


BIG DATA ANALYTICS

LAB ASSIGNMENT 3

Mahesh Pachare

FINAL YEAR B.TECH IT

191080054



AIM: Create mapreduce code for wordcount and execute it on multi-node cluster

THEORY:

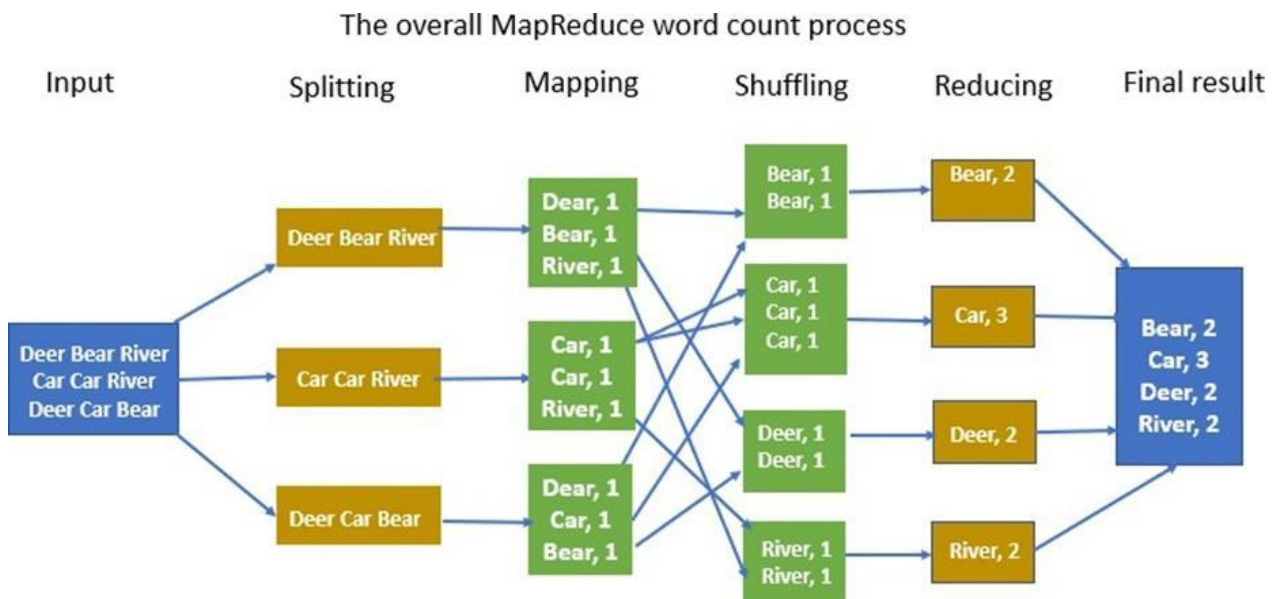
MapReduce is a parallel processing framework that is used to process large amounts of data across a cluster of computers. It was originally developed by Google and has since become a popular tool for big data processing.

The MapReduce programming model is based on two main operations: Map and Reduce. The Map operation takes in a dataset and maps each element in the dataset to a set of intermediate key-value pairs. The Reduce operation takes the intermediate key-value pairs and aggregates them into a final set of outputs. The map and reduce operations are performed in parallel across a large number of nodes in the cluster, allowing for efficient and scalable processing of large datasets.

The wordcount example is a classic use case for the MapReduce programming model. The goal of the word count is to count the number of occurrences of each word in a large dataset. The Map operation in this case maps each word in the dataset to a key-value pair, with the word as the key and the number of occurrences as the value. The Reduce operation then aggregates

these key-value pairs to produce a final count of the number of occurrences of each word in the dataset.

By executing this mapreduce code on a multi-node cluster, the processing can be distributed across many nodes, allowing for a large increase in processing power and a reduction in processing time.



MapReduce Architecture

MapReduce process has the following phases:

- Input Splits
- Mapping
- Shuffling and Sorting
- Reducing



Input Splits

MapReduce splits the input into smaller chunks called input splits, representing a block of work with a single mapper task.

Mapping

The input data is processed and divided into smaller segments in the mapper phase, where the number of mappers is equal to the number of input splits. RecordReader produces a key-value pair of the input splits using TextFormat, which Reducer later uses as input. The mapper then processes these key-value pairs using coding logic to produce an output of the same form.

Shuffling

In the shuffling phase, the output of the mapper phase is passed to the reducer phase by removing duplicate values and grouping the values. The output remains in the form of keys and values in the mapper phase. Since shuffling can begin even before the mapper phase is complete, it saves time.

Sorting

Sorting is performed simultaneously with shuffling. The Sorting phase involves merging and sorting the output generated by the mapper. The intermediate key-value pairs are sorted by key before starting the reducer phase, and the values can take any order.

Sorting by value is done by secondary sorting.



Reducing

In the reducer phase, the intermediate values from the shuffling phase are reduced to produce a single output value that summarizes the entire dataset. HDFS is then used to store the final output.

Limitations of MapReduce

MapReduce also faces some limitations, and they are as follows:

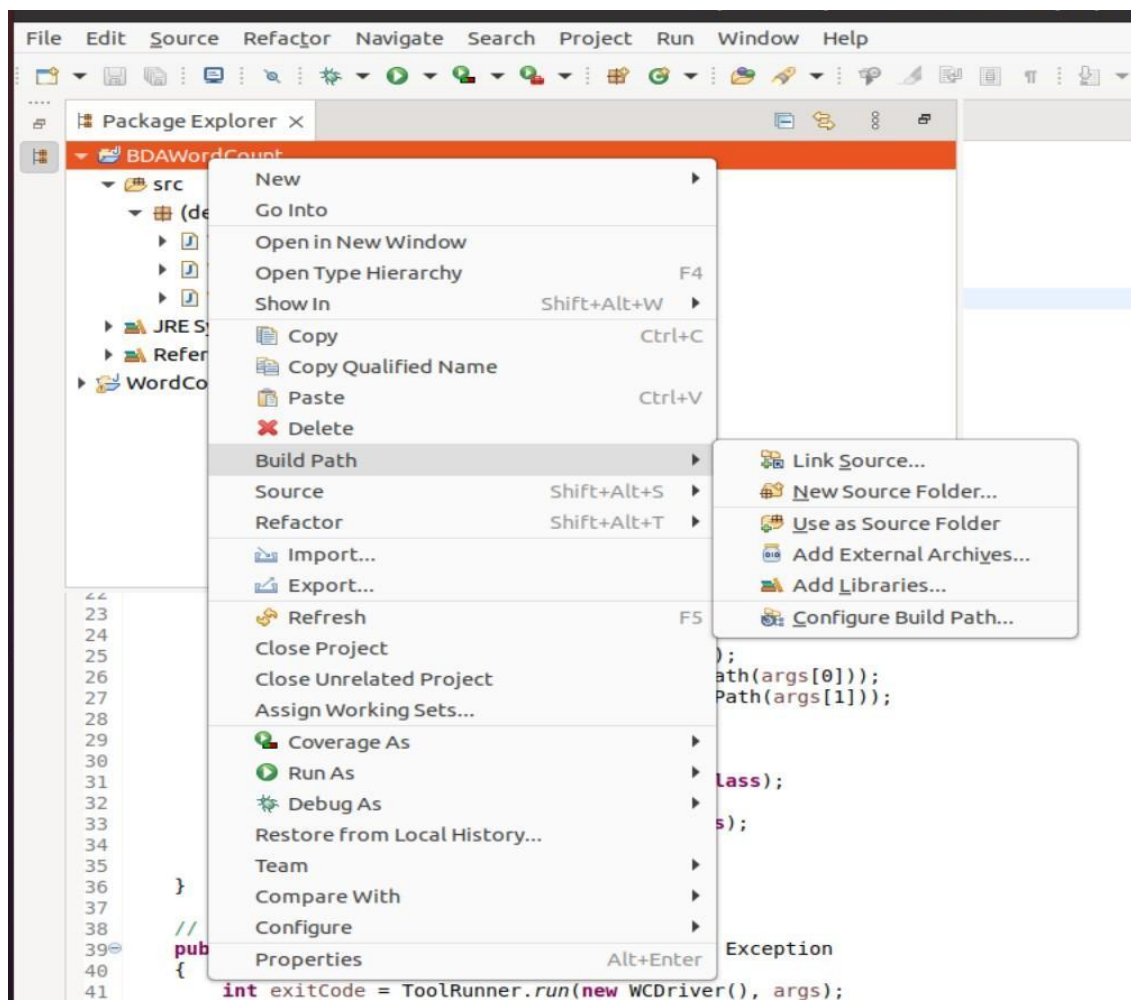
- MapReduce is a low-level programming model which involves a lot of writing code.
- The batch-based processing nature of MapReduce makes it unsuitable for real-time processing.
- It does not support data pipelining or overlapping of Map and Reduce phases.
- Task initialization, coordination, monitoring, and scheduling take up a large chunk of MapReduce's execution time and reduce its performance.
- MapReduce cannot cache the intermediate data in memory, thereby diminishing Hadoop's performance.

PROCEDURE:

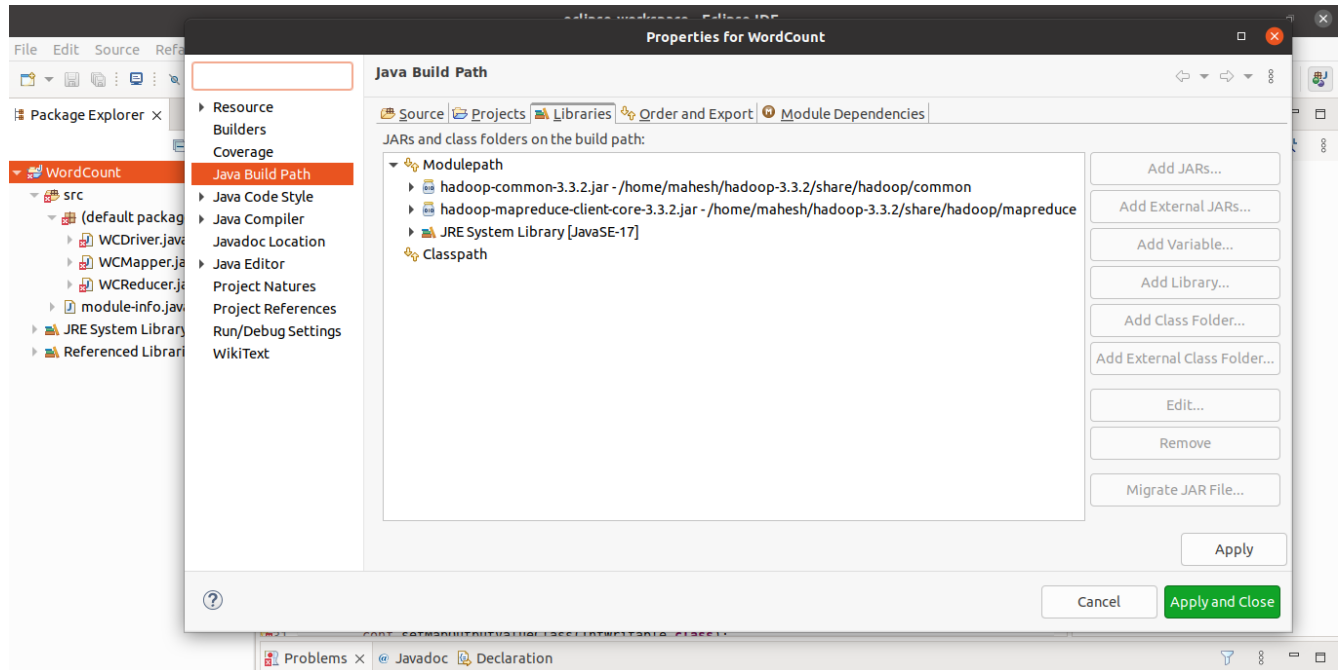
- 1) Starting the multi-node hadoop cluster on master and slave nodes
- 2) Install Eclipse

```
maresh@master:~$ sudo snap install --classic eclipse
[sudo] password for maresh:
Download snap "eclipse" (66) from channel "stable"
17% 2.42MB/s Download snap "eclipse" (66) from channel "stable"
Download snap "eclipse" (66) from channel "stable"
Download snap "eclipse" (66) from channel "stable"
eclipse 2022-12 from Snapcrafters installed
maresh@master:~$ sudo apt install default-jre
Reading package lists... Done
```

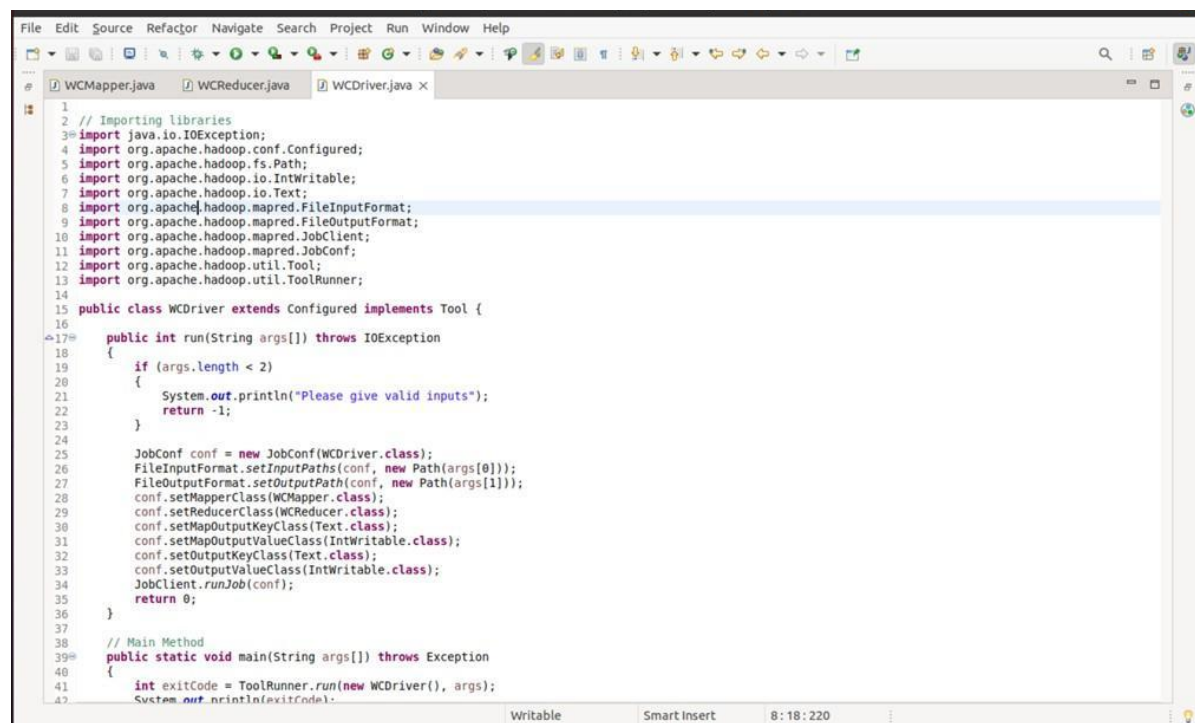
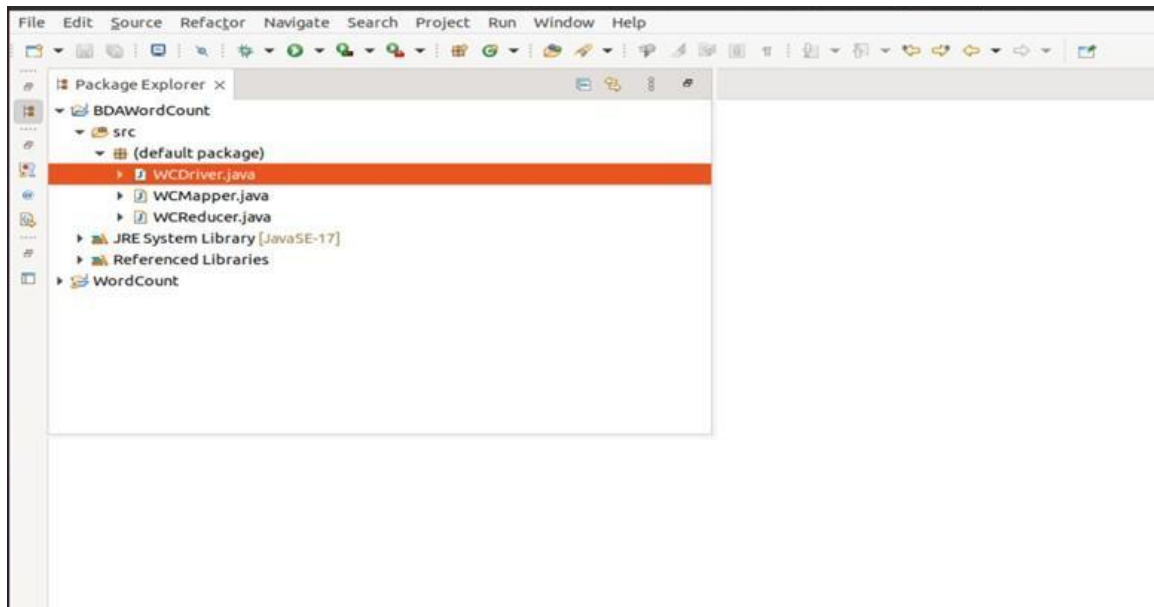
- 3) Now start the eclipse and create a java project.
- 4) Import necessary jar files from hadoop. (Go to Build Path -> Configure Build Path)



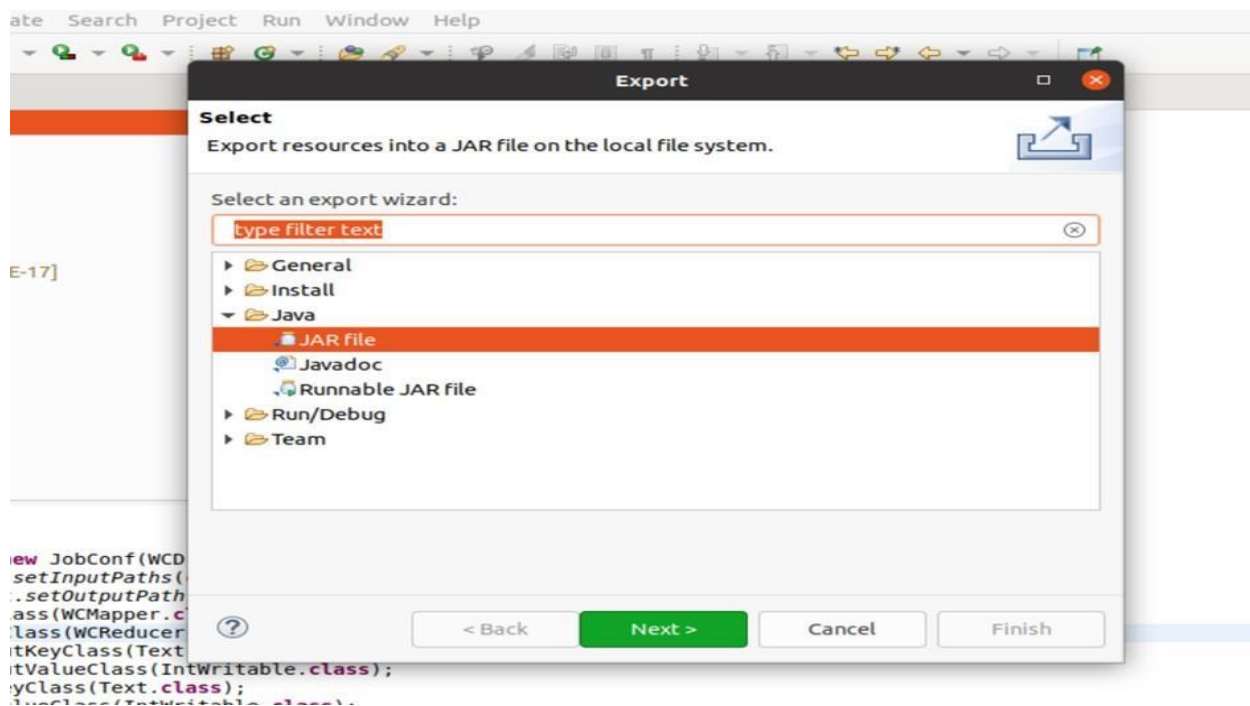
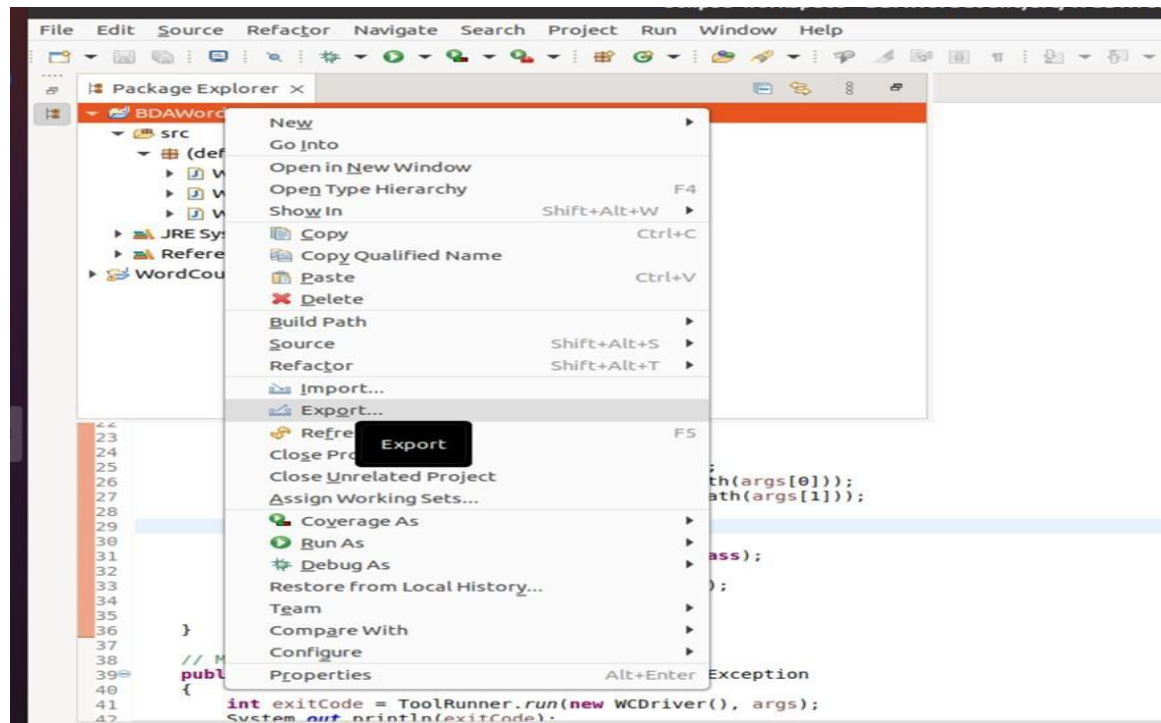
(Go to Add External JARs and then add JAR files present in hadoop directory)

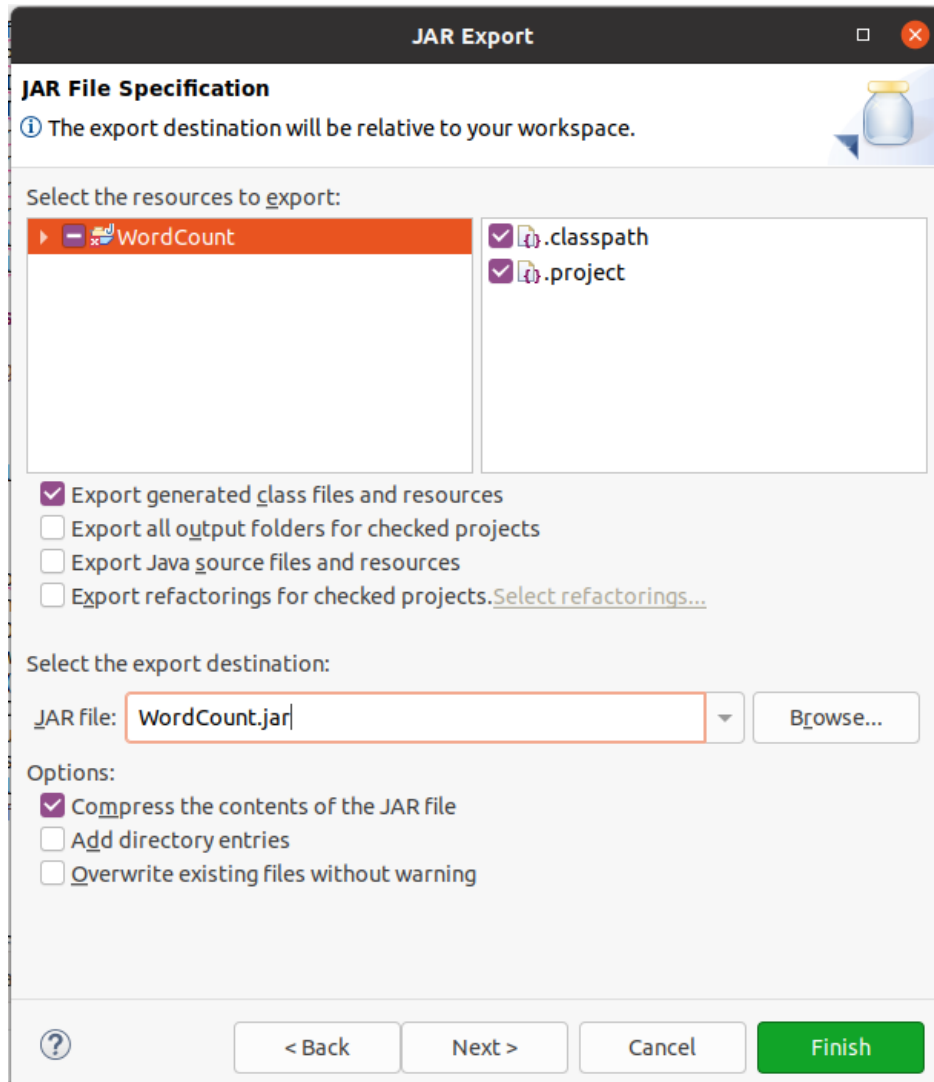


5) Write the code for Word Count MapReduce to count the number of words.



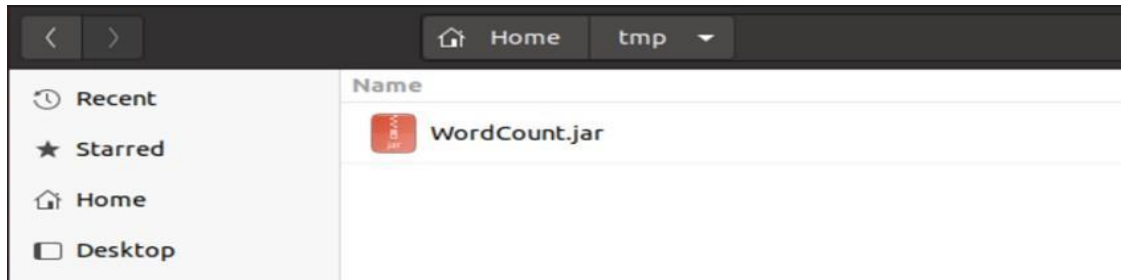
6) Export the code as a jar file to run it using hadoop.





(I have moved the jar file from eclipse workspace to another directory in /home/tmp directory)

7) Download the data as a text file to count the number of words in it.



```
maresh@master:~/hadoop-3.3.2/data$ wget https://www.gutenberg.org/files/972/972-0.txt
--2023-05-07 17:56:26-- https://www.gutenberg.org/files/972/972-0.txt
Resolving www.gutenberg.org (www.gutenberg.org)... 152.19.134.47, 2610:28:3090:3000:0:
bad:cafe:47
Connecting to www.gutenberg.org (www.gutenberg.org)|152.19.134.47|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 393851 (385K) [text/plain]
Saving to: '972-0.txt'

972-0.txt          100%[=====>] 384.62K  313KB/s   in 1.2s
2023-05-07 17:56:28 (313 KB/s) - '972-0.txt' saved [393851/393851]
```

- 8) Copying the data from "data" folder to "/files" folder of hadoop using hadoop dfs command in hadoop directory and checking the access of data file directory.

```
mahesh@master:~/hadoop-3.3.2/sbin$ cd ..
mahesh@master:~/hadoop-3.3.2$ hadoop dfs -copyFromLocal data /files
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

mahesh@master:~/hadoop-3.3.2$ hadoop dfs -ls /files
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

Found 1 items
-rw-r--r--  2 mahesh supergroup    393851 2023-05-07 17:59 /files/972-0.txt
```

- 9) Viewing the data in hadoop directory using web interface.

The screenshot shows the Hadoop web interface at localhost:9870/explorer.html#/files. A modal window titled "File information - 972-0.txt" is open, displaying details for the file. The modal includes options to "Download", "Head the file (first 32K)", and "Tail the file (last 32K)". Below these, a section titled "Block information -- Block 0" shows the following details:

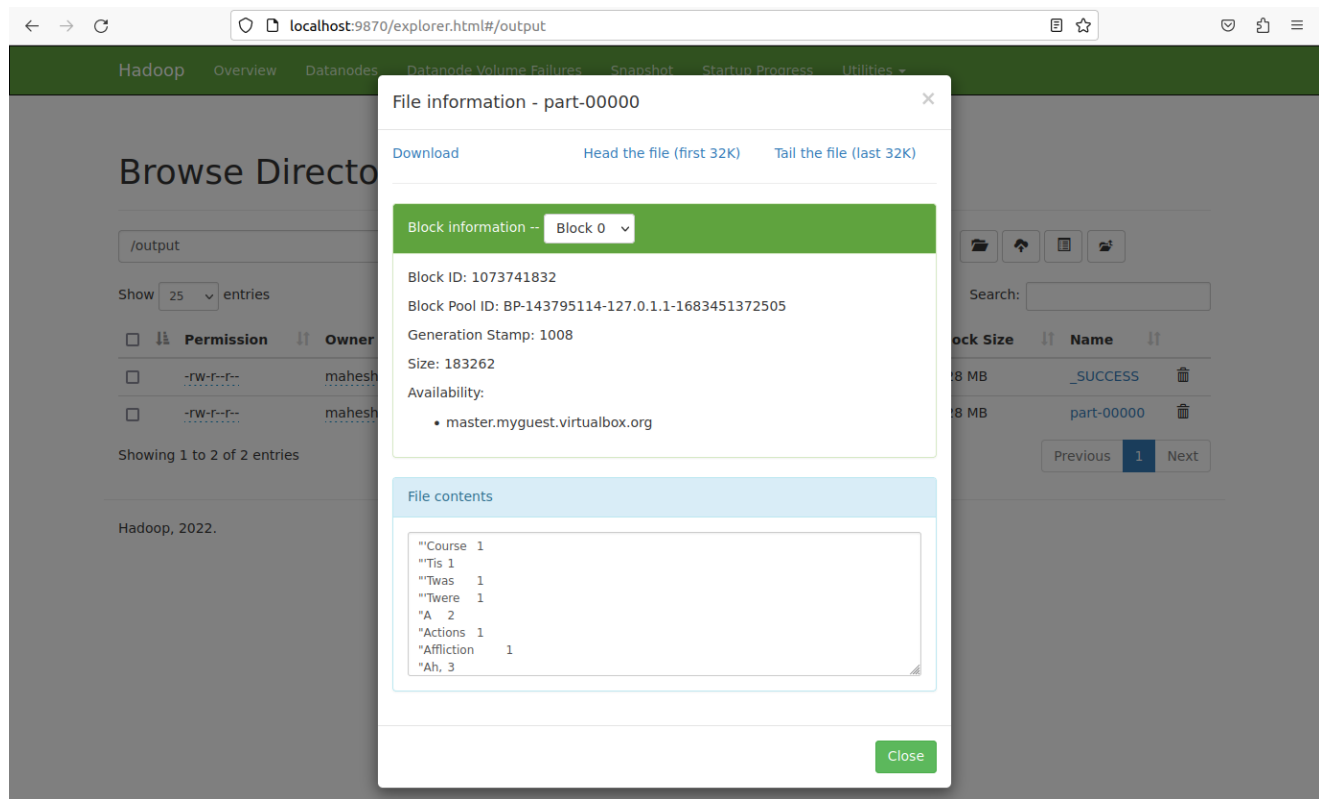
- Block ID: 1073741825
- Block Pool ID: BP-143795114-127.0.1.1-1683451372505
- Generation Stamp: 1001
- Size: 393851
- Availability:
 - master.myguest.virtualbox.org

The background interface shows a "Browse Directory" view for the path "/files". It includes a search bar, a table with columns for "Block Size" and "Name", and a list of files. The file "972-0.txt" is listed with a size of 28 MB. The interface also shows navigation buttons like "Previous", "1", and "Next".

10) Running the mapReduce task by supplying the Word Count jar file created above and providing input and output directories.

```
mahesh@master:~/hadoop-3.3.2$ hadoop jar /home/mahesh/tmp/WordCount.jar WCDriver /files /output
2023-05-07 18:20:25,212 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at master/127.0.1.1:8032
2023-05-07 18:20:25,446 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at master/127.0.1.1:8032
2023-05-07 18:20:26,041 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute
your application with ToolRunner to remedy this.
2023-05-07 18:20:26,101 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/mahesh/.staging/job_1683462450
968_0001
2023-05-07 18:20:27,107 INFO mapred.FileInputFormat: Total input files to process : 1
2023-05-07 18:20:27,232 INFO mapreduce.JobSubmitter: number of splits:2
2023-05-07 18:20:27,492 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1683462450968_0001
2023-05-07 18:20:27,492 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-05-07 18:20:27,740 INFO conf.Configuration: resource-types.xml not found
2023-05-07 18:20:27,741 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2023-05-07 18:20:28,972 INFO impl.YarnClientImpl: Submitted application application_1683462450968_0001
2023-05-07 18:20:29,087 INFO mapreduce.Job: The url to track the job: http://master.myguest.virtualbox.org:8088/proxy/application_1683462450968_0001/
2023-05-07 18:20:29,088 INFO mapreduce.Job: Running job: job_1683462450968_0001
2023-05-07 18:21:20,435 INFO mapreduce.Job: Job job_1683462450968_0001 running in uber mode : false
2023-05-07 18:21:20,438 INFO mapreduce.Job: map 0% reduce 0%
2023-05-07 18:21:29,642 INFO mapreduce.Job: map 100% reduce 0%
2023-05-07 18:21:37,720 INFO mapreduce.Job: map 100% reduce 100%
2023-05-07 18:21:37,747 INFO mapreduce.Job: Job job_1683462450968_0001 completed successfully
2023-05-07 18:21:37,990 INFO mapreduce.Job: Counters: 54
  File System Counters
    FILE: Number of bytes read=750378
    FILE: Number of bytes written=2326271
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=398119
    HDFS: Number of bytes written=183262
    HDFS: Number of read operations=11
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
    HDFS: Number of bytes read erasure-coded=0
  Job Counters
    Launched map tasks=2
    Launched reduce tasks=1
    Data-local map tasks=2
    Total time spent by all maps in occupied slots (ms)=12967
    Total time spent by all reduces in occupied slots (ms)=5380
    Total time spent by all map tasks (ms)=12967
    Total time spent by all reduce tasks (ms)=5380
    Total vcore-milliseconds taken by all map tasks=12967
    Total vcore-milliseconds taken by all reduce tasks=5380
    Total megabyte-milliseconds taken by all map tasks=13278208
    Total megabyte-milliseconds taken by all reduce tasks=5509120
  Map-Reduce Framework
    Map input records=9514
    Map output records=63160
    Map output bytes=624052
    Map output materialized bytes=750384
    Input split bytes=172
    Combine input records=0
    Combine output records=0
    Reduce input groups=17430
    Reduce shuffle bytes=750384
    Reduce input records=63160
    Reduce output records=17430
    Spilled Records=126320
    Shuffled Maps =2
    Failed Shuffles=0
    Merged Map outputs=2
    GC time elapsed (ms)=301
    CPU time spent (ms)=3060
    Physical memory (bytes) snapshot=708968448
    Virtual memory (bytes) snapshot=7591571456
    Total committed heap usage (bytes)=594018304
    Peak Map Physical memory (bytes)=272478208
    Peak Map Virtual memory (bytes)=2528903168
    Peak Reduce Physical memory (bytes)=174379008
    Peak Reduce Virtual memory (bytes)=2535436288
  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=397947
  File Output Format Counters
    Bytes Written=183262
0
mahesh@master:~/hadoop-3.3.2$
```

- 11) After completing the Map Reduce job, viewing the output in the web interface which displays the occurrence of each word in the provided input directory.



CONCLUSION:

In this experiment, map reduce code for word count has been done on eclipse, jar file for word count has been created, and then executed on a multi node cluster.