# HADOOP WORD COUNT PROJET REPORT

FERHAT KAMALI 171805056

ALİ BİÇİCİ 191805075 MUHAMMED FURKAN YILMAZ 181805069

# CONTENTS

This project created as a final exam group project. As a group we will try to show you Hadoop word count algorithm's basic installation, configuration and using. Hadoop word count algorithm tries to calculate how many times the word used in a document.

**First Part Installation & Configuration**

**FIRST STEP: INSTALLATION**

1. Download the Java 8 Package[1].
2. Extract the Java file
3. Edit the environment variables on control panel.
4. Download the Hadoop files[2] an extract them.
5. Do the configurations[3] on core-site, hdfs-site, mapred-site, yarn-site files.
6. Create new folder as "data" in Hadoop file also you need to create "datanode" and "namenode" files in the data file.
7. Then you can check the situation on command with "2adoop version" code

**SECOND STEP: USING THE WORD COUNT ALGORITHM**

1. First you need to create a Java Project on Eclipse
2. Then you need to create 3 different classes as WCDriver, WCMapper and WCReducer.
3. Check the Java build Path configurations.
4. After that you should field the classes with code which you can found on Hadoop website[4]
5. You need to make a jar file. On project settings you should choose export as "Select export destination as Jar file then save it
6. After all you can start to use Hadoop on your terminal

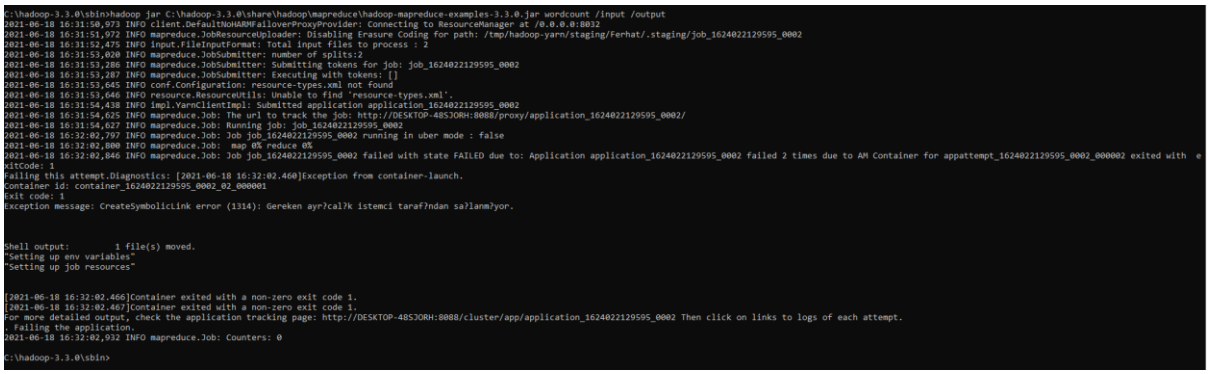
NOTE: we could not do that like write in upside you can read Error 1

**CODES THAT WE USE**

```
//////For create an input direcory in HDFS

hadoop fs -mkdir /input

//////Copy the input text file named input_file.txt in the input directory (input_dir)of
HDFS.

hadoop fs -put C:/inputtxt.txt /input

\\\\\\Verify content of the copied file.

hadoop fs -cat /input_dir/inputtxt.txt
//////Run MapReduceClient.jar and also provide input and out directories.

hadoop jar C:\hadoop-3.3.0\share\hadoop\mapreduce\hadoop-mapreduce-examples-3.3.0.jar
wordcount /input /output
```

**Finally we have experienced a fail like this:**

While the code we wrote here should create the word counts and the settings folder, it only created the settings folder and did not create the output folder and file. As a result of our long research on the internet, we could not come to a conclusion and unfortunately we decided to submit the assignment in this form.




**ERRORS**

**ERROR 1**

First of all, we tried to run the program using the code given below, but even though we tried in ubuntu and windows 10, we could not create the class files and then turn them into jars.

While looking for a solution to this, we learned that the "hadoop-mapreduce-examples-3.3.0.jar" file contains the wordcount feature and we decided to do the homework with it.

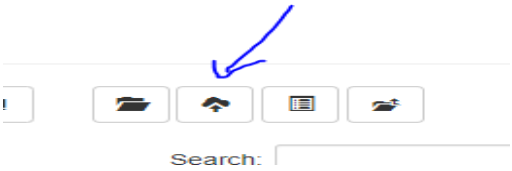
```
import java.io.IOException;

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;
```

```
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.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

**ERROR 2**

When add txt file to the hadoop system, although we had trouble at first, we could not load it to the system in any way after we wrote the "hdfs namenode -format" code when we tried again the next day. We also tried to load the file from the key you see below, but this time we got the "PERMISSION DENIED" error.




As a result of our investigations, we resolved this error as follows:

We have open “C:\hadoop-3.3.0\etc\hadoop\hdfs-site.xml” with editor and we add code down below

```
<configuration>
    <property>
        <name>dfs.permissions</name>
        <value>false</value>
    </property>
</configuration>
```

After that we tried again upload file from Hadoop system and we succeed this time.

**ERROR 3**

During the configuration part we got an error as “Could not find or load main class” that was basic issue because it is occurring our computer’s username. For example my username was “Ali Biçici” because the space of two vocabulary.

Solution was easy too. You should open the Hadoop folder then etc folder then you should open Hadoop folder and inside of the folder you should edit the “Hadoop-env” folder. In the source code last line was
“set HADOOP_IDENT_STRING=%USERNAME%”
and you need change as
“set HADOOP_IDENT_STRING=AliBiçici”
that means you should write your username as without space.

# Conclusion

After the Hadoop project we had some idea about Why we need to use Hadoop?
First, ability to store and process huge amounts of any kind of data, quickly and that makes Hadoop as Hadoop. Computing power is the second features to choose Hadoop. Hadoop's distributed computing model processes big data fast. Unlike the other databases, you don't have to preprocess data before storing it. You can easily grow your system to handle more data simply by adding nodes. Little administration is required. Data and application processing are protected against hardware failure. If a node goes down, jobs are automatically redirected to other nodes to make sure the distributed computing does not fail. Multiple copies of all data are stored automatically.
Final point, Hadoop has everything you need at the same time they giving this thing for free. High flexibility and scability and fault tolerance makes your project easier. You easily can word count analysis wit MapReduce.

**Distribution of tasks**

Codes written with Ferhat's computer errors solved from with All team members
Muhammed made Fundraising job and send them to Ferhat
Ali and Ferhat prepared Report
I can confidently say that everyone in the team worked and now has more or less knowledge of hadoop

**References**

[1] https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html#license-lightbox
[2] https://hadoop.apache.org/releases.html
[3] https://drive.google.com/file/d/1AMqV4F5ybPF4ab4CeK8B3AsjdGtQCdvy/view
[4]https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

| | |
|---|---|
| Ferhat KAMALI | 171805056 |
| Ali Biçici | 191805075 |
| Muhammed Furkan Yılmaz | 181805069 |