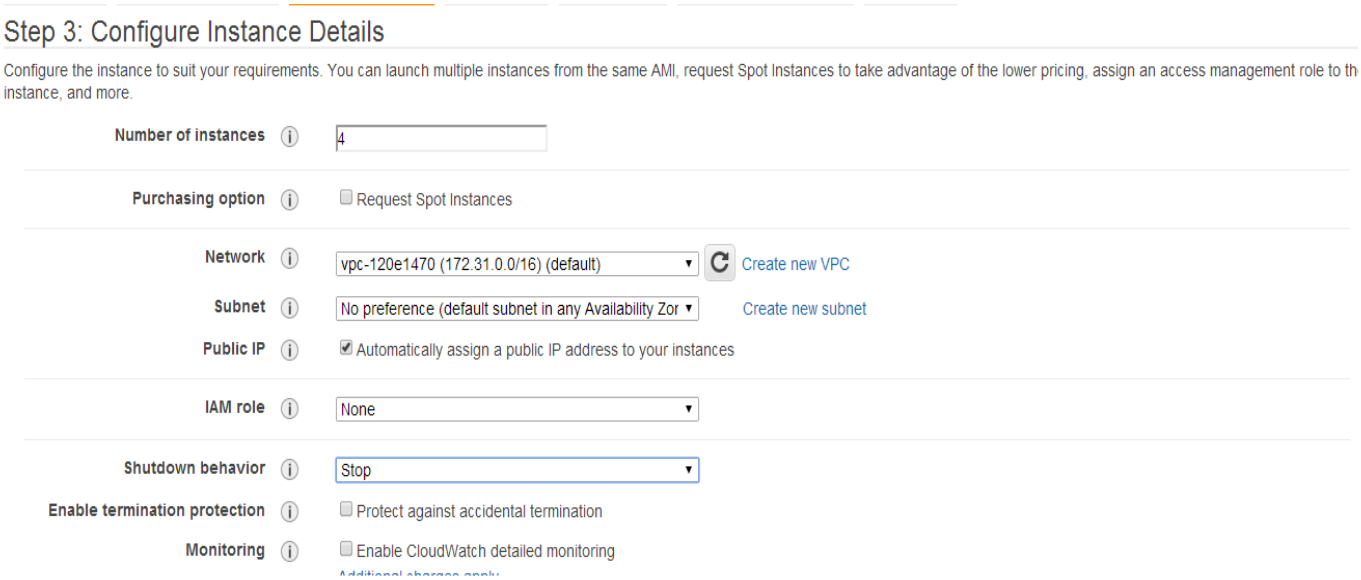
Select AMI



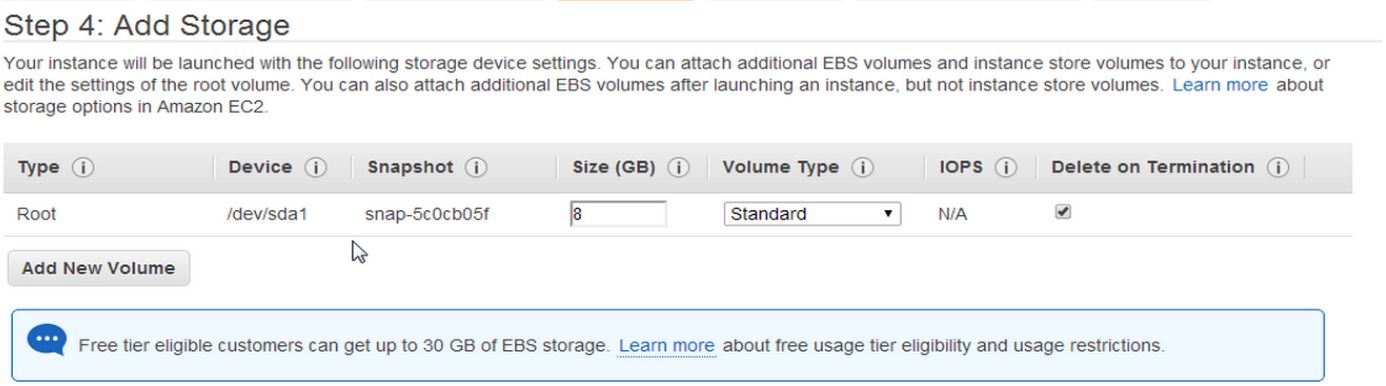
Select Instance Type



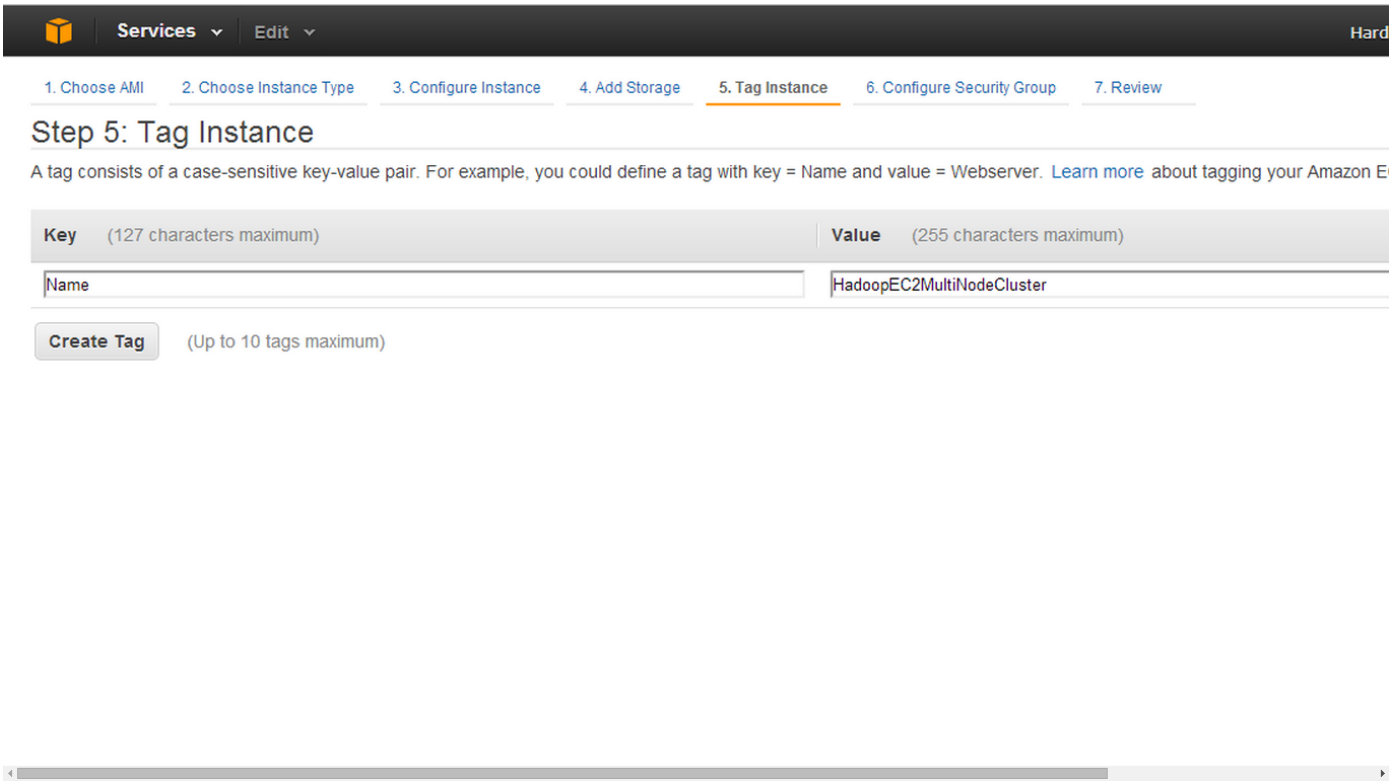
Configure Number of Instances(We are creating 4 instances(clusters))



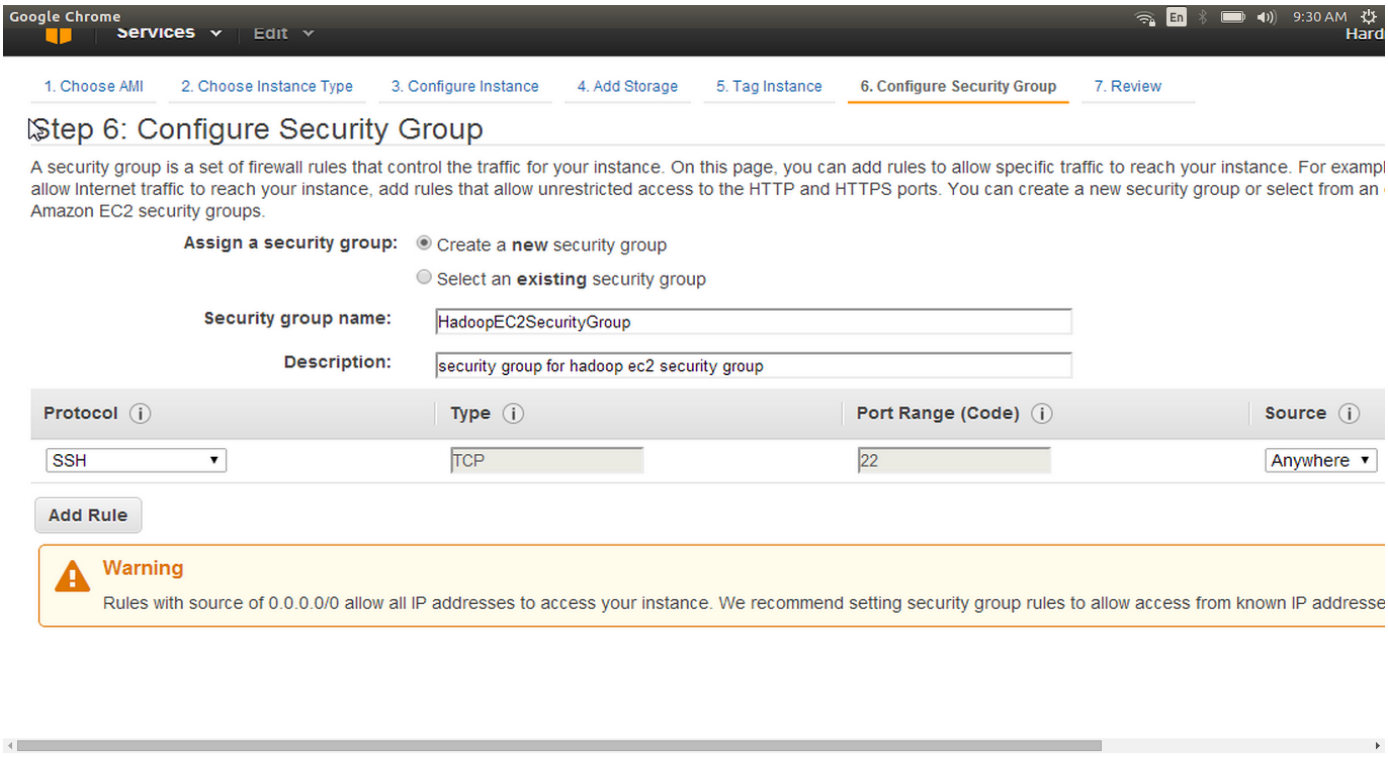
Add Storage(We are adding 8GB)



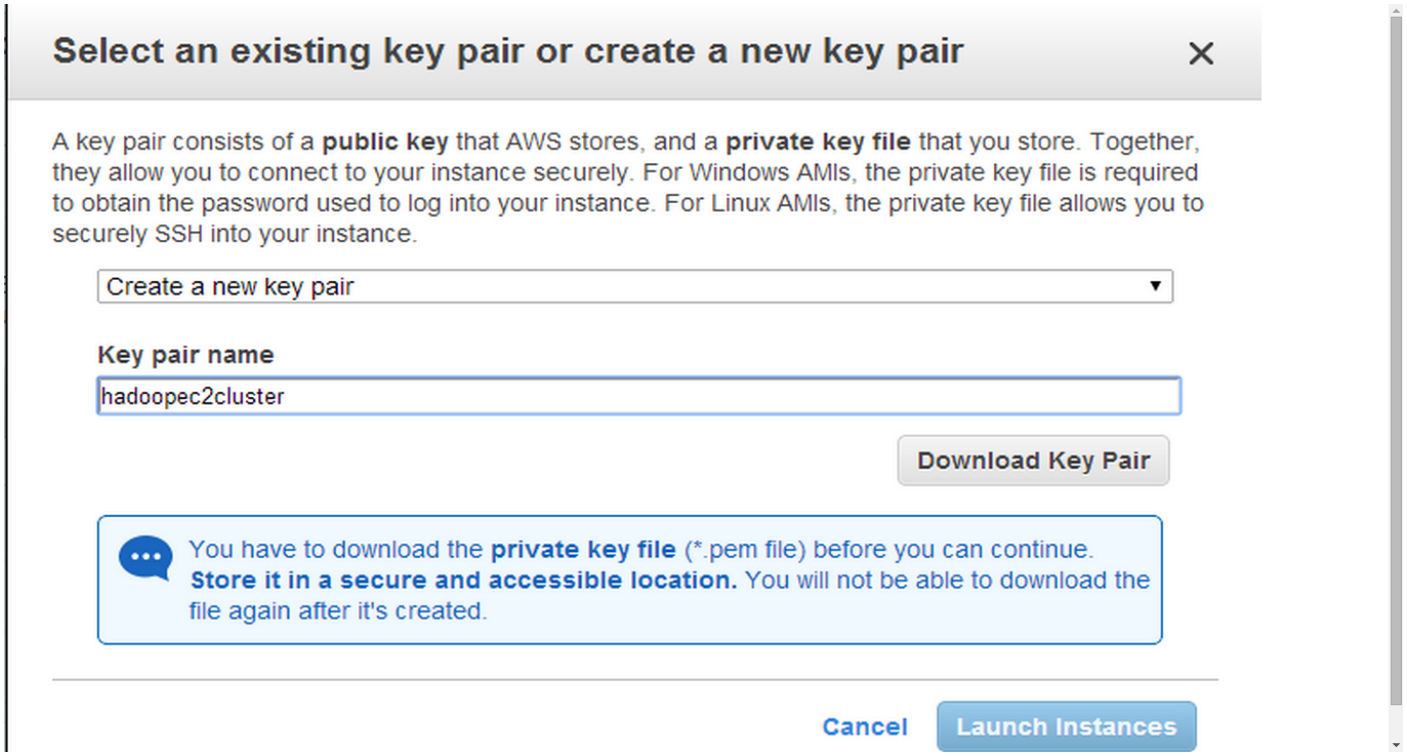
Instance Description



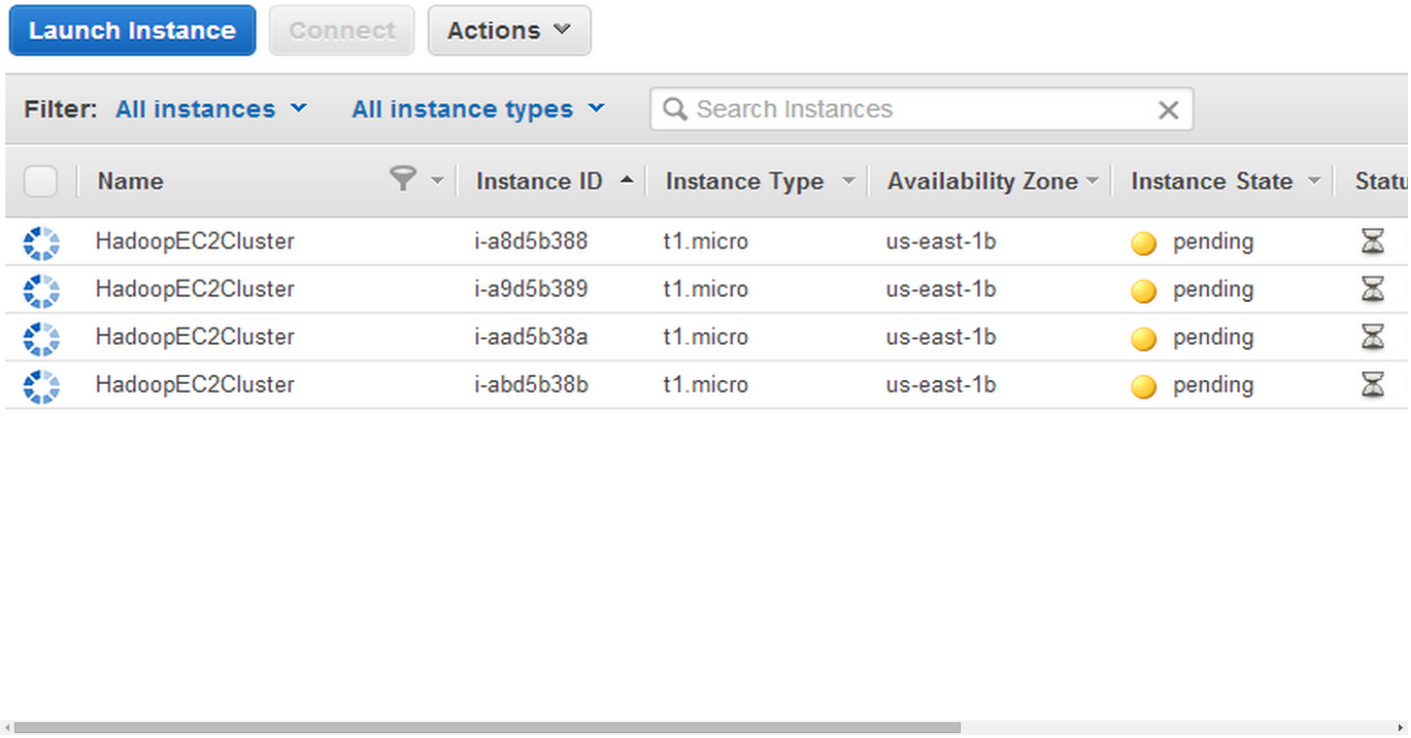
Define a Security Group



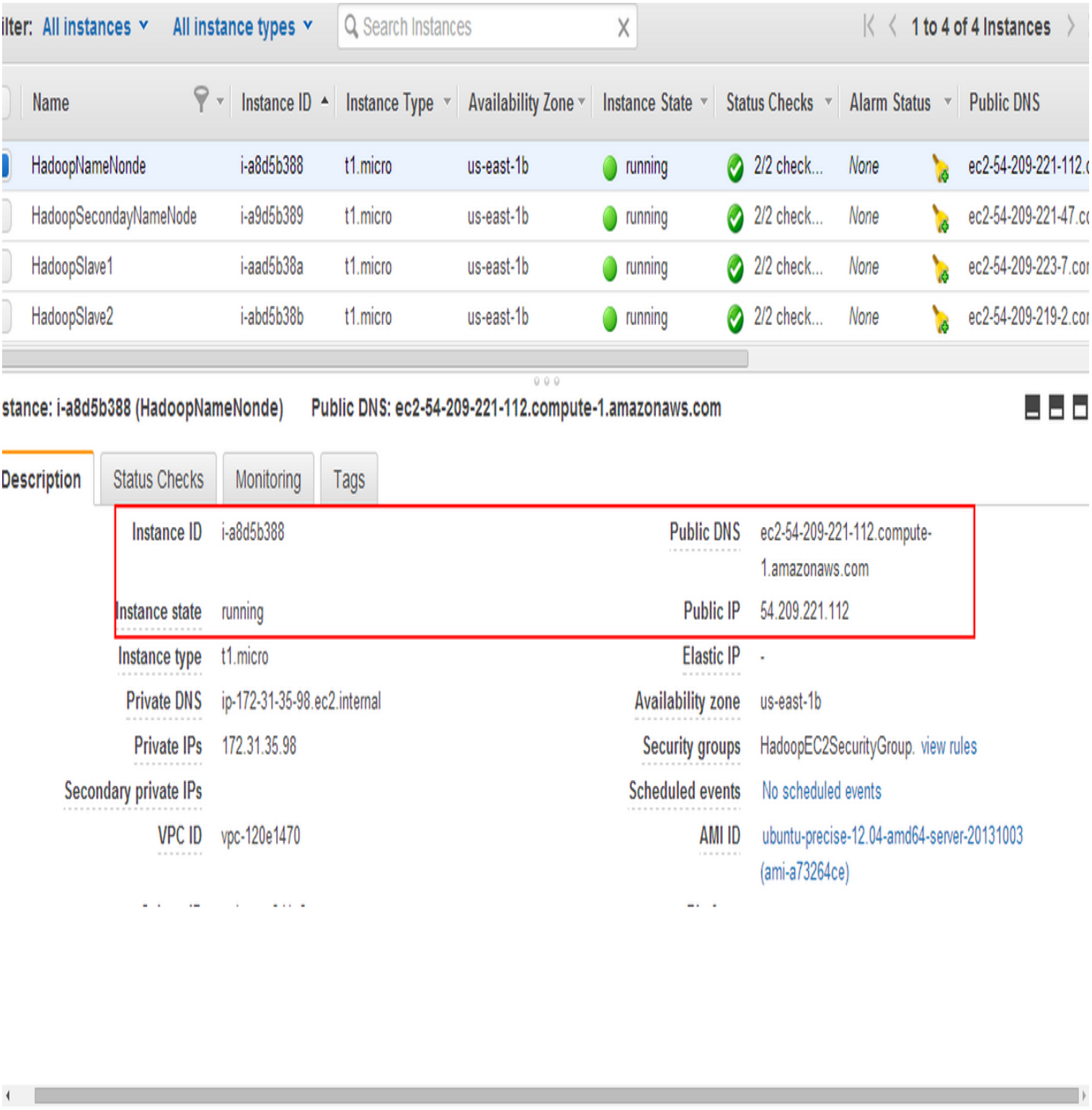
Launch Instance and Create Security Pair



Launching Instances

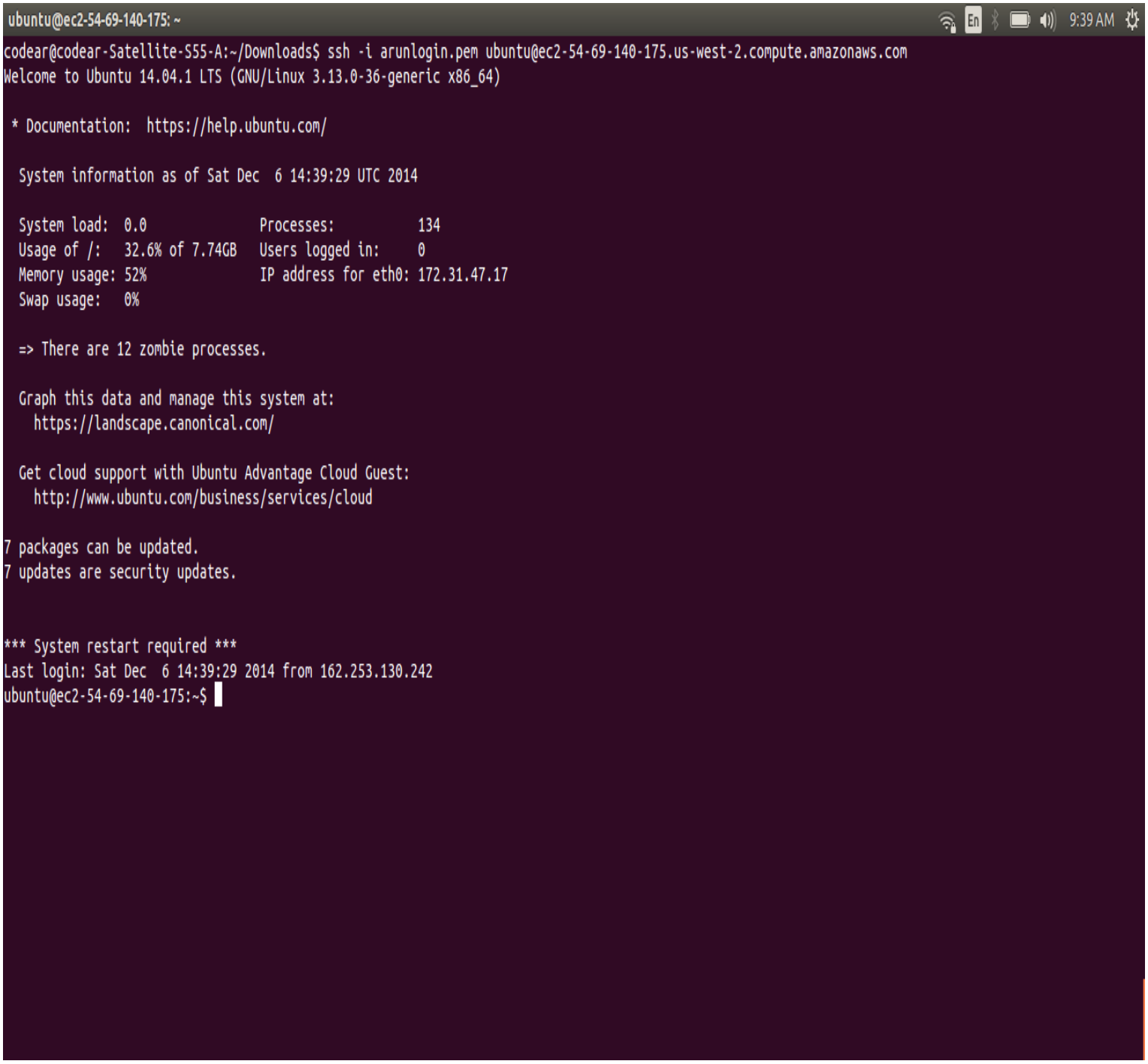


Viewing Connection Details:



Once the instances are up and running we can ssh using the authentication pem

key and connect to the cluster directly.



The health of the cluster and details of the running and completed jobs can be viewed using these web interfaces provided by hadoop.

Quick Links

ec2-54-69-140-175 Hadoop Map/Reduce Administration

State: RUNNING
Started: Sat Dec 06 13:33:01 UTC 2014
Version: 1.2.1, r1503152
Compiled: Mon Jul 22 15:23:09 PDT 2013 by mattf
Identifier: 201412061332
SafeMode: OFF

Cluster Summary (Heap Size is 15.06 MB/1.89 GB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes	Excluded Nodes
0	0	0	3	0	0	0	0	6	6	4.00	0	0	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

none

Retired Jobs

ec2-54-69-140-175 Hadoop Machine List

Active Task Trackers

Task Trackers													
Name	Host	# running tasks	Max Map Tasks	Max Reduce Tasks	Task Failures	Directory Failures	Node Health Status	Seconds Since Node Last Healthy	Total Tasks Since Start	Succeeded Tasks Since Start	Total Tasks Last Day	Succeeded Tasks Last Day	Total Tasks Last Hour
tracker_ec2-54-69-140-175.us-west-2.compute.amazonaws.com:127.0.0.1/127.0.0.1:50584	ec2-54-69-140-175.us-west-2.compute.amazonaws.com	0	2	2	67	0	N/A	0	0	0	0	0	0
tracker_ec2-54-69-179-42.us-west-2.compute.amazonaws.com:127.0.0.1/127.0.0.1:38075	ec2-54-69-179-42.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	0	0	0	0	0
tracker_ec2-54-68-74-228.us-west-2.compute.amazonaws.com:127.0.0.1/127.0.0.1:43319	ec2-54-68-74-228.us-west-2.compute.amazonaws.com	0	2	2	0	0	N/A	0	0	0	0	0	0

Highest Failures: tracker_ec2-54-69-140-175.us-west-2.compute.amazonaws.com:127.0.0.1/127.0.0.1:50584 with 67 failures

This is Apache Hadoop release 1.2.1

ec2-54-69-140-175 Hadoop Map/Reduce History Viewer

Filter (username:jobname) Filter!

Specify [user][:jobname keyword(s)][;MM/DD/YYYY] . Each of the three components is optional. Filter components are conjunctive.

Example: 'smith' will display jobs submitted by user 'smith'. 'smith:sort' will display jobs from user 'smith' having a 'sort' keyword in the jobname. ';07/04/2010' restricts to July 4, 2010

Available Jobs in History (*Displaying **100** jobs from **1** to **100** out of **242** jobs*) [get more results] [show in one page] [first page] [last page]

< 1 2 3 Next >

Job submit time	Job Id	Name	User
Sat Dec 06 12:59:16 UTC 2014	job_201412060302_0002	wordcount	ubuntu
Sat Dec 06 03:03:41 UTC 2014	job_201412060302_0001	wordcount	ubuntu
Sat Dec 06 03:01:16 UTC 2014	job_201412051302_0239	wordcount	ubuntu
Sat Dec 06 02:48:50 UTC 2014	job_201412051302_0238	wordcount	ubuntu
Sat Dec 06 02:45:52 UTC 2014	job_201412051302_0237	wordcount	ubuntu
Sat Dec 06 02:03:24 UTC 2014	job_201412051302_0236	wordcount	ubuntu
Sat Dec 06 01:24:12 UTC 2014	job_201412051302_0235	wordcount	ubuntu
Sat Dec 06 01:19:09 UTC 2014	job_201412051302_0234	wordcount	ubuntu
Sat Dec 06 01:16:19 UTC 2014	job_201412051302_0233	wordcount	ubuntu
Sat Dec 06 00:36:13 UTC 2014	job_201412051302_0232	wordcount	ubuntu

Google Chrome 9:45 AM

Hadoop Job job_201412060302_0002 on History Viewer

User: ubuntu
JobName: wordcount
JobConf: hdfs://ec2-54-69-140-175.us-west-2.compute.amazonaws.com:8020/home/ubuntu/hdfstmp/mapred/staging/ubuntu/.staging/job_201412060302_0002/job.xml
Job-ACLs: All users are allowed
Submitted At: 6-Dec-2014 12:59:16
Launched At: 6-Dec-2014 12:59:16 (0sec)
Finished At: 6-Dec-2014 12:59:33 (16sec)
Status: SUCCESS
Failure Info:
Analyse This Job

Kind	Total Tasks(successful+failed+killed)	Successful tasks	Failed tasks	Killed tasks	Start Time	Finish Time
Setup	1	1	0	0	6-Dec-2014 12:59:17	6-Dec-2014 12:59:19 (2sec)
Map	2	2	0	0	6-Dec-2014 12:59:19	6-Dec-2014 12:59:21 (2sec)
Reduce	1	1	0	0	6-Dec-2014 12:59:21	6-Dec-2014 12:59:31 (9sec)
Cleanup	1	1	0	0	6-Dec-2014 12:59:31	6-Dec-2014 12:59:33 (2sec)

	Counter	Map	Reduce	Total
Job Counters	Launched reduce tasks	0	0	1
	SLOTS_MILLIS_MAPS	0	0	8,498
	Total time spent by all reduces waiting after reserving slots (ms)	0	0	0
	Total time spent by all maps waiting after reserving slots (ms)	0	0	0
	Rack-local map tasks	0	0	1
	Launched map tasks	0	0	2
	Data-local map tasks	0	0	1
	SLOTS_MILLIS_REDUCES	0	0	9,257

Design Decisions:

1.Hadoop map reduce was used only for the tasks which were time consuming like counting of key value pairs.

2.The algorithm was divided into two parts data manipulation and rules generation. Data manipulation part which is time consuming was implemented using map reduce and rules generation part was implemented using normal Java.

3.HDFS Stream,PHP java bridge and Web HDFS frameworks were used to develop the front end to provide a simple graphical interface for my class mates to have a glimpse of my output.

4.Apache common logging framework was used for debugging purposes.

Implementation:(Back end)

All the code is explained as comments inside the code.

```
package org.myorg;
import org.apache.hadoop.fs.FileSystem;
import java.util.*;
import java.io.*;
import java.lang.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.ContentSummary;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

public class tr {
public static String newline=System.getProperty("line.separator");
/*
The callJobs Function
return type:void
Input:Integer parameter
The job:
The job of this function is to initiate a map reduce job which
counts the number of yes and no values in the input data set.
The location of the yes and no values is in column number 4 of the
input data set(Columns are counted from 0).The output is written as
two part files in HDFS format*/
```

```
public static void callJobs(int j) throws Exception
{
String x[]={"/final/input","/final/output"};

 JobConf conf = new JobConf(tr.class);
    conf.setJobName("wordcount");
    conf.set("test","4");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(Map.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    Path outputDir=new Path("/final/output");
    FileSystem fs = FileSystem.get(conf);
    fs.delete(outputDir, true);
    FileInputFormat.setInputPaths(conf, new Path(x[j]));
    FileOutputFormat.setOutputPath(conf, new Path(x[j+1]));
    JobClient.runJob(conf);
System.out.println(newline+"Iteration 1 over:"+newline);
Path pt=new Path("/final/output/part-00000");
System.out.println(newline+"We just counted the number of yes and no values in
column 5 of the input");
System.out.println();
fs = FileSystem.get(new Configuration());
                        BufferedReader br=new BufferedReader(new
InputStreamReader(fs.open(pt)));
                        String line;
                        line=br.readLine();
                        while (line != null){
                                System.out.println(line);
                                line=br.readLine();
                        }br.close();
//System.out.println("We just counted the number of yes and no values in column 5
pf the input");

}

/*The totalEntropy Function
return type:total entropy value as double
Input parameter:An integer
The job:
The totalEntropy function opens the part files written by the
call Jobs function and gets the total yes and no values
and calculates total entropy using the formula
Total entopy=-(number of yes values/Total number of values*-
log(base 2)(number of yes values/Total number of values)-
(number of no values/Total number of values*-
log(base 2)(number of no values/Total number of values)*/

public static double totalEntropy(int z) throws Exception
{
if(z==5)
System.out.println(newline+"Calculating total Entropy"+newline);
Path pt=new Path("/final/output/part-00000");
FileSystem fs = FileSystem.get(new Configuration());
BufferedReader br=new BufferedReader(new InputStreamReader(fs.open(pt)));
String s="";
```

```
double pos=0;
double sum=0;
double neg=0;
s=br.readLine();
StringTokenizer st=new StringTokenizer(s);
st.nextToken();
pos=Integer.parseInt(st.nextToken());
s=br.readLine();
st=new StringTokenizer(s);
st.nextToken();
neg=Integer.parseInt(st.nextToken());
sum=pos+neg;
double entropy=-((pos/sum)*(Math.log(pos/sum)/Math.log(2)))-
((neg/sum)*(Math.log(neg/sum)/Math.log(2)));
if(z==5)
System.out.println("The Total Entropy of the given input is "+entropy+""+newline);
return entropy;
}
/*The individualEntropy() function
return type:void
Input parameters:none
The job:
The individualEntropy function creates jobs which
counts the number of occurrence of yes and no values of each attribute in
each column.The output directory names are appended with {a,b,c,d} depending
on the column number namely attributes of column 0 will have a appended to
the output directory name and attributes of column 1 will be appended by b.
This naming convention is used  for easy file manipulation and separating the
output based on the column numbers.The map reduce jobs are actually not created
here instead
a call to utit method is made which does the job creation and configuration
part.*/

public static void individualEntropy() throws Exception
{


for(int i=0;i<4;i++)
{

JobConf conf = new JobConf(tr.class);
    conf.setJobName("wordcount");
    conf.set("test",i+"");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);

    conf.setMapperClass(Map.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);

    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    Path outputDir=new Path("/final/output"+i);
    FileSystem fs = FileSystem.get(conf);
    fs.delete(outputDir, true);

   FileInputFormat.setInputPaths(conf, new Path("/final/input"));
    FileOutputFormat.setOutputPath(conf, outputDir);
```

```
    JobClient.runJob(conf);
}
String fn[]={"a","b","c","d"};
for(int i=0;i<4;i++)
{
java.util.Map<String,Integer> temp=new HashMap<String,Integer>();
Path pt=new Path("/final/output"+i+"/part-00000");
FileSystem fs = FileSystem.get(new Configuration());
BufferedReader br=new BufferedReader(new InputStreamReader(fs.open(pt)));
String s="";
while((s=br.readLine())!=null)
{
StringTokenizer st=new StringTokenizer(s);
while(st.hasMoreTokens())
{
temp.put(st.nextToken(),Integer.parseInt(st.nextToken()));
}
}
int t=0;

for(java.util.Map.Entry<String,Integer> entry:temp.entrySet())
{
 String key=entry.getKey().trim();
 utit(key,"yes",fn[i]+""+t,i);
 t++;
 utit(key,"no",fn[i]+""+(t),i);
t++;
}
}
}

/*
The fileCorrector method:
return type:void
Input parameters:none
The job:
File corrector function is used to include 0 if the count of a particular
combination is zero.Map reduce doesn't write output of those words which has a
count of zero.So this function opens all the files written by individual entropy
function and writes the value as 0.
*/

public static void filecorrector() throws Exception
{

String x[]={"a","b","c","d"};
for(int i=0;i<4;i++)
{

for(int j=0;j<6;j++)
{
Path pt=new Path("/final/output"+x[i]+""+j+"/part-00000");
FileSystem fs = FileSystem.get(new Configuration());
if(fs.exists(pt))
{
FileStatus fp=fs.getFileStatus(pt);
ContentSummary cSummary = fs.getContentSummary(pt);
long length = cSummary.getLength();
System.out.println(length);
```

```
if(length==0)
{
int u=j-1;
 Path pt1=new Path("/final/output"+x[i]+""+u+"/part-00000");
 FileSystem fs1 = FileSystem.get(new Configuration());
 BufferedReader br1=new BufferedReader(new InputStreamReader(fs1.open(pt1)));
 BufferedWriter br=new BufferedWriter(new OutputStreamWriter(fs.create(pt,true)));
 StringTokenizer st1=new StringTokenizer(br1.readLine());
 br.write(st1.nextToken()+" "+"0");
 br.close();
}
}
}
}
}
/*
The fileMerger method
return type:void
Input parameters:none
The job:
Hadoop writes output in individual files.This function combines the output of
each column in HDFS format and converts it into a single file and stores it
locally.
Each file is the complete output of each column in the input.
*/
public static void fileMerger() throws Exception
{
String x[]={"a","b","c","d"};
int i=0,j=0;
java.util.Map<String,ArrayList<Integer>> m1=new
HashMap<String,ArrayList<Integer>>();
ArrayList<Integer> a1=new ArrayList<Integer>();
for(i=0;i<4;i++)
{
m1=new HashMap<String,ArrayList<Integer>>();
PrintWriter pr=new PrintWriter("/usr/local/hadoop/list"+i+".txt");
for(j=0;j<6;)
{
  Path pt=new Path("/final/output"+x[i]+""+j+"/part-00000");
  Path pt1=new Path("/final/output"+x[i]+""+(j+1)+"/part-00000");
  FileSystem fs = FileSystem.get(new Configuration());
    FileSystem fs1 = FileSystem.get(new Configuration());
if(fs.exists(pt))
{
 a1=new ArrayList<Integer>();
 BufferedReader br=new BufferedReader(new InputStreamReader(fs.open(pt)));
 BufferedReader br1=new BufferedReader(new InputStreamReader(fs1.open(pt1)));
 String s="";
 String z1="";
 String z2="";
 StringTokenizer st=new StringTokenizer(br.readLine());
 StringTokenizer st1=new StringTokenizer(br1.readLine());
 String xx=st.nextToken();
 st1.nextToken();
 a1.add(Integer.parseInt(st.nextToken()));
 a1.add(Integer.parseInt(st1.nextToken()));
 m1.put(xx,a1);

}
```

```
j=j+2;
}
for(java.util.Map.Entry<String,ArrayList<Integer>> entry:m1.entrySet())
{
ArrayList<Integer> t1=entry.getValue();
pr.println(entry.getKey()+" "+t1.get(0)+" "+t1.get(1));
}
pr.close();
}
System.out.println(newline+"Starting Iteration 2:"+newline);
System.out.println(newline+"Output file contents after second Iteration
are :"+newline);
for(i=0;i<4;i++)
{
BufferedReader br=new BufferedReader(new
FileReader("/usr/local/hadoop/list"+i+".txt"));
String gg="";
while((gg=br.readLine())!=null)
{
System.out.println(gg+""+newline);
}
br.close();
}
}
/*File merger1 method
return type:void
Input parameters:none
The Job:
It's work is same as file merger method but does it for files from iteration 2*/

public static void fileMerger1() throws Exception
{
java.util.Map<String,ArrayList<Integer>> m1=new
HashMap<String,ArrayList<Integer>>();
ArrayList<Integer> a1=new ArrayList<Integer>();
String x[]={"aa","bb"};
for(int i=0;i<2;i++)
{
for(int k=1;k<4;k++)
{
m1=new HashMap<String,ArrayList<Integer>>();
PrintWriter pr=new PrintWriter("/usr/local/hadoop/list1"+i+""+k+".txt");
for(int j=0;j<6;)
{
Path pt=new Path("/final/output"+x[i]+""+k+""+j+"/part-00000");
FileSystem fs = FileSystem.get(new Configuration());
j++;
Path pt1=new Path("/final/output"+x[i]+""+k+""+j+"/part-00000");
j++;
FileSystem fs1=FileSystem.get(new Configuration());
if(fs.exists(pt))
{
a1=new ArrayList<Integer>();
 BufferedReader br=new BufferedReader(new InputStreamReader(fs.open(pt)));
 BufferedReader br1=new BufferedReader(new InputStreamReader(fs1.open(pt1)));
 String s="";
 String z1="";
 String z2="";
 StringTokenizer st=new StringTokenizer(br.readLine());
```

```
 StringTokenizer st1=new StringTokenizer(br1.readLine());
 String xx=st.nextToken();
 st1.nextToken();
 a1.add(Integer.parseInt(st.nextToken()));
 a1.add(Integer.parseInt(st1.nextToken()));
 m1.put(xx,a1);
}
}
for(java.util.Map.Entry<String,ArrayList<Integer>> entry:m1.entrySet())
{
ArrayList<Integer> t1=entry.getValue();
pr.println(entry.getKey()+" "+t1.get(0)+" "+t1.get(1));
}
pr.close();
}
}
}
/*The selectWinner1 method
return type:int
Input parameter:none
The Job:
The selectWinner1 function calculates information gain of all the columns
and finds the one with the highest gain and returns the column number.
*/

public static int selectWinner1() throws Exception
{
double entropy[]=new double[4];
double main1=totalEntropy(0);
String categoryNames[]={"Outlook","Temperature","Humidity","Wind","Decision"};
for(int i=0;i<4;i++)
{
BufferedReader br=new BufferedReader(new
FileReader("/usr/local/hadoop/list"+i+".txt"));
String s="";
while((s=br.readLine())!=null)
{
StringTokenizer st=new StringTokenizer(s);
//System.out.println(s);
st.nextToken();
double x[]=new double[2];
double c=0.0;
x[0]=Double.parseDouble(st.nextToken());
x[1]=Double.parseDouble(st.nextToken());
c+=(-((x[0]/(x[0]+x[1]))*(Math.log(x[0]/(x[0]+x[1]))/Math.log(2))));
if(x[1]!=0.0)
c+=(-((x[1]/(x[0]+x[1]))*(Math.log(x[1]/(x[0]+x[1]))/Math.log(2))));
entropy[i]+=((x[0]+x[1])/14)*c;
}
entropy[i]=main1-entropy[i];
System.out.println(newline+" "+"The Information Gain of "+categoryNames[i]+" is
"+entropy[i]+""+newline);
}
int maxInformationgain=0,max=(int) entropy[0];
for(int i=1;i<entropy.length;i++)
{
if(max<((int) entropy[i]))
maxInformationgain=i;
}
```

```
System.out.println(newline+"The winner is "+categoryNames[maxInformationgain]
+newline+" "+newline);
return maxInformationgain;
}

/*
The winnerEntropies method
return type:HashMap of string and double
Input parameter:none
The Job:
It stores the names of the winners of iteration 1 and entropies in a1
hashmap and returns it.*/

public static HashMap<String,Double> winnerEntropies() throws Exception
{
ArrayList<String> x=winnerNames();
HashMap<String,Double> m1=new HashMap<String,Double>();
int nodeSplit=0;
BufferedReader br=new BufferedReader(new
FileReader("/usr/local/hadoop/list"+nodeSplit+".txt"));
String s="";
while((s=br.readLine())!=null)
{
StringTokenizer st=new StringTokenizer(s);
String z=st.nextToken();
if(x.contains(z))
{
double x2=Double.parseDouble(st.nextToken());
double y2=Double.parseDouble(st.nextToken());
double x3=-((x2/(x2+y2))*(Math.log(x2/(x2+y2))/Math.log(2)))-((y2/
(x2+y2))*(Math.log(y2/(x2+y2))/Math.log(2)));
m1.put(z+" "+(x2+y2),x3);
}
}
/*for(java.util.Map.Entry<String,Double> entry:m1.entrySet())
{

System.out.println(entry.getKey()+" "+entry.getValue());

}*/
return m1;
}
/*The stest2() method
return type:ArrayList of Integers
Input Parameter:none
The Job:
The stest2 method finds the columns with highest information gain
for iteration 3 and stores the column numbers of the winners in
arraylist and returns the arraylist.*/

public static ArrayList<Integer> stest2() throws Exception
{
//double entropy[]=new double[3];
ArrayList<Integer> ans1=new ArrayList<Integer>();
ArrayList<String> names=winnerNames();
ArrayList<String> names1=new ArrayList<String>();
String categoryNames[]={"Outlook","Temperature","Humidity","Wind","Decision"};

double totalen[]=new double[2];
```

```
int nodeSplit=0,p=0;
java.util.Map<String,Double> m1=winnerEntropies();
for(java.util.Map.Entry<String,Double> entry:m1.entrySet())
{
totalen[p]=entry.getValue();
p++;
}
//double main1=totalEntropy();
//System.out.println(main1);
for(int k=0;k<2;k++)
{
double entropy[]=new double[3];
int t=0;
for(int i=0;i<4;i++)
{
if(i!=nodeSplit)
{
BufferedReader br=new BufferedReader(new
FileReader("/usr/local/hadoop/list1"+k+""+i+".txt"));
String s="";
while((s=br.readLine())!=null)
{
StringTokenizer st=new StringTokenizer(s);
//System.out.println(s);
names1.add(st.nextToken());
double x[]=new double[2];
double c=0.0;
x[0]=Double.parseDouble(st.nextToken());
x[1]=Double.parseDouble(st.nextToken());
if(x[0]!=0.0)
c+=(-((x[0]/(x[0]+x[1]))*(Math.log(x[0]/(x[0]+x[1]))/Math.log(2))));
if(x[1]!=0.0)
c+=(-((x[1]/(x[0]+x[1]))*(Math.log(x[1]/(x[0]+x[1]))/Math.log(2))));
entropy[t]+=((x[0]+x[1])/5)*c;
//System.out.println(x[0]+" "+x[1]);
}
entropy[t]=totalen[k]-entropy[t];
t++;
}
}
int ll=0;
for(double rr:entropy)
{
System.out.println("The information gain of "+names.get(k)+"
"+categoryNames[ll+1]+" is "+rr+""+newline);
ll++;
}
int listPosition=1;
double max=entropy[0];
for(int y1=1;y1<entropy.length;y1++)
{
if(max<entropy[y1])
{
max=entropy[y1];
listPosition=y1;

}
}
ans1.add(listPosition);
```

```
System.out.println(newline+"The winner for "+names.get(k)+" is
"+categoryNames[listPosition+1]+""+newline);
names1=new ArrayList<String>();
}
return ans1;
}

public static void filecorrector1() throws Exception
{

String x[]={"aa","bb"};
for(int i=0;i<2;i++)
{
for(int k=0;k<4;k++)
{
for(int j=0;j<6;)
{
Path pt=new Path("/final/output"+x[i]+""+k+""+j+"/part-00000");
FileSystem fs = FileSystem.get(new Configuration());
j++;
Path pt1=new Path("/final/output"+x[i]+""+k+""+j+"/part-00000");
j++;
FileSystem fs1=FileSystem.get(new Configuration());
if(fs.exists(pt))
{
FileStatus fp=fs.getFileStatus(pt);
ContentSummary cSummary = fs.getContentSummary(pt);
long length = cSummary.getLength();
FileStatus fp1=fs1.getFileStatus(pt1);
ContentSummary cSummary1 = fs1.getContentSummary(pt1);
long length1 = cSummary1.getLength();
if(length==0 && length1==0){
fs.delete(pt,true);
fs1.delete(pt1,true);
}
else if(length==0 || length1==0)
{
if(length1==0)
{

 BufferedReader br1=new BufferedReader(new InputStreamReader(fs.open(pt)));
 BufferedWriter br=new BufferedWriter(new
OutputStreamWriter(fs1.create(pt1,true)));
 StringTokenizer st=new StringTokenizer(br1.readLine());
 br.write(st.nextToken()+" "+"0");
 br.close();


}
else if(length==0)
{

 BufferedReader br1=new BufferedReader(new InputStreamReader(fs1.open(pt1)));
 BufferedWriter br=new BufferedWriter(new OutputStreamWriter(fs.create(pt,true)));
 StringTokenizer st=new StringTokenizer(br1.readLine());
 br.write(st.nextToken()+" "+"0");
 br.close();
}
}
```

```
}
}
}}
}
/*The winner Names method
return type:ArrayList<String>
Input parameters:none
The Job:
The winnerNames method returns the name of the winners in the first iteration*/

public static ArrayList<String> winnerNames() throws Exception
{
int nodeSplit=0;
BufferedReader br=new BufferedReader(new
FileReader("/usr/local/hadoop/list"+nodeSplit+".txt"));
ArrayList<String> select1=new ArrayList<String>();
String s="";
while((s=br.readLine())!=null)
{
int flag=0;
StringTokenizer st=new StringTokenizer(s);
String key=st.nextToken();
while(st.hasMoreTokens())
{
int x=Integer.parseInt(st.nextToken());
if(x==0)
flag=1;
}
if(flag!=1)
{
select1.add(key);
}
}

return select1;
}
/*The selectWinner2 method
return type:void
Input parameters:none
The job:
Same as selectwinner1 but calculates the columns with highest
information gain for the second iteration.

*/

public static void selectWinner2() throws Exception
{
int nodeSplit=0;
BufferedReader br=new BufferedReader(new FileReader("list"+nodeSplit+".txt"));
ArrayList<String> select1=new ArrayList<String>();
String s="";
while((s=br.readLine())!=null)
{
int flag=0;
StringTokenizer st=new StringTokenizer(s);
String key=st.nextToken();
while(st.hasMoreTokens())
{
```

```
int x=Integer.parseInt(st.nextToken());
if(x==0)
flag=1;
}
if(flag!=1)
{
select1.add(key);
}
}


String fn[]={"aa","bb"};
int yy=0;

for(String r:select1)
{

for(int i=0;i<4;i++)
{
int t=0;
if(i!=nodeSplit)
{
br=new BufferedReader(new FileReader("list"+i+".txt"));
String keys="";
while((keys=br.readLine())!=null)
{
StringTokenizer st=new StringTokenizer(keys);
String sm=st.nextToken();
utit1(r,"yes",sm,fn[yy]+""+i+""+t,nodeSplit,i);
t++;
//System.out.println(yy+" "+sm);
utit1(r,"no",sm,fn[yy]+""+i+""+t,nodeSplit,i);
t++;
}
}
}
yy++;
}
}
/*
The utit method
Input parameters:
The string x and y stores the subsets based on which the map should count.
String y holds the ouput file name in which the output is written.int pos
holds the position of the input values.
For ex:The call utit("yes","o1","sunny",4) create map reduce jobs which counts
the total number of occurrence of  sunny and yes values and writes the output in
o1/part-00000 file.Part files are automatically created by hadoop map reduce and
we
can only specify the directory.

*/
public static void utit(String x,String y,String z,int pos) throws Exception
{
JobConf conf = new JobConf(tr.class);
    conf.setJobName("wordcount");
    conf.set("test",x);
    conf.set("test1",y);
    conf.set("test2",pos+"");
```

```
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(myMap.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    Path outputDir=new Path("/final/output"+z);
    FileSystem fs = FileSystem.get(conf);
    fs.delete(outputDir, true);
    FileInputFormat.setInputPaths(conf, new Path("/final/input"));
    FileOutputFormat.setOutputPath(conf, outputDir);
    JobClient.runJob(conf);
}
/*The utit1 method
return type:void
Input parameters:String x,key1 and z holds the criteria and y holds the
file name.pos and pos 1 are positions in the input.It works same like
utit but with one extra string to match.*/

public static void utit1(String x,String y,String key1,String z,int pos,int pos1)
throws Exception
{
JobConf conf = new JobConf(tr.class);
    conf.setJobName("wordcount");
    conf.set("test",x);
    conf.set("test1",y);
    conf.set("test3",key1);
    conf.set("test4",pos1+"");
    conf.set("test2",pos+"");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(twoMap.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    Path outputDir=new Path("/final/output"+z);
    FileSystem fs = FileSystem.get(conf);
    fs.delete(outputDir, true);
    FileInputFormat.setInputPaths(conf, new Path("/final/input"));
    FileOutputFormat.setOutputPath(conf, outputDir);
    JobClient.runJob(conf);
}
/*
The twoMap is mapper class used by utit1 function.
*/


public static class twoMap extends MapReduceBase implements Mapper<LongWritable,
Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
private static String N;
private static String N1;
private static String N3;
private static Long N2;
private static Long N4;
public void configure(JobConf job) {
```

```
    N = job.get("test");
    N1=job.get("test1");
    N3=job.get("test3");
    N2 = Long.parseLong(job.get("test2"));
    N4=Long.parseLong(job.get("test4"));
}

    public void map(LongWritable key, Text value, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {
      String line = value.toString();

      String x[]=line.split(",");
    if(x[4].equals(N1) && x[Long.valueOf(N2).intValue()].equals(N) &&
x[Long.valueOf(N4).intValue()].equals(N3))
{
        word.set(x[Long.valueOf(N4).intValue()]);
        output.collect(word, one);
}
}
}


/*
The myMap is a mapper class used by utit function*/

public static class myMap extends MapReduceBase implements Mapper<LongWritable,
Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
private static String N;
private static String N1;
private static Long N2;
private static final Log l1= LogFactory.getLog(Map.class);
public void configure(JobConf job) {
    N = job.get("test");
    N1=job.get("test1");
    N2 = Long.parseLong(job.get("test2"));

}

    public void map(LongWritable key, Text value, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {
      String line = value.toString();
      String x[]=line.split(",");
    if(x[4].equals(N1) && x[Long.valueOf(N2).intValue()].equals(N))
{
     System.out.println(newline+"Map key "+ key);
        word.set(x[Long.valueOf(N2).intValue()]);
        output.collect(word, one);
}
}
}

/*
The map is a mapper class used by call Jobs function*/

  public static class Map extends MapReduceBase implements Mapper<LongWritable,
Text, Text, IntWritable> {
     private final static IntWritable one = new IntWritable(1);
```

```
    private Text word = new Text();
private static Long N;
private static final Log l1= LogFactory.getLog(Map.class);

public void configure(JobConf job) {
    N = Long.parseLong(job.get("test"));
}

    public void map(LongWritable key, Text value, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {
     System.out.println(newline+"Map key "+ key);

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

        word.set(x[Long.valueOf(N).intValue()]);
        output.collect(word, one);

    }
  }
/*The Reduce is the common reducer class used by all mapper classes*/
  public static class Reduce extends MapReduceBase implements Reducer<Text,
IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values,
OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
      int sum = 0;
      while (values.hasNext()) {
        sum += values.next().get();
      }
      output.collect(key, new IntWritable(sum));
    }
  }
/*The generate Rules Method
Input parameters:none
return type:void
The job:
This method opens the list files created by the previous iterations and finds
rules from those
files and writes it in a file called rules.txt in hdfs format.

*/
public static void generateRules() throws Exception
{
ArrayList<String> rules=new ArrayList<String>();
String categoryNames[]={"Outlook","Temperature","Humidity","Wind","Decision"};
PrintWriter pr=new PrintWriter("/usr/local/hadoop/rules.txt");
int winner1=selectWinner1();
ArrayList<String> winnerNames=winnerNames();
ArrayList<Integer> winnerPos=stest2();
//System.out.println(categoryNames[winner1]+" ");
int j=0;
/*for(String rt:winnerNames)
{
System.out.println(rt +" "+categoryNames[winnerPos.get(j)+1]);
j++;
}*/
BufferedReader br=new BufferedReader(new
FileReader("/usr/local/hadoop/list"+winner1+".txt"));
String z="";
```

```
while((z=br.readLine())!=null)
{
StringTokenizer st=new StringTokenizer(z);
String t1=st.nextToken();
if(!(winnerNames.contains(t1)))
{
int yes_value=Integer.parseInt(st.nextToken());
int no_value=Integer.parseInt(st.nextToken());
if(yes_value!=0 && no_value==0)
rules.add(categoryNames[winner1]+" "+t1+" "+"yes");
else if(no_value!=0 && yes_value==0)
rules.add(categoryNames[winner1]+" "+t1+" "+"no");
}
}
int l=0;
for(int g:winnerPos)
{
br=new BufferedReader(new FileReader("/usr/local/hadoop/list1"+l+""+
(g+1)+".txt"));
while((z=br.readLine())!=null)
{
StringTokenizer st=new StringTokenizer(z);
String t1=st.nextToken();
int yes_value=Integer.parseInt(st.nextToken());
int no_value=Integer.parseInt(st.nextToken());
if(no_value==0)
rules.add(categoryNames[winner1]+" "+winnerNames.get(l)+" "+categoryNames[g+1]+"
"+t1+" "+"yes");
else if(yes_value==0)
rules.add(categoryNames[winner1]+" "+winnerNames.get(l)+" "+categoryNames[g+1]+"
"+t1+" "+"no");
}
l++;
}

 Path pt=new Path("/final/Rules_Generated/rule.txt");
 FileSystem fs = FileSystem.get(new Configuration());
if(fs.exists(pt))
fs.delete(pt, true);
 BufferedWriter br1=new BufferedWriter(new
OutputStreamWriter(fs.create(pt,true)));
System.out.println("The generated rules are :"+newline);
for(String h1:rules)
{
System.out.println(h1+""+newline);
br1.write(h1);
br1.newLine();
}
br1.close();
}

  public static void main(String[] args) throws Exception {
Path pt=new Path("/final/input/input.txt");
System.out.println(newline+"The sample input is :"+newline);
                        FileSystem fs = FileSystem.get(new Configuration());
                        BufferedReader br=new BufferedReader(new
InputStreamReader(fs.open(pt)));
                        String line;
                        line=br.readLine();
```

```
                        while (line != null){
                                System.out.println(line+""+newline);
                                line=br.readLine();
                        }br.close();
System.out.println();
 callJobs(0);
    totalEntropy(5);
    individualEntropy();
   filecorrector();

   fileMerger();
selectWinner1();
selectWinner2();
filecorrector1();
fileMerger1();
stest2();
generateRules();
//winnerEntropies();
 }
}
```

The main problem in using Apache map reduce is that each output from mapper and reducer are stored as separate files in different directories. So file handling has to be done manually and all the output files has to be merged. These jobs are done by the file* functions in the code.The utit,utit1 and callJob functions are used for Map reduce job configuration and initialization.Function with same names and different numbers in the end like selectWinner1 and selectWinner2 means they perform the same job but with results from that particular iteration like 1 and 2.

Implementation(Front end):

Web Hdfs framework and php-java bridge is used to invoke the back end java code and output the results in a simple web interface. The user query in the web interface invokes only the call Jobs,total Entropy and generate rules function in the back end java code to produce fast results and avoid long waiting times. The full back end java code takes 10-15 minutes for complete execution. The link to the front end is:

http://ec2-54-69-140-175.us-west-2.compute.amazonaws.com:2145/xampp/t2.php

Please select the required conditions and click predict and wait for 10-15 seconds to see the ouput.

HTML Front end code:

```
<html>
<head>
<style>
body
{

background-color:#E8f0f0

}
h1 {
    color: #0033CC;
    text-align:center;
    size:24px;
    font-family:Sans-serif;
}
h2
{
color:#FF0000;
text-align:center;
size:22px;
font-family:Sans-serif;


}

#mydiv
{
text-align:center;
font-family:Georgia, serif;
font-size:20px;
}
 #btn {
  background: #3498db;
  background-image: -webkit-linear-gradient(top, #3498db, #2980b9);
  background-image: -moz-linear-gradient(top, #3498db, #2980b9);
  background-image: -ms-linear-gradient(top, #3498db, #2980b9);
  background-image: -o-linear-gradient(top, #3498db, #2980b9);
  background-image: linear-gradient(to bottom, #3498db, #2980b9);
  -webkit-border-radius: 28;
  -moz-border-radius: 28;
  border-radius: 28px;
  font-family: Georgia;
  color: #ffffff;
```

```
  font-size: 20px;
  padding: 8px 35px 8px 35px;
  text-decoration: none;
  margin-top:45px;
}
select#soflow {
   -webkit-appearance: button;
   -webkit-border-radius: 2px;
   -webkit-box-shadow: 0px 1px 3px rgba(0, 0, 0, 0.1);
   -webkit-padding-end: 20px;
   -webkit-padding-start: 2px;
   -webkit-user-select: none;
   background-image: url(http://i62.tinypic.com/15xvbd5.png), -webkit-linear-
gradient(#FAFAFA, #F4F4F4 40%, #E5E5E5);
   background-position: 97% center;
   background-repeat: no-repeat;
   border: 1px solid #AAA;
   color: #555;
   font-size: 20px;
   margin-top: 35px;
   overflow: hidden;
   padding: 5px 10px;
   text-overflow: ellipsis;
   white-space: nowrap;
   width: 150px;
   border:1px solid;
   border-radius: 18px;


}

#btn:hover {
  background: #3cb0fd;
  background-image: -webkit-linear-gradient(top, #3cb0fd, #3498db);
  background-image: -moz-linear-gradient(top, #3cb0fd, #3498db);
  background-image: -ms-linear-gradient(top, #3cb0fd, #3498db);
  background-image: -o-linear-gradient(top, #3cb0fd, #3498db);
  background-image: linear-gradient(to bottom, #3cb0fd, #3498db);
  text-decoration: none;
}
</style>
</head>
<body>
<h1>Decison Making System Using ID3</h1>
<h2>Select suitable attributes below to predict if outdoor play is possible:</h2>
<form name="myform" action="/xampp/t11.php" method="POST">
<div id="mydiv"> Outlook:<select name="s1" id="soflow">
  <option value="sunny">Sunny</option>
  <option value="overcast">Overcast</option>
  <option value="rain">Rain</option>
</select>
   Temperature:
<select name="s2" id="soflow">
  <option value="hot">Hot</option>
  <option value="mild">Mild</option>
  <option value="cool">Cool</option>
</select>
       Humidity:
<select name="s3" id="soflow">
```

```
  <option value="high">High</option>
  <option value="normal">Normal</option>
</select>    Wind:
<select name="s4" id="soflow">
  <option value="strong">Strong</option>
  <option value="weak">Weak</option>
</select><br />
<input type="submit" value="PREDICT" id="btn"></div>
</form>
</body>
```

PHP Front End:

```
<html>
<head>
<style>
body {
    background-color: #E8F0F0;
}


h1 {
    color: #0033CC;
    text-align:center;
    size:24px;
    font-family:Sans-serif;
}
h2
{
color:#FF0000;
text-align:left;
size:22px;
font-family:Sans-serif;


}

p {
    color:#000000;
    font-family: "Georgia, serif";
    font-size: 26px;
}
#info
{

  color:#000000;
    size:26px;
  font-family:Georgia, serif;


}
#rules-head
{
  color: #FF0000;
  size:28px;
```

```
  font-family:Georgia, serif;


}
#rules
{
  color: #000000;
  size:26px;
  font-family:Georgia, serif;


}
#jspe
{
position: absolute;
right: 0px;
padding-top:75px;
width:480px;
height:850px;

}
</style>
</head>
<body>
<div id="jspe"><embed src="http://ec2-54-69-140-175.us-west-
2.compute.amazonaws.com:50030/jobtracker.jsp" width="450" height="850" /></div>
<?php
$cmd = "java JavaRunCommand";
ob_implicit_flush(true);ob_end_flush();
$descriptorspec = array(
   0 => array("pipe", "r"),   // stdin is a pipe that the child will read from
   1 => array("pipe", "w"),   // stdout is a pipe that the child will write to
   2 => array("pipe", "w")    // stderr is a pipe that the child will write to
);
echo "<h1><b>Real Time ID3 Decision Tree Algorithm Execution</b></h1>";
echo "<h2><b>Input</b></h2>";
flush();
$process = proc_open($cmd, $descriptorspec, $pipes, realpath('./'), array());
echo "<pre>";
if (is_resource($process)) {
    while ($s = fgets($pipes[1])) {
        print $s;
        flush();
    }
}
echo "</pre>";

proc_close($process);
$x1=$_POST["s1"];
$x2=$_POST["s2"];
$x3=$_POST["s3"];
$x4=$_POST["s4"];
echo "<p>The given user query is ".$x1." ".$x2." ".$x3." ".$x4.".";
if($x1=="overcast")
echo("You can play Outside.</p>");
else
if($x1=="sunny")
{
if($x3=="normal")
```

```
echo "You can play Outside.</p>";
else
if($x3=="high")
echo "You cannot play Outside.</p>";
}
else
if($x1=="rain")
{
if($x3=="normal")
echo "You can play Outside.</p>";
else
if($x3=="high")
echo "You cannot play Outside.</p>";
}
else
if($x1=="rain")
{
if($x4=="strong")
echo "You cannot play Outside.</p>";
else
if($x4=="weak")
echo "You can play Outside.</p>";
}


?>
</body>
</html>
```

Java Front End:

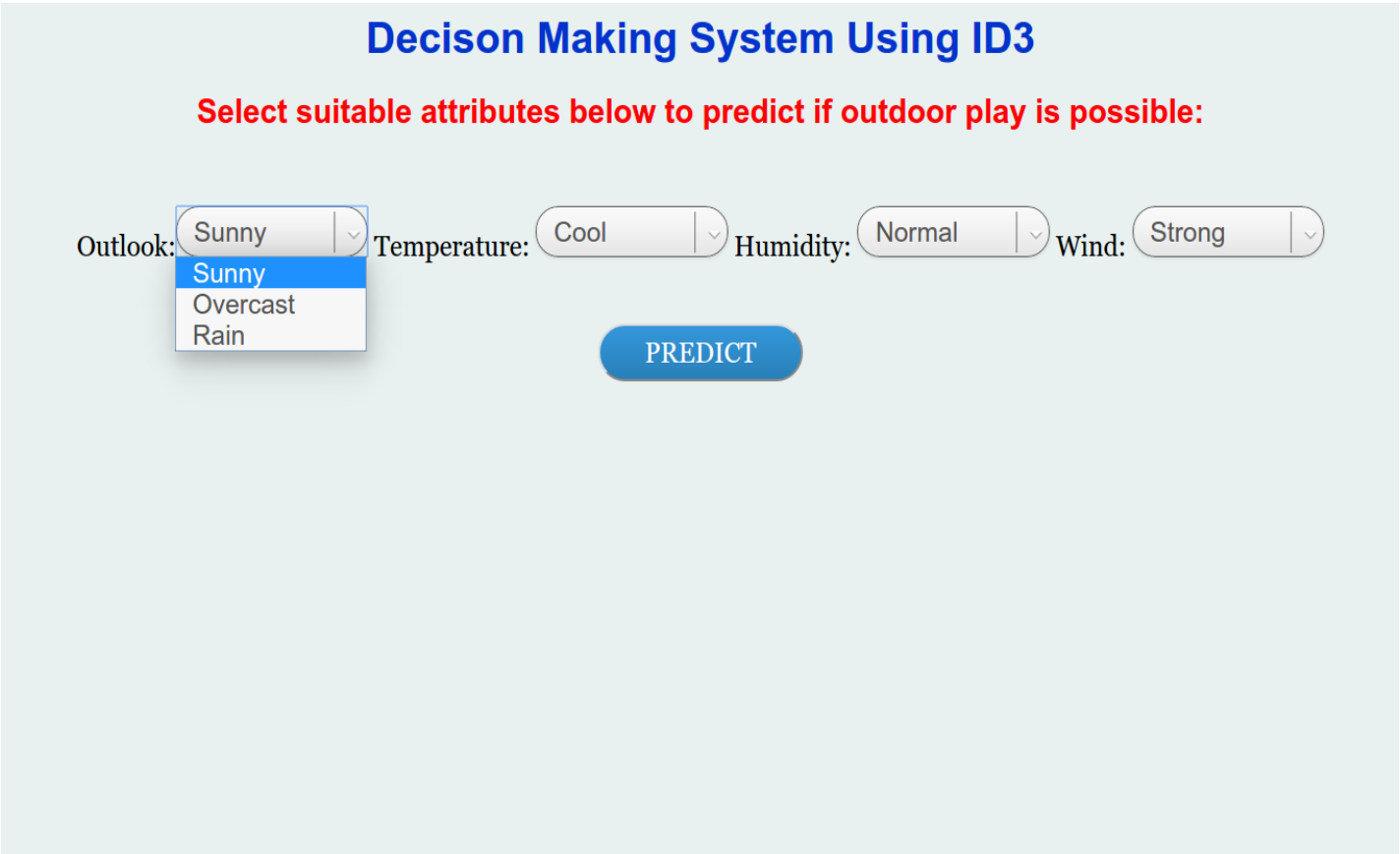
```
import java.io.*;
public class JavaRunCommand {

    public static void main(String args[]) {

        String s = null;

        try {


            // using the Runtime exec method:
            Process p = Runtime.getRuntime().exec("/home/ubuntu/hadoop/bin/hadoop
jar /home/ubuntu/hadoop/tr.jar org.myorg.tr");

            BufferedReader stdInput = new BufferedReader(new
                 InputStreamReader(p.getInputStream()));

            BufferedReader stdError = new BufferedReader(new
```

```
                InputStreamReader(p.getErrorStream()));

            // read the output from the command

            while ((s = stdInput.readLine()) != null) {
                System.out.println(s);
            }

            // read any errors from the attempted command

            while ((s = stdError.readLine()) != null) {
                System.out.println(s);
            }

            System.exit(0);
        }
        catch (IOException e) {
            e.printStackTrace();
            System.exit(-1);
        }

    }}
```

The java code calls the back end java class and streams the ouput from the command line and the ouput is buffered to user screen in real time using php. CSS is used for styling purposes.

Front End Screen Shots:

Front end calls only three functions in the back end java class and it provides answer for the user query and also shows the output of the map reduce job initiated to answer that query. Output of the whole code is shown later in this report.



The user should select the suitable attributes from the drop down and click predict to run the code.When the user clicks predict new map reduce job is created and the complete execution is shown in **realtime**.

User Output:

The right side shows the job creation and other information.

Real Time ID3 Decision Tree Algorithm Execution

Input

The sample input is :

sunny hot high weak no

sunny hot high strong no

overcast hot high weak yes

rain mild high weak yes

rain cool normal weak yes

rain cool normal strong no

overcast cool normal strong yes

Quick Links

ec2-54-69-140-175 Hadoop Map/Reduce Administration

State: RUNNING
Started: Sat Dec 06 13:33:01 UTC 2014
Version: 1.2.1, r1503152
Compiled: Mon Jul 22 15:23:09 PDT 2013 by mattf
Identifier: 201412061332
SafeMode: OFF

Cluster Summary (Heap Size is 15.12 MB/1.89 GB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	O
0	0	20	3	0	0

Scheduling Information

Google Chrome 4:15 PM

Output:

Please wait for Output.....

Iteration 1 over:
We just counted the number of yes and no values in column 5 of the input
no 5
yes 9

Calculating total Entropy
The Total Entropy of the given input is 0.9402859586706309

Starting Iteration 2:Output file contents after second Iteration are :

File 0:


Google Chrome
4:15 PM
File 0:
sunny 2 3
overcast 4 0
rain 3 2
File 1:
cool 3 1
hot 2 2
mild 4 2
File 2:
normal 6 1


high 3 4
File 3:
strong 3 3
weak 6 2
The information gain of sunny Temperature is 0.5709505944546686
The information gain of sunny Humidity is 0.9709505944546686
The information gain of sunny Wind is 0.01997309402197489

The winner for sunny is Humidity

The information gain of rain Temperature is 0.01997309402197489

The information gain of rain Humidity is 0.01997309402197489

The information gain of rain Wind is 0.9709505944546686

The winner for rain is Wind

The generated rules are :

```
14/12/06 21:13:39 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=8572
14/12/06 21:13:39 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
14/12/06 21:13:39 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
14/12/06 21:13:39 INFO mapred.JobClient:     Rack-local map tasks=1
14/12/06 21:13:39 INFO mapred.JobClient:     Launched map tasks=2
14/12/06 21:13:39 INFO mapred.JobClient:     Data-local map tasks=1
14/12/06 21:13:39 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=9245
14/12/06 21:13:39 INFO mapred.JobClient:   File Input Format Counters
14/12/06 21:13:39 INFO mapred.JobClient:     Bytes Read=561
14/12/06 21:13:39 INFO mapred.JobClient:   File Output Format Counters
14/12/06 21:13:39 INFO mapred.JobClient:     Bytes Written=11
14/12/06 21:13:39 INFO mapred.JobClient:   FileSystemCounters
14/12/06 21:13:39 INFO mapred.JobClient:     FILE_BYTES_READ=44
14/12/06 21:13:39 INFO mapred.JobClient:     HDFS_BYTES_READ=833
14/12/06 21:13:39 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=169857
14/12/06 21:13:39 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=11
14/12/06 21:13:39 INFO mapred.JobClient:   Map-Reduce Framework
14/12/06 21:13:39 INFO mapred.JobClient:     Map output materialized bytes=50
14/12/06 21:13:39 INFO mapred.JobClient:     Map input records=14
14/12/06 21:13:39 INFO mapred.JobClient:     Reduce shuffle bytes=50
14/12/06 21:13:39 INFO mapred.JobClient:     Spilled Records=8
14/12/06 21:13:39 INFO mapred.JobClient:     Map output bytes=107
14/12/06 21:13:39 INFO mapred.JobClient:     Total committed heap usage (bytes)=291577856
14/12/06 21:13:39 INFO mapred.JobClient:     CPU time spent (ms)=1340
14/12/06 21:13:39 INFO mapred.JobClient:     Map input bytes=374
14/12/06 21:13:39 INFO mapred.JobClient:     SPLIT_RAW_BYTES=272
14/12/06 21:13:39 INFO mapred.JobClient:     Combine input records=14
14/12/06 21:13:39 INFO mapred.JobClient:     Reduce input records=4
14/12/06 21:13:39 INFO mapred.JobClient:     Reduce input groups=2
14/12/06 21:13:39 INFO mapred.JobClient:     Combine output records=4
14/12/06 21:13:39 INFO mapred.JobClient:     Physical memory (bytes) snapshot=419389440
14/12/06 21:13:39 INFO mapred.JobClient:     Reduce output records=2
14/12/06 21:13:39 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=1911922688
14/12/06 21:13:39 INFO mapred.JobClient:     Map output records=14

The given user query is sunny hot normal strong.You can play Outside.
```

BACK END(Full Output):

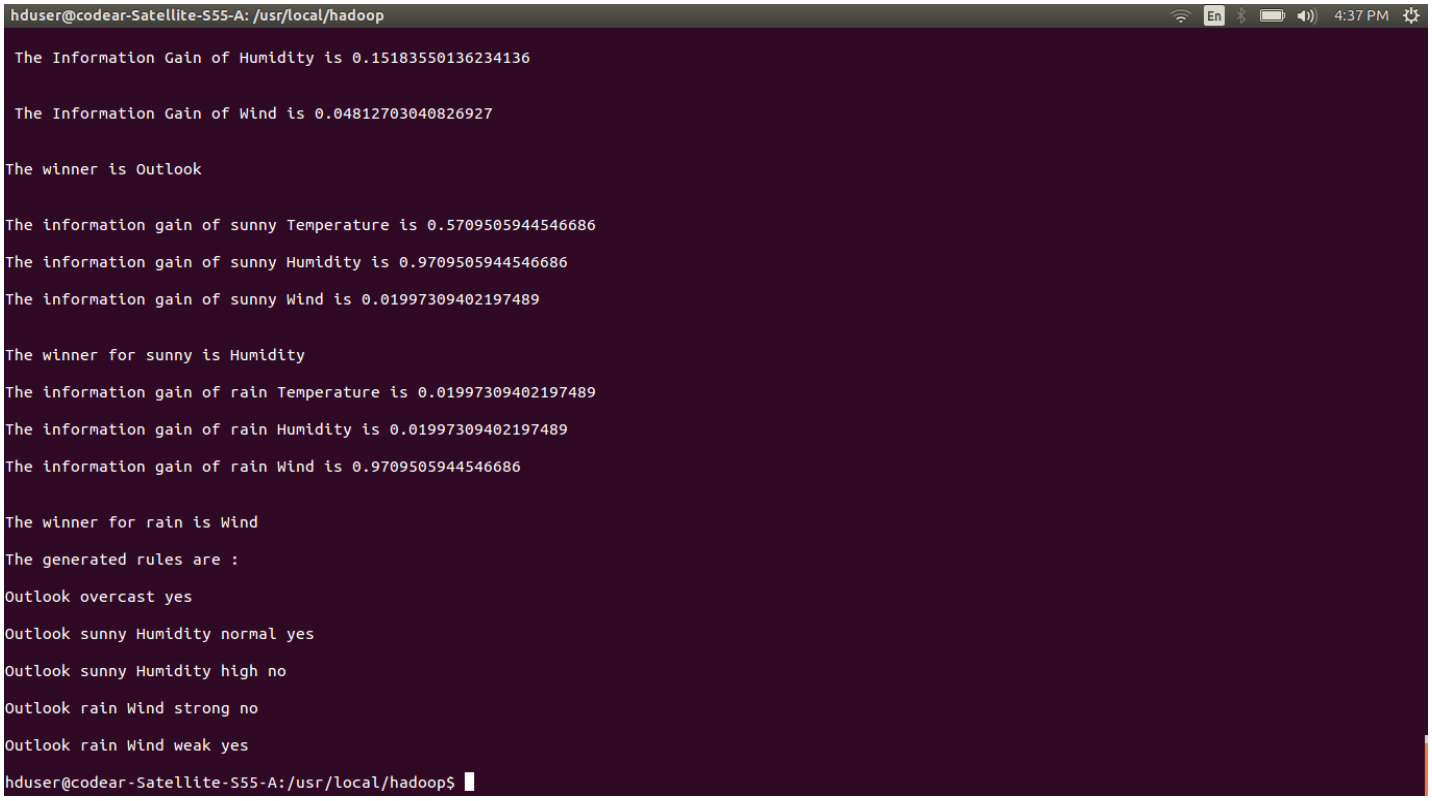
The Backend Java code takes 10-15 minutes to produce the full output so only a sample of the output is shown. Totally 160-180 map reduce jobs are created and their status are printed so only a output of few of them are shown.

```
hduser@codear-Satellite-S55-A: /usr/local/hadoop
hduser@codear-Satellite-S55-A:/usr/local/hadoop$ bin/hadoop jar tr.jar org.myorg.tr

The sample input is :

sunny,hot,high,weak,no

sunny,hot,high,strong,no

overcast,hot,high,weak,yes

rain,mild,high,weak,yes

rain,cool,normal,weak,yes

rain,cool,normal,strong,no

overcast,cool,normal,strong,yes

sunny,mild,high,weak,no

sunny,cool,normal,weak,yes

rain,mild,normal,weak,yes

sunny,mild,normal,strong,yes

overcast,mild,high,strong,yes

overcast,hot,normal,weak,yes

rain,mild,high,strong,no


14/12/06 16:24:08 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
14/12/06 16:24:08 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/12/06 16:24:08 WARN snappy.LoadSnappy: Snappy native library not loaded
14/12/06 16:24:08 INFO mapred.FileInputFormat: Total input paths to process : 1
14/12/06 16:24:09 INFO mapred.JobClient: Running job: job_201412050548_0311
14/12/06 16:24:10 INFO mapred.JobClient:  map 0% reduce 0%
14/12/06 16:24:13 INFO mapred.JobClient:  map 100% reduce 0%
```

```
hduser@codear-Satellite-S55-A: /usr/local/hadoop
14/12/06 16:24:22 INFO mapred.JobClient:     FILE_BYTES_READ=44
14/12/06 16:24:22 INFO mapred.JobClient:     HDFS_BYTES_READ=755
14/12/06 16:24:22 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=164780
14/12/06 16:24:22 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=11
14/12/06 16:24:22 INFO mapred.JobClient:   Map-Reduce Framework
14/12/06 16:24:22 INFO mapred.JobClient:     Map output materialized bytes=50
14/12/06 16:24:22 INFO mapred.JobClient:     Map input records=14
14/12/06 16:24:22 INFO mapred.JobClient:     Reduce shuffle bytes=50
14/12/06 16:24:22 INFO mapred.JobClient:     Spilled Records=8
14/12/06 16:24:22 INFO mapred.JobClient:     Map output bytes=107
14/12/06 16:24:22 INFO mapred.JobClient:     Total committed heap usage (bytes)=603979776
14/12/06 16:24:22 INFO mapred.JobClient:     CPU time spent (ms)=2030
14/12/06 16:24:22 INFO mapred.JobClient:     Map input bytes=374
14/12/06 16:24:22 INFO mapred.JobClient:     SPLIT_RAW_BYTES=194
14/12/06 16:24:22 INFO mapred.JobClient:     Combine input records=14
14/12/06 16:24:22 INFO mapred.JobClient:     Reduce input records=4
14/12/06 16:24:22 INFO mapred.JobClient:     Reduce input groups=2
14/12/06 16:24:22 INFO mapred.JobClient:     Combine output records=4
14/12/06 16:24:22 INFO mapred.JobClient:     Physical memory (bytes) snapshot=551284736
14/12/06 16:24:22 INFO mapred.JobClient:     Reduce output records=2
14/12/06 16:24:22 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=2095427584
14/12/06 16:24:22 INFO mapred.JobClient:     Map output records=14

Iteration 1 over:


We just counted the number of yes and no values in column 5 of the input

no      5
yes     9

Calculating total Entropy

The Total Entropy of the given input is 0.9402859586706309

14/12/06 16:24:22 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
14/12/06 16:24:22 INFO mapred.FileInputFormat: Total input paths to process : 1
14/12/06 16:24:22 INFO mapred.JobClient: Running job: job_201412050548_0312
14/12/06 16:24:23 INFO mapred.JobClient:  map 0% reduce 0%
14/12/06 16:24:26 INFO mapred.JobClient:  map 100% reduce 0%
```

```
hduser@codear-Satellite-S55-A: /usr/local/hadoop                    4:25 PM
14/12/06 16:25:15 INFO mapred.JobClient:   Job Counters
14/12/06 16:25:15 INFO mapred.JobClient:     Launched reduce tasks=1
14/12/06 16:25:15 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=5259
14/12/06 16:25:15 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
14/12/06 16:25:15 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
14/12/06 16:25:15 INFO mapred.JobClient:     Launched map tasks=2
14/12/06 16:25:15 INFO mapred.JobClient:     Data-local map tasks=2
14/12/06 16:25:15 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=8170
14/12/06 16:25:15 INFO mapred.JobClient:   File Input Format Counters
14/12/06 16:25:15 INFO mapred.JobClient:     Bytes Read=561
14/12/06 16:25:15 INFO mapred.JobClient:   File Output Format Counters
14/12/06 16:25:15 INFO mapred.JobClient:     Bytes Written=16
14/12/06 16:25:15 INFO mapred.JobClient:   FileSystemCounters
14/12/06 16:25:15 INFO mapred.JobClient:     FILE_BYTES_READ=54
14/12/06 16:25:15 INFO mapred.JobClient:     HDFS_BYTES_READ=755
14/12/06 16:25:15 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=164803
14/12/06 16:25:15 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=16
14/12/06 16:25:15 INFO mapred.JobClient:   Map-Reduce Framework
14/12/06 16:25:15 INFO mapred.JobClient:     Map output materialized bytes=60
14/12/06 16:25:15 INFO mapred.JobClient:     Map input records=14
14/12/06 16:25:15 INFO mapred.JobClient:     Reduce shuffle bytes=60
14/12/06 16:25:15 INFO mapred.JobClient:     Spilled Records=8
14/12/06 16:25:15 INFO mapred.JobClient:     Map output bytes=138
14/12/06 16:25:15 INFO mapred.JobClient:     Total committed heap usage (bytes)=603979776
14/12/06 16:25:15 INFO mapred.JobClient:     CPU time spent (ms)=1810
14/12/06 16:25:15 INFO mapred.JobClient:     Map input bytes=374
14/12/06 16:25:15 INFO mapred.JobClient:     SPLIT_RAW_BYTES=194
14/12/06 16:25:15 INFO mapred.JobClient:     Combine input records=14
14/12/06 16:25:15 INFO mapred.JobClient:     Reduce input records=4
14/12/06 16:25:15 INFO mapred.JobClient:     Reduce input groups=2
14/12/06 16:25:15 INFO mapred.JobClient:     Combine output records=4
14/12/06 16:25:15 INFO mapred.JobClient:     Physical memory (bytes) snapshot=556621824
14/12/06 16:25:15 INFO mapred.JobClient:     Reduce output records=2
14/12/06 16:25:15 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=2103590912
14/12/06 16:25:15 INFO mapred.JobClient:     Map output records=14
14/12/06 16:25:15 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
14/12/06 16:25:15 INFO mapred.FileInputFormat: Total input paths to process : 1
14/12/06 16:25:16 INFO mapred.JobClient: Running job: job_201412050548_0316
14/12/06 16:25:17 INFO mapred.JobClient:  map 0% reduce 0%
14/12/06 16:25:20 INFO mapred.JobClient:  map 100% reduce 0%
```

```
hduser@codear-Satellite-S55-A: /usr/local/hadoop                    4:37 PM
14/12/06 16:35:58 INFO mapred.JobClient:     CPU time spent (ms)=2720
14/12/06 16:35:58 INFO mapred.JobClient:     Map input bytes=374
14/12/06 16:35:58 INFO mapred.JobClient:     SPLIT_RAW_BYTES=194
14/12/06 16:35:58 INFO mapred.JobClient:     Combine input records=0
14/12/06 16:35:58 INFO mapred.JobClient:     Reduce input records=0
14/12/06 16:35:58 INFO mapred.JobClient:     Reduce input groups=0
14/12/06 16:35:58 INFO mapred.JobClient:     Combine output records=0
14/12/06 16:35:58 INFO mapred.JobClient:     Physical memory (bytes) snapshot=565194752
14/12/06 16:35:58 INFO mapred.JobClient:     Reduce output records=0
14/12/06 16:35:58 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=2098397184
14/12/06 16:35:58 INFO mapred.JobClient:     Map output records=0
The information gain of sunny Temperature is 0.5709505944546686

The information gain of sunny Humidity is 0.9709505944546686

The information gain of sunny Wind is 0.01997309402197489


The winner for sunny is Humidity

The information gain of rain Temperature is 0.01997309402197489

The information gain of rain Humidity is 0.01997309402197489

The information gain of rain Wind is 0.9709505944546686


The winner for rain is Wind


 The Information Gain of Outlook is 0.2467498197744391


 The Information Gain of Temperature is 0.029222565658954647


 The Information Gain of Humidity is 0.15183550136234136


 The Information Gain of Wind is 0.04812703040826927
```



Output Verification Methods:

(Some cool ways to check my output.All links shown below will be active so please check it out if time permits)

1.To run a Job please use the web interface. Even though it doesn't give the full output you will get the feel of what's happening behind and also the output of the map reduce job initiated by you is displayed.

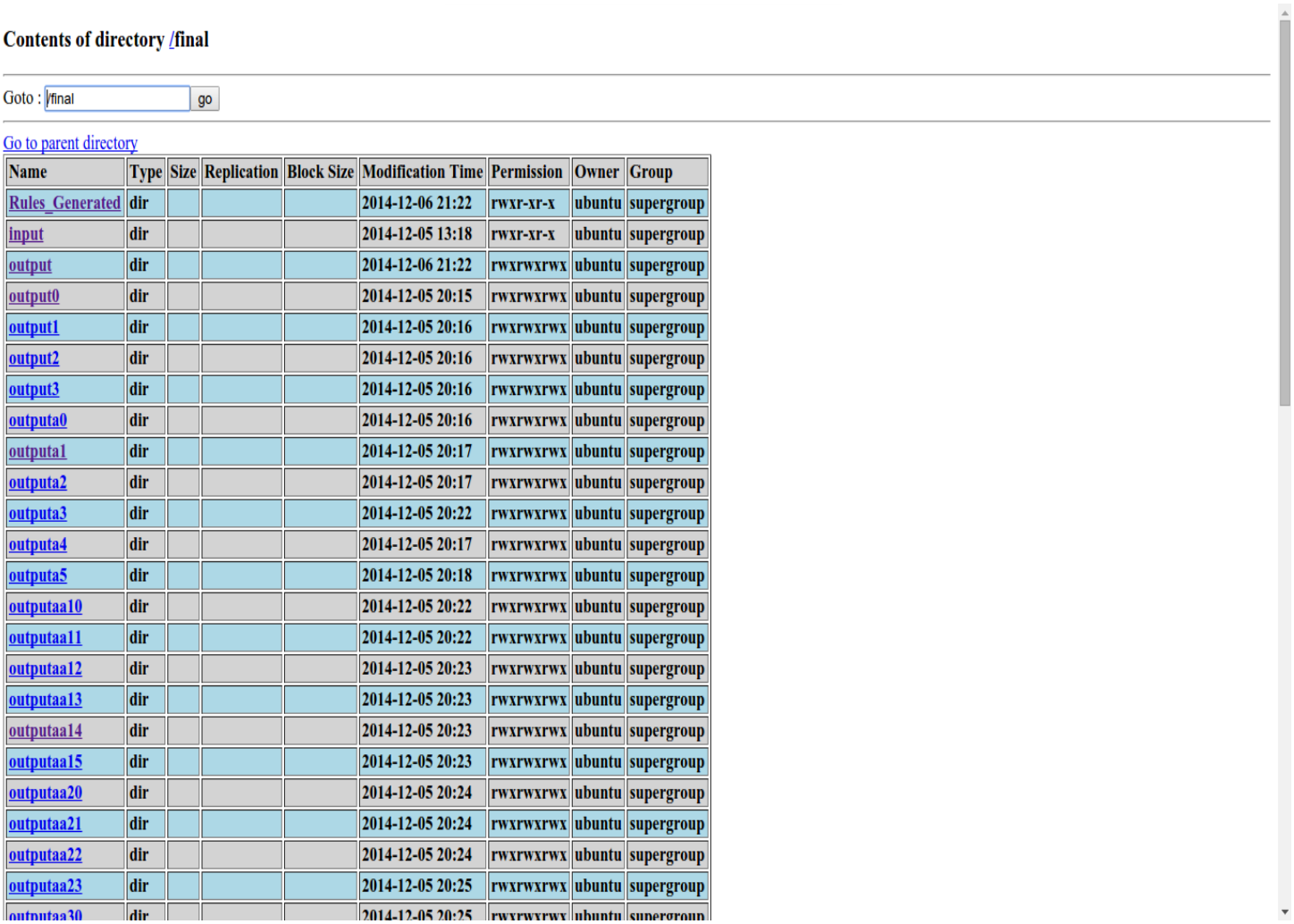
http://ec2-54-69-140-175.us-west-2.compute.amazonaws.com:2145/xampp/t2.php

2.To see the full list of output files generated by the back end java code and to view the contents of each and every file use the following link and click on the folders and files to view the content. The Rules_Generated folder contains the

final output in a rule.txt file and all other folders are intermediately results generated by the mapper and reducer classes. The naming used is as follows. If a folder has letter a appended to it once its the output produced for the first iteration for column 1.If it has double a then results of column 1 for second iteration. Inside all intermediate folders there is a file called part-00000 which has the results. The link to this listing is

http://ec2-54-69-179-42.us-west-2.compute.amazonaws.com:50075/browseDirectory.jsp?dir=%2Ffinal&namenodeInfoPort=50070

A sample screen shot is shown below.



Contents of directory /final

Goto : /final go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
Rules_Generated	dir				2014-12-06 21:22	rwxr-xr-x	ubuntu	supergroup
input	dir				2014-12-05 13:18	rwxr-xr-x	ubuntu	supergroup
output	dir				2014-12-06 21:22	rwxrwxrwx	ubuntu	supergroup
output0	dir				2014-12-05 20:15	rwxrwxrwx	ubuntu	supergroup
output1	dir				2014-12-05 20:16	rwxrwxrwx	ubuntu	supergroup
output2	dir				2014-12-05 20:16	rwxrwxrwx	ubuntu	supergroup
output3	dir				2014-12-05 20:16	rwxrwxrwx	ubuntu	supergroup
outputa0	dir				2014-12-05 20:16	rwxrwxrwx	ubuntu	supergroup
outputa1	dir				2014-12-05 20:17	rwxrwxrwx	ubuntu	supergroup
outputa2	dir				2014-12-05 20:17	rwxrwxrwx	ubuntu	supergroup
outputa3	dir				2014-12-05 20:22	rwxrwxrwx	ubuntu	supergroup
outputa4	dir				2014-12-05 20:17	rwxrwxrwx	ubuntu	supergroup
outputa5	dir				2014-12-05 20:18	rwxrwxrwx	ubuntu	supergroup
outputaa10	dir				2014-12-05 20:22	rwxrwxrwx	ubuntu	supergroup
outputaa11	dir				2014-12-05 20:22	rwxrwxrwx	ubuntu	supergroup
outputaa12	dir				2014-12-05 20:23	rwxrwxrwx	ubuntu	supergroup
outputaa13	dir				2014-12-05 20:23	rwxrwxrwx	ubuntu	supergroup
outputaa14	dir				2014-12-05 20:23	rwxrwxrwx	ubuntu	supergroup
outputaa15	dir				2014-12-05 20:23	rwxrwxrwx	ubuntu	supergroup
outputaa20	dir				2014-12-05 20:24	rwxrwxrwx	ubuntu	supergroup
outputaa21	dir				2014-12-05 20:24	rwxrwxrwx	ubuntu	supergroup
outputaa22	dir				2014-12-05 20:24	rwxrwxrwx	ubuntu	supergroup
outputaa23	dir				2014-12-05 20:25	rwxrwxrwx	ubuntu	supergroup
outputaa30	dir				2014-12-05 20:25	rwxrwxrwx	ubuntu	supergroup

3.Totally 4 nodes or systems were created in amazon aws and the links to each cluster is listed below. It shows the job history and mapper and reducer logs and other important informations.

Master Node:(No data node just calls map reduce jobs and process the outputs)

http://ec2-54-69-140-175.us-west-2.compute.amazonaws.com:50030/jobtracker.jsp

Secondary Name Node:(Supplementary node starts only on failure of Master Node)
http://ec2-54-148-117-133.us-west-2.compute.amazonaws.com:50030/

Slave1:(Data Node 1)
http://ec2-54-68-74-228.us-west-2.compute.amazonaws.com:50060/tasktracker.jsp

Slave 2:(Data Node 2)
http://ec2-54-69-179-42.us-west-2.compute.amazonaws.com:50060/tasktracker.jsp

The master node only creates the mapper and reducer jobs and schedules them which are executer by slaves and the output is again processed at master node and it remains idle at all other times.

Sample screen shots are shown below.

Completed Jobs

Quick Links

Jobid	Started	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information
job_201412061332_0001	Sat Dec 06 15:01:29 UTC 2014	NORMAL	ubuntu	wordcount	100.00%	2	2	100.00%	1	1	NA
job_201412061332_0002	Sat Dec 06 16:01:57 UTC 2014	NORMAL	ubuntu	wordcount	100.00%	2	2	100.00%	1	1	NA
job_201412061332_0003	Sat Dec 06 16:08:42 UTC 2014	NORMAL	ubuntu	wordcount	100.00%	2	2	100.00%	1	1	NA
job_201412061332_0004	Sat Dec 06 17:17:11 UTC 2014	NORMAL	ubuntu	wordcount	100.00%	2	2	100.00%	1	1	NA
job_201412061332_0005	Sat Dec 06 17:18:24 UTC	NORMAL	ubuntu	wordcount	100.00%	2	2	100.00%	1	1	NA

Hadoop job_201412061332_0001 on ec2-54-69-140-175

User: ubuntu
Job Name: wordcount
Job File: hdfs://ec2-54-69-140-175.us-west-2.compute.amazonaws.com:8020/home/ubuntu/hdfstmp/mapred/staging/ubuntu/.staging/job_201412061332_0001/job.xml
Submit Host: ec2-54-69-140-175.us-west-2.compute.amazonaws.com
Submit Host Address: 172.31.47.17
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Sat Dec 06 15:01:29 UTC 2014
Finished at: Sat Dec 06 15:01:46 UTC 2014
Finished in: 16sec
Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	2	0	0	2	0	0 / 0
reduce	100.00%	1	0	0	1	0	0 / 0

	Counter	Map	Reduce	Total
File Input Format Counters	Bytes Read	0	0	561
	SLOTS_MILLIS_MAPS	0	0	8,557
	Launched reduce tasks	0	0	1

Difference between Apache Map Reduce and Elastic Map Reduce?

Apache Map Reduce is an Open source implementation of Google Map Reduce by the Apache foundation. Elastic Map reduce is a closed source implementation by Amazon.

Apache Map Reduce runs on Apache Hadoop which should be installed and configured manually on all amazon ec2 clusters before writing map reduce programs in Java. If we use Elastic Map reduce we need not create Amazon EC2 clusters we can use Amazon EMR to host virtual instances with no work and it will be up in seconds.

Apache Map Reduce and Hadoop doesn't provide any fancy input interfaces and just some default jsp's to view the status of the running jobs and the list of files generated. Elastic Mapreduce is supported by Amazon user interface where everything is click and go and users can add jobs and delete jobs using a nice GUI.

I have used Apache Map Reduce and Hadoop because I hate closed source softwares.

Difference between Apache Mahout and Apache Map Reduce?

Apache Mahout is an open source implementation of all the famous data mining algorithms in map reduce format and all the source codes are available in github and also jar files are available.

Apache Map Reduce is just a java package available through Apache Hadoop which supports coding in map reduce paradigm. Apache mahout also uses Apache Map Reduce.

I have used Apache Map reduce because Id3 is not implemented in Apache Mahout and I wanted to code it from scratch.

Conclusion:

The main lesson learned after using hadoop and map reduce is that all algorithms can't be implemented in a map reduce fashion in an optimal way. Hadoop Map reduce is a bad platform for recursion algorithms because it doesn't support recursion. The only form of recursion supported by Hadoop is tail recursion in which each call is made giving output of the previous call as input so the state is saved at each stage. Implementation of tail recursion for big algorithms is a bit tricky. HDFS file system being the main reason for distributed processing of data has one drawback which is the outputs are return in separate files and hence the user should write separate code to merge these files. Another problem with hadoop is that it is still initial stages of development and each release has many features included and many features removed so there is a huge learning curve.