# Experiment No.: 03

## Aim:

Experiment on Hadoop Map-Reduce. Write a program to Implement the Wordcount program using

Map-Reduce.

## Theory:

In Hadoop, MapReduce is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of tasks can be joined together to compute final results.

MapReduce consists of 2 steps:

- **Map Function –** It takes a set of data and converts it into another set of data, where individual elements are broken down into tuples
  (Key-Value pair).
- **Reduce Function** – Takes the output from Map as an input and combines those data tuples into a smaller set of tuples.

Workflow of MapReduce consists of 5 steps:

1. Splitting – The splitting parameter can be anything, e.g. splitting by space, comma, semicolon, or even by a new line ('\n').
2. Mapping – as explained above.
3. Intermediate splitting – the entire process in parallel on different clusters. In order to group them in "Reduce Phase" the similar
   KEY data should be on the same cluster.
4. Reduce – it is nothing but mostly group by phase.
5. Combining – The last phase where all the data (individual result set from each cluster) is combined to form a result.

## Steps for executing Wordcount program on Hadoop

1. Open Eclipse> File > New > Java Project >(Name it – Wordcount) > Next.

2. Import laibraries>Add external JARs>File Sysytem>user>lib>hadoop

3. Select all the JAR files>ok

4. Again, click Add external JARs>client>select all JAR files>ok>finish

5. Now to create a class src>new>Class>Give file name (Wordcount)>finish

6. Open the program for wordcount in hadoop and paste it in the file (Wordcount) and save the file.

7. To export the JAR file

    a) Export>JAVA>JAR file>next
    b) Browse the location>home>cloudera>assign the nameWordcount.jar>ok>finish

8. Now to check output> open terminal>check the working directory> create a file to process using command:

    a) Cat > /home/cloudera/Processfile1.txt
    b) Add the content in the file **3.** Click ctrl+z to create the file.

9. Create a folder inputfolder1

10. Move this to HDFS using hdfs dfs –put /home/cloudera/Processfile1.txt/inputfolder1 hdfs dfs –cat/inputfolder1/Processfile1.txt

11. Use following command to run MapReduce function hadoop jar /home/cloudera/Wordcount.jar Wordcount /inputfolder1/Processfile1.txt /out1.txt /out1

12. Finally, to see the output:

    hdfs dfs –cat /out1/part-r-00000

**Program:**

```
import java.io.IOException;
import java.util.StringTokenizer
; import org.apache.hadoop.conf.
Configuration; import
org.apache.hadoop.fs.Pat h; import
org.apache.hadoop.io.Int
Writable; import

org.apache.hadoop.io.Te xt; import
org.apache.hadoop.mapr educe.Job;
import org.apache.hadoop.mapr
educe.Mapper; import
org.apache.hadoop.mapr
educe.Reducer; import
org.apache.hadoop.mapr
educe.lib.input.FileInput Format;
import org.apache.hadoop.mapr
educe.lib.output.FileOut
putFormat;


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();     Job job = Job.getInstance(conf,
"word count"); job.setJarByClass(WordCount.class);
job.setMapperClass(TokenizerMapper.
class); job.setCombinerClass(IntSumReducer.
class); job.setReducerClass(IntSumReducer.c
lass); job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.c lass);
     FileInputFormat.addInputPath(job, new Path(args[0]));

       FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

 }

}
```

## Output:

```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/WordCount.jar WordCount /Wordcount/text.txt /op
21/09/19 04:15:21 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
21/09/19 04:15:22 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your appl
21/09/19 04:15:22 INFO input.FileInputFormat: Total input paths to process : 1
21/09/19 04:15:23 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:952)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:690)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:879)
21/09/19 04:15:23 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:952)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:690)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:879)
21/09/19 04:15:23 INFO mapreduce.JobSubmitter: number of splits:1
21/09/19 04:15:23 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1632040711600_0002
21/09/19 04:15:24 INFO impl.YarnClientImpl: Submitted application application_1632040711600_0002
21/09/19 04:15:24 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1632040711600_0002/
21/09/19 04:15:24 INFO mapreduce.Job: Running job: job_1632040711600_0002
21/09/19 04:15:38 INFO mapreduce.Job: Job job_1632040711600_0002 running in uber mode : false
21/09/19 04:15:38 INFO mapreduce.Job:  map 0% reduce 0%
21/09/19 04:15:49 INFO mapreduce.Job:  map 100% reduce 0%
21/09/19 04:16:01 INFO mapreduce.Job:  map 100% reduce 100%
21/09/19 04:16:01 INFO mapreduce.Job: Job job_1632040711600_0002 completed successfully
21/09/19 04:16:02 INFO mapreduce.Job: Counters: 49
```

[Apache Hadoop 3.3.1... | cloudera | cloudera@quickstart:~

File Edit View Search Terminal Help

```
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=8642
                Total time spent by all reduces in occupied slots (ms)=9984
                Total time spent by all map tasks (ms)=8642
                Total time spent by all reduce tasks (ms)=9984
                Total vcore-milliseconds taken by all map tasks=8642
                Total vcore-milliseconds taken by all reduce tasks=9984
                Total megabyte-milliseconds taken by all map tasks=8849408
                Total megabyte-milliseconds taken by all reduce tasks=10223616
        Map-Reduce Framework
                Map input records=15
                Map output records=14
                Map output bytes=137
                Map output materialized bytes=41
                Input split bytes=115
                Combine input records=14
                Combine output records=3
                Reduce input groups=3
                Reduce shuffle bytes=41
                Reduce input records=3
                Reduce output records=3
                Spilled Records=6
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=352
                CPU time spent (ms)=2310
                Physical memory (bytes) snapshot=368549888
                Virtual memory (bytes) snapshot=3015217152
                Total committed heap usage (bytes)=226365440
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=87
        File Output Format Counters
                Bytes Written=23
[cloudera@quickstart ~]$ hadoop fs -ls /op
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2021-09-19 04:16 /op/_SUCCESS
-rw-r--r--   1 cloudera supergroup         23 2021-09-19 04:15 /op/part-r-00000
 [cloudera@quickstart ~]$ hdfs dfs -cat /out1/part-r-00000
  is      6
  this    6
```

**Conclusion**: Thus, the WordCount MapReduce program is successfully executed in Cloudera.