Annamalai University

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B. E. (C.S.E.) Data Science

V Semester

DSCP507 – MapReduce Programming with Hadoop Lab Manual

Dr.M. THILLAIKARASI
Assistant Professor
Department of Computer Science and Engineering
Annamalai University

Staff-in-charge

Annamalai University
Faculty of Engineering and Technology
Department of Computer Science and Engineering
B. E. (C.S.E.) Data Science - V Semester
DSCP507 - MapReduce Programming with Hadoop Lab

List of Exercises

1. Setting up single node Hadoop cluster in ubuntu.
2. Write a MapReduce application in java to count the number of occurrences of words in a dataset and run it using single node Hadoop cluster.
3. Write a MapReduce application in java to find the highest temperature and lowest temperature of the year.
4. Write a MapReduce application in java to calculate simple aggregate metrics about the weblog dataset.
5. Write a MapReduce application in java to group web server log data and calculate number of hits.
6. Write a MapReduce application in java to calculate frequency distribution for the hit counts of web server log data.
7. Write a MapReduce application in java to calculate histogram of web server log data.
8. Write a MapReduce application in java to calculate the correlation between number of hits and message size of weblog dataset using scatter plot.
9. Write a MapReduce application in java to parse the Tomcat e-mail list dataset that has complex data format using Hadoop by writing an input formatter.
10. Write a MapReduce application in java to join two MBOX-formatted e-mail datasets and show the result in scatter plot.

Assignments

11. Setting up multi node Hadoop cluster in ubuntu.
12. Write a MapReduce application in java to count the number of occurrences of words in a dataset and run it using multi node Hadoop cluster.
13. Study of Execution of Word Count Map Reduce application (count the number of occurrences of words in a dataset) on Amazon Elastic Map Reduce (EMR).
14. Write a MapReduce application in java to perform content-based recommendations for the Amazon product co-purchasing network metadata dataset.
15. Write a MapReduce application in java to assign advertisements to keywords using the AdWords balance algorithm for the Amazon product co-purchasing network metadata dataset.
16. Write a MapReduce application in Python to clean and extract data from the 20news dataset and store the data as a tab-separated file and remove duplicate mail records.

Dr. M.THILLAIKARASI

Staff-in-charge

Ex. No. : 1 Setting up single node Hadoop cluster in Ubuntu

Date :

Aim:

To install and configure a single node Hadoop cluster in Ubuntu operating system, and start and stop services such as dfs and yarn.

Procedure:

Step 1 : Commands for removing lock

```
$ sudo rm /var/lib/apt/lists/lock
$ sudo rm /var/cache/apt/archives/lock
$ sudo rm /var/lib/dpkg/lock
```

Step 2 : Installation of JAVA

# Update the source list

```
$ sudo apt-get update
```

# The OpenJDK project is the default version of Java
# that is provided from a supported Ubuntu repository.

```
$ sudo apt-get install default-jdk

$ java -version
```

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

Step 3 : Adding a dedicated Hadoop user

```
$ sudo addgroup hadoop
```

Adding group `hadoop' (GID 1002) ...
Done.

```
$ sudo adduser --ingroup hadoop hduser
```

```
Adding user `hduser' ...
Adding new user `hduser' (1001) with group `hadoop' ...
Creating home directory `/home/hduser' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hduser
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
```

```
$ sudo adduser hduser sudo
```

```
[sudo] password for Ragupathy:
Adding user `hduser' to group `sudo' ...
Adding user hduser to group sudo
Done.
```

Step 4 : Installing SSH, Create and Setup SSH Certificates

#installing SSH

```
$ sudo apt-get install ssh
```

#verification of SSH and SSHD

```
        $ which ssh
/usr/bin/ssh

        $ which sshd
/usr/sbin/sshd
```

#Switching to hduser

```
        $ su hduser

Password:
```

#SSH certificate generation

```
$ ssh-keygen -t rsa -P ""
```

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
50:6b:f3:fc:0f:32:bf:30:79:c2:41:71:26:cc:7d:e3 hduser@laptop
The key's randomart image is:
+--[ RSA 2048]--- +
/       .oo.o   /
/      . .o=. o /
/     . + .  o . /
/       o =   E /
/       S +     /
/        . +    /
/          O +   /
/           O o  /
/            o..  /
+-----------------+
```

# added to list of authorized keys so that ssh can be used without prompting for a password

```
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

# to check SSH works or not

```
$ ssh localhost
```

Step 5 : Download and Install Hadoop

#switch to hduser and download hadoop 2.6.5

```
$ wget https://archive.apache.org/dist/hadoop/common/hadoop-2.6.5/hadoop-2.6.5.tar.gz
```

#unzip hadoop-2.6.5.tar.gz

```
$ tar xvzf hadoop-2.6.5.tar.gz
```

#make a hadoop dir under usr/local dir

```
$ sudo mkdir -p /usr/local/hadoop
```

# move to hadoop-2.6.5 and move all files and folders in hadoop-2.6.5 dir to hadoop dir

```
$ cd hadoop-2.6.5

$ sudo mv * /usr/local/hadoop
```

#change ownership rights of all files and folders recursively to hduser in hadoop group

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

Step 6 : Hadoop Setup Configuration Files

#The following files should to be modified to complete the Hadoop setup:

# ~/.bashrc
# /usr/local/hadoop/etc/hadoop/hadoop-env.sh
# /usr/local/hadoop/etc/hadoop/core-site.xml
# /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
# /usr/local/hadoop/etc/hadoop/hdfs-site.xml
# /usr/local/hadoop/etc/hadoop/yarn-site.xml

# JAVA_HOME can be found from following command
# from the responce copy /usr/lib/jvm/java-8-openjdk-i386 only for JAVA_HOME

```
$ update-alternatives --config java
```

Step 6.1 : Edit ~/.bashrc file

```
$ sudo gedit ~/.bashrc
```

# insert the following HADOOP VARIABLE export commands in that file

```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-i386
fexport HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export
HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
export HADOOP_CLASSPATH=$(hadoop classpath)
#HADOOP VARIABLES END
```

#execute .bashrc

```
$ source ~/.bashrc
```

Step 6.2 : Edit hadoop-env.sh file

```
$ sudo gedit /usr/local/hadoop/etc/hadoop/hadoop-env.sh
```

#insert the following export command in that file

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-i386
```

Step 6.3 : To override configuration settings of core-site.xml

# create the /app/hadoop/tmp directory to be used to override default settings that Hadoop starts

```
$ sudo mkdir -p /app/hadoop/tmp
```

#change the ownership to hduser in hadoop group

```
$ sudo chown hduser:hadoop /app/hadoop/tmp
```

Step 6. 4 : Edit core-site.xml file

```
$ sudo gedit /usr/local/hadoop/etc/hadoop/core-site.xml
```

#insert the following statements in that file in between <configuration> </configuration>

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

<property>
 <name>fs.default.name</name>
 <value>hdfs://localhost:54310</value>
 <description>The name of the default file system. A URI whose
 scheme and authority determine the FileSystem implementation. The
 uri's scheme determines the config property (fs.SCHEME.impl) naming
 the FileSystem implementation class. The uri's authority is used to
 determine the host, port, etc. for a filesystem.</description>
</property>
```

Step 6.5 : Edit mapred-site.xml

# copy mapred-site.xml.template to mapred-site.xml

```
$ cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
/usr/local/hadoop/etc/hadoop/mapred-site.xml
```

#edit core-site.xml file

```
$ sudo gedit /usr/local/hadoop/etc/hadoop/mapred-site.xml
```

#insert the following statements in that file in between <configuration> </configuration>

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

<property>
 <name>mapreduce.framework.name</name>
 <value>yarn</value>
</property>
```

Step 6.6 : Create namenode, datanode and change the ownership of hadoop_store to hduser in hadoop group

```
$ sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
$ sudo chown -R hduser:hadoop /usr/local/hadoop_store
```

Step 6.7 : Edit hdfs-site.xml

```
$ sudo gedit /usr/local/hadoop/etc/hadoop/hdfs-site.xml
```

#insert the following statements in that file in between <configuration> </configuration>

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is
created.
  The default is used if replication is not specified in create time.
  </description>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
 </property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
 </property>
```

Step 6.8 : Format the New Hadoop Filesystem

```
$ hadoop namenode -format
```

Step 6.9 : Edit yarn-site.xml

```
$ sudo gedit /usr/local/hadoop/etc/hadoop/yarn-site.xml
```

#insert the following statements in that file in between <configuration> </configuration>

```
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>

<property>
<name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
```

Step 7 : Starting Hadoop

```
$ start-all.ah
```

or

```
$ start-dfs.sh
$ start-yarn.sh
```

# to check the execution

```
$ jps
```

# display as follows
14306 DataNode
14660 ResourceManager
14505 SecondaryNameNode
14205 NameNode

14765 NodeManager
15166 Jps

#web UI of the NameNode daemon - Type http://localhost:50070/ as url into our browser
#SecondaryNameNode - Type in http://localhost:50090/status.jsp as url into our browser
#logs - Type in http://localhost:50090/logs/ as url into our browser
#Resouece mamager - Type http://localhost:8088/ as url into our browser

Step 8 : Stoping Hadoop

```
$ stop-all.ah
```

or

```
$ stop-dfs.sh
$ stop-yarn.sh
```

Result:

Thus, a single node Hadoop cluster in Ubuntu operating system has been installed and configured, and started, verified the execution and stopped services such as dfs and yarn.

Ex. No. : 2 Number of occurrences of words in a book dataset

Date :

Aim:

To write a MapReduce application in java to count the number of occurrences of words in a dataset and run it on single node Hadoop cluster.

Source Code:

```
// save the following code in WordCount.java

import java.io.IOException;
import java.util.StringTokenizer;

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


public class WordCount {

public static class TokenizerMapper
      extends Mapper<Object, Text, Text, IntWritable>{

     private final static IntWritable one = new IntWritable(1);
     private Text word = new Text();


     public void map(Object key, Text value, Context context)
                 throws IOException, InterruptedException {
          StringTokenizer itr = new
                           StringTokenizer(value.toString());
          while (itr.hasMoreTokens()) {
          word.set(itr.nextToken());
          context.write(word, one);
      }
    }
  }
  public static class IntSumReducer
         extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

   public void reduce(Text key, Iterable<IntWritable> values,
                                               Context context)
               throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }


public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new
                GenericOptionsParser(conf,args).getRemainingArgs();
    if (otherArgs.length < 2) {
      System.err.println("Usage: wordcount <in> <out>");
      System.exit(2);
    }

    Job job = new Job(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Procedure:

Preparing input – book dataset

Step 1 : Download input files The Hunger Games.txt and Mockingjay.txt from following website and rename as input01.txt and input02.txt and store them in /lab/wc directory

https://sites.google.com/site/the74thhungergamesbyced/download-the-hunger-games-trilogy-e-book-txt-file

Executing a WordCount MapReduce program in Hadoop

Note : Create /lab/wc directory and save source code in WordCount.java in it.

Step 1 : Compilation of a java program

```
$ javac -classpath $HADOOP_CLASSPATH WordCount.java
```

Step 2 : Creation of jar file

```
$ jar -cvf wc.jar *.class
```

Step 3 : Creation of directories

```
$ hdfs dfs -mkdir /user
$ hdfs dfs -mkdir /user/wc
$ hdfs dfs -mkdir /user/wc/input
```

Step 4 : Copying inputfiles from local directory to Hadoop

```
$ hadoop fs -copyFromLocal wcinput1.txt /user/wc/input
$ hadoop fs -copyFromLocal wcinput2.txt /user/wc/input
```

Step 5 : Executing job in hadoop

```
$ hadoop jar wc.jar WordCount /user/wc/input /user/wc/output
```

Step 6 : Copying output files from Hadoop to local directory

```
$ hadoop fs -copyToLocal /user/wc/output/*
```

Step 7 : Viewing the output file

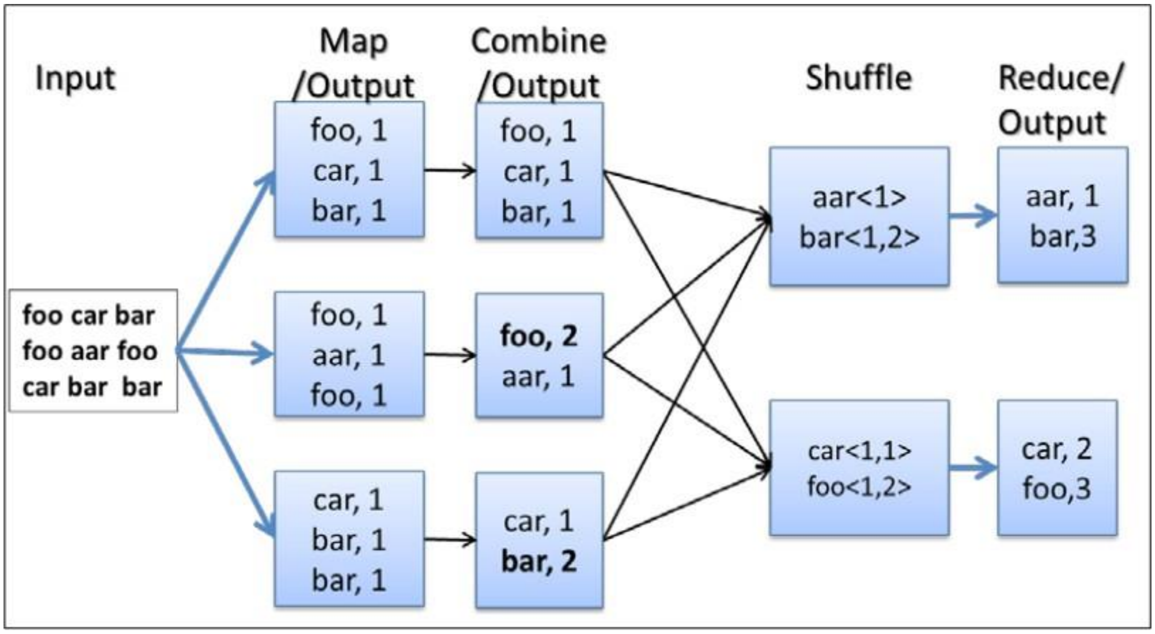
```
$ gedit part-r-00000
```

Step 8 : Remove the output filles and directory from hadoop

```
$ hdfs dfs -rm /user/wc/output/*
$ hdfs dfs -rmdir /user/wc/output
```

Note : only by removing the output files and directory from hadoop, we can use the above procedure for executing the job again. If input need to be changed, remove the input files and directory also, and do changes in Step 3 and 4.

Illustration of Entire Process:



Sample Input and Output:

Sample Content of input file : wcinput1.txt

```
The Hunger Games


The Hunger Games 1by Suzanne Collins

PART I"THE TRIBUTES"

1.

When I wake up, the other side of the bed is cold. My fingers
stretch out, seeking Prims warmth but finding only the rough canvas
cover of the mattress. She must have had bad dreams and climbed in
with our mother. Of course, she did. This is the day of the
reaping. I prop myself up on one elbow. Theres enough light in the
bedroom to see them. My little sister, Prim, curled up on her side,
cocooned in my mothers body, their cheeks pressed together. In
sleep, my mother looks younger, still worn but not so beaten-down.
Prims face is as fresh as a raindrop, as lovely as the primrose for
which she was named. My mother was very beautiful once, too. Or so
.
.
.
```

Sample Content of input file : wcinput2.txt

```
MOCKINGJAY
SUZANNE COLLINS

PART I
"THE ASHES"
1
I stare down at my shoes, watching as a fine layer of
ash settles on the worn leather. This is where the bed I
shared with my sister, Prim, stood. Over there was the
kitchen table. The bricks of the chimney, which collapsed in
a charred heap, provide a point of reference for the rest of
the house. How else could I orient myself in this sea of
gray?
Almost nothing remains of District 12. A month ago,
the Capitol's firebombs obliterated the poor coal miners'
.
.
.
```

Sample Content of output file : part-r-00000

```
Awfully    1
Axminster  1
Ay    1
Ay!   2
Ay,   9
Ay.   4
Ayes  1
Azazel,    1
Azotes.    1
Aztec 1
Aztecs,    1
B     6
B,    2
B.    27
B.)   3
B.,   2
B.A.  1
B.C.  1
BABES 1
BABY  1
BALANCE    1
BANTAM     1
BARBER:    1
BARRY:     5
BATTLES    1
BAWD: 8
BE    1
BEATITUDES:     1
BEAUFOY:   4
BEFORE     1
BELIEF     1
BELLA,     2
BELLA:     23
BELLHANGER:     1
BELLINGHAM:     5
.
.
.
```

Result:

Thus, a MapReduce application has been developed in java to count the number of occurrences of words in a dataset, executed on single node Hadoop cluster and responses have been verified.

Ex. No. : 3 Highest and lowest temperatures of a weather dataset

Date :

Aim:

To write a MapReduce application in java to find the highest temperature and lowest temperature of the year from a weather dataset and run it on single node Hadoop cluster.

Source Code:

```
// save the following code in WeatherJob.java

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

import java.io.IOException;

public class WeatherJob {

    public static void main(String[] args) throws Exception {
        if (args == null || args.length < 2) {
            System.err.println("Parameter Errors!
                               Usages:<inputpath> <outputpath>");
            System.exit(-1);
        }

        Path inputPath = new Path(args[0]);
        Path outputPath = new Path(args[1]);

        Configuration conf = new Configuration();
        String jobName = WeatherJob.class.getSimpleName();
        Job job = Job.getInstance(conf, jobName);
        job.setJarByClass(WeatherJob.class);

        FileInputFormat.setInputPaths(job, inputPath);
        job.setInputFormatClass(TextInputFormat.class);
        job.setMapperClass(WeatherMapper.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(DoubleWritable.class);
        outputpath.getFileSystem(conf).delete(outputpath, true);
        FileOutputFormat.setOutputPath(job, outputPath);
        job.setOutputFormatClass(TextOutputFormat.class);
        job.setReducerClass(WeatherReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(DoubleWritable.class);
        job.setNumReduceTasks(1);
        job.waitForCompletion(true);
    }

    public static class WeatherMapper extends
                 Mapper<LongWritable, Text, Text, DoubleWritable> {
        @Override
      protected void map(LongWritable k1, Text v1, Context context)
                     throws IOException, InterruptedException {
            String line = v1.toString();
            Double max = null;
            Double min = null;
            try {
                max = Double.parseDouble(line.substring(103, 108));
                min = Double.parseDouble(line.substring(111, 116));
            } catch (NumberFormatException e) {
                return;
            }

           context.write(new Text("MAX"), new DoubleWritable(max));
           context.write(new Text("MIN"), new DoubleWritable(min));
        }
    }

    public static class WeatherReducer extends
              Reducer<Text, DoubleWritable, Text, DoubleWritable> {
        @Override
        protected void reduce(Text k2, Iterable<DoubleWritable>
                                           v2s, Context context)
                       throws IOException, InterruptedException {
            double max = Double.MIN_VALUE;
            double min = Double.MAX_VALUE;
            if ("MAX".equals(k2.toString())) {
                for (DoubleWritable v2 : v2s) {
                    double tmp = v2.get();
                    if (tmp > max) {
                        max = tmp;
                    }
                }
            } else {
                for (DoubleWritable v2 : v2s) {
                    double tmp = v2.get();
                    if (tmp < min) {
                        min = tmp;
                    }
                }
            }
            context.write(k2, "MAX".equals(k2.toString()) ? new
                    DoubleWritable(max) : new DoubleWritable(min));
        }
    }
}
```

Procedure:

Preparing input - weather dataset

Step 1 : Download input files from following website

```
$ wget ftp://ftp.ncdc.noaa.gov/pub/data/gsod/2021/010010-
      99999-2021.op.gz
$ wget ftp://ftp.ncdc.noaa.gov/pub/data/gsod/2021/010014-
      99999-2021.op.gz
```

Step 2 : Extract them using gunzip

```
$ gunzip -d 010010-99999-2021.op.gz
$ gunzip -d 010014-99999-2021.op.gz
```

Step 3 : Move them in /home/hduser/lab/weather directory

```
$ mv  010010-99999-2021.op /home/hduser/lab/weather
$ mv  010014-99999-2021.op /home/hduser/lab/weather
```

Executing a WeatherJob MapReduce program in Hadoop

Note : Create /lab/wc directory and save source code in WeatherJob.java in it.

Step 1 : Compilation of a java program

```
$ javac -classpath $HADOOP_CLASSPATH WeatherJob.java
```

Step 2 : Creation of jar file

```
$ jar -cvf weather.jar *.class
```

Step 3 : Creation of directories

```
$ hdfs dfs -mkdir /user/weather
$ hdfs dfs -mkdir /user/weather/input
```

Step 4 : Copying inputfiles from local directory to Hadoop

```
$ hadoop fs -copyFromLocal 010010-99999-2021.op
        /user/weather/input
$ hadoop fs -copyFromLocal 010014-99999-2021.op
        /user/weather/input
```

Step 5 : Executing job in hadoop

```
$ hadoop jar weather.jar WeatherJob /user/weather/input
         /user/weather/output
```

Step 6 : Copying output files from Hadoop to local directory

```
$ hadoop fs -copyToLocal /user/weather/output/*
```

Step 7 : Viewing the output file

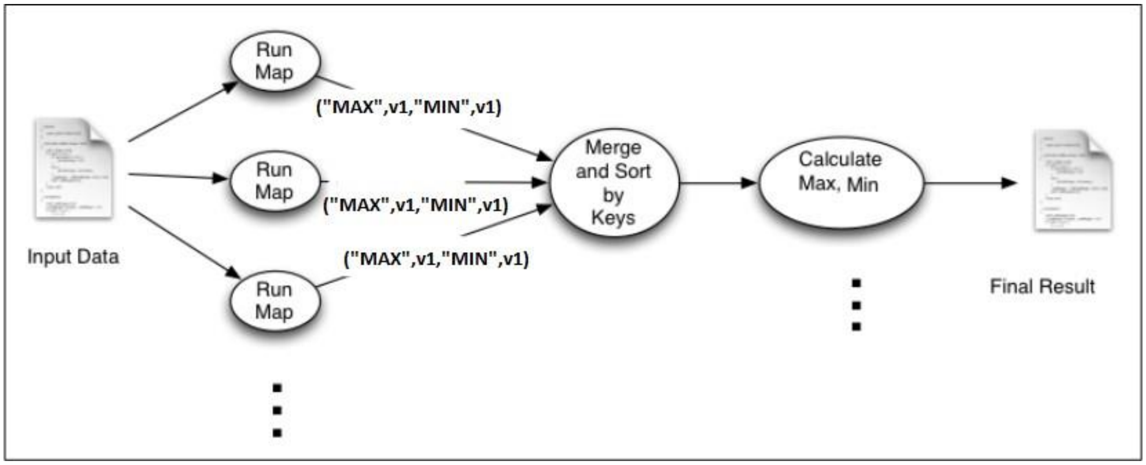
```
$ gedit part-r-00000
```

Step 8 : Remove the output filles and directory from hadoop

```
$ hdfs dfs -rm /user/weather/output/*
$ hdfs dfs -rmdir /user/weather/output
```

Note : only by removing the output files and directory from hadoop, we can use the above procedure for executing the job again. If input need to be changed, remove the input files and directory also, and do changes in Step 3 and 4.

Illustration of Entire Process:



Sample Input and Output:

Sample Content of input file : 010010-99999-2021

```
STN--- WBAN    YEARMODA     TEMP       DEWP       SLP        STP
VISIB    WDSP    MXSPD    GUST      MAX      MIN    PRCP    SNDP
FRSHTT
010010 99999  20210101    23.4 24    15.7 24  1017.7 24  1016.5 24
28.0  6   10.9 24   22.1   39.4    26.2    20.1   0.04G 999.9
000000
010010 99999  20210102    31.5 23    30.1 23  1017.4 23  1016.2 23
4.9  6   12.5 23   21.6   25.8    33.8    20.1   0.06G 999.9
011000
010010 99999  20210103    35.0 24    34.3 24  1011.4 24  1010.2 24
3.1  6   12.2 24   19.0   24.9    39.9    32.7   0.11G 999.9
010000
010010 99999  20210104    35.4 22    29.8 22  1007.6 22  1006.4 22
6.1  6   23.1 22   41.4   56.7    37.9    33.6   0.01G 999.9
010000
010010 99999  20210105    27.5 24    19.4 24  1015.1 24  1013.9 24
13.5  6   12.8 24   25.1   37.9    35.4    21.7   0.00G 999.9
001000
010010 99999  20210106    23.6 24    16.2 24  1015.1 24  1013.9 24
8.6  6   17.0 24   35.0   51.3    25.0    21.7  99.99  999.9
001000
010010 99999  20210107    18.7 24    11.7 24  1015.1 24  1013.9 24
4.1  6   20.7 24   36.5   56.1    25.5    14.4   0.03G 999.9
001000
▪
▪
▪
```

Sample Content of input file : 010014-99999-2021

```
STN--- WBAN   YEARMODA    TEMP       DEWP      SLP        STP
VISIB      WDSP     MXSPD   GUST    MAX     MIN   PRCP   SNDP
FRSHTT
010014 99999  20210101    34.7  6    30.2  6  9999.9  0  9999.9  0
6.2  6    4.7  6    7.0  999.9    35.6*   33.8*  0.00I 999.9
000000
010014 99999  20210103    35.1  8    22.8  8  9999.9  0  9999.9  0
6.2  8    3.4  8    7.0  999.9    37.4*   30.2*  0.00I 999.9
000000
010014 99999  20210104    30.6 14    22.5 14  9999.9  0  9999.9  0
6.2 14    3.6 14    7.0  999.9    35.6*   26.6*  0.00I 999.9
000000
010014 99999  20210105    31.7  7    19.9  7  9999.9  0  9999.9  0
6.2  7    2.1  7    2.9  999.9    35.6*   30.2*  0.00I 999.9
000000
010014 99999  20210106    29.4 11    21.9 11  9999.9  0  9999.9  0
6.2 11    7.7 11    9.9  999.9    32.0*   28.4*  0.00I 999.9
000000
010014 99999  20210107    28.7 11    18.4 11  9999.9  0  9999.9  0
6.2 11    9.2 11   11.1  999.9    30.2*   28.4*  0.00I 999.9
000000
010014 99999  20210108    29.2  9    17.2  9  9999.9  0  9999.9  0
6.2  9    5.5  9    8.9  999.9    32.0*   26.6*  0.00I 999.9
000000
.
.
.
```

Sample Content of output file : part-r-00000

```
MAX   999.9
MIN   9.1
```

Result:

Thus, a MapReduce application in java has been developed to find the highest temperature and lowest temperature of the year from a weather dataset, executed on single node Hadoop cluster and responses have been verified.

Ex. No. : 4 Simple aggregate metrics about the weblog dataset

Date :

Aim:

To write a MapReduce application in java to calculate simple aggregate metrics about the weblog dataset executed on single node Hadoop cluster and run it on single node Hadoop cluster.

Source Code:

```
// save the following code in MsgSizeAggregateMapReduce.java

import java.io.IOException;
import java.util.Iterator;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

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


public class MsgSizeAggregateMapReduce
          extends Configured implements Tool {

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

     @Override
     public int run(String[] args) throws Exception {

          if (args.length != 2) {
               System.err.println("Usage:  <input_path>
                                         <output_path>");
               System.exit(-1);
          }
```

```
        String inputPath = args[0];
        String outputPath = args[1];

        Job job = Job.getInstance(getConf(),
                                 "WebLogMessageSizeAggregator");
        job.setJarByClass(MsgSizeAggregateMapReduce.class);
        job.setMapperClass(AMapper.class);
        job.setReducerClass(AReducer.class);
        job.setNumReduceTasks(1);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.setInputPaths(job, new Path(inputPath));
        FileOutputFormat.setOutputPath(job, new
                                            Path(outputPath));

        int exitStatus = job.waitForCompletion(true) ? 0 : 1;
        return exitStatus;
    }

    public static class AMapper extends
                 Mapper<Object, Text, Text, IntWritable> {
        public static final Pattern httplogPattern = Pattern
                   .compile("([^\\s]+) - - \\[(.+)\\]\"([^\\s]+)
                   (/[^\\s]*) HTTP/[^\\s]+\" [^\\s]+ ([0-9]+)");

        public void map(Object key, Text value, Context context)
                   throws IOException, InterruptedException {
            Matcher matcher =
                       httplogPattern.matcher(value.toString());
            if (matcher.matches()) {
                int size =
                             Integer.parseInt(matcher.group(5));
                context.write(new Text("msgSize"), new
                                             IntWritable(size));
            }
        }
    }

    public static class AReducer extends
                Reducer<Text, IntWritable, Text, IntWritable> {
        public void reduce(Text key, Iterable<IntWritable>
                      values,Context context)
                    throws IOException, InterruptedException {
            double tot = 0;
            int count = 0;
            int min =  Integer.MAX_VALUE;
            int max = 0;
            Iterator<IntWritable> iterator =
                                             values.iterator();
                while (iterator.hasNext()) {
                      int value = iterator.next().get();
                      tot = tot + value;
                      count++;
                      if (value < min) {
                           min = value;
                      }
                      if (value > max) {
                           max = value;
                      }
                }
                context.write(new Text("Mean"), new
                                   IntWritable((int) tot / count));
                context.write(new Text("Max"), new
                                                 IntWritable(max));
                context.write(new Text("Min"), new
                                                 IntWritable(min));
           }
      }
}
```

Procedure:

Preparing input - weblog dataset

Step 1 : Download the NASA weblog dataset

```
$ wget ftp://ita.ee.lbl.gov/traces/NASA_access_log_Jul95.gz
```

Step 2 : Unzip the gz file

```
$ gunzip -d  NASA_access_log_Jul95.gz
```

Step 3 : Move the NASA_access_log_Jul95 to /home/hduser/lab/aggregate directory

```
$ mv NASA_access_log_Jul95 /home/hduser/lab/aggregate
```

Executing a Simple Aggregate MapReduce program in Hadoop

Note : Create /lab/aggregate directory and save source code in MsgSizeAggregateMapReduce.java in it.

Step 1 : Compilation of a java program

```
$ javac -classpath $HADOOP_CLASSPATH MsgSizeAggregateMapReduce.java
```

Step 2 : Creation of jar file

```
$ jar -cvf aggregate.jar *.class
```

Step 3 : Creation of directories

```
$ hdfs dfs -mkdir /user/ag
$ hdfs dfs -mkdir /user/ag/input
```

Step 4 : Copying inputfiles from local directory to Hadoop

```
$ hadoop fs -copyFromLocal NASA_access_log_Jul95 /user/ag/input
```

Step 5 : Executing job in hadoop

```
$ hadoop jar aggregate.jar MsgSizeAggregateMapReduce
         /user/ag/input /user/ag/output
```

Step 6 : Copying output files from Hadoop to local directory

```
$ hadoop fs -copyToLocal /user/ag/output/*
```

Step 7 : Viewing the output file

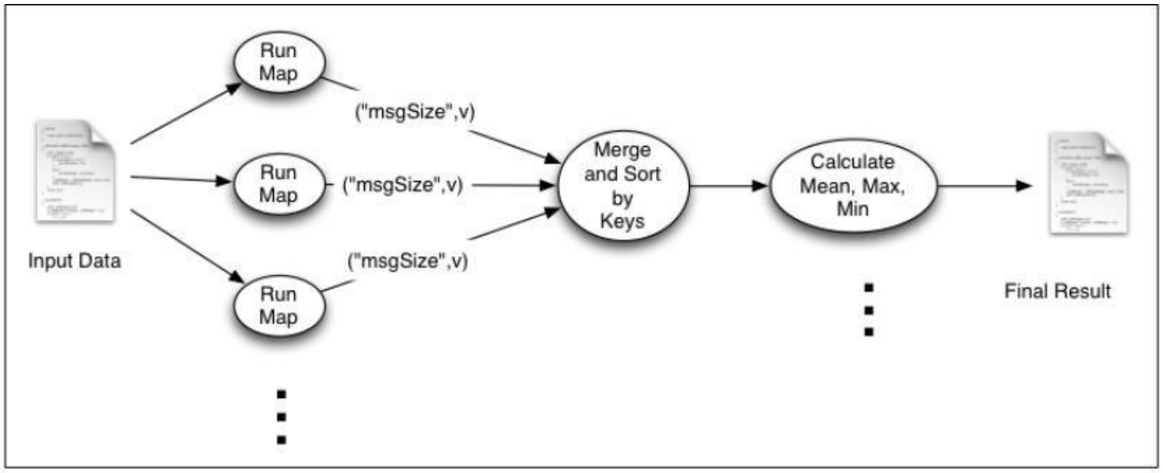
```
$ geidt part-r-00000
```

Step 8 : Remove the output filles and directory from hadoop

```
$ hdfs dfs -rm /user/ag/output/*
$ hdfs dfs -rmdir /user/ag/output
```

Note : only by removing the output files and directory from hadoop, we can use the above procedure for executing the job again. If input need to be changed, remove the input files and directory also, and do changes in Step 3 and 4.

Illustration of Entire Process:



Sample Input and Output:

Sample Content of input file : NASA_access_log_Jul95

```
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/
HTTP/1.0" 200 6245
unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET
/shuttle/countdown/ HTTP/1.0" 200 3985
199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET
/shuttle/missions/sts-73/mission-sts-73.html HTTP/1.0" 200 4085
burger.letters.com - - [01/Jul/1995:00:00:11 -0400] "GET
/shuttle/countdown/liftoff.html HTTP/1.0" 304 0
199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET
/shuttle/missions/sts-73/sts-73-patch-small.gif HTTP/1.0" 200 4179
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET
/images/NASA-logosmall.gif HTTP/1.0" 304 0
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET
/shuttle/countdown/video/livevideo.gif HTTP/1.0" 200 0
205.212.115.106 - - [01/Jul/1995:00:00:12 -0400] "GET
/shuttle/countdown/countdown.html HTTP/1.0" 200 3985
```

```
d104.aa.net - - [01/Jul/1995:00:00:13 -0400] "GET
/shuttle/countdown/ HTTP/1.0" 200 3985
129.94.144.152 - - [01/Jul/1995:00:00:13 -0400] "GET / HTTP/1.0"
200 7074
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET
/shuttle/countdown/count.gif HTTP/1.0" 200 40310
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET
/images/NASA-logosmall.gif HTTP/1.0" 200 786
...
```

Sample Content of output file : part-r-00000

```
Mean  1150
Max   6823936
Min   0
```

Result:

Thus, a MapReduce application has been developed in java to count the number of occurrences of words in a dataset, executed on single node Hadoop cluster and responses have been verified.

Ex. No. : 5 Grouping of web server log data and calculating number of hits
Date :

Aim:

To write a MapReduce application in java to group web server log data and calculate number of hits and run it on single node Hadoop cluster.

Source Code:

```
// save the following code in HitCountMapReduce.java

import java.io.IOException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

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


public class HitCountMapReduce extends Configured implements Tool {
      public static void main(String[] args) throws Exception {
           int res = ToolRunner.run(new Configuration(), new
                                        HitCountMapReduce(), args);
           System.exit(res);
      }

      @Override
      public int run(String[] args) throws Exception {

           if (args.length < 2) {
                 System.err.println("Usage:  <input_path>
                               <output_path> <num_reduce_tasks>");
                 System.exit(-1);
           }

           String inputPath = args[0];
           String outputPath = args[1];
           int numReduce = 1;
```

```
        if (args.length == 3)
            numReduce = Integer.parseInt(args[2]);

        Job job = Job.getInstance(getConf(),
                                   "WeblogHitsByLinkProcessor");
        job.setJarByClass(HitCountMapReduce.class);
        job.setMapperClass(AMapper.class);
        job.setReducerClass(AReducer.class);
        job.setNumReduceTasks(numReduce);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.setInputPaths(job, new Path(inputPath));
        FileOutputFormat.setOutputPath(job, new
                                               Path(outputPath));

        int exitStatus = job.waitForCompletion(true) ? 0 : 1;
        return exitStatus;
    }

    public static class AMapper extends
                   Mapper<Object, Text, Text, IntWritable> {
        public static final Pattern httplogPattern = Pattern
                   .compile("([^\\s]+) - - \\[(.+)\\]\"([^\\s]+)
                   (/[^\\s]*) HTTP/[^\\s]+\" [^\\s]+ ([0-9]+)");

        private final static IntWritable one = new
                                                 IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context)
                    throws IOException, InterruptedException {
            Matcher matcher =
                      httplogPattern.matcher(value.toString());
            if (matcher.matches()) {
                String linkUrl = matcher.group(4);
                word.set(linkUrl);
                context.write(word, one);
            }
        }
    }

    public static class AReducer extends
                   Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable>
                    values, Context context)
                    throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }
}
```

Procedure:

Preparing input - weblog dataset

Step 1 : Download the NASA weblog dataset

```
$ wget ftp://ita.ee.lbl.gov/traces/NASA_access_log_Jul95.gz
```

Step 2 : Unzip the gz file

```
$ gunzip -d  NASA_access_log_Jul95.gz
```

Step 3 : Move the NASA_access_log_Jul95 to /home/hduser/lab/hc directory

```
$ mv NASA_access_log_Jul95 /home/hduser/lab/hc
```

Executing a hit count MapReduce program in Hadoop

Note : Create /lab/hc directory and save source code in HitCountMapReduce.java in it.

Step 1 : Compilation of a java program

```
$ javac -classpath $HADOOP_CLASSPATH HitCountMapReduce.java
```

Step 2 : Creation of jar file

```
$ jar -cvf hc.jar *.class
```

Step 3 : Creation of directories

```
$ hdfs dfs -mkdir /user/hc
$ hdfs dfs -mkdir /user/hc/input
```

Step 4 : Copying input files from local directory to Hadoop

```
$ hadoop fs -copyFromLocal NASA_access_log_Jul95 /user/hc/input
```

Step 5 : Executing job in hadoop

```
$  hadoop  jar  hc.jar  HitCountMapReduce  /user/hc/input
           /user/hc/output
```

Step 6 : Copying output files from Hadoop to local directory

```
$ hadoop fs -copyToLocal /user/hc/output/*
```

Step 7 : Viewing the output file

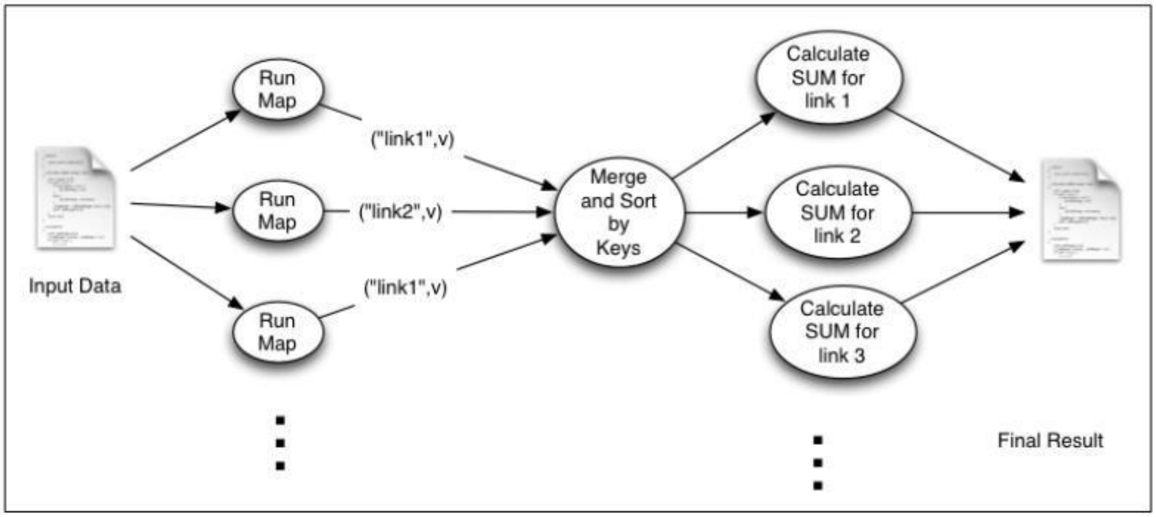
```
$ gedit part-r-00000
```

Step 8 : Remove the output filles and directory from hadoop

```
$ hdfs dfs -rm /user/hc/output/*
$ hdfs dfs -rmdir /user/hc/output
```

Note : only by removing the output files and directory from hadoop, we can use the above procedure for executing the job again. If input need to be changed, remove the input files and directory also, and do changes in Step 3 and 4.

Illustration of Entire Process:



Sample Input and Output:

Sample Content of input file : NASA_access_log_Jul95

```
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/
HTTP/1.0" 200 6245
unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET
/shuttle/countdown/ HTTP/1.0" 200 3985
199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET
/shuttle/missions/sts-73/mission-sts-73.html HTTP/1.0" 200 4085
burger.letters.com - - [01/Jul/1995:00:00:11 -0400] "GET
/shuttle/countdown/liftoff.html HTTP/1.0" 304 0
199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET
/shuttle/missions/sts-73/sts-73-patch-small.gif HTTP/1.0" 200 4179
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET
/images/NASA-logosmall.gif HTTP/1.0" 304 0
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET
/shuttle/countdown/video/livevideo.gif HTTP/1.0" 200 0
205.212.115.106 - - [01/Jul/1995:00:00:12 -0400] "GET
/shuttle/countdown/countdown.html HTTP/1.0" 200 3985
d104.aa.net - - [01/Jul/1995:00:00:13 -0400] "GET
/shuttle/countdown/ HTTP/1.0" 200 3985
129.94.144.152 - - [01/Jul/1995:00:00:13 -0400] "GET / HTTP/1.0"
200 7074
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET
/shuttle/countdown/count.gif HTTP/1.0" 200 40310
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET
/images/NASA-logosmall.gif HTTP/1.0" 200 786
...
```

Sample Content of output file : part-r-00000

```
//biomed/climate/gif/f16pcfinmed.gif  1
//biomed/gif/   1
//biomed/gif/aerial.gif    1
//elv/bakgro.gif 2
//elv/elvhead2.gif    1
//elv/elvhead3.gif    1
//elv/elvpage.htm     2
//elv/endball.gif     1
//elv/vidpicp.htm     1
//elv/whnew.htm 1
//facilities/spaceport.html 1
//ksc.html 3
//shuttle/missions/missions.html  4
//shuttle/missions/sts-70/images/woodpecker-on-et.jpg  1
/Harvest/  2
/Harvest/brokers/WWW/admin/admin.html 3
/Harvest/brokers/WWW/query.html  27
/Harvest/brokers/WWW/summary.html     4
/Harvest/brokers/queryhelp.html  1
.
.
.
```

Result:

Thus, a MapReduce application has been developed in java to count the number of occurrences of words in a dataset, executed on single node Hadoop cluster and responses have been verified.

Ex. No. : 6 Frequency distribution for the hit counts of web server log data
Date :

Aim:

To write a MapReduce application in java to calculate frequency distribution for the hit counts of web server log data and run it on single node Hadoop cluster.

Source Code:

```
// save the following code in FrequencyDistributionMapReduce.java

import java.io.IOException;
import java.util.Iterator;

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


public class FrequencyDistributionMapReduce
             extends Configured implements Tool {

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

      @Override
      public int run(String[] args) throws Exception {

           if (args.length != 2) {
                System.err.println("Usage:  <input_path>
                                            <output_path>");
                System.exit(-1);
           }
```

```
        String inputPath = args[0];
        String outputPath = args[1];

        Job job = Job.getInstance(getConf(),
                    "WeblogFrequencyDistributionProcessor");
        job.setJarByClass(FrequencyDistributionMapReduce.class);
        job.setMapperClass(AMapper.class);
        job.setReducerClass(AReducer.class);
        job.setNumReduceTasks(1);

        job.setMapOutputKeyClass(IntWritable.class);
        job.setMapOutputValueClass(Text.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.setInputPaths(job, new Path(inputPath));
        FileOutputFormat.setOutputPath(job, new
                                    Path(outputPath));

        int exitStatus = job.waitForCompletion(true) ? 0 : 1;
        return exitStatus;
    }

    public static class AMapper
             extends Mapper<Object, Text, IntWritable, Text> {

        public void map(Object key, Text value, Context context)
                 throws IOException, InterruptedException {
            String[] tokens = value.toString().split("\\s");
            context.write(new
                   IntWritable(Integer.parseInt(tokens[1])),
                                  new Text(tokens[0]));
        }
    }


    public static class AReducer
         extends Reducer<IntWritable, Text, Text, IntWritable> {
        public void reduce(IntWritable key, Iterable<Text>
                 values, Context context)
                 throws IOException,  InterruptedException {
            Iterator<Text> iterator = values.iterator();
            while (iterator.hasNext()) {
                context.write(iterator.next(), key);
            }
        }
    }
}
```

Procedure:

Executing a frequency distribution MapReduce program in Hadoop

Note : Create /lab/fd directory and save source code in FrequencyDistributionMapReduce.java in it.

Step 1 : Compilation of a java program

```
$ javac -classpath $HADOOP_CLASSPATH
        FrequencyDistributionMapReduce.java
```

Step 2 : Creation of jar file

```
$ jar -cvf fd.jar *.class
```

Step 3 : Creation of directories

```
$ hdfs dfs -mkdir /user/fd
$ hdfs dfs -mkdir /user/fd/input
```

Step 4 : Copying input file from local directory to Hadoop

```
$ cp /home/hduser/lab/hc/part-r-00000
    /home/hduser/lab/fd/hit-count-output
$ hadoop fs -copyFromLocal hit-count-output
        /user/fd/input
```

Step 5 : Executing job in hadoop

```
$ hadoop jar fd.jar FrequencyDistributionMapReduce
           /user/fd/input /user/fd/output
```

Step 6 : Copying output files from Hadoop to local directory

```
$ hadoop fs -copyToLocal /user/fd/output/*
```

Step 7 : Viewing the output file

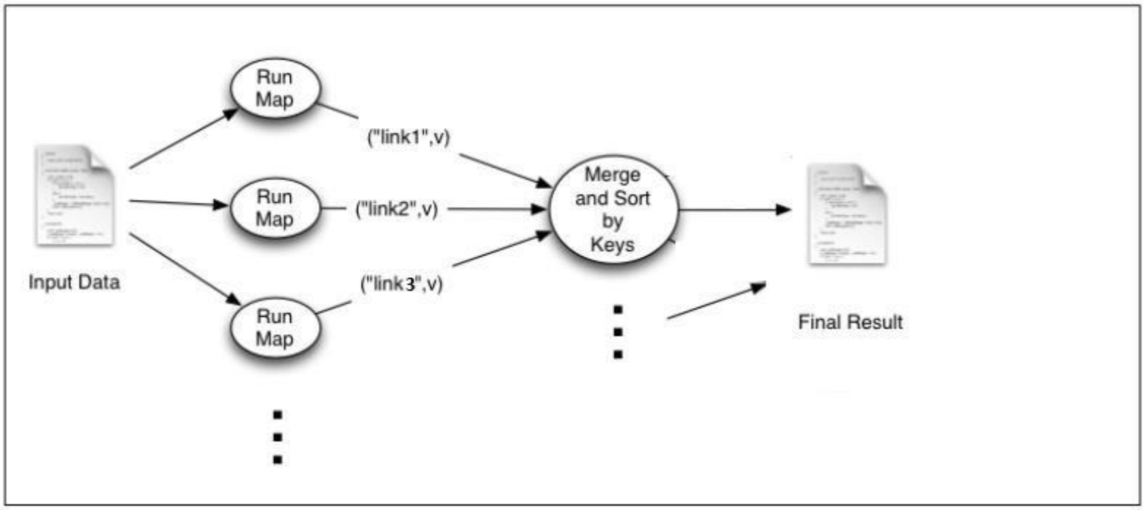
```
$ gedit part-r-00000
```

Step 8 : Remove the output filles and directory from hadoop

```
$ hdfs dfs -rm /user/fd/output/*
$ hdfs dfs -rmdir /user/fd/output
```

Note : only by removing the output files and directory from hadoop, we can use the above procedure for executing the job again. If input need to be changed, remove the input files and directory also, and do changes in Step 3 and 4.

Illustration of Entire Process:



Sample Input and Output:

Sample Content of input file : hit-count-output

```
//biomed/climate/gif/f16pcfinmed.gif  1
//biomed/gif/   1
//biomed/gif/aerial.gif    1
//elv/bakgro.gif 2
//elv/elvhead2.gif    1
//elv/elvhead3.gif    1
//elv/elvpage.htm     2
//elv/endball.gif     1
//elv/vidpicp.htm     1
//elv/whnew.htm 1
```

```
//facilities/spaceport.html 1
//ksc.html 3
//shuttle/missions/missions.html  4
//shuttle/missions/sts-70/images/woodpecker-on-et.jpg  1
/Harvest/  2
/Harvest/brokers/WWW/admin/admin.html  3
/Harvest/brokers/WWW/query.html  27
/Harvest/brokers/WWW/summary.html      4
/Harvest/brokers/queryhelp.html  1
.
.
.
```

Sample Content of output file : part-r-00000

```
/shuttle/missions/sts-70/images/KSC-95EC-0622.jpg 152
/cgi-bin/geturlstats.pl    153
/msfc/onboard/redball.gif  154
/msfc/onboard/colorbar.gif 154
/software/ 155
/shuttle/missions/sts-43/sts-43-patch-small.gif  155
/shuttle/movies/ 156
/elv/ATLAS_CENTAUR/atlcent.htm   156
/shuttle/missions/sts-44/sts-44-patch-small.gif  156
/shuttle/missions/sts-54/mission-sts-54.html     156
/history/apollo/apollo-11/images/69HC905.GIF     157
/history/apollo/apollo-12/ 157
/shuttle/missions/sts-67/images/KSC-95EC-0392.jpg157
/images/crawler.gif   158
/history/apollo/apollo-1/apollo-1-patch.jpg   159
/history/apollo/apollo-8/images/  160
/shuttle/missions/sts-56/sts-56-patch-small.gif  160
/facts/faq01.html     160
/shuttle/countdown/lps/c-1/c-1.html    161
/shuttle/missions/sts-70/images/KSC-95EC-0852.txt161
/images/vab-medium.gif     161
.
.
.
```

Result:

Thus, a MapReduce application has been developed in java to calculate frequency distribution for the hit counts of web server log data, executed on single node Hadoop cluster and responses have been verified.

Ex. No. : 7 Histogram of web server log data

Date :

Aim:

To write a MapReduce application in java to calculate histogram of web server log data and run it on single node Hadoop cluster.

Source Code:

```
// save the following code in HistogramGenerationMapReduce.java

import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

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


public class HistogramGenerationMapReduce
             extends Configured implements Tool {

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

```
@Override
public int run(String[] args) throws Exception {

      if (args.length != 2) {
           System.err.println("Usage:  <input_path>
                           <output_path> <num_reduce_tasks>");
           System.exit(-1);
      }

      String inputPath = args[0];
      String outputPath = args[1];
      int numReduce = 1;
      if (args.length == 3)
           numReduce = Integer.parseInt(args[2]);

      Job job = Job.getInstance(getConf(),
                          "WeblogTimeOfDayHistogramCreator");
      job.setJarByClass(HistogramGenerationMapReduce.class);
      job.setMapperClass(AMapper.class);
      job.setReducerClass(AReducer.class);
      job.setNumReduceTasks(numReduce);

      job.setMapOutputKeyClass(IntWritable.class);
      job.setMapOutputValueClass(IntWritable.class);
      FileInputFormat.setInputPaths(job, new Path(inputPath));
      FileOutputFormat.setOutputPath(job, new
                                           Path(outputPath));

      int exitStatus = job.waitForCompletion(true) ? 0 : 1;
      return exitStatus;
}

public static class AMapper
      extends Mapper<Object, Text, IntWritable, IntWritable> {
      public static final Pattern httplogPattern = Pattern
                 .compile("([^\\s]+) - - \\[(.+)\\]\"([^\\s]+)
                 (/[^\\s]*) HTTP/[^\\s]+\" [^\\s]+ ([0-9]+)");
      public static SimpleDateFormat dateFormatter = new
                 SimpleDateFormat("dd/MMMMM/yyyy:hh:mm:ss z");

      private final static IntWritable one = new
                                              IntWritable(1);

      public void map(Object key, Text value, Context context)
                  throws IOException, InterruptedException {
           try {
                Matcher matcher =
                    httplogPattern.matcher(value.toString());
                if (matcher.matches()) {
                     String timeAsStr = matcher.group(2);
                    Date time =
                            dateFormatter.parse(timeAsStr);
                    Calendar calendar =
                            GregorianCalendar.getInstance();
                    calendar.setTime(time);
                    int hour =
                         calendar.get(Calendar.HOUR_OF_DAY);
                    context.write(new IntWritable(hour),
                                                         one);
                }
            } catch (ParseException e) {
                e.printStackTrace();
            }
        }
    }


    public static class AReducer
                   extends Reducer<IntWritable, IntWritable,
                   IntWritable, IntWritable> {
        public void reduce(IntWritable key,
                     Iterable<IntWritable> values,
                     Context context) throws IOException,
                     InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            context.write(key, new IntWritable(sum));
        }
    }
}
```

Procedure:

Preparing input - weblog dataset

Step 1 : Download the NASA weblog dataset

```
$ wget ftp://ita.ee.lbl.gov/traces/NASA_access_log_Jul95.gz
```

Step 2 : Unzip the gz file

```
$ gunzip -d  NASA_access_log_Jul95.gz
```

Step 3 : Move the NASA_access_log_Jul95 to /home/hduser/lab/histo directory

```
$ mv NASA_access_log_Jul95 /home/hduser/lab/histo
```

Installing gunplot

Step 1 : Update package list

```
$ sudo apt-get update -y
```

Step 2 : Download and install gunplot

```
$ sudo apt-get install -y gnuplot
```

Executing a histogram MapReduce program in Hadoop

Note : Create /lab/histo directory and save source code in HistogramGenerationMapReduce.java in it.

Step 1 : Compilation of a java program

```
$ javac -classpath $HADOOP_CLASSPATH
        HistogramGenerationMapReduce.java
```

Step 2 : Creation of jar file

```
$ jar -cvf histo.jar *.class
```

Step 3 : Creation of directories

```
$ hdfs dfs -mkdir /user/histo
$ hdfs dfs -mkdir /user/histo/input
```

Step 4 : Copying input file from local directory to Hadoop

```
$ hadoop fs -copyFromLocal NASA_access_log_Jul95 /user/histo/input
```

Step 5 : Executing job in hadoop

```
$ hadoop jar histo.jar HistogramGenerationMapReduce
         /user/histo/input /user/histo/output
```

Step 6 : Copying output files from Hadoop to local directory

```
$ hadoop fs -copyToLocal /user/histo/output/*
```

Step 7 : Viewing the output file

```
$ gedit part-r-00000
```

Step 8 : Create a histobyhour.plot file with following settings

```
set terminal png
set output "hitsobyHour.png"

set title "Hits by Hour of Day";
set ylabel "Number of Hits";
set xlabel "Hour";
set key left top
set log y

plot "part-r-00000" using 1:2 title "2 Node" with linespoints
```

Step 9 : Generate the plot by running the following command and It will generate a file called hitsbyHour.png ,

```
$ gnuplot histobyhour.plot
```

Step 10 : View hitsbyHour.png using an image viewer [(Eye of Gnome) eog is the default image viewer in ubuntu]

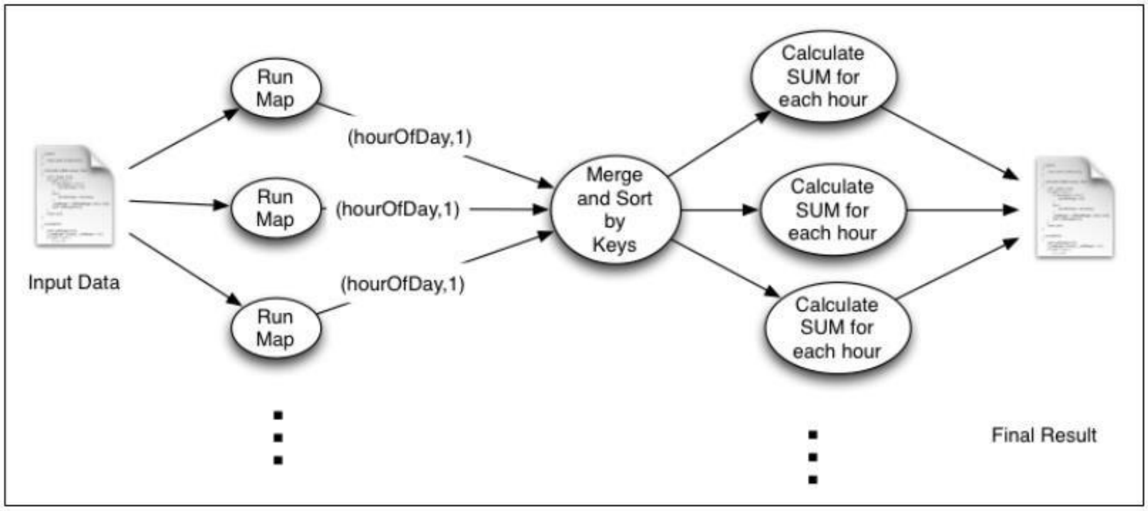
```
$ eog hitsbyHour.png
```

Step 11 : Remove the output filles and directory from hadoop

```
$ hdfs dfs -rm /user/histo/output/*
$ hdfs dfs -rmdir /user/histo/output
```

Note : only by removing the output files and directory from hadoop, we can use the above procedure for executing the job again. If input need to be changed, remove the input files and directory also, and do changes in Step 3 and 4.

Illustration of Entire Process:



Sample Input and Output:

Sample Content of input file : NASA_access_log_Jul95

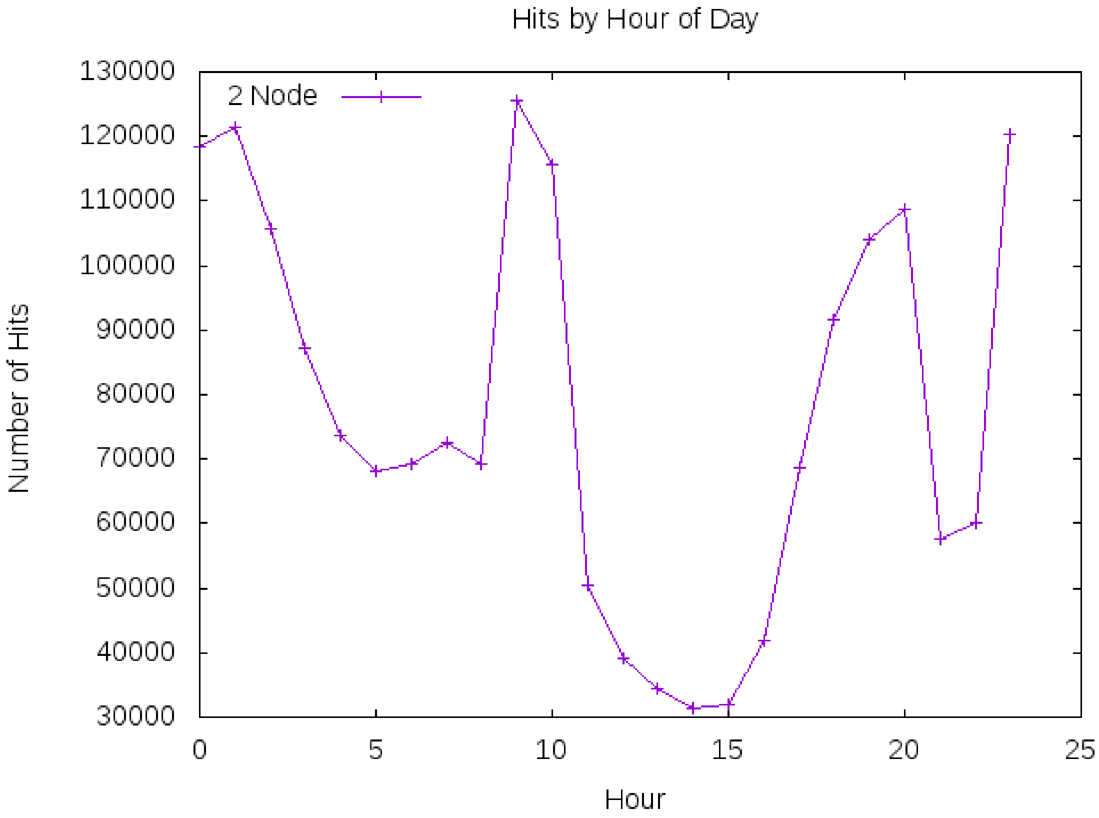
```
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/
HTTP/1.0" 200 6245
unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET
/shuttle/countdown/ HTTP/1.0" 200 3985
199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET
/shuttle/missions/sts-73/mission-sts-73.html HTTP/1.0" 200 4085
burger.letters.com - - [01/Jul/1995:00:00:11 -0400] "GET
/shuttle/countdown/liftoff.html HTTP/1.0" 304 0
199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET
/shuttle/missions/sts-73/sts-73-patch-small.gif HTTP/1.0" 200 4179
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET
/images/NASA-logosmall.gif HTTP/1.0" 304 0
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET
/shuttle/countdown/video/livevideo.gif HTTP/1.0" 200 0
205.212.115.106 - - [01/Jul/1995:00:00:12 -0400] "GET
```

```
/shuttle/countdown/countdown.html HTTP/1.0" 200 3985
d104.aa.net - - [01/Jul/1995:00:00:13 -0400] "GET
/shuttle/countdown/ HTTP/1.0" 200 3985
129.94.144.152 - - [01/Jul/1995:00:00:13 -0400] "GET / HTTP/1.0"
200 7074
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET
/shuttle/countdown/count.gif HTTP/1.0" 200 40310
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET
/images/NASA-logosmall.gif HTTP/1.0" 200 786
 .
 .
 .
```

Sample Content of output file : part-r-00000

```
0     118413
1     121372
2     105636
3     87193
4     73699
5     68237
6     69308
7     72662
8     69303
9     125652
10    115723
11    50487
12    39158
13    34285
14    31350
15    31985
16    41805
17    68593
18    91563
19    104110
20    108600
21    57652
22    60208
23    120279
```

Generated histogram of output file part-r-00000



Result:

Thus, a MapReduce application has been developed in java to calculate histogram of web server log data, executed on single node Hadoop cluster and responses have been verified.

Ex. No. : 8 Correlation between number of hits and message size of weblog dataset using scatter plot

Date :

Aim:

To write a MapReduce application in java to calculate the correlation between number of hits and message size of weblog dataset using scatter plot and run it on single node Hadoop cluster.

Source Code:

```
// save the following code in MsgSizeScatterMapReduce.java

import java.io.IOException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

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


public class MsgSizeScatterMapReduce
             extends Configured implements Tool {

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

      @Override
      public int run(String[] args) throws Exception {

            if (args.length < 2) {
                  System.err.println("Usage:  <input_path>
                                 <output_path> <num_reduce_tasks>");
                  System.exit(-1);
            }
```

```
           String inputPath = args[0];
           String outputPath = args[1];
           int numReduce = 1;
           if (args.length == 3)
                numReduce = Integer.parseInt(args[2]);

           Job job = Job.getInstance(getConf(),
                               "WeblogMessagesizevsHitsProcessor");
           job.setJarByClass(MsgSizeScatterMapReduce.class);
           job.setMapperClass(AMapper.class);
           job.setReducerClass(AReducer.class);
           job.setNumReduceTasks(numReduce);

           job.setMapOutputKeyClass(IntWritable.class);
           job.setMapOutputValueClass(IntWritable.class);
           FileInputFormat.setInputPaths(job, new Path(inputPath));
           FileOutputFormat.setOutputPath(job, new
                                                 Path(outputPath));

           int exitStatus = job.waitForCompletion(true) ? 0 : 1;
           return exitStatus;
      }

    public static class AMapper
           extends Mapper<Object, Text, IntWritable, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        public static final Pattern httplogPattern = Pattern
                .compile("([^\\s]+) - - \\[(.+)\\] \"([^\\s]+)
                 (/[^\\s]*) HTTP/[^\\s]+\" [^\\s]+ ([0-9]+)");

        public void map(Object key, Text value, Context context)
                    throws IOException, InterruptedException {
            Matcher matcher =
                         httplogPattern.matcher(value.toString());
            if (matcher.matches()) {
                int size = Integer.parseInt(matcher.group(5));
                context.write(new IntWritable(size / 1024), one);
            }
        }
    }

    public static class AReducer extends Reducer
             <IntWritable, IntWritable, IntWritable, IntWritable> {
        public void reduce(IntWritable key, Iterable<IntWritable>
                    values, Context context)
                    throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            context.write(key, new IntWritable(sum));
        }
    }
}
```

Procedure:

Preparing input - weblog dataset

Step 1 : Download the NASA weblog dataset

```
$ wget ftp://ita.ee.lbl.gov/traces/NASA_access_log_Jul95.gz
```

Step 2 : Unzip the gz file

```
$ gunzip -d  NASA_access_log_Jul95.gz
```

Step 3 : Move the NASA_access_log_Jul95 to /home/hduser/lab/histo directory

```
$ mv NASA_access_log_Jul95 /home/hduser/lab/scatter
```

Installing gunplot

Step 1 : Update package list

```
$ sudo apt-get update -y
```

Step 2 : Download and install gunplot

```
$ sudo apt-get install -y gnuplot
```

Executing a scatter ploat MapReduce program in Hadoop

Note : Create /lab/scatter directory and save source code in MsgSizeScatterMapReduce.java in it.

Step 1 : Compilation of a java program

```
$ javac -classpath $HADOOP_CLASSPATH MsgSizeScatterMapReduce.java
```

Step 2 : Creation of jar file

```
$ jar -cvf scatter.jar *.class
```

Step 3 : Creation of directories

```
$ hdfs dfs -mkdir /user/scatter
$ hdfs dfs -mkdir /user/scatter/input
```

Step 4 : Copying input file from local directory to Hadoop

```
$ hadoop fs -copyFromLocal NASA_access_log_Jul95
        /user/scatter/input
```

Step 5 : Executing job in hadoop

```
$ hadoop jar scatter.jar MsgSizeScatterMapReduce
        /user/scatter/input /user/scatter/output 3
```

Step 6 : Copying output files from Hadoop to local directory

```
$ hadoop fs -copyToLocal /user/scatter/output/*
```

Step 7 : 3 output files are created because number of reducer is set to 3 while executing the job. View the output files using gedit command as follows

```
$ gedit part-r-00000
$ gedit part-r-00001
$ gedit part-r-00002
```

Step 8 : Create a hitsvsmsgsize.plot file with following settings

```
set terminal png
set output "hitsbymsgsize1.png"

set title "Hits by Size of the Message";
set ylabel "Number of Hits";
set xlabel "Size of the Message (X1000) bytes";
set key left top
set log y
set log x

plot "part-r-00000" using 1:2 title "2 Node" with points
```

Step 9 : Generate the plot by running the following command and It will generate a file called hitsbyHour.png ,

```
$ gnuplot hitsvsmsgsize.plot
```

Step 10 : View hitsbyHour.png using an image viewer [(Eye of Gnome) eog is the default image viewer in ubuntu]

```
$ eog hitsbymsgsize1.png
```

Step 11 : Change input file name in Step 8 as part-r-00001 and output file name as hitsbymsgsize2 and repeat Step 9 and 10.

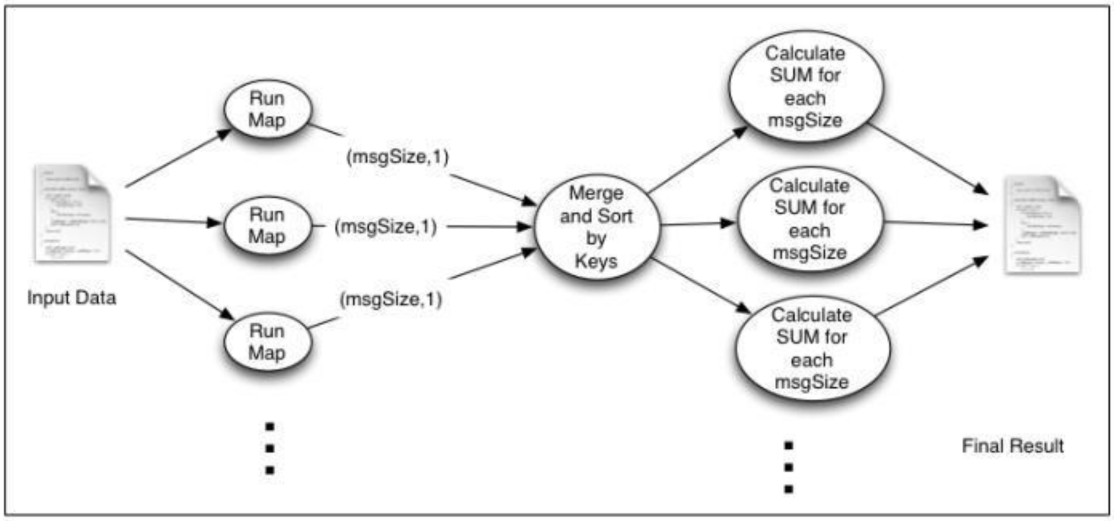
Step 12 : Change input file name in Step 8 as part-r-00002 and output file name as hitsbymsgsize3 and repeat Step 9 and 10.

Step 13 : Remove the output filles and directory from hadoop

```
$ hdfs dfs -rm /user/scatter/output/*
$ hdfs dfs -rmdir /user/scatter/output
```

Note : only by removing the output files and directory from hadoop, we can use the above procedure for executing the job again. If input need to be changed, remove the input files and directory also, and do changes in Step 3 and 4.

Illustration of Entire Process:



Sample Input and Output:

Sample Content of input file : NASA_access_log_Jul95

```
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/
HTTP/1.0" 200 6245
unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET
/shuttle/countdown/ HTTP/1.0" 200 3985
199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET
/shuttle/missions/sts-73/mission-sts-73.html HTTP/1.0" 200 4085
burger.letters.com - - [01/Jul/1995:00:00:11 -0400] "GET
/shuttle/countdown/liftoff.html HTTP/1.0" 304 0
199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET
/shuttle/missions/sts-73/sts-73-patch-small.gif HTTP/1.0" 200 4179
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET
/images/NASA-logosmall.gif HTTP/1.0" 304 0
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET
/shuttle/countdown/video/livevideo.gif HTTP/1.0" 200 0
205.212.115.106 - - [01/Jul/1995:00:00:12 -0400] "GET
/shuttle/countdown/countdown.html HTTP/1.0" 200 3985
d104.aa.net - - [01/Jul/1995:00:00:13 -0400] "GET
/shuttle/countdown/ HTTP/1.0" 200 3985
129.94.144.152 - - [01/Jul/1995:00:00:13 -0400] "GET / HTTP/1.0"
200 7074
 .
 .
 .
```

Sample Content of output file : part-r-00000

```
0     554526
3     131824
6     100802
9     24774
12    26328
15    7243
18    5608
21    1555
24    9527
27    6008
30    3621
 .
 .
 .
```

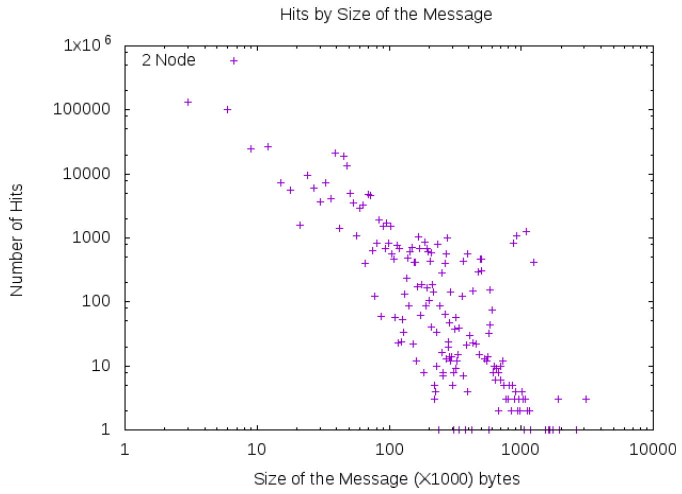
Sample Content of output file : part-r-00001

```
1     248295
4     109586
7     31875
10    10163
13    30266
16    7739
19    5844
22    1459
25    1284
28    3106
31    4868
 .
 .
 .
```

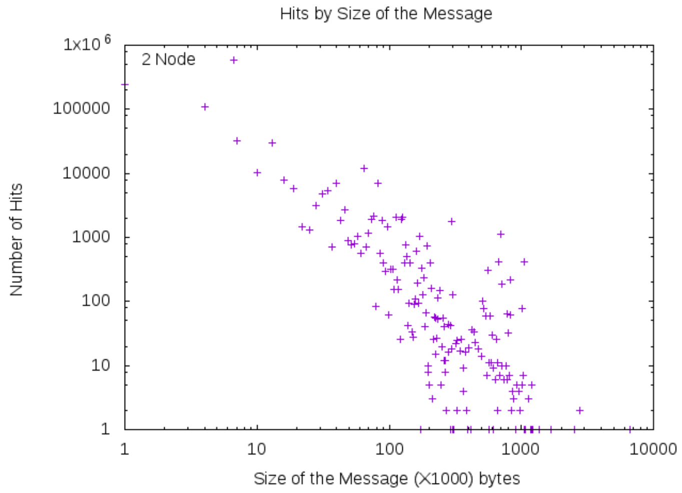
Sample Content of output file : part-r-00002

```
2     57652
5     74123
8     33810
11    51831
14    10164
17    33811
20    7465
23    1622
26    2691
29    6948
 .
 .
 .
```

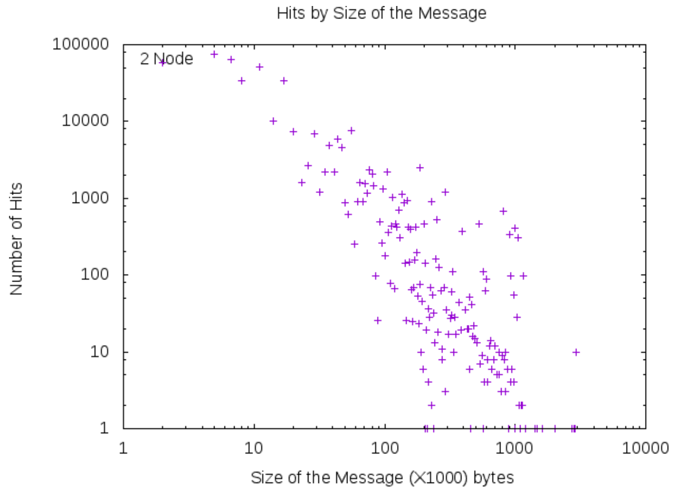
Generated scatter plot of output file part-r-00000



Generated scatter plot of output file part-r-00001



Generated scatter plot of output file part-r-00002



Result:

Thus, a MapReduce application has been developed in java to calculate the correlation between number of hits and message size of weblog dataset using scatter plot, executed on single node Hadoop cluster and responses have been verified.

Ex. No. : 9 Parsing the Tomcat e-mail list dataset with complex data format
Date :

Aim:

To write a MapReduce application in java to parse the Tomcat e-mail list dataset that has complex data format by writing an input formatter and run it on single node Hadoop cluster.

Source Code:

```
// save the following code in MBoxFileInputFormat.java
// content of MBoxFileInputFormat.java starts here

import java.io.IOException;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.InputSplit;
import org.apache.hadoop.mapreduce.RecordReader;
import org.apache.hadoop.mapreduce.TaskAttemptContext;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;



public class MBoxFileInputFormat
             extends FileInputFormat<Text, Text>{
    private MBoxFileReader boxFileReader = null;

    @Override
    public RecordReader<Text, Text> createRecordReader(
            InputSplit inputSplit, TaskAttemptContext attempt)
            throws IOException,InterruptedException {
        boxFileReader = new MBoxFileReader();
        boxFileReader.initialize(inputSplit, attempt);
        return boxFileReader;
    }

}

// content of MBoxFileInputFormat.java ends here
// save the following code in MBoxFileReader.java
// content of MBoxFileReader.java starts here

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.InputSplit;
import org.apache.hadoop.mapreduce.RecordReader;
import org.apache.hadoop.mapreduce.TaskAttemptContext;
import org.apache.hadoop.mapreduce.lib.input.FileSplit;

public class MBoxFileReader
             extends RecordReader<Text, Text> {
    private static Pattern pattern1 = Pattern.compile(
                   "From .*tomcat.apache.org@tomcat.apache.org.*");
    private BufferedReader reader;
    private int count = 0;
    private Text key;
    private Text value;
    private StringBuffer email = new StringBuffer();
    String line = null;

    public MBoxFileReader() {
    }

    @Override
    public void initialize(InputSplit inputSplit,
                TaskAttemptContext attempt)
                throws IOException, InterruptedException {
        Path path = ((FileSplit) inputSplit).getPath();

        FileSystem fs = FileSystem.get(attempt.getConfiguration());
        FSDataInputStream fsStream = fs.open(path);
        reader = new BufferedReader(new
                                    InputStreamReader(fsStream));

        while ((line = reader.readLine()) != null) {
            Matcher matcher = pattern1.matcher(line);
            if (matcher.matches()) {
                email.append(line).append("\n");
                break;
            }
        }
    }
    @Override
    public boolean nextKeyValue()
                   throws IOException, InterruptedException {
        if (email == null) {
            return false;
        }
        count++;
        while ((line = reader.readLine()) != null) {
            Matcher matcher = pattern1.matcher(line);
            if (!matcher.matches()) {
                email.append(line).append("\n");
            } else {
                parseEmail(email.toString());
                email = new StringBuffer();
                email.append(line).append("\n");
                return true;
            }
        }
        parseEmail(email.toString());
        email = null;
        return true;
    }

    @Override
    public Text getCurrentKey()
                throws IOException, InterruptedException {
        return key;
    }



    @Override
    public Text getCurrentValue()
                throws IOException, InterruptedException {
        return value;
    }

    @Override
    public float getProgress()
                 throws IOException, InterruptedException {
        return count;
    }

    @Override
    public void close() throws IOException {
        reader.close();
    }
    public void parseEmail(String email) {
        String[] tokens = email.split("\n");
        String from = null;
        String subject = null;
        String date = null;

        for (String token : tokens) {
            if (token.contains(":")) {
                if (token.startsWith("From:")) {
                    from = token.substring(5).replaceAll
                            ("<.*>|\\\"|,|=[0-9]*", "")
                            .replaceAll("\\[.*?\\]", "")
                            .replaceAll("\\s", "_").trim();
                } else if (token.startsWith("Subject:")) {
                    subject = token.substring(8).trim();
                } else if (token.startsWith("Date:")) {
                    date = token.substring(5).trim();
                }
            }
        }

        key = new Text(String.valueOf((from + subject +
                                       date).hashCode()));
        value = new Text(from + "#" + subject + "#" + date);
    }
}

// content of MBoxFileReader.java ends here



// save the following code in CountReceivedRepliesMapReduce.java
// content of CountReceivedRepliesMapReduce.java starts here

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TreeMap;

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

```
public class CountReceivedRepliesMapReduce
          extends Configured implements Tool {
      public static void main(String[] args) throws Exception {
            int res = ToolRunner.run(new Configuration(), new
                              CountReceivedRepliesMapReduce(), args);
            System.exit(res);
      }

      @Override
      public int run(String[] args) throws Exception {

            if (args.length < 2) {
                  System.err.println("Usage:  <input_path>
                                 <output_path> <num_reduce_tasks>");
                  System.exit(-1);
            }

            String inputPath = args[0];
            String outputPath = args[1];
            int numReduce = 1;
            if (args.length == 3)
                  numReduce = Integer.parseInt(args[2]);

            Job job = Job.getInstance(getConf(),
                                          "MLReceiveReplyProcessor");
            job.setJarByClass(CountReceivedRepliesMapReduce.class);
            job.setMapperClass(AMapper.class);
            job.setReducerClass(AReducer.class);
            job.setNumReduceTasks(numReduce);

            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(Text.class);
            job.setInputFormatClass(MBoxFileInputFormat.class);
            FileInputFormat.setInputPaths(job, new Path(inputPath));
            FileOutputFormat.setOutputPath(job, new
                                                    Path(outputPath));

            int exitStatus = job.waitForCompletion(true) ? 0 : 1;
            return exitStatus;
      }

      public static class AMapper
                        extends Mapper<Object, Text, Text, Text> {

            public void map(Object key, Text value, Context context)
                        throws IOException, InterruptedException {
                  String[] tokens = value.toString().split("#");
                  String from = tokens[0];
                  String subject = tokens[1];
                  String date = tokens[2].replaceAll(",", "");
            subject = subject.replaceAll("Re:", "");
            context.write(new Text(subject),
                      new Text(date + "#" + from));
        }
    }

    public static class AReducer
                   extends Reducer<Text, Text, Text, Text> {
        public static SimpleDateFormat dateFormatter = new
              SimpleDateFormat("EEEE dd MMM yyyy hh:mm:ss z");

        public void reduce(Text key, Iterable<Text> values,
                 Context context)
                 throws IOException, InterruptedException {
            try {
                TreeMap<Long, String> replyData = new
                              TreeMap<Long, String>();
                for (Text val : values) {
                    String[] tokens =
                             val.toString().split("#");
                    if (tokens.length != 2) {
                        throw new IOException("Unexpected
                              token " + val.toString());
                    }
                    String from = tokens[1];
                    Date date =
                          dateFormatter.parse(tokens[0]);
                    replyData.put(date.getTime(), from);
                }
                String owner =
                      replyData.get(replyData.firstKey());
                int replyCount = replyData.size();
                int selfReplies = 0;
                for (String from : replyData.values()) {
                    if (owner.equals(from)) {
                        selfReplies++;
                    }
                }
                replyCount = replyCount - selfReplies;

                context.write(new Text(owner), new
                     Text(replyCount + "#" + selfReplies));
            } catch (Exception e) {
                System.out.println("ERROR:" +
                                     e.getMessage());
                return;
            }
        }
    }
}
// content of CountReceivedRepliesMapReduce.java ends here
```

Procedure:

Preparing input - Tomcat email archives

Note : create /lab/email/input directory

Step 1 : Download Tomcat email archives for the year 2012 under the /lab/email/input directory

```
$ wget http://mail-archives.apache.org/mod_mbox/tomcat-dev/201201
$ wget http://mail-archives.apache.org/mod_mbox/tomcat-dev/201202
$ wget http://mail-archives.apache.org/mod_mbox/tomcat-dev/201203
                                                              :
                                                              .
$ wget http://mail-archives.apache.org/mod_mbox/tomcat-dev/201212
```

Executing email parser MapReduce program in Hadoop

Note : Create /lab/email directory and save source code namely in MBoxFileReader.java, MBoxFileInputFormat.java, CountReceivedRepliesMapReduce.java

Step 1 : Compilation of a java program

```
$ javac -classpath $HADOOP_CLASSPATH  MBoxFileReader.java

$ javac -classpath $HADOOP_CLASSPATH:.  MBoxFileInputFormat.java

$ javac -classpath $HADOOP_CLASSPATH:.
                   CountReceivedRepliesMapReduce.java
```

Step 2 : Creation of jar file

```
          $ jar -cvf email.jar *.class
```

Step 3 : Creation of directories

```
          $ hdfs dfs -mkdir /user/email
          $ hdfs dfs -mkdir /user/email/input
```

Step 4 : Copying inputfiles from /user/email/input directory to Hadoop

```
$ hadoop fs -copyFromLocal * /user/email/input
```

Step 5 : Executing job in hadoop

```
$  hadoop  jar  email.jar  CountReceivedRepliesMapReduce
           /user/email/input /user/email/output
```

Step 6 : Copying output files from Hadoop to local directory

```
$ hadoop fs -copyToLocal /user/email/output/*
```

Step 7 : Viewing the output file

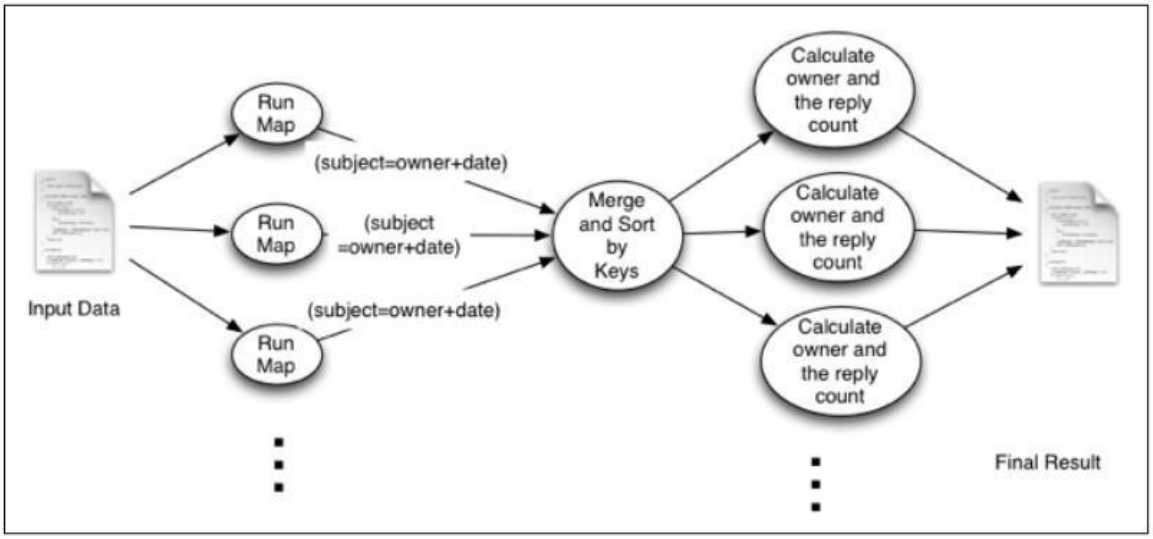
```
$ gedit part-r-00000
```

Step 8 : Remove the output filles and directory from hadoop

```
$ hdfs dfs -rm /user/email/output/*
$ hdfs dfs -rmdir /user/email/output
```

Note : only by removing the output files and directory from hadoop, we can use the above procedure for executing the job again. If input need to be changed, remove the input files and directory also, and do changes in Step 3 and 4.

Illustration of Entire Process:



Sample Input and Output:

Sample Content of input file : 201201

```
From dev-return-123324-apmail-tomcat-dev-
archive=tomcat.apache.org@tomcat.apache.org  Sun Jan  1 02:30:46
2012
Return-Path: <dev-return-123324-apmail-tomcat-dev-
archive=tomcat.apache.org@tomcat.apache.org>
X-Original-To: apmail-tomcat-dev-archive@www.apache.org
Delivered-To: apmail-tomcat-dev-archive@www.apache.org
Received: from mail.apache.org (hermes.apache.org [140.211.11.3])
     by minotaur.apache.org (Postfix) with SMTP id 32FF398E8
     for <apmail-tomcat-dev-archive@www.apache.org>; Sun,  1 Jan
2012 02:30:46 +0000 (UTC)
Received: (qmail 49555 invoked by uid 500); 1 Jan 2012 02:30:44 -
0000
Delivered-To: apmail-tomcat-dev-archive@tomcat.apache.org
Received: (qmail 49500 invoked by uid 500); 1 Jan 2012 02:30:44 -
0000
Mailing-List: contact dev-help@tomcat.apache.org; run by ezmlm
Precedence: bulk
List-Help: <mailto:dev-help@tomcat.apache.org>
List-Unsubscribe: <mailto:dev-unsubscribe@tomcat.apache.org>
List-Post: <mailto:dev@tomcat.apache.org>
.
.
.
```

Sample Content of input file : 201202

```
From dev-return-124294-apmail-tomcat-dev-
archive=tomcat.apache.org@tomcat.apache.org  Wed Feb  1 00:18:55
2012
Return-Path: <dev-return-124294-apmail-tomcat-dev-
archive=tomcat.apache.org@tomcat.apache.org>
X-Original-To: apmail-tomcat-dev-archive@www.apache.org
Delivered-To: apmail-tomcat-dev-archive@www.apache.org
Received: from mail.apache.org (hermes.apache.org [140.211.11.3])
      by minotaur.apache.org (Postfix) with SMTP id 65828940E
      for <apmail-tomcat-dev-archive@www.apache.org>; Wed,  1 Feb
2012 00:18:55 +0000 (UTC)
Received: (qmail 87578 invoked by uid 500); 1 Feb 2012 00:18:54 -
0000
Delivered-To: apmail-tomcat-dev-archive@tomcat.apache.org
Received: (qmail 87393 invoked by uid 500); 1 Feb 2012 00:18:53 -
0000
.
.
.
```

Sample Content of input file : 201203

```
.
.
.
```

Sample Content of input file : 201212

```
From dev-return-133405-apmail-tomcat-dev-
archive=tomcat.apache.org@tomcat.apache.org  Sat Dec  1 03:35:59
2012
Return-Path: <dev-return-133405-apmail-tomcat-dev-
archive=tomcat.apache.org@tomcat.apache.org>
X-Original-To: apmail-tomcat-dev-archive@www.apache.org
Delivered-To: apmail-tomcat-dev-archive@www.apache.org
Received: from mail.apache.org (hermes.apache.org [140.211.11.3])
      by minotaur.apache.org (Postfix) with SMTP id 642A8D7CA
      for <apmail-tomcat-dev-archive@www.apache.org>; Sat,  1 Dec
2012 03:35:59 +0000 (UTC)
Received: (qmail 82905 invoked by uid 500); 1 Dec 2012 03:35:58 -
0000
Delivered-To: apmail-tomcat-dev-archive@tomcat.apache.org
Received: (qmail 82533 invoked by uid 500); 1 Dec 2012 03:35:55 -
0000
.
.
.
```

Sample Content of output file : part-r-00000

```
_Rainer_Jung_   2#1
_Mark_Thomas_   0#1
_Konstantin_Kolinko_  1#2
_Henri_Gomez_   12#3
_Jeanfrancois_Arcand_ 0#1
_Mark_Thomas_   2#1
_Mark_Thomas_   20#2
_Rainer_Jung_   0#1
_Rainer_Jung_   0#1
_Konstantin_Kolinko_  1#2
_Ivan_     0#1
_Mark_Thomas_   1#2
_Konstantin_Kolinko_  8#2
_Willem_Fibbe_-_Realworks_BV_   2#1
_Mark_Thomas_   1#1
_Konstantin_Kolinko_  0#1
 _Filip_Hanik_Mailing_Lists_ 1#1
_Konstantin_Kolinko_  2#2
_Konstantin_Kolinko_  4#2
_sebb_     0#1
_Mark_Thomas_   0#1
_sebb_     2#1
 .
 .
 .
```

Result:

Thus, a MapReduce application has been developed in java to parse the Tomcat e-mail list dataset that has complex data format by writing an input formatter, executed on single node Hadoop cluster and responses have been verified.

Ex. No. : 10 Joining of two MBOX-formatted e-mail datasets

Date :

Aim:

To write a MapReduce application in java to join two MBOX-formatted e-mail datasets and show the result in scatter plot, and run it on single node Hadoop cluster.

Source Code:

```
// save the following code in MBoxFileInputFormat.java
// content of MBoxFileInputFormat.java starts here

import java.io.IOException;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.InputSplit;
import org.apache.hadoop.mapreduce.RecordReader;
import org.apache.hadoop.mapreduce.TaskAttemptContext;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;



public class MBoxFileInputFormat
             extends FileInputFormat<Text, Text>{
    private MBoxFileReader boxFileReader = null;

    @Override
    public RecordReader<Text, Text> createRecordReader(
            InputSplit inputSplit, TaskAttemptContext attempt)
            throws IOException,InterruptedException {
        boxFileReader = new MBoxFileReader();
        boxFileReader.initialize(inputSplit, attempt);
        return boxFileReader;
    }

}

// content of MBoxFileInputFormat.java ends here
// save the following code in MBoxFileReader.java
// content of MBoxFileReader.java starts here

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.InputSplit;
import org.apache.hadoop.mapreduce.RecordReader;
import org.apache.hadoop.mapreduce.TaskAttemptContext;
import org.apache.hadoop.mapreduce.lib.input.FileSplit;

public class MBoxFileReader
             extends RecordReader<Text, Text> {
    private static Pattern pattern1 = Pattern.compile(
                   "From .*tomcat.apache.org@tomcat.apache.org.*");
    private BufferedReader reader;
    private int count = 0;
    private Text key;
    private Text value;
    private StringBuffer email = new StringBuffer();
    String line = null;

    public MBoxFileReader() {
    }

    @Override
    public void initialize(InputSplit inputSplit,
                TaskAttemptContext attempt)
                throws IOException, InterruptedException {
        Path path = ((FileSplit) inputSplit).getPath();

        FileSystem fs = FileSystem.get(attempt.getConfiguration());
        FSDataInputStream fsStream = fs.open(path);
        reader = new BufferedReader(new
                                    InputStreamReader(fsStream));

        while ((line = reader.readLine()) != null) {
            Matcher matcher = pattern1.matcher(line);
            if (matcher.matches()) {
                email.append(line).append("\n");
                break;
            }
        }
    }
```

```
        @Override
        public boolean nextKeyValue()
                        throws IOException, InterruptedException {
            if (email == null) {
                return false;
            }
            count++;
            while ((line = reader.readLine()) != null) {
                Matcher matcher = pattern1.matcher(line);
                if (!matcher.matches()) {
                    email.append(line).append("\n");
                } else {
                    parseEmail(email.toString());
                    email = new StringBuffer();
                    email.append(line).append("\n");
                    return true;
                }
            }
            parseEmail(email.toString());
            email = null;
            return true;
        }

        @Override
        public Text getCurrentKey()
                    throws IOException, InterruptedException {
            return key;
        }



        @Override
        public Text getCurrentValue()
                    throws IOException, InterruptedException {
            return value;
        }

        @Override
        public float getProgress()
                     throws IOException, InterruptedException {
            return count;
        }

        @Override
        public void close() throws IOException {
            reader.close();
        }
```

```
    public void parseEmail(String email) {
        String[] tokens = email.split("\n");
        String from = null;
        String subject = null;
        String date = null;

        for (String token : tokens) {
            if (token.contains(":")) {
                if (token.startsWith("From:")) {
                    from = token.substring(5).replaceAll
                            ("<.*>|\\\"|,|=[0-9]*", "")
                            .replaceAll("\\[.*?\\]", "")
                            .replaceAll("\\s", "_").trim();
                } else if (token.startsWith("Subject:")) {
                    subject = token.substring(8).trim();
                } else if (token.startsWith("Date:")) {
                    date = token.substring(5).trim();
                }
            }
        }

        key = new Text(String.valueOf((from + subject +
                                       date).hashCode()));
        value = new Text(from + "#" + subject + "#" + date);
    }
}

// content of MBoxFileReader.java ends here



// save the following code in CountReceivedRepliesMapReduce.java
// content of CountReceivedRepliesMapReduce.java starts here

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TreeMap;

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

```
public class CountReceivedRepliesMapReduce
            extends Configured implements Tool {
      public static void main(String[] args) throws Exception {
            int res = ToolRunner.run(new Configuration(), new
                              CountReceivedRepliesMapReduce(), args);
            System.exit(res);
      }

      @Override
      public int run(String[] args) throws Exception {

            if (args.length < 2) {
                  System.err.println("Usage:  <input_path>
                                 <output_path> <num_reduce_tasks>");
                  System.exit(-1);
            }

            String inputPath = args[0];
            String outputPath = args[1];
            int numReduce = 1;
            if (args.length == 3)
                  numReduce = Integer.parseInt(args[2]);

            Job job = Job.getInstance(getConf(),
                                    "MLReceiveReplyProcessor");
            job.setJarByClass(CountReceivedRepliesMapReduce.class);
            job.setMapperClass(AMapper.class);
            job.setReducerClass(AReducer.class);
            job.setNumReduceTasks(numReduce);

            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(Text.class);
            job.setInputFormatClass(MBoxFileInputFormat.class);
            FileInputFormat.setInputPaths(job, new Path(inputPath));
            FileOutputFormat.setOutputPath(job, new
                                          Path(outputPath));

            int exitStatus = job.waitForCompletion(true) ? 0 : 1;
            return exitStatus;
      }

      public static class AMapper
                  extends Mapper<Object, Text, Text, Text> {

            public void map(Object key, Text value, Context context)
                        throws IOException, InterruptedException {
                  String[] tokens = value.toString().split("#");
                  String from = tokens[0];
                  String subject = tokens[1];
                  String date = tokens[2].replaceAll(",", "");
```

```
                subject = subject.replaceAll("Re:", "");
                context.write(new Text(subject),
                       new Text(date + "#" + from));
            }
        }

        public static class AReducer
                   extends Reducer<Text, Text, Text, Text> {
            public static SimpleDateFormat dateFormatter = new
                  SimpleDateFormat("EEEE dd MMM yyyy hh:mm:ss z");

            public void reduce(Text key, Iterable<Text> values,
                     Context context)
                     throws IOException, InterruptedException {
                try {
                    TreeMap<Long, String> replyData = new
                                   TreeMap<Long, String>();
                    for (Text val : values) {
                        String[] tokens =
                                 val.toString().split("#");
                        if (tokens.length != 2) {
                            throw new IOException("Unexpected
                                  token " + val.toString());
                        }
                        String from = tokens[1];
                        Date date =
                              dateFormatter.parse(tokens[0]);
                        replyData.put(date.getTime(), from);
                    }
                    String owner =
                           replyData.get(replyData.firstKey());
                    int replyCount = replyData.size();
                    int selfReplies = 0;
                    for (String from : replyData.values()) {
                        if (owner.equals(from)) {
                            selfReplies++;
                        }
                    }
                    replyCount = replyCount - selfReplies;

                    context.write(new Text(owner), new
                         Text(replyCount + "#" + selfReplies));
                } catch (Exception e) {
                    System.out.println("ERROR:" +
                                     e.getMessage());
                    return;
                }
            }
        }
    }
    // content of CountReceivedRepliesMapReduce.java ends here
```

```
// save the following code in CountSentRepliesMapReduce.java
// content of CountSentRepliesMapReduce.java starts here

import java.io.IOException;

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

public class CountSentRepliesMapReduce
             extends Configured implements Tool {

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

      @Override
      public int run(String[] args) throws Exception {

            if (args.length < 2) {
                  System.err.println("Usage:  <input_path>
                                 <output_path> <num_reduce_tasks>");
                  System.exit(-1);
            }

            String inputPath = args[0];
            String outputPath = args[1];
            int numReduce = 1;
            if (args.length == 3)
                  numReduce = Integer.parseInt(args[2]);

            Job job = Job.getInstance(getConf(),
                                     "MLSendReplyProcessor");
            job.setJarByClass(CountReceivedRepliesMapReduce.class);
            job.setMapperClass(AMapper.class);
            job.setReducerClass(AReducer.class);
            job.setNumReduceTasks(numReduce);

            job.setMapOutputKeyClass(Text.class);
            job.setMapOutputValueClass(Text.class);
```

```
           job.setOutputKeyClass(Text.class);
           job.setOutputValueClass(IntWritable.class);
           job.setInputFormatClass(MBoxFileInputFormat.class);
           FileInputFormat.setInputPaths(job, new Path(inputPath));
           FileOutputFormat.setOutputPath(job, new
                                                 Path(outputPath));

           int exitStatus = job.waitForCompletion(true) ? 0 : 1;
           return exitStatus;
      }

    public static class AMapper
                        extends Mapper<Object, Text, Text, Text> {
        public void map(Object key, Text value, Context context)
                    throws IOException, InterruptedException {
            String[] tokens = value.toString().split("#");
            String from = tokens[0];
            String subject = tokens[1];
            String date = tokens[2];
            System.out.println(from + "=" + date);
            context.write(new Text(from), new Text(date));
        }
    }


    public static class AReducer
                   extends Reducer<Text, Text, Text, IntWritable> {

        public void reduce(Text key, Iterable<Text> values,
                        Context context)
                        throws IOException, InterruptedException {
            int sum = 0;
            for (Text val : values) {
                sum = sum + 1;
            }
            context.write(key, new IntWritable(sum));
        }
    }
}

// content of CountSentRepliesMapReduce.java ends here
// save the following code in JoinSentReceivedReplies.java
// content of JoinSentReceivedReplies.java starts here

import java.io.IOException;
import java.text.SimpleDateFormat;

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


public class JoinSentReceivedReplies
          extends Configured implements Tool {

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

     @Override
     public int run(String[] args) throws Exception {

          if (args.length < 2) {
               System.err.println("Usage:  <input_path>
                         <output_path> <num_reduce_tasks>");
               System.exit(-1);
          }

          String inputPath = args[0];
          String outputPath = args[1];
          int numReduce = 1;
          if (args.length == 3)
               numReduce = Integer.parseInt(args[2]);

          Job job = Job.getInstance(getConf(),
                              "MLJoinSendReceiveReplies");
          job.setJarByClass(JoinSentReceivedReplies.class);
          job.setMapperClass(AMapper.class);
          job.setReducerClass(AReducer.class);
```

```
       job.setNumReduceTasks(numReduce);

       job.setMapOutputKeyClass(Text.class);
       job.setMapOutputValueClass(Text.class);
       job.setOutputKeyClass(IntWritable.class);
       job.setOutputValueClass(IntWritable.class);
       FileInputFormat.setInputPaths(job, new Path(inputPath));
       FileOutputFormat.setOutputPath(job, new
                                             Path(outputPath));

       int exitStatus = job.waitForCompletion(true) ? 0 : 1;
       return exitStatus;
  }

public static class AMapper
                    extends Mapper<Object, Text, Text, Text> {

    public void map(Object key, Text value, Context context)
                throws IOException, InterruptedException {
        String[] tokens = value.toString().split("\\s");
        String from = tokens[0];
        String replyData = tokens[1];
        context.write(new Text(from), new Text(replyData));
    }
}

public static class AReducer
        extends Reducer<Text, Text, IntWritable, IntWritable> {

    public void reduce(Text key, Iterable<Text> values,
                Context context)
                throws IOException, InterruptedException {
        StringBuffer buf = new StringBuffer("[");

        try {
            int sendReplyCount = 0;
            int receiveReplyCount = 0;
            for (Text val : values) {
                String strVal = val.toString();
                buf.append(strVal).append(",");
                if (strVal.contains("#")) {
                    String[] tokens = strVal.split("#");
                    int repliesOnThisThread =
                                   Integer.parseInt(tokens[0]);
                    int selfRepliesOnThisThread =
                                   Integer.parseInt(tokens[1]);
                    receiveReplyCount = receiveReplyCount +
                                           repliesOnThisThread;
                    sendReplyCount = sendReplyCount -
                                       selfRepliesOnThisThread;
                } else {
                        sendReplyCount = sendReplyCount +
                                    Integer.parseInt(strVal);
                    }
                }
                context.write(new IntWritable(sendReplyCount), new
                                IntWritable(receiveReplyCount));
                buf.append("]");
            } catch (NumberFormatException e) {
                System.out.println("ERROR " + e.getMessage() + " "
                                                            + buf);
            }
        }
    }
}

// content of JoinSentReceivedReplies.java ends here
```

Procedure:

Preparing input - Tomcat email archives

Note : create /lab/join2email/input directory

Step 1 : download Tomcat email archives for the year 2012 under the /lab/join2email/input directory

```
$ wget http://mail-archives.apache.org/mod_mbox/tomcat-dev/201201
$ wget http://mail-archives.apache.org/mod_mbox/tomcat-dev/201202
$ wget http://mail-archives.apache.org/mod_mbox/tomcat-dev/201203
                                                               :
                                                               .
$ wget http://mail-archives.apache.org/mod_mbox/tomcat-dev/201212
```

Executing email parser MapReduce program in Hadoop

Note : Create /lab/join2email directory and save source code namely in MBoxFileReader.java, MBoxFileInputFormat.java, CountReceivedRepliesMapReduce.java, CountSentRepliesMapReduce.java, JoinSentReceivedReplies.java

Step 1 : Compilation of a java program

```
$ javac -classpath $HADOOP_CLASSPATH  MBoxFileReader.java

$ javac -classpath $HADOOP_CLASSPATH:.  MBoxFileInputFormat.java

$ javac -classpath $HADOOP_CLASSPATH:.
```

```
                    CountReceivedRepliesMapReduce.java

$ javac -classpath $HADOOP_CLASSPATH:.
                    CountSentRepliesMapReduce.java

$ javac -classpath $HADOOP_CLASSPATH:.
                    JoinSentReceivedReplies.java
```

Step 2 : Creation of jar file

```
$ jar -cvf join2email.jar *.class
```

Step 3 : Creation of directories

```
$ hdfs dfs -mkdir /user/join2email
$ hdfs dfs -mkdir /user/join2email/input
$ hdfs dfs -mkdir /user/join2email/jinput
```

Step 4 : Copying inputfiles from /user/join2email/input directory to Hadoop

```
$ hadoop fs -copyFromLocal * /user/join2email/input
```

Step 5 : Executing jobs in hadoop

```
$ hadoop jar join2email.jar CountReceivedRepliesMapReduce
               /user/join2email/input /user/join2email/routput

$ hadoop jar join2email.jar CountSentRepliesMapReduce
               /user/join2email/input /user/join2email/soutput
```

copy the output file as input to joinSentReceiveReplies

```
$ hdfs dfs -cp /user/join2email/routput/part-r-00000
                     /user/join2email/jinput/1.data

$ hdfs dfs -cp /user/join2email/soutput/part-r-00000
                      /user/join2email/jinput/2.data

$ hadoop jar join2email.jar JoinSentReceivedReplies
          /user/join2email/jinput /user/join2email/joutput
```

Step 6 : Copying output files from Hadoop to local directory

```
$ hadoop fs -copyToLocal /user/join2email/joutput/*
```

Step 7 : Viewing the output file

```
$ gedit part-r-00000
```

Step 8 : Create a sendvsreceive.plot file with following settings

```
set terminal png
set output "sendreceive.png"

set title "Replies Sent vs. Replies Received";
set ylabel "Replies Received";
set xlabel "Replies Sent (without self replies)";
set key left top
set pointsize 2
set log x
set log y

plot  "part-r-00000"  using  1:2  title  "Scatter"  with  points
       pointtype 6
```

Step 9 : Generate the plot by running the following command and It will generate a file called hitsbyHour.png ,

```
$ gnuplot sendvsreceive.plot
```

Step 10 : view hitsbyHour.png using an image viewer [(Eye of Gnome) eog is the default image viewer in ubuntu]

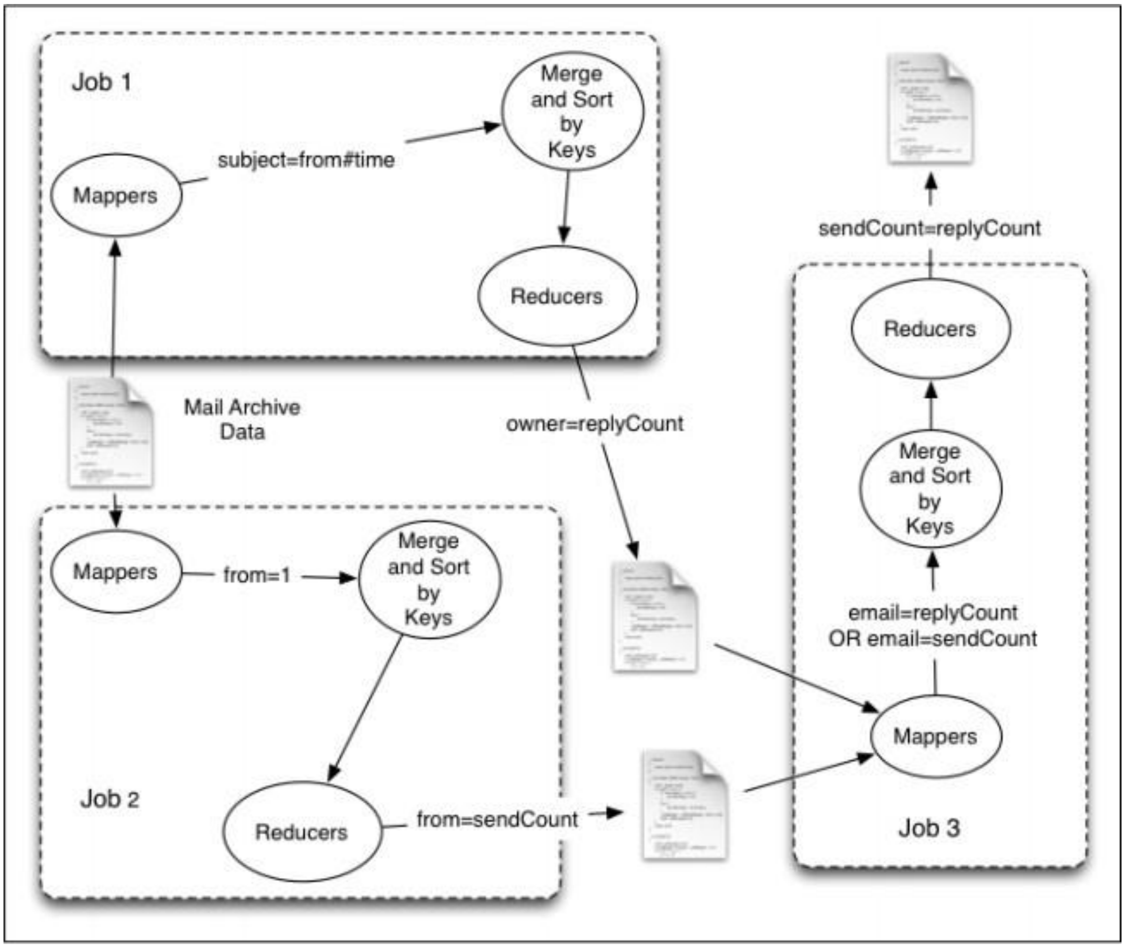
```
$ eog sendreceive.png
```

Step 11 : Remove the output filles and directory from hadoop

```
$ hdfs dfs -rm /user/join2email/jinput/*
$ hdfs dfs -rmdir /user/join2email/jinput
$ hdfs dfs -rm /user/join2email/routput/*
$ hdfs dfs -rmdir /user/join2email/routput
$ hdfs dfs -rm /user/join2email/soutput/*
$ hdfs dfs -rmdir /user/join2email/soutput
$ hdfs dfs -rm /user/join2email/joutput/*
$ hdfs dfs -rmdir /user/join2email/joutput
```

Note : only by removing the output files and directory from hadoop, we can use the above procedure for executing the job again. If input need to be changed, remove the input files and directory also, and do changes in Step 3 and 4.

Illustration of Entire Process:



Sample Input and Output:

Sample Content of input file : 201201

```
From dev-return-123324-apmail-tomcat-dev-
archive=tomcat.apache.org@tomcat.apache.org  Sun Jan  1 02:30:46
2012
Return-Path: <dev-return-123324-apmail-tomcat-dev-
archive=tomcat.apache.org@tomcat.apache.org>
X-Original-To: apmail-tomcat-dev-archive@www.apache.org
Delivered-To: apmail-tomcat-dev-archive@www.apache.org
Received: from mail.apache.org (hermes.apache.org [140.211.11.3])
      by minotaur.apache.org (Postfix) with SMTP id 32FF398E8
      for <apmail-tomcat-dev-archive@www.apache.org>; Sun,  1 Jan
2012 02:30:46 +0000 (UTC)
Received: (qmail 49555 invoked by uid 500); 1 Jan 2012 02:30:44 -
0000
Delivered-To: apmail-tomcat-dev-archive@tomcat.apache.org
Received: (qmail 49500 invoked by uid 500); 1 Jan 2012 02:30:44 -
0000
Mailing-List: contact dev-help@tomcat.apache.org; run by ezmlm
Precedence: bulk
List-Help: <mailto:dev-help@tomcat.apache.org>
List-Unsubscribe: <mailto:dev-unsubscribe@tomcat.apache.org>
List-Post: <mailto:dev@tomcat.apache.org>
.
.
.
```

Sample Content of input file : 201202

```
From dev-return-124294-apmail-tomcat-dev-
archive=tomcat.apache.org@tomcat.apache.org  Wed Feb  1 00:18:55
2012
Return-Path: <dev-return-124294-apmail-tomcat-dev-
archive=tomcat.apache.org@tomcat.apache.org>
X-Original-To: apmail-tomcat-dev-archive@www.apache.org
Delivered-To: apmail-tomcat-dev-archive@www.apache.org
Received: from mail.apache.org (hermes.apache.org [140.211.11.3])
      by minotaur.apache.org (Postfix) with SMTP id 65828940E
      for <apmail-tomcat-dev-archive@www.apache.org>; Wed,  1 Feb
2012 00:18:55 +0000 (UTC)
Received: (qmail 87578 invoked by uid 500); 1 Feb 2012 00:18:54 -
0000
Delivered-To: apmail-tomcat-dev-archive@tomcat.apache.org
Received: (qmail 87393 invoked by uid 500); 1 Feb 2012 00:18:53 -
0000
.
.
.
```

Sample Content of input file : 201203

```
.
.
.
```

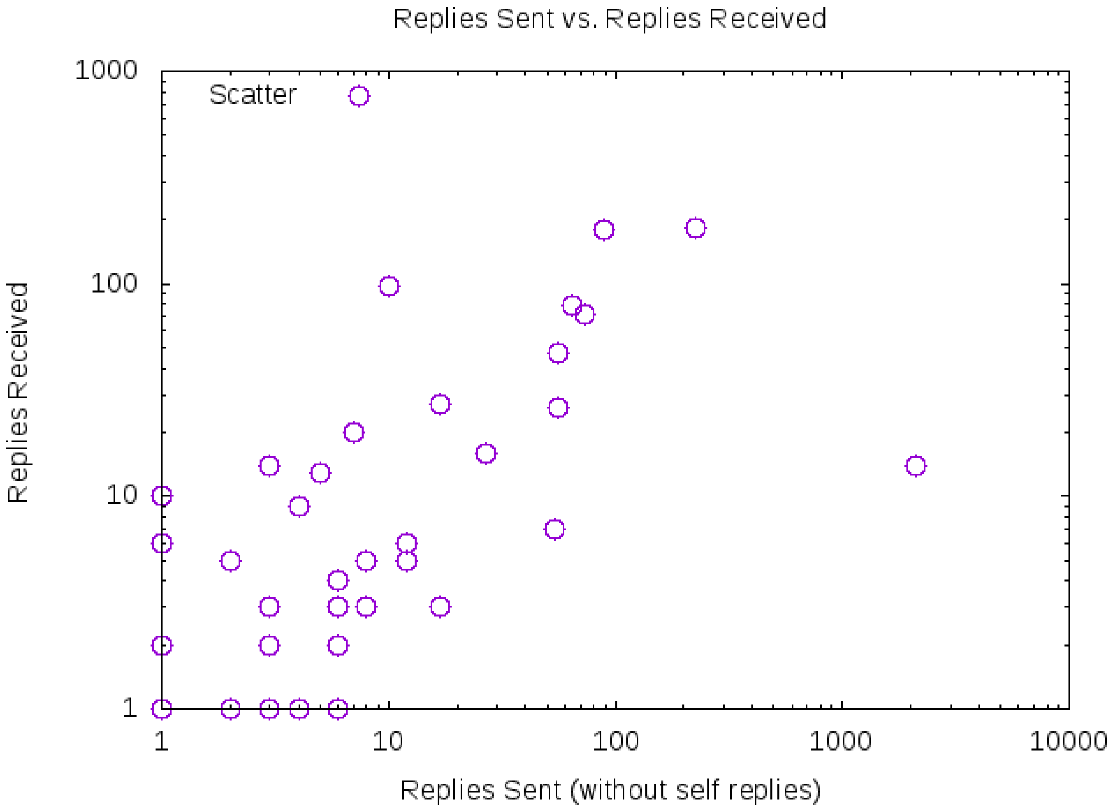
Sample Content of input file : 201212

```
From dev-return-133405-apmail-tomcat-dev-
archive=tomcat.apache.org@tomcat.apache.org  Sat Dec  1 03:35:59
2012
Return-Path: <dev-return-133405-apmail-tomcat-dev-
archive=tomcat.apache.org@tomcat.apache.org>
X-Original-To: apmail-tomcat-dev-archive@www.apache.org
Delivered-To: apmail-tomcat-dev-archive@www.apache.org
Received: from mail.apache.org (hermes.apache.org [140.211.11.3])
     by minotaur.apache.org (Postfix) with SMTP id 642A8D7CA
     for <apmail-tomcat-dev-archive@www.apache.org>; Sat,  1 Dec
2012 03:35:59 +0000 (UTC)
Received: (qmail 82905 invoked by uid 500); 1 Dec 2012 03:35:58 -
0000
.
.
.
```

Sample Content of output file : part-r-00000

```
           0     0
           2     0
           0     2
           4     1
           0     2
           0     0
           0     0
           2     0
           0     0
           0     0
           0     0
           0     0
           2     0
           1     0
           0     0
           8     5
           3     1
           17    27
           .
           .
           .
```

Generated scatter plot of output file part-r-00000



Result:

Thus, a MapReduce application has been developed in java to join two MBOX-formatted e-mail datasets and show the result in scatter plot, executed on single node Hadoop cluster and responses have been verified.