24/4/21 Experiment -16

Aim:- To Perform Simple analytics using Map Reduce

Program:-

```
import java.io.IOException;
import java.util.Iterator;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
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.File
                                    InputFormat;
import org.apache.hadoop.mapreduce.lib.output.
                                    FileOutputFormat;
Import org.apache.hadoop.util.GenericOptionsParser;
Public class WebLogMessageSizeAggregator
{
    Public static final Pattern httplogPattern = Pattern.
        compile("([^\\s]+) -- \\[(.+)\\] \"([^\\s]+)
        (/[^\\s]*) HTTP/[^\\s]+ \"[^\\s]+ ([0-9]+)");
    Public static class AMapper extends Mapper
                    <Object, Text, Text, IntWritable>
    {
```

```
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("msg size"), 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));
    }
}
Public static void main(String[] args) throws
                                    Exception
{
    JobConf conf = new JobConf();
    String[] otherArgs = new GenericOptionsParser
            (conf, args).getRemainingArgs();
    if (otherArgs.length != 2)
    {
        System.err.println("Usage: <in> <out>");
        System.exit(2);
    }
    Job job = new Job(conf, "WebLogMessage Size Aggregator");
    job.setJarByClass(WebLogMessageSizeAggregator.class);
    job.setMapperClass(AMapper.class);
```

```
job.setReducerClass(AReducer.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(
                                otherArgs[0]));
FileOutputFormat.setOutputPath(job, new
                                Path(otherArgs[1]));
System.exit(job.waitForCompletion(true)?0:1);
}
}
```

steps to execute the Hadoop application:

1. gedit WebLogMessageSizeAggregator.java
2. export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
3. ~~jar cf wcjar WebLogMessageSize~~
3. hadoop com.sun.tools.javac.Main WebLogMessageSizeAggregator.java
4. jar cf wc.jar WebLogMessageSizeAggregator*.class
5. hadoop fs -mkdir -p /wc
6. hadoop fs -copyFromLocal input /wc/
7. hadoop jar wc.jar weblog MessageSizeAggregator /wc/input /word/out16
8. hadoop fs -ls /word/out16
9. hadoop fs -cat /word/out16/part-r-00000

Output:-

Mean 1150

Max 6823936

Min 0

11/12/21

Experiment -17

Aim:- Performing Group-By using Map Reduce.

Program:-

```
import java.io.IOException;
import java.util.Iterator;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
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.FileIn
                                    -putFormat;
import org.apache.hadoop.mapreduce.lib.output.
                                    FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

Public class WeblogHitsByLinkProcessor
{
  Public static final Pattern httplogPattern = Pattern
      .compile("([^\\s]+)--\\[(.+)\\]\"([^\\s]+)(/[^
      \\s]*)HTTP/[^\\s]+\"[^\\s]+([0-9]+)");

  Public static class AMapper 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
{
  Matcher matcher = httplog Pattern.matcher (value.to
                                    String());
    if (matcher.matches())
    {   String  link Url = 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);
    }
}
```

```
Public static void main (String[] args) throws
                                    Exception
{
  JobConf conf = new JobConf();
  String[] otherArgs = new GenericOptionsParser (conf,
                        args).getRemainingArgs();

  if (otherArgs.length != 2)
  {
    System.err.println("Usage : <in> <out>");
    System.exit(2);
  }
  Job job = new Job (Conf, "weblog Hits By Link Processor");
  job.setJarByClass(WeblogHitsByLinkProcessor.class);
  job.setMapperClass (AMapper.class);
  job.setReducerClass (AReducer.class);
  Job.setMapOutputKeyClass (Text.class);
  job.setMapOutputValueclass(IntWritable.class);
  FileInputFormat.addInputPath(job, new Path(other
                              _Args[0]));
  FileOutputFormat.setOutputPath (job, new Path
                              (otherArgs[1]));
  System.exit(job.waitforCompletion (true) ? 0 : 1);
}
}
```

Steps to execute the Hadoop application:

① gedit WeblogHitsByLinkProcessor.java

② export HADOOP_CLASSPATH = ${JAVA_HOME}/lib/tools.jar

③ ~~jar cf~~ hadoop com.sun.tools.javac.Main WeblogHitsByLinkProcessor.java

④ jar cf wc.jar WeblogHitsByLinkProcessor*.class

⑤ hadoop fs -mkdir -p /wc

⑥ hadoop fs -copyFromLocal input /wc

⑦ hadoop jar wc.jar WeblogHitsByLinkProcessor /wc/input /word/out17

⑧ hadoop fs -ls ~~/wcfin~~ /word/out17

⑨ hadoop fs -cat /word/out17/part-r-00000

Output:-

/~downs/www/summary.html	1
/~downs/www/query.html	3
/~downs/www/index.html	2
/~downs/www/	3
/~downs/home.html	41
/~downs/launch.gif	37
/~downs/harvest-1.2/brokers/www/query.html	8

8/12/21

Experiment – 18

Aim:- Calculating webloghits using Map Reduce.

Program:-

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

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

```
Private final static IntWritable one =
                    new IntWritable(1);
Public void map (Object key, Text value, Context
        context) throws IoException, Interrupted
                        Exception
{
  Matcher matcher = httplog Pattern. 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, Interrupted Exception
    {
        int sum = 0;
        for (IntWritable val : values)
        { sum += val. get();
        }
        context. write (key, new IntWritable (sum));
    }
}
```

```
Public static void main(String[] args) throws
Exception
{ JobConf conf = new JobConf();
  String[] otherArgs = new GenericOptionsParser
        (conf, args).getRemainingArgs();
  if (otherArgs.length != 2)
  { System.err.println("Usage: <in> <out>");
    System.exit(2);
  }
  Job job = new Job(conf, "weblog Messagesize US
        - Hits Processor");
  job.setJarByClass(Webloghits.class);
  job.setMapperClass(AMapper.class);
  job.setReducerClass(AReducer.class);
  job.setMapOutputKeyClass(IntWritable.class);
  job.setMapOutputValueClass(IntWritable.class);
  FileInputFormat.addInputPath(job, new Path
        (otherArgs[0]));
  FileOutputFormat.setOutputPath(job, new Path
        (otherArgs[1]));
  System.exit(job.waitForCompletion(true) ? 0 : 1);
 }
}
```

Input :- accesslog.txt - which contains website clicks and IPaddresses.

Execution steps:-

① gedit Webloghits.java

② export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar.

③ hadoop com.sun.tools.javac.Main Webloghits.java

④ jar cf wc.jar Webloghits*.class

⑤ hadoop fs -mkdir -p /wc

⑥ hadoop fs -copyFromLocal accesslog /wc

⑦ hadoop jar wc.jar Webloghits /wc/accesslog /word/out19

⑧ hadoop fs -ls /word/out19

⑨ hadoop fs -cat /word/out19/part-r-00000

Output:-

0	554526
1	248295
2	57652
3	131824
4	109586
5	74123
6	100802
7	31875
8	33810
9	24774
10	10163
11	51831
12	26328

13	30266
14	10164
15	7243
16	7739
17	33811
18	5608
19	5844
20	7465
21	1555
22	1459
23	1622
24	9527
25	1284
26	26911
27	6008
28	3106
29	6948
30	3621

and Soon...

8/12/21 Experiment -19

Aim :- Plotting the Hadoop results using GNU Plot

Download the results of the last recipe to a local computer by running the following command

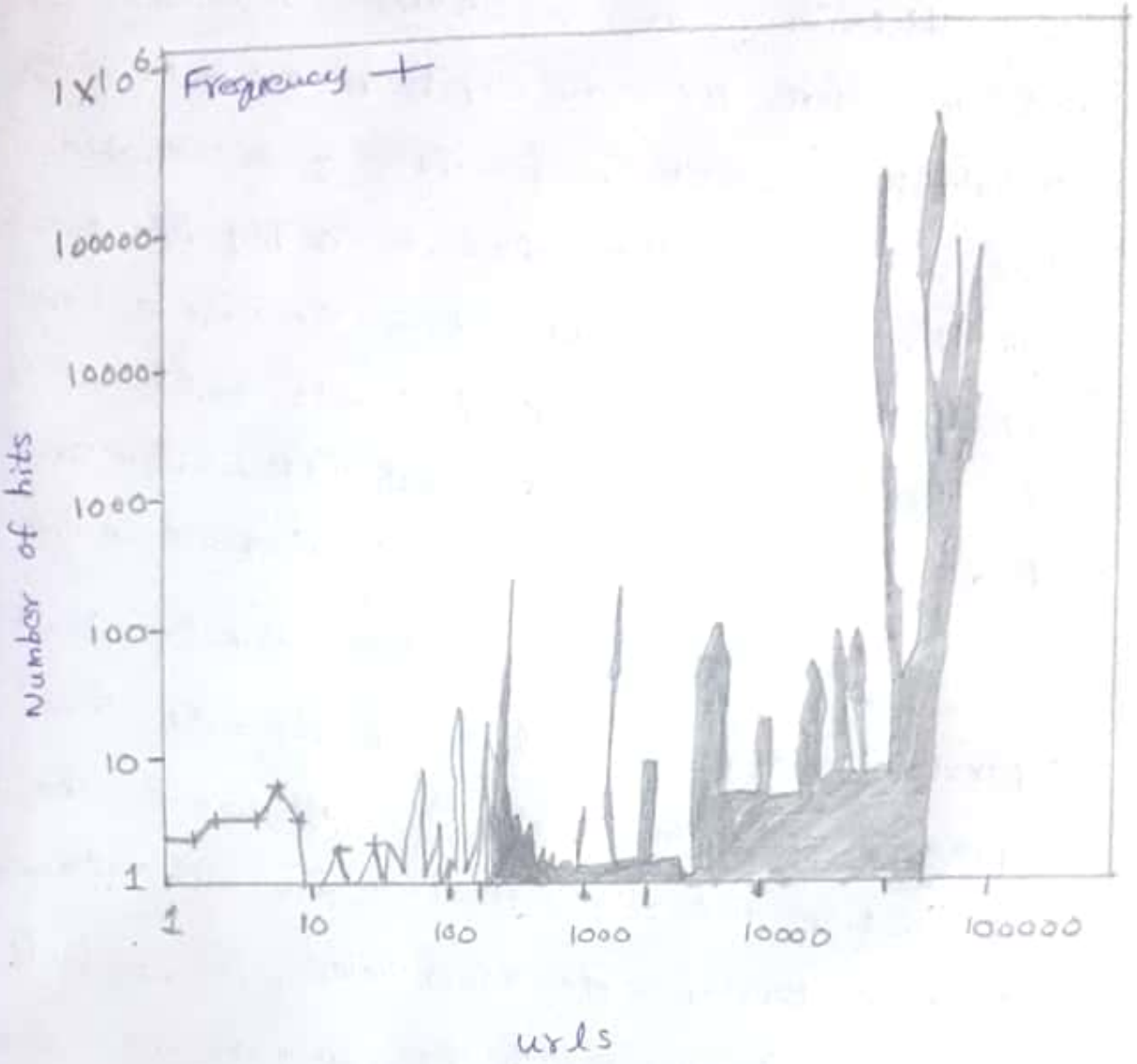
→ hadoop fs -get /word/output17/part-r-00000 2.data

→ gnuplot (Sudo gedit http-freqdist.plot

```
set terminal png
set output "Frequency.png"
Set title "Frequency Distribution of Hits by URL"
Set ylabel "Number of Hits"
Set xlabel "urls"
Set key left top
Set log y
Set log x
plot "2.data" using 2 title "Frequency" with linespoints.
```

output:-

Frequency Distribution of Hits by URL



15/12/24 Experiment - 20

Aim:- Calculating histograms using Map Reduce.

Histograms are a common visualization technique that gives an empirical estimate of the probability density function (Pdf) of a variable. Histograms are well-suited to a big data environment, because they can reduce the size of raw input data to a vector of counts. Each count is the number of observations that falls within each of a set of contiguous, numeric intervals or bins.

The map reduce function computes counts separately on multiple chunks of the data. Then mapreduce sums the counts from all chunks. The map function and reduce function are both extremely simple in this example. Nevertheless, you can build flexible visualizations with the summary information that they collect.

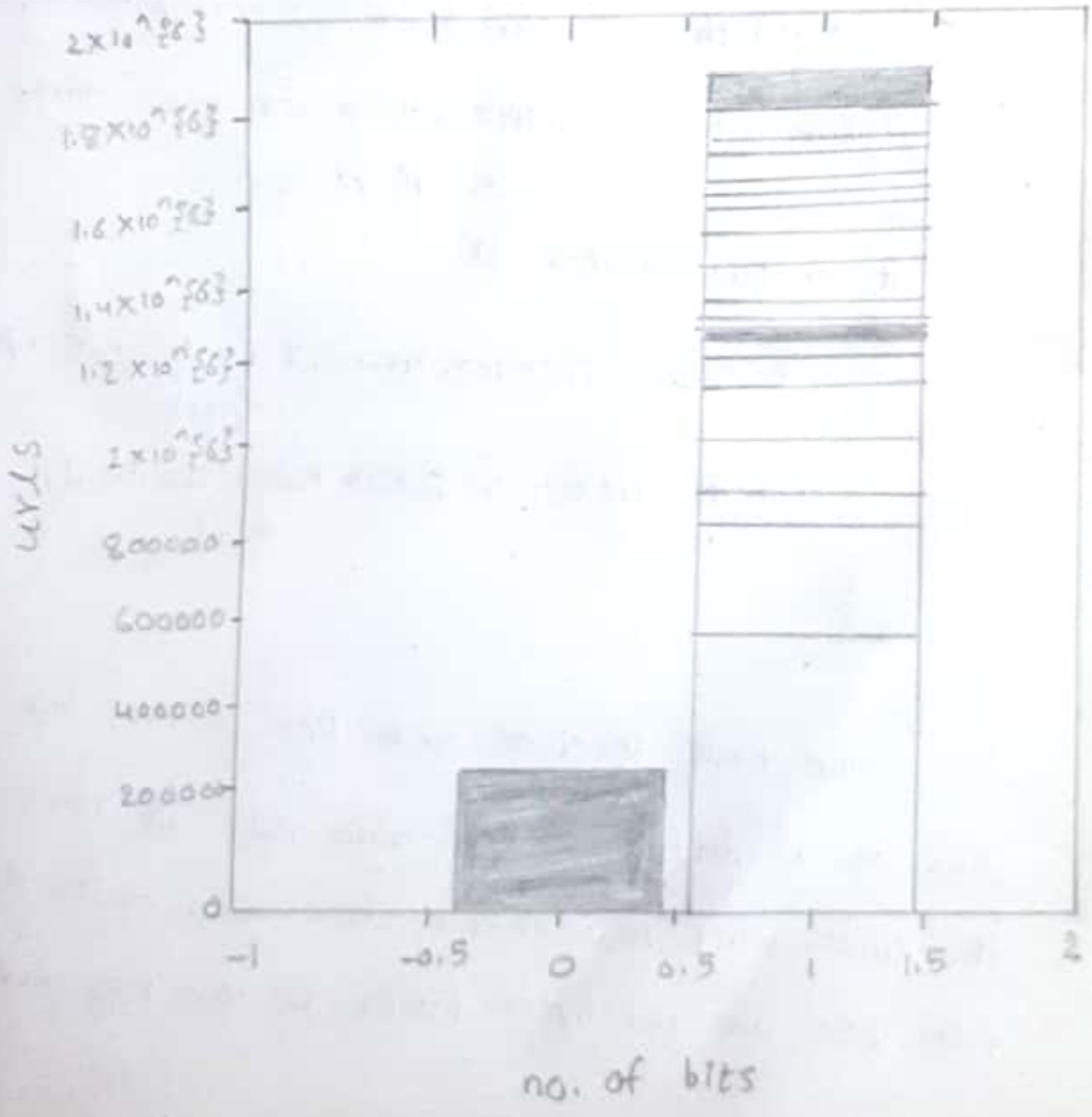
Visualize Results:

Plot the raw bin counts using the whole range of the data in gnuplot by using following commands having input from previous program output i.e.,

→ hadoop fs -get /word/out19/part-r-00000 hist.data

```
set term png medium
set output "histogram.png"
set xlabel "number of bits"
set ylabel "urls"
set style histogram clustered gap 2
set style histogram columnstacked
set boxwidth 0.9 relative
set style data histograms
set style fill solid 1.0 border-1
plot "hist.data" using 1, "hist.data" using 2
```

output:-



15/12/21 Experiment - 21

Aim :- Calculating scatter plots using Map Reduce

Another useful tool while analyzing data is a Scatter plot : scatter plot is used to find the relationship between two measurements (dimensions). It plots the two dimensions against each other.

The following code segment shows the code for the mapper.

```
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);
    }
}
```

Map task receives each line in the log files as a different key-value pair. It parse the lines using regular expressions and emits the file size as 1024-bytes blocks as the key and

one as the values. Then, Hadoop collects the key-value pairs sorts them, and then invokes the reducer once for each key. Each reducer walks through the values and calculates the count of page accesses for each file size.

```
Public void reduce(IntWritable key, Iterable values,
    Context context) throws IOException, Interrupted
  -Exception
{
    int sum = 0;
    for(IntWritable val : values)
    {   sum += val.get();
    }
    context.write(key, new IntWritable(sum));
}
```

The following commands are used for plotter graph between the size of the web pages and the number of hits received by the web page in gnuplot.

```
Set output "Hitsby Message"
Set title "Hits by size of the message"
Set ylabel "no of Hits"
Set xlabel "size of the message"
Set key left top
Set log y
```

set log x

plot "hist.data" using 1:2 title "2 Node" with points

Output :-