Word Count in java using Map Reduce

Report submitted to the SASTRA Deemed to be University as the requirement for the course

BCSCCS701R03: BIG DATA ANALYTICS

Submitted by

N G DATTA VARMA

(Reg No.: 121003179, B.Tech CSE)

FEBRUARY 2021

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 OF THE UGC ACT, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 OF THE UGC ACT, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA-613 401

BONAFIDE CERTIFICATE

This is to certify that the report titled "**Word Count in Java using Map Reduce**" submitted as a requirement for the course, **BCSCCS701R03 : BIG DATA ANALYTICS** for B.Tech. is a bonafide record of the work done by **Mr. N G Datta Varma** (**Reg. No. 121003179, B. Tech, CSE**) during the academic year 2020-21, in the School of Computing.

Project *Viva Voice* to be held on : 15/02/2021

Examiner 1 **Examiner 2**

LIST OF FIGURES

ABBREVIATIONS

HDFS	Hadoop Distributed File System
Fig	Figure
JAR	JAVA Archive
VM	Virtual Machine
SSH	Secure Shell

ABSTRACT

The main objective of the project "**Word Count in Java using Hadoop Map reduce"** is to actualize Word Count model code in Map Reduce to check the quantity of words of a given word in the input text file. In the mechanical world there are number of innovations which are creating a huge measure of information step by step that prompts arrangement of an innovation called Big Data. Big Data manages the huge and unstructured information that can be computationally dissected to uncover the patterns and examples of an information. In this paper the essential program called Word Count Map Reduce program executed in apache hadoop. Adjusting the info and diminishing the quantity of tasks that rolls out the improvements in execution of a program. The point of this paper is running the Word Count program with various parameters.

Keywords : Map Reduce, Word Count, Hadoop, Big Data.

Big Data Tools used:

- HDFS
- Hadoop

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Big Data term defines the collection of large datasets, where the data is in structured and unstructured formats. Structured data can present in the form of table format. So the data can be easy to analyze and processed by using data mining tools. Unstructured data refers the data does not have any table format i.e. it does not have any structure. So it is not resided in any traditional databases. Several challenges are encountered in big data while processing, storing and analyzing the data. To fast process the large volume of data within the short period of time, a tool is required which is called Hadoop. Hadoop is open source software which is developed by Apache for reliable, scalable and distributed computing. Apache Hadoop is a framework it uses simple programming models for distributed processing of large volume of data sets through the clusters of computers. It is designed to set up from single server to group of machines, each system offers the storage and local computation. HDFS and Map Reduce are the two major concepts of Hadoop. Both the Map Reduce and Hadoop are related to distributed computation. Basically Hadoop architecture is same as distributed master slave architecture I.e. in master slave only one system acts as master and remaining systems acts like servers. The use of Hadoop Distributed File System is for distributed and storage for computational capabilities. The purpose of Hadoop is for partitioning the data and it perform parallel computing for large data sets. In Map Reduce master schedules the work on the slave nodes. And the HDFS master is responsible partitioning and computing the data from slaves and it keeps track of the data where it is located.

1.1 What is MAP REDUCE?

This particular Map Reduce algorithm consists of two steps. One is mapping, where the data gets split and mapped to their respective keys. And the other is reducer, where the output from mapper function will be fed into this function to shuffle the data and compress them with respect to their particular keys.

1.2 IMPORTANCE OF MAP REDUCE

Due to Map Reduce, Hadoop is more powerful and efficient. This is what a small overview of Map Reduce, take a look on know how to divide work into sub work and how Map Reduce works. In this process, the total work divided into small divisions. Each division will process in parallel on the clusters of servers which can give individual output. And finally, these individual outputs give the final output. It is scalable and can use across many computers.

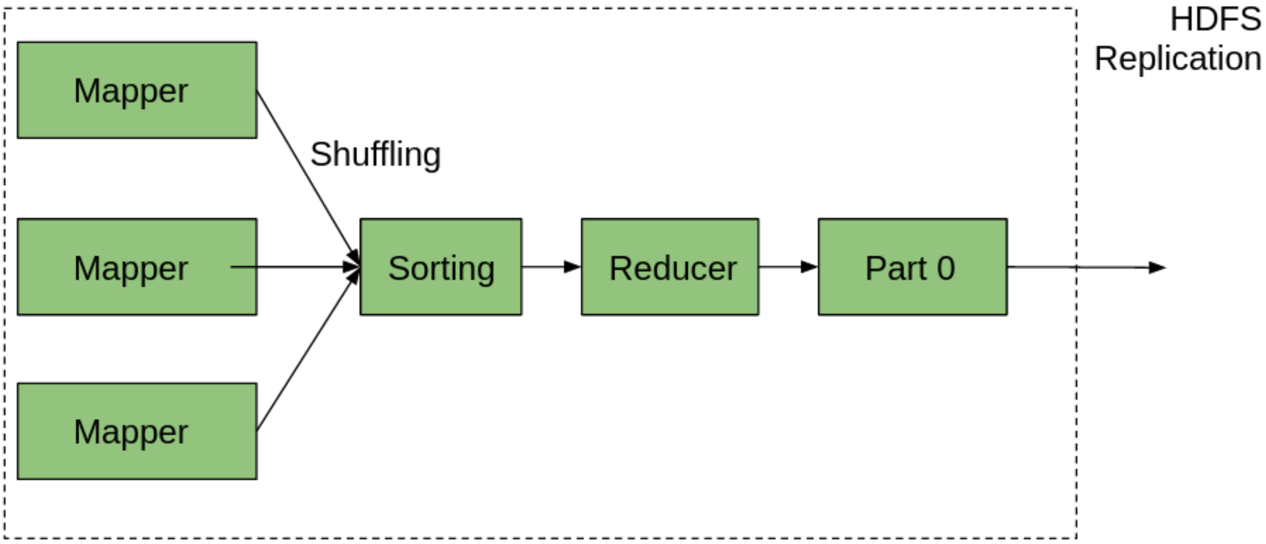


Fig 1.1 : Map Reduce Work Flow

1.3 TECHNICAL REQUIREMENTS

Following are the technical requirements:

1. **Google Cloud Platform :** GCP is used for Data Proc service which creates HDFS cluster easily and faster.
2. **Hadoop :** Hadoop is an open-source software that provides several services along with data storage. Distributed storage and processing of data in huge size can be achieved through Hadoop.
3. **HDFS:** The distributed file system which follows master-slave architecture and stores huge amount of data. HDFS also provides parallel processing for the applications.

1.4 OBJECTIVE

To Count the frequency of each word in any given input file, considering large dataset (book) as text input file. This process involves Map Reduce process. The job of Mapper is to plan the keys to the current instances and the part of Reducer is to total the keys of regular instances. Thus, everything is addressed as Key-value pair. Adjusting the info and diminishing the quantity of tasks that rolls out the improvements in execution of a program. The point of this paper is running the Word Count program with various parameters.

1.5 METHODOLOGY

In Map reduce, The job of Mapper is to plan the keys to the current instances and the part of Reducer is to total the keys of regular instances. Thus, everything is addressed as Key-value pair. It involves three steps.

1) Mapper Phase
2) Shuffle and Sort Phase
3) Reducer Phase

1.5.1 MAPPER PHASE

Hadoop Mapper is a capacity or undertaking which is utilized to deal with all information records from a text document and create the yield which functions as contribution for Reducer. It creates the yield by returning new key-pair sets. The information must be changed over to key-esteem sets as Mapper can not handle the crude information records or tuples(key-value sets). The mapper additionally creates some little blocks of information while handling the information records as a key-esteem pair. Key highlights and how the key-pair sets are produced in the Mapper are shown in below figure.

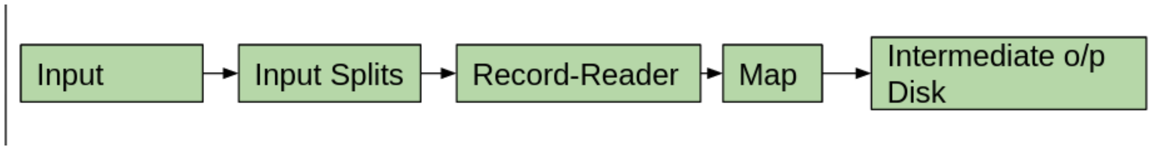


Fig 1.2 : Mapper Phase

1.5.2 SHUFFLE AND SORT PHASE

This phase consumes the output of mapping phase. Its task is to consolidate the relevant records from Mapping phase output. In our example, the same words are clubbed together along with their respective frequency.

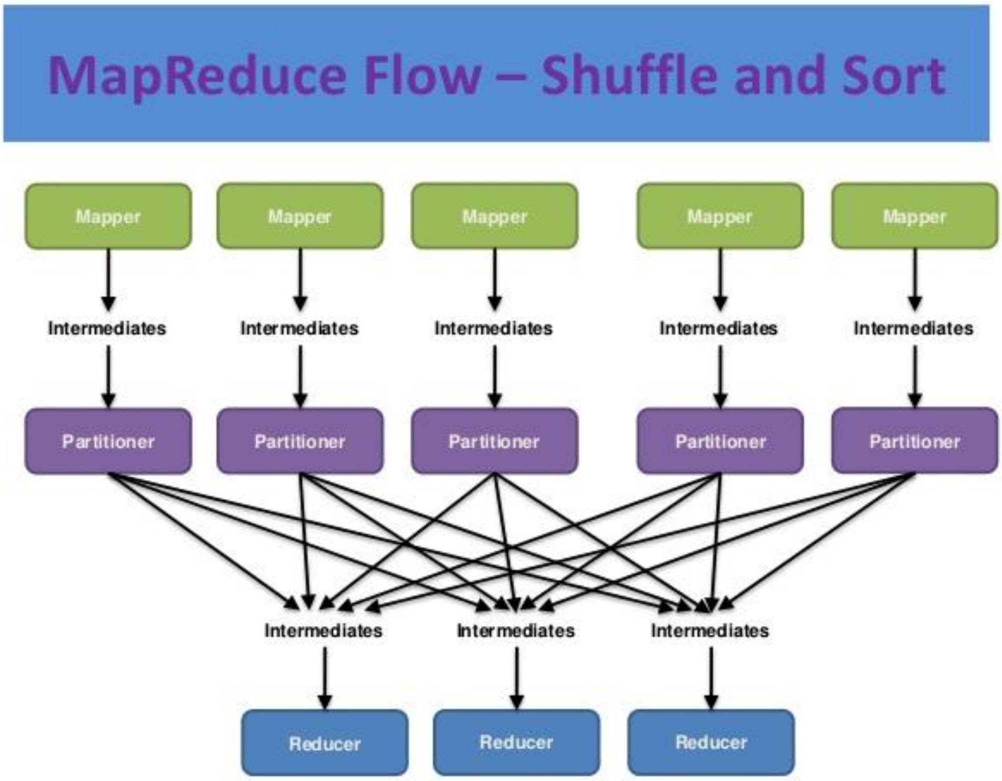


Fig 1.3 : Shuffle and sort Phase

1.5.3 Reducer Phase

Hadoop Reducer takes a bunch of a intermediate key-value pair delivered by the mapper as the information and runs a Reducer work on every one of them. One can total channel and consolidate this information (key, value) in various ways for a wide scope of processing. Reducer first processes the intermediate values for specific key created by the map function and afterward produces the yield (at least zero key-value pair).

1.6 Word Count - Map Reduce Application

The content from the information text document is tokenized into words to frame a key value pair with all the words present in the info text record. The key is the word from the information record and worth is '1'.

For example in the event that you think about the sentence "Bus Car Train Train Plane Car Bus Bus Plane". The mapper stage in the Word Count model will part the string into singular tokens for example words. After the map phase execution is finished effectively, shuffle stage is executed consequently wherein the key-value sets produced in the map stage are taken as information and afterward arranged in sequential order alphabetically.

In the reduce stage, all the keys are gathered and the qualities for comparable keys are amounted to discover the events for a specific word. It resembles a total stage for the keys produced by the guide stage. The reducer stage takes the yield of shuffle stage as info and afterward reduce the key-pair sets to unique keys with values added up. This is how the MapReduce word count program executes and outputs the number of occurrences of a word in any given input file.

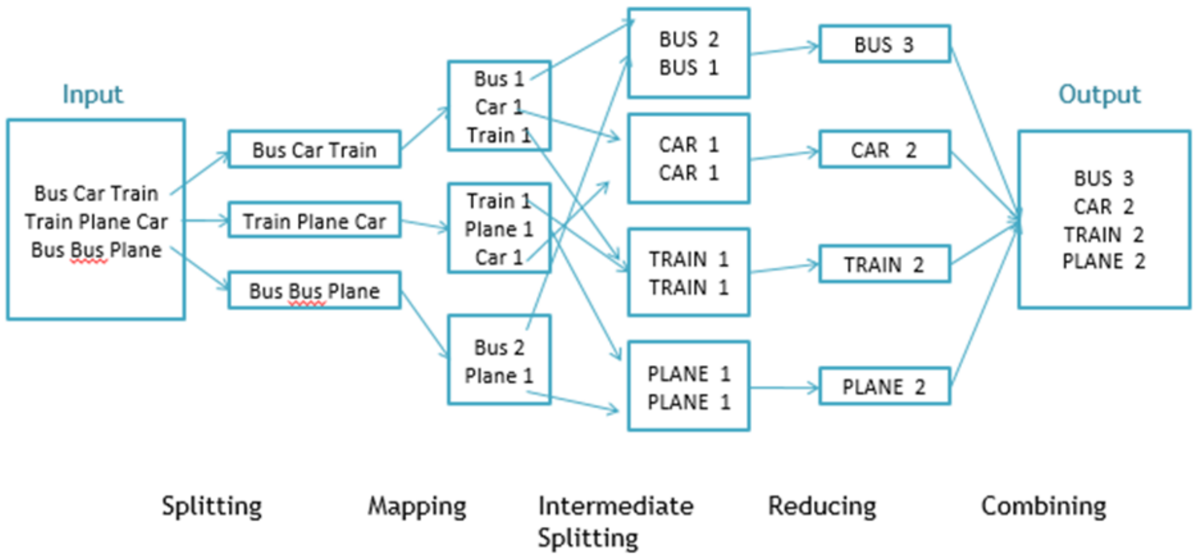


Fig 1.4 Word Count Example using Map Reduce

We will consider Big Data while implementing the program.

DATASET : Amazon Reviews as .txt file(1GB)

Link : https://www.dropbox.com/s/knfofa5z9lnvcrd/processedTExt.txt

Merits :

Merits of using Map Reduce technique for Word Count are

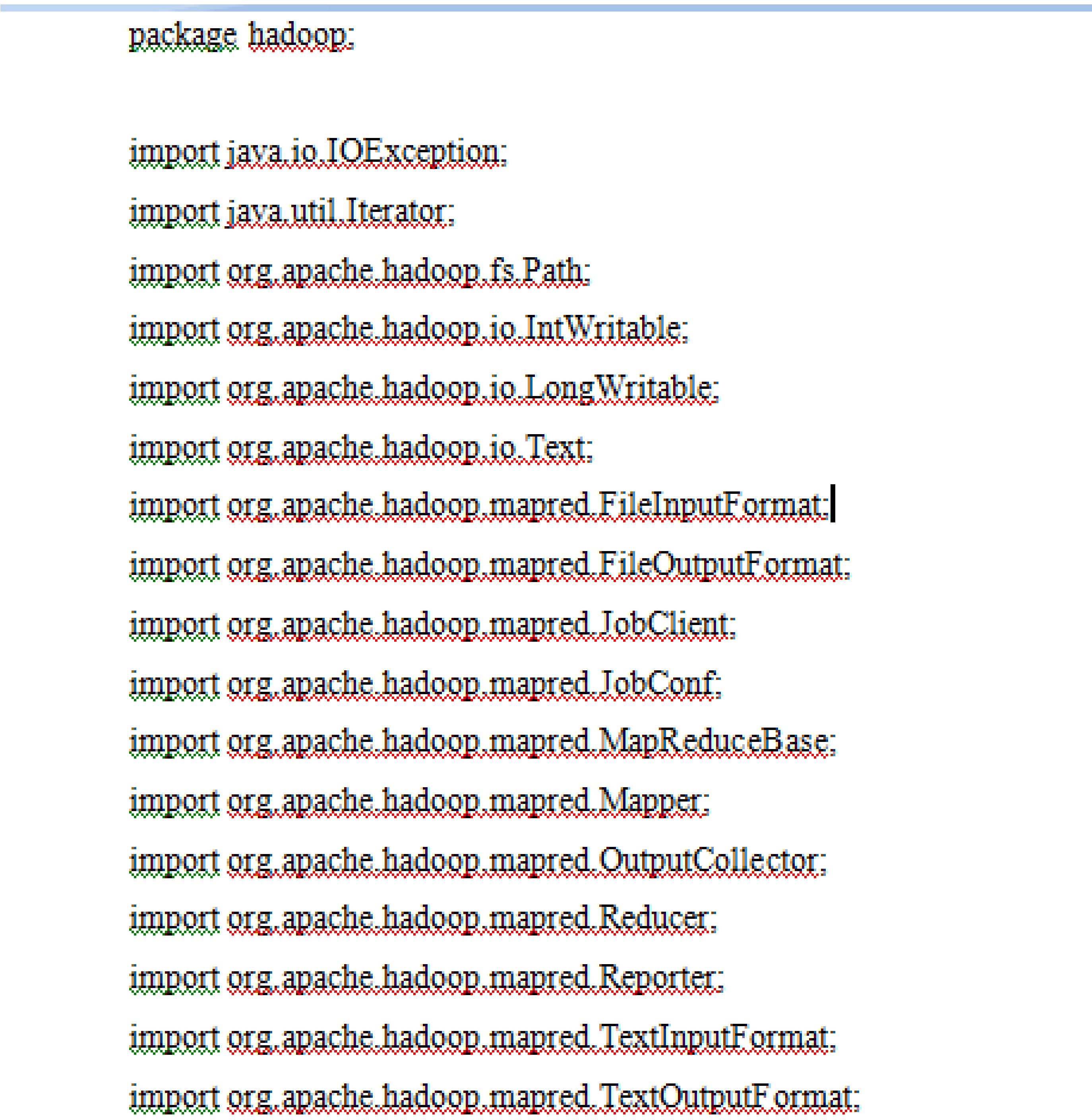
1) Scalabilty
2) Flexibility
3) Parallel Programming
4) Cost Effective
5) Fast

Demerits :

- It's not always very easy to implement each and everything as a MR program.
- When your intermediate jobs run in isolation (need to talk to each other)
- When your processing requires lot of data to be shuffled over the network.

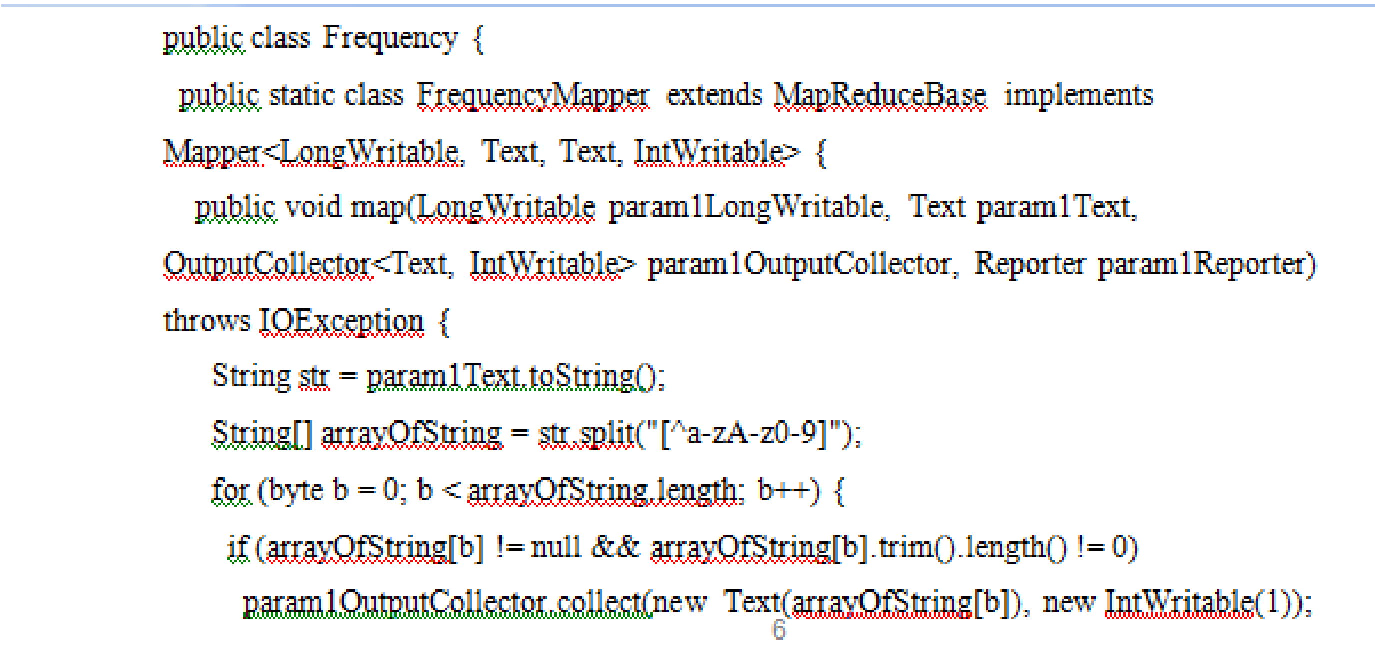
CHAPTER 2

SOURCE CODE



```
package hadoop;


import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
```

```
  public static void main(String[] paramArrayOfString) throws Exception {
    JobConf jobConf = new JobConf(Frequency.class);
    jobConf.setJobName("Frequency_Generator");
    jobConf.setOutputKeyClass(Text.class);
    jobConf.setOutputValueClass(IntWritable.class);
    jobConf.setMapperClass(FrequencyMapper.class);
    jobConf.setCombinerClass(FrequencyReducer.class);
    jobConf.setReducerClass(FrequencyReducer.class);
    jobConf.setInputFormat(TextInputFormat.class);
    jobConf.setOutputFormat(TextOutputFormat.class);
    FileInputFormat.setInputPaths(jobConf, new Path[] { new Path(paramArrayOfString[0]) });
    FileOutputFormat.setOutputPath(jobConf, new Path(paramArrayOfString[1]));
    JobClient.runJob(jobConf);
  }
}




    }
  }
}

public static class FrequencyReducer extends MapReduceBase implements Reducer<Text,
IntWritable, Text, IntWritable> {
  public void reduce(Text param1Text, Iterator<IntWritable> param1Iterator,
OutputCollector<Text, IntWritable> param1OutputCollector, Reporter param1Reporter)
throws IOException {
    int i = 0;
    while (param1Iterator.hasNext())
      i += ((IntWritable)param1Iterator.next()).get();
    param1OutputCollector.collect(param1Text, new IntWritable(i));
  }
}
```



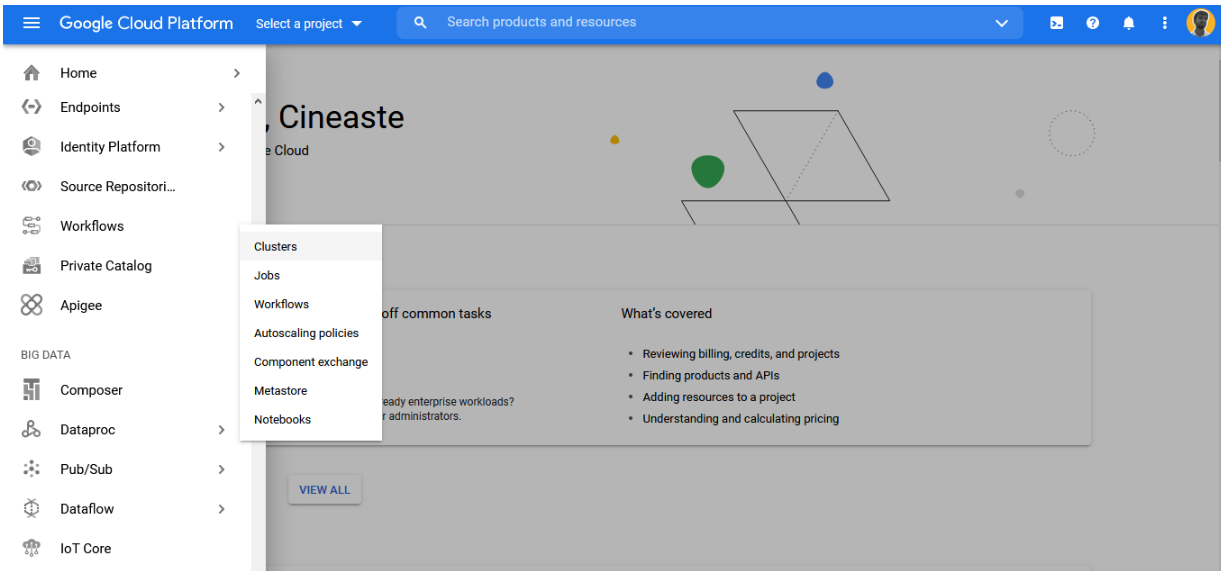
```
public class Frequency {
  public static class FrequencyMapper extends MapReduceBase implements
Mapper<LongWritable, Text, Text, IntWritable> {
    public void map(LongWritable param1LongWritable, Text param1Text,
OutputCollector<Text, IntWritable> param1OutputCollector, Reporter param1Reporter)
throws IOException {
      String str = param1Text.toString();
      String[] arrayOfString = str.split("[^a-zA-z0-9]");
      for (byte b = 0; b < arrayOfString.length; b++) {
        if (arrayOfString[b] != null && arrayOfString[b].trim().length() != 0)
          param1OutputCollector.collect(new Text(arrayOfString[b]), new IntWritable(1));
```

CHAPTER 3

SNAPSHOTS

3.1 Creating HDFS Cluster on Google Cloud Platform:

We have to create HDFS cluster to run Map Reduce Job.



What's covered

- Reviewing billing, credits, and projects
- Finding products and APIs
- Adding resources to a project
- Understanding and calculating pricing

Fig 3.1 : Creating HDFS Cluster

Configuring Nodes :

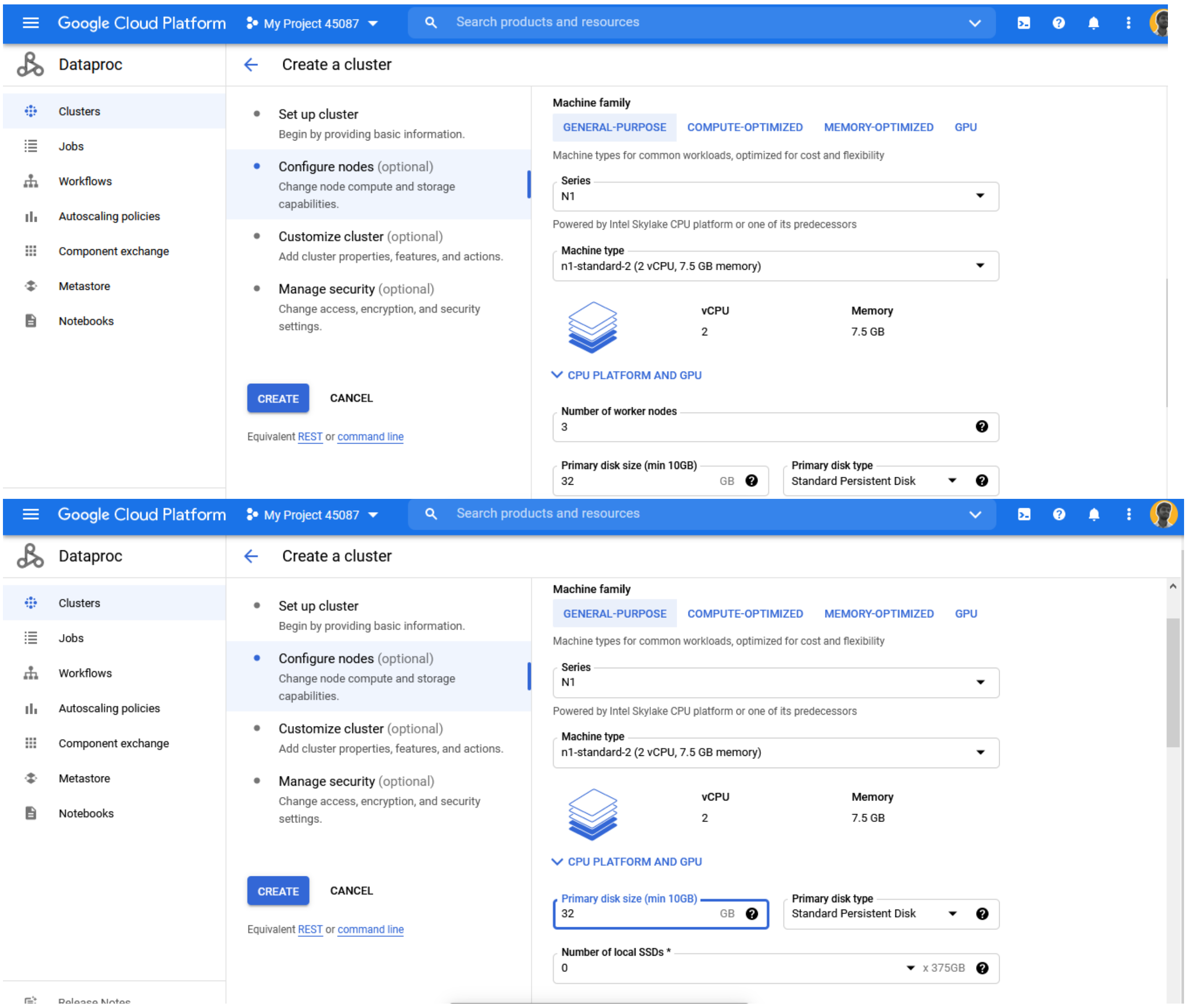
Machine Type : n1-standard-2(2 vCPU, 7.5 GB Memory)

Primary Disk Size : 32 GB

Worker Nodes

No. of Worker Nodes : 3

Primary Disk Size : 32 GB.



Create a cluster

- Set up cluster
  Begin by providing basic information.
- Configure nodes (optional)
  Change node compute and storage capabilities.
- Customize cluster (optional)
  Add cluster properties, features, and actions.
- Manage security (optional)
  Change access, encryption, and security settings.

Equivalent REST or command line

Machine family

Machine types for common workloads, optimized for cost and flexibility

Series: N1

Powered by Intel Skylake CPU platform or one of its predecessors

Machine type: n1-standard-2 (2 vCPU, 7.5 GB memory)

vCPU: 2

Memory: 7.5 GB

Number of worker nodes: 3

Primary disk size (min 10GB): 32 GB

Primary disk type: Standard Persistent Disk

Create a cluster

- Set up cluster
  Begin by providing basic information.
- Configure nodes (optional)
  Change node compute and storage capabilities.
- Customize cluster (optional)
  Add cluster properties, features, and actions.
- Manage security (optional)
  Change access, encryption, and security settings.

Equivalent REST or command line

Machine family

Machine types for common workloads, optimized for cost and flexibility

Series: N1

Powered by Intel Skylake CPU platform or one of its predecessors

Machine type: n1-standard-2 (2 vCPU, 7.5 GB memory)

vCPU: 2

Memory: 7.5 GB

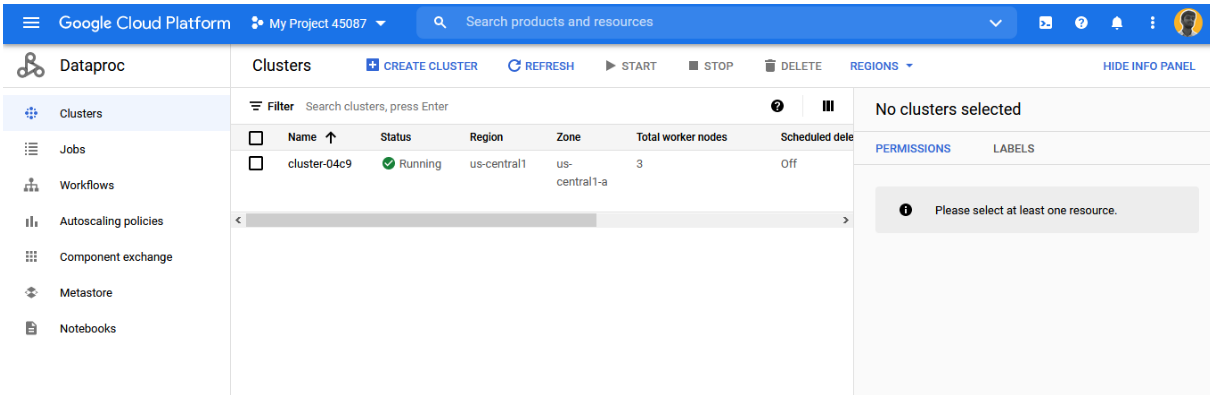
Primary disk size (min 10GB): 32 GB

Primary disk type: Standard Persistent Disk

Number of local SSDs *: 0 x 375GB

Fig 3.2 : Configuring Cluster

3.2 Cluster Created :



Clusters

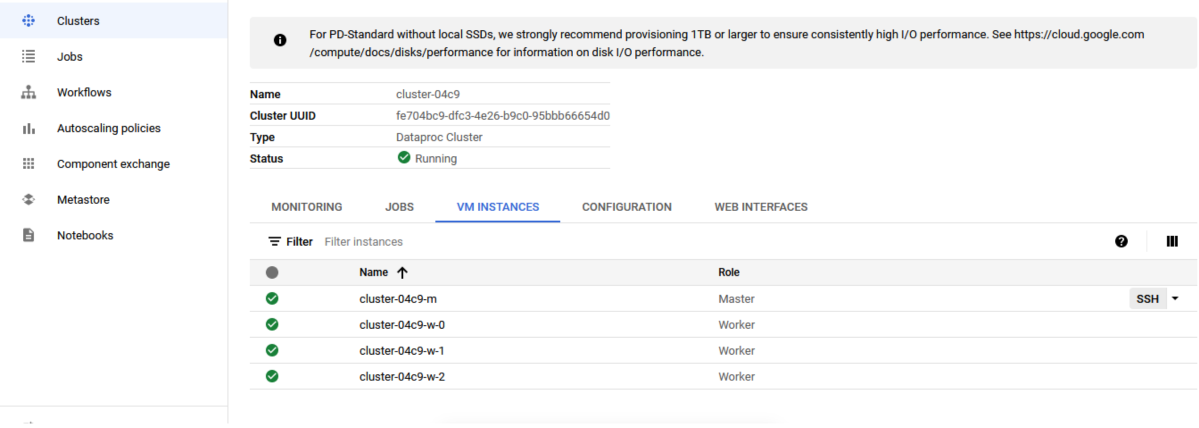
☐	Name ↑	Status	Region	Zone	Total worker nodes	Scheduled dele
☐	cluster-04c9	✓ Running	us-central1	us-central1-a	3	Off

No clusters selected

Please select at least one resource.

Fig 3.3 Cluster created

Connection SSH to VM :



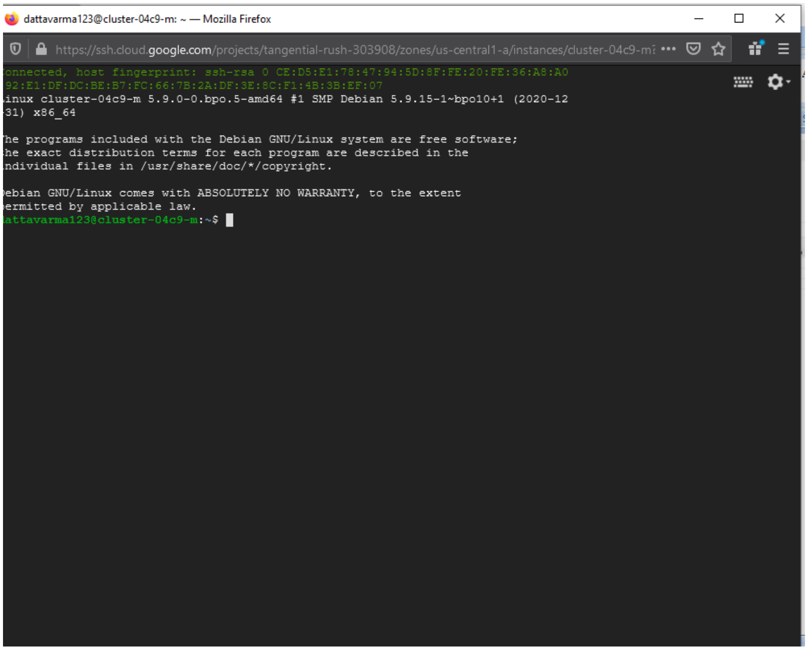
For PD-Standard without local SSDs, we strongly recommend provisioning 1TB or larger to ensure consistently high I/O performance. See https://cloud.google.com /compute/docs/disks/performance for information on disk I/O performance.

Name	cluster-04c9
Cluster UUID	fe704bc9-dfc3-4e26-b9c0-95bbb66654d0
Type	Dataproc Cluster
Status	✓ Running

●	Name ↑	Role	
✓	cluster-04c9-m	Master	SSH ▾
✓	cluster-04c9-w-0	Worker	
✓	cluster-04c9-w-1	Worker	
✓	cluster-04c9-w-2	Worker	

Fig 3.4: Connecting SSH to VM

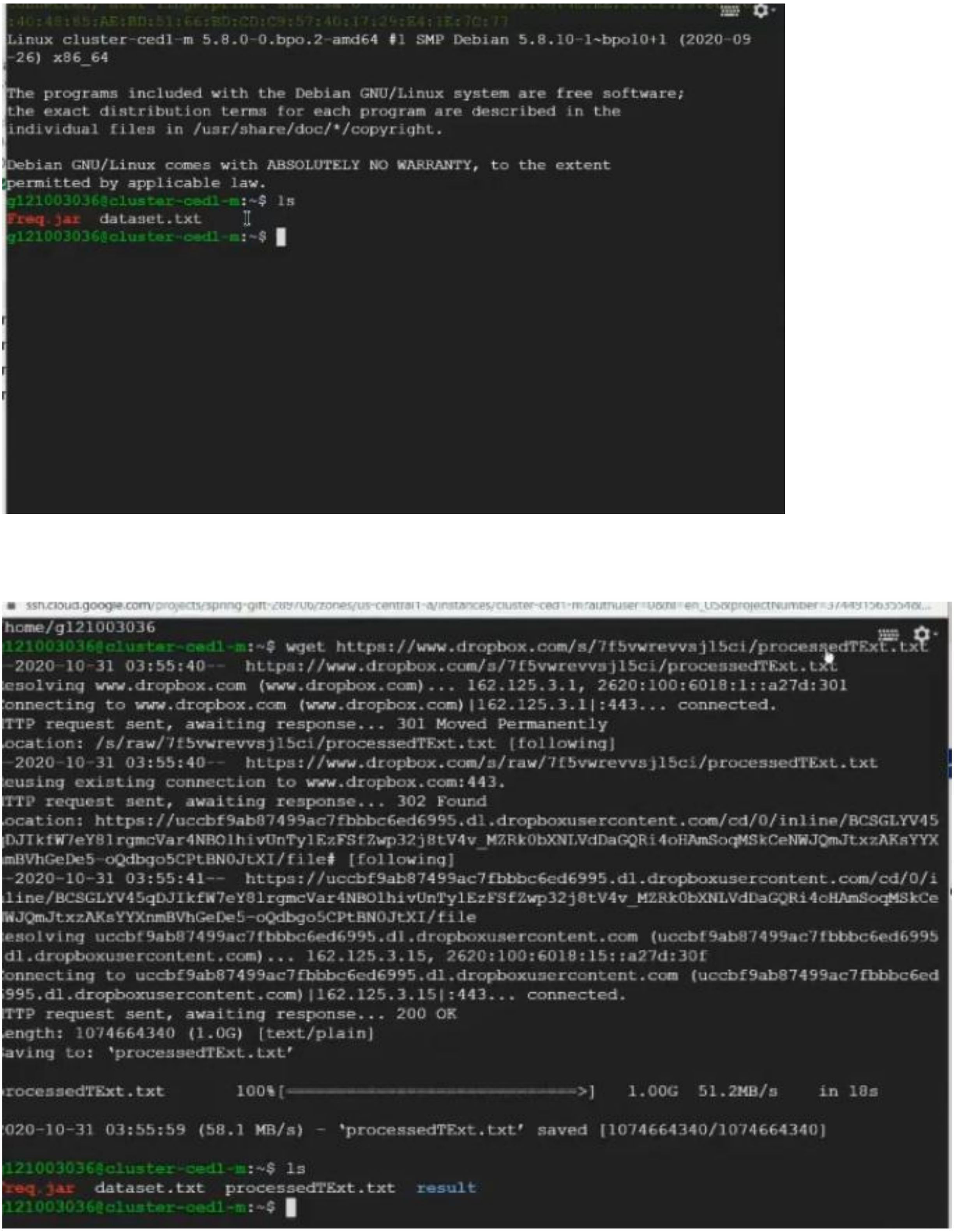
SSH Command Prompt:


dattavarma123@cluster-04c9-m: ~ — Mozilla Firefox
https://ssh.cloud.google.com/projects/tangential-rush-303908/zones/us-central1-a/instances/cluster-04c9-m?

```
onnected, host fingerprint: ssh-rsa 0 CE:D5:E1:78:47:94:5D:8F:FE:20:FE:36:A8:A0
92:E1:DF:DC:BE:B7:FC:66:7B:2A:DF:3E:8C:F1:4B:3B:EF:07
inux cluster-04c9-m 5.9.0-0.bpo.5-amd64 #1 SMP Debian 5.9.15-1~bpo10+1 (2020-12
31) x86_64

he programs included with the Debian GNU/Linux system are free software;
he exact distribution terms for each program are described in the
ndividual files in /usr/share/doc/*/copyright.

ebian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
ermitted by applicable law.
attavarma123@cluster-04c9-m:~$
```

Fig 3.5 : SSH Command Prompt

3.3 Uploading Jar File and Data Set

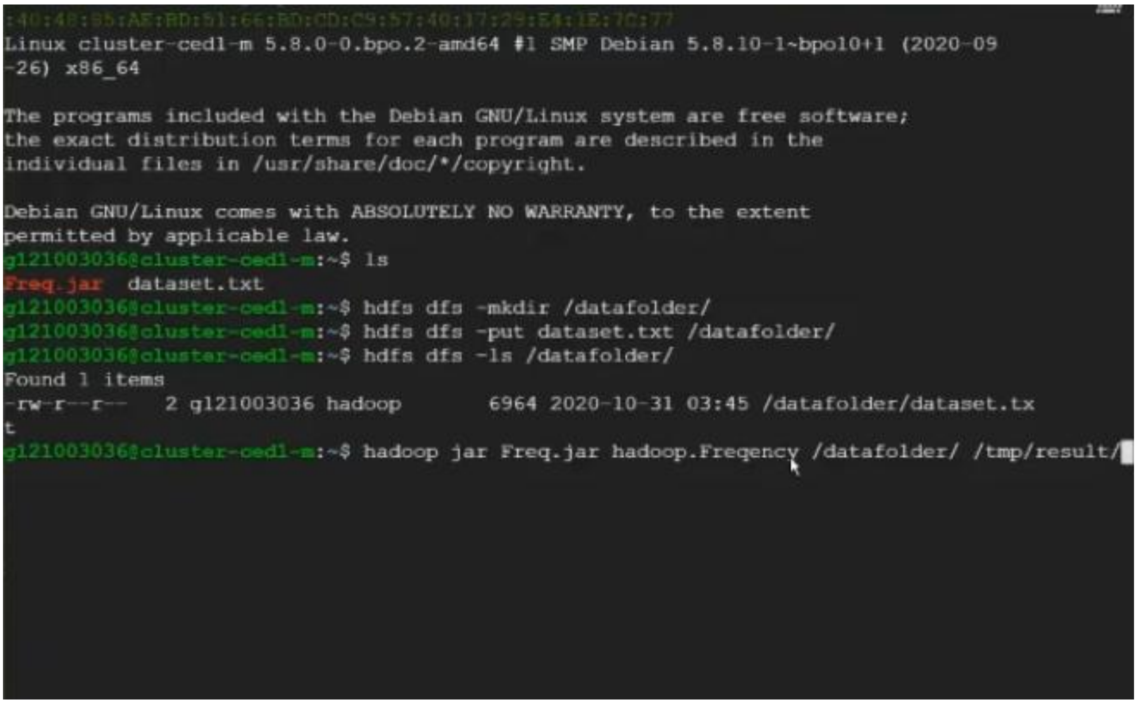


```
:40:48:85:AE:BD:51:66:BD:CD:C9:57:40:17:29:E4:1E:7C:77
Linux cluster-cedl-m 5.8.0-0.bpo.2-amd64 #1 SMP Debian 5.8.10-1~bpo10+1 (2020-09
-26) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
g121003036@cluster-cedl-m:~$ ls
Freq.jar  dataset.txt
g121003036@cluster-cedl-m:~$
```

```
home/g121003036
121003036@cluster-cedl-m:~$ wget https://www.dropbox.com/s/7f5vwrevvsj15ci/processedTExt.txt
-2020-10-31 03:55:40--  https://www.dropbox.com/s/7f5vwrevvsj15ci/processedTExt.txt
esolving www.dropbox.com (www.dropbox.com)... 162.125.3.1, 2620:100:6018:1::a27d:301
onnecting to www.dropbox.com (www.dropbox.com)|162.125.3.1|:443... connected.
TTP request sent, awaiting response... 301 Moved Permanently
ocation: /s/raw/7f5vwrevvsj15ci/processedTExt.txt [following]
-2020-10-31 03:55:40--  https://www.dropbox.com/s/raw/7f5vwrevvsj15ci/processedTExt.txt
eusing existing connection to www.dropbox.com:443.
TTP request sent, awaiting response... 302 Found
ocation: https://uccbf9ab87499ac7fbbbc6ed6995.dl.dropboxusercontent.com/cd/0/inline/BCSGLYV45
qDJIkfW7eY8lrgmcVar4NBOlhivUnTylEzFSfZwp32j8tV4v_MZRk0bXNLVdDaGQRi4oHAmSoqMSkCeNWJQmJtxzAKsYYX
mBVhGeDe5-oQdbgo5CPtBN0JtXI/file# [following]
-2020-10-31 03:55:41--  https://uccbf9ab87499ac7fbbbc6ed6995.dl.dropboxusercontent.com/cd/0/i
line/BCSGLYV45qDJIkfW7eY8lrgmcVar4NBOlhivUnTylEzFSfZwp32j8tV4v_MZRk0bXNLVdDaGQRi4oHAmSoqMSkCe
WJQmJtxzAKsYYXnmBVhGeDe5-oQdbgo5CPtBN0JtXI/file
esolving uccbf9ab87499ac7fbbbc6ed6995.dl.dropboxusercontent.com (uccbf9ab87499ac7fbbbc6ed6995
dl.dropboxusercontent.com)... 162.125.3.15, 2620:100:6018:15::a27d:30f
onnecting to uccbf9ab87499ac7fbbbc6ed6995.dl.dropboxusercontent.com (uccbf9ab87499ac7fbbbc6ed
995.dl.dropboxusercontent.com)|162.125.3.15|:443... connected.
TTP request sent, awaiting response... 200 OK
ength: 1074664340 (1.0G) [text/plain]
aving to: 'processedTExt.txt'

rocessedTExt.txt       100%[===============================>]   1.00G  51.2MB/s    in 18s

020-10-31 03:55:59 (58.1 MB/s) - 'processedTExt.txt' saved [1074664340/1074664340]

121003036@cluster-cedl-m:~$ ls
req.jar  dataset.txt  processedTExt.txt  result
121003036@cluster-cedl-m:~$
```

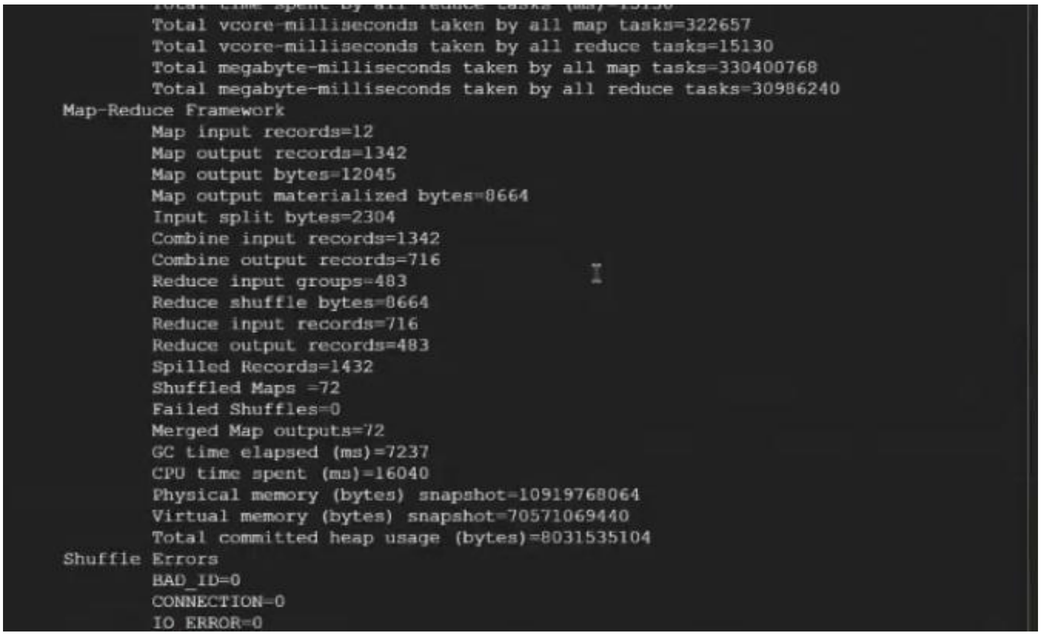
Fig 3.6 : Snapshot of uploading Jar File and Data Set

Creating HDFS Directory



```
:40:48:85:AE:BD:51:66:BD:CD:C9:57:40:17:29:E4:1E:7C:77
Linux cluster-cedl-m 5.8.0-0.bpo.2-amd64 #1 SMP Debian 5.8.10-1~bpo10+1 (2020-09
-26) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
g121003036@cluster-cedl-m:~$ ls
Freq.jar  dataset.txt
g121003036@cluster-cedl-m:~$ hdfs dfs -mkdir /datafolder/
g121003036@cluster-cedl-m:~$ hdfs dfs -put dataset.txt /datafolder/
g121003036@cluster-cedl-m:~$ hdfs dfs -ls /datafolder/
Found 1 items
-rw-r--r--   2 g121003036 hadoop       6964 2020-10-31 03:45 /datafolder/dataset.tx
t
g121003036@cluster-cedl-m:~$ hadoop jar Freq.jar hadoop.Freqency /datafolder/ /tmp/result/
```

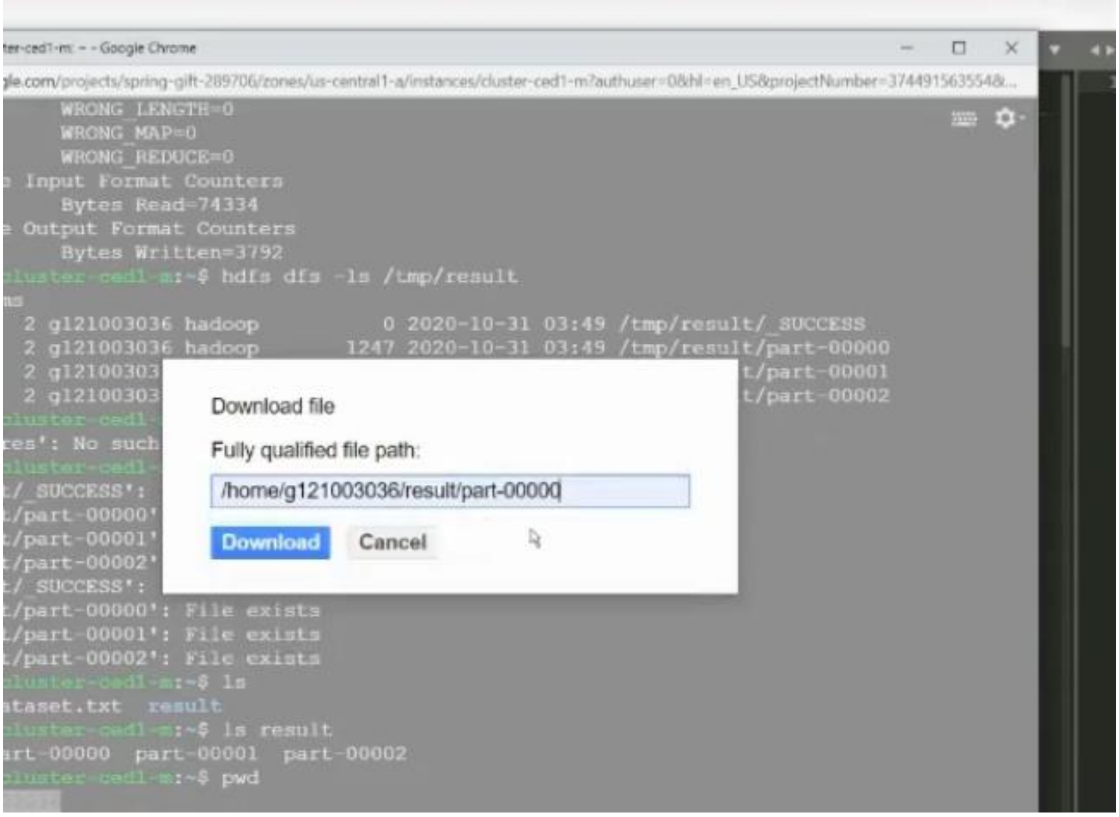
3.4 Running Map Reduce Job



```
                Total time spent by all reduce tasks (ms)=15130
                Total vcore-milliseconds taken by all map tasks=322657
                Total vcore-milliseconds taken by all reduce tasks=15130
                Total megabyte-milliseconds taken by all map tasks=330400768
                Total megabyte-milliseconds taken by all reduce tasks=30986240
        Map-Reduce Framework
                Map input records=12
                Map output records=1342
                Map output bytes=12045
                Map output materialized bytes=8664
                Input split bytes=2304
                Combine input records=1342
                Combine output records=716
                Reduce input groups=483
                Reduce shuffle bytes=8664
                Reduce input records=716
                Reduce output records=483
                Spilled Records=1432
                Shuffled Maps =72
                Failed Shuffles=0
                Merged Map outputs=72
                GC time elapsed (ms)=7237
                CPU time spent (ms)=16040
                Physical memory (bytes) snapshot=10919768064
                Virtual memory (bytes) snapshot=70571069440
                Total committed heap usage (bytes)=8031535104
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
```

Fig 3.7 Snapshot of Map Reduce Job

Transfering Output files from HDFS to Local Storage



ter-ced1-m: - - Google Chrome
gle.com/projects/spring-gift-289706/zones/us-central1-a/instances/cluster-ced1-m?authuser=0&hl=en_US&projectNumber=374491563554&...

```
      WRONG_LENGTH=0
      WRONG_MAP=0
      WRONG_REDUCE=0
e Input Format Counters
      Bytes Read=74334
e Output Format Counters
      Bytes Written=3792
cluster-ced1-m:~$ hdfs dfs -ls /tmp/result
ms
  2 g121003036 hadoop          0 2020-10-31 03:49 /tmp/result/_SUCCESS
  2 g121003036 hadoop       1247 2020-10-31 03:49 /tmp/result/part-00000
  2 g12100303                                        t/part-00001
  2 g12100303                                        t/part-00002
cluster-ced1-
res': No such
cluster-ced1-
t/_SUCCESS':
t/part-00000'
t/part-00001'
t/part-00002'
t/_SUCCESS':
t/part-00000': File exists
t/part-00001': File exists
t/part-00002': File exists
cluster-ced1-m:~$ ls
ataset.txt  result
cluster-ced1-m:~$ ls result
art-00000  part-00001  part-00002
cluster-ced1-m:~$ pwd
```

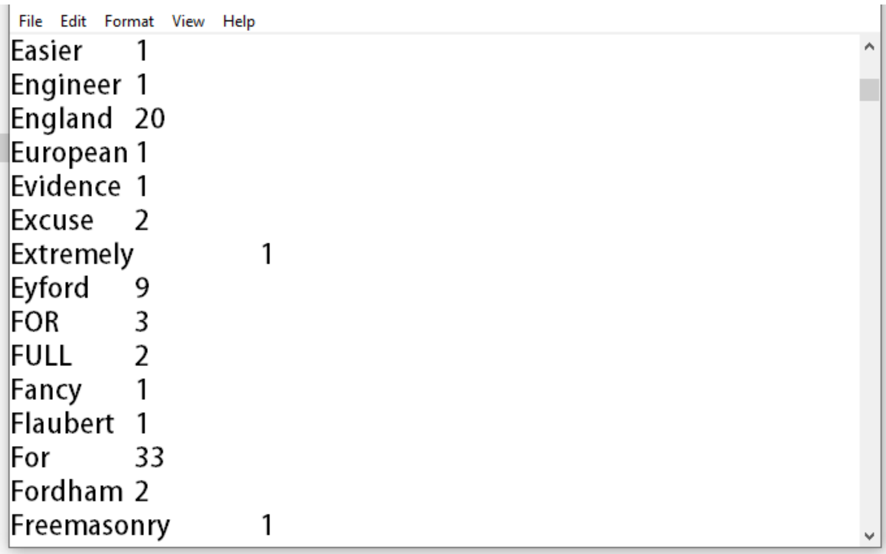
Download file

Fully qualified file path:

/home/g121003036/result/part-00000

After downloading the file change format to text.

3.5 Output File



```
Easier      1
Engineer  1
England   20
European 1
Evidence  1
Excuse     2
Extremely            1
Eyford      9
FOR         3
FULL        2
Fancy       1
Flaubert   1
For          33
Fordham 2
Freemasonry         1
```

Fig 3.8 Snapshot of Output

CHAPTER 4

CONCLUSION AND FUTURE PLANS

In this project we counted the frequency of words in a Big Data which is 1 GB text file.

Tasks done step by step

1) Created HDFS Cluster on GCP.
2) Uploaded Jar File and Data Set to HDFS.
3) Run Map Reduce Job on Cluster using SSH command prompt.
4) Transferred Output files from HDFS to local file System.
5) Frequency of words is in Output file.

Hadoop Map-Reduce is versatile and can likewise be utilized across numerous PCs. Numerous huge machines can be utilized to process jobs that couldn't be handled by a large machine. Future work can be done for Live Streaming Analysis using Map Reduce.

CHAPTER 5

REFERENCES

[1]http://hadoop.apache.org/,ApacheHadoop

[2] Maurya, M., & Mahajan, S. (2012, October). Performance analysis of MapReduce programs on Hadoop cluster. In Information and Communication Technologies (WICT), 2012 World CongressOn(pp.505-510).IEEE.

[3] Yang, X. and Sun, J., 2011, September. An analytical performance model of mapreduce. In Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on (pp. 306-310). IEEE.M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, “High resolution fiber distributed measurements with coherent OFDR,” in Proc. ECOC’00, 2000, paper11.3.4,p.109.

[4]Zaharia, Matei, et al. "Improving MapReduce performance in heterogeneous environments." Osdi. Vol. 8. No. 4. 2008. (2002) The IEEE website. [Online]. Available: http://www.ieee.org/

[5]https://www.tutorialspoint.com/hadoop/MapReducetutorial

[6] Chavan, Vibhavari, and Rajesh N. Phursule. "Survey paper on big data." Int. J. Comput. Sci.Inf.Technol5,no.6(2014):7932-7939.

[7] Blazhievsky, S., 2013. Introduction to Hadoop, MapReduce and HDFS for Big Data Applications.SNIAEducation.

[8] Arora, Suman, and Dr Madhu Goel. "Survey paper on Scheduling in Hadoop." International Journal of Advanced Research in Computer Science and Software Engineering 4.5(2014).

[9] Tan, Jian, Xiaoqiao Meng, and Li Zhang. "Performance analysis of coupling scheduler for mapreduce/hadoop." INFOCOM, 2012 Proceedings IEEE. IEEE, 2012.