Experiment No. 4
Experiment on Hadoop Map-Reduce
Date of Performance: 14/08/2023
Date of Submission: 21/08/2023

Aim: To write a program to implement a word count program using MapReduce.

Theory:

WordCount is a simple program which counts the number of occurrences of each word in a given text input data set. WordCount fits very well with the MapReduce programming model making it a great example to understand the Hadoop Map/Reduce programming style. The implementation consists of three main parts:

1. Mapper

2. Reducer

3. Driver

Step-1. Write a Mapper

A Mapper overrides the ―map‖ function from the Class "org.apache.hadoop.mapreduce.Mapper" which provides <key, value> pairs as the input. A Mapper implementation may output <key,value> pairs using the provided Context .

Input value of the WordCount Map task will be a line of text from the input data file and the key would be the line number <line_number, line_of_text> . Map task outputs <word, one> for each word in the line of text.

Pseudo-code

```
void Map (key,value){

for each word x in

value:

output.collect(x,1);

    }
```

Step-2. Write a Reducer

A Reducer collects the intermediate <key,value> output from multiple map tasks and assemble a single result. Here, the WordCount program will sum up the occurrence of each word to pairs as <word, occurrence>.

Pseudo-code

```
void Reduce (keyword, <list of value>){
for each x in <list of value>:
    sum+=x;
    final_output.collect(keyword, sum);
}
```

Code:

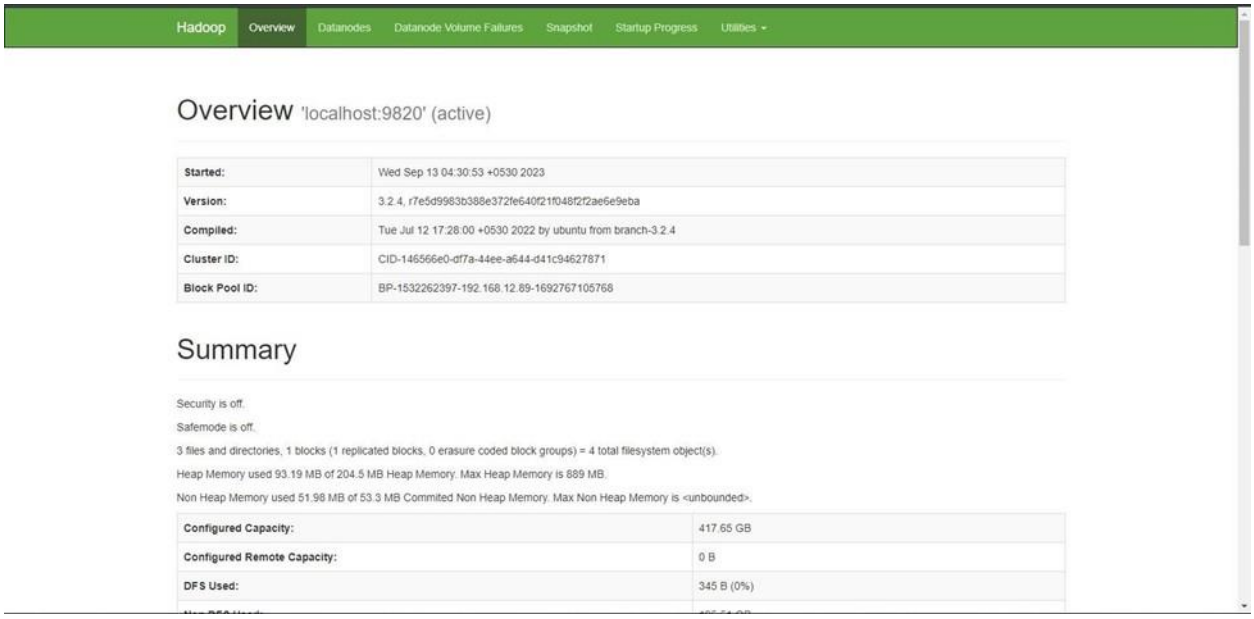
```
import
java.io.IOException;
import
java.util.StringTokenizer;
import
org.apache.hadoop.io.IntWritable;
import
org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import
org.apache.hadoop.mapreduce.Mapper;
import
org.apache.hadoop.mapreduce.Reducer;
import
```

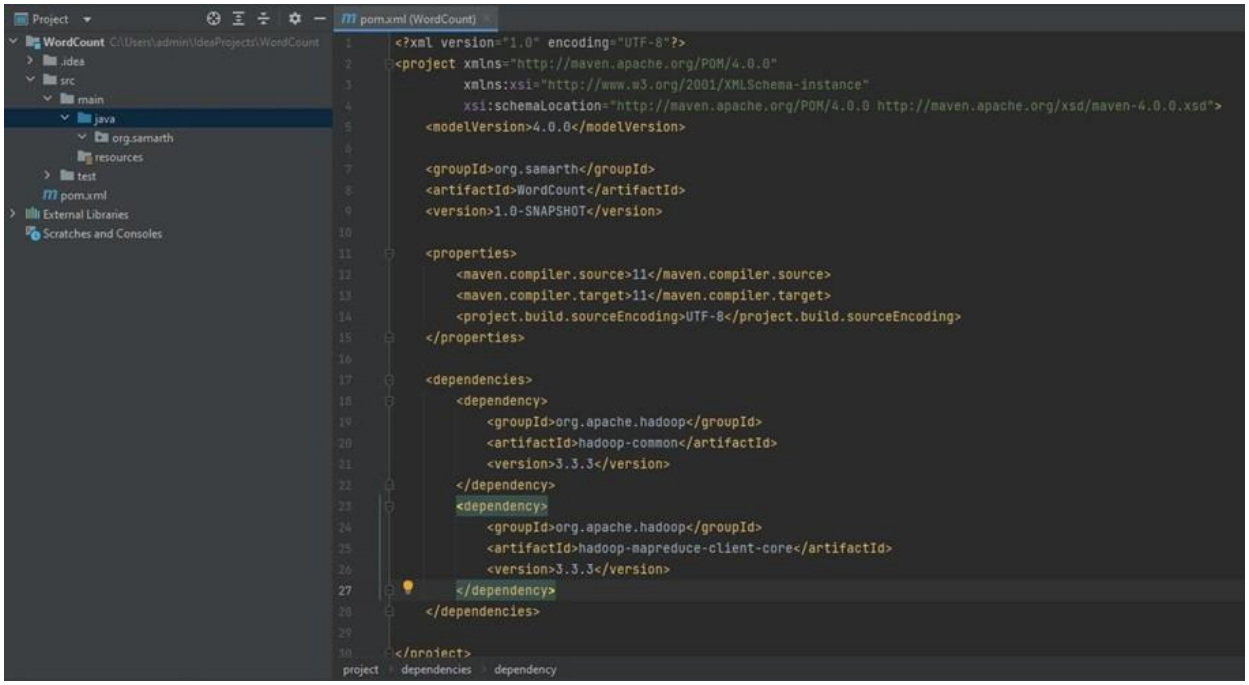
```
org.apache.hadoop.conf.Configuration;
import
org.apache.hadoop.mapreduce.Job;
import
org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import
org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.fs.Path;

public class WordCount
{
public static class Map extends
Mapper<LongWritable,Text,Text,IntWritable> { public void
map(LongWritable key, Text value,Context context) throws
IOException,InterruptedException{
String line = value.toString();
StringTokenizer tokenizer = new
StringTokenizer(line); while
(tokenizer.hasMoreTokens()) {
value.set(tokenizer.nextToken());
context.write(value, new IntWritable(1));
```

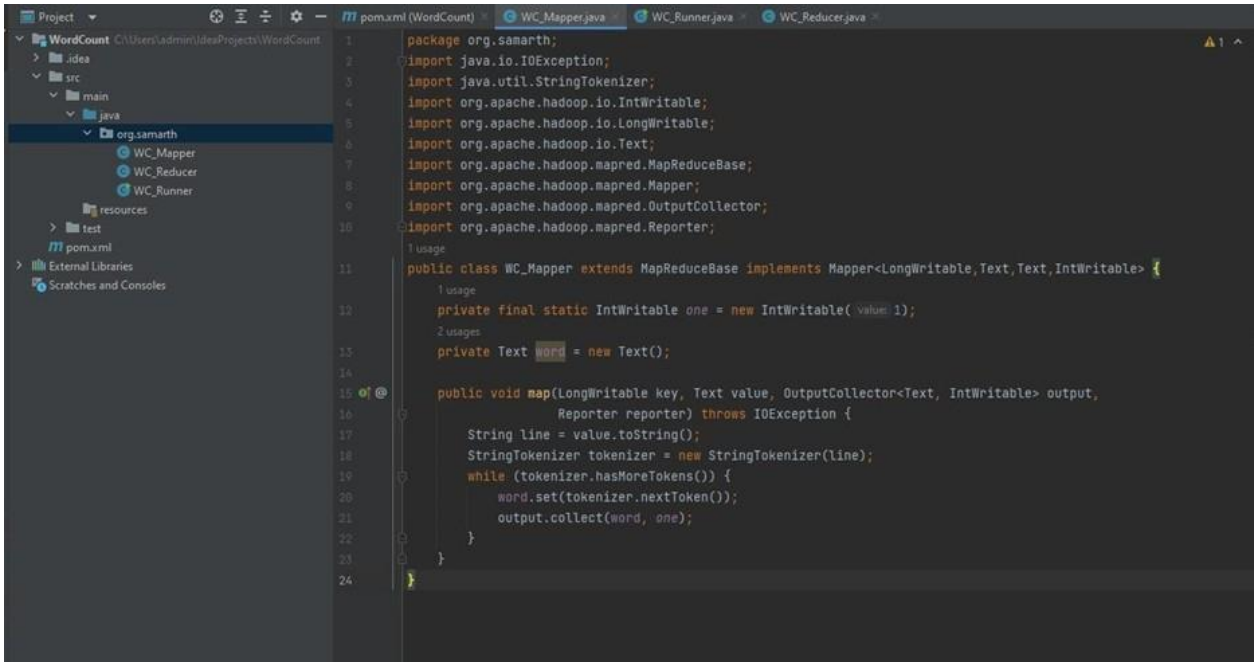
```
}

}

}

public static class Reduce extends

Reducer<Text,IntWritable,Text,IntWritable> { public void reduce(Text

key, Iterable<IntWritable> values,Context context) throws

IOException,InterruptedException {

int sum=0;

for(IntWritable x:

values)

{

sum+=x.get();

}

context.write(key, new IntWritable(sum));

}

}

public static void main(String[] args) throws

Exception { Configuration conf= new

Configuration();

Job job = new Job(conf,"My Word Count

Program"); job.setJarByClass(WordCount.class);

job.setMapperClass(Map.class);
```

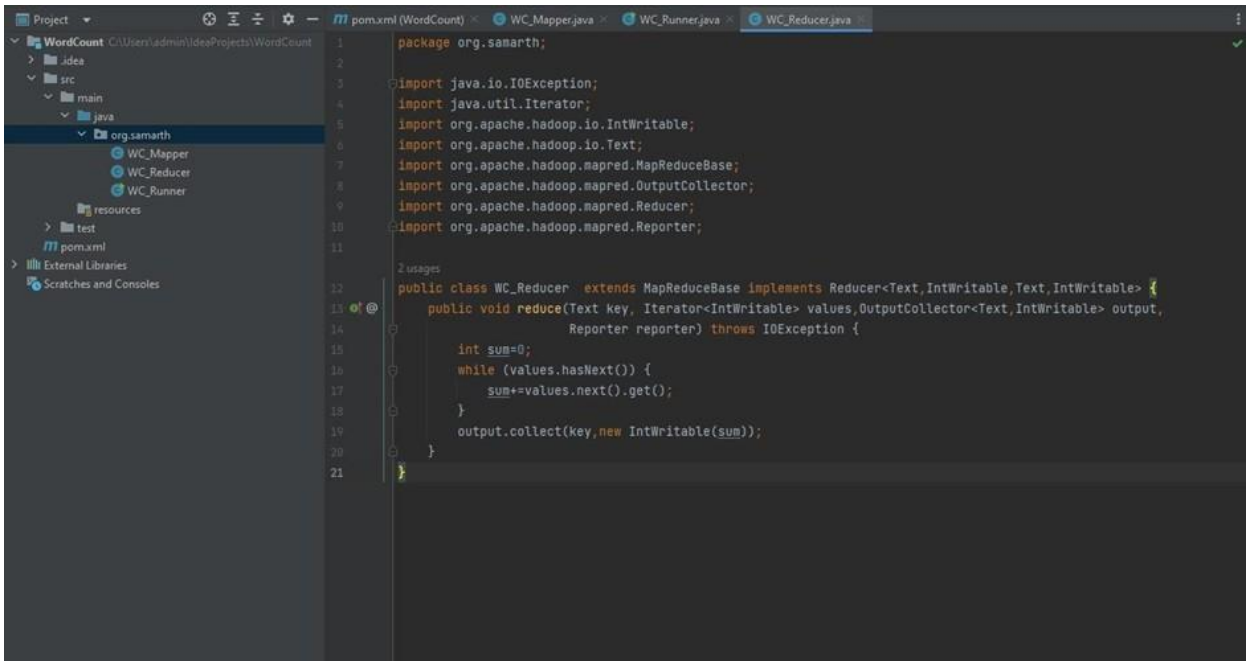
```
job.setReducerClass(Reduce.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.cla
ss); Path outputPath = new Path(args[1]);

//Configuring the input/output path from the filesystem into the job
FileInputFormat.addInputPath(job, new Path(args[0]));

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

//deleting the output path automatically from hdfs so that we don't have
to delete it explicitly
outputPath.getFileSystem(conf).delete(outputPath);

//exiting the job only if the flag value becomes false
System.exit(job.waitForCompletion(true) ? 0 : 1);

}
}
```

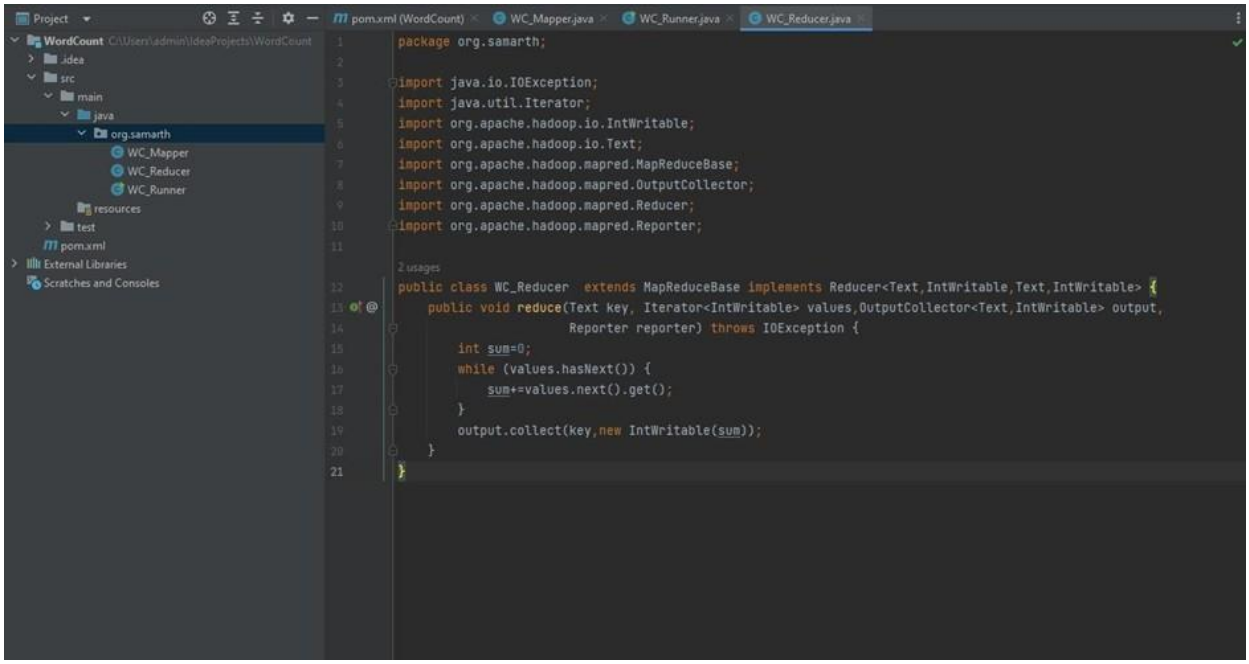
Output:

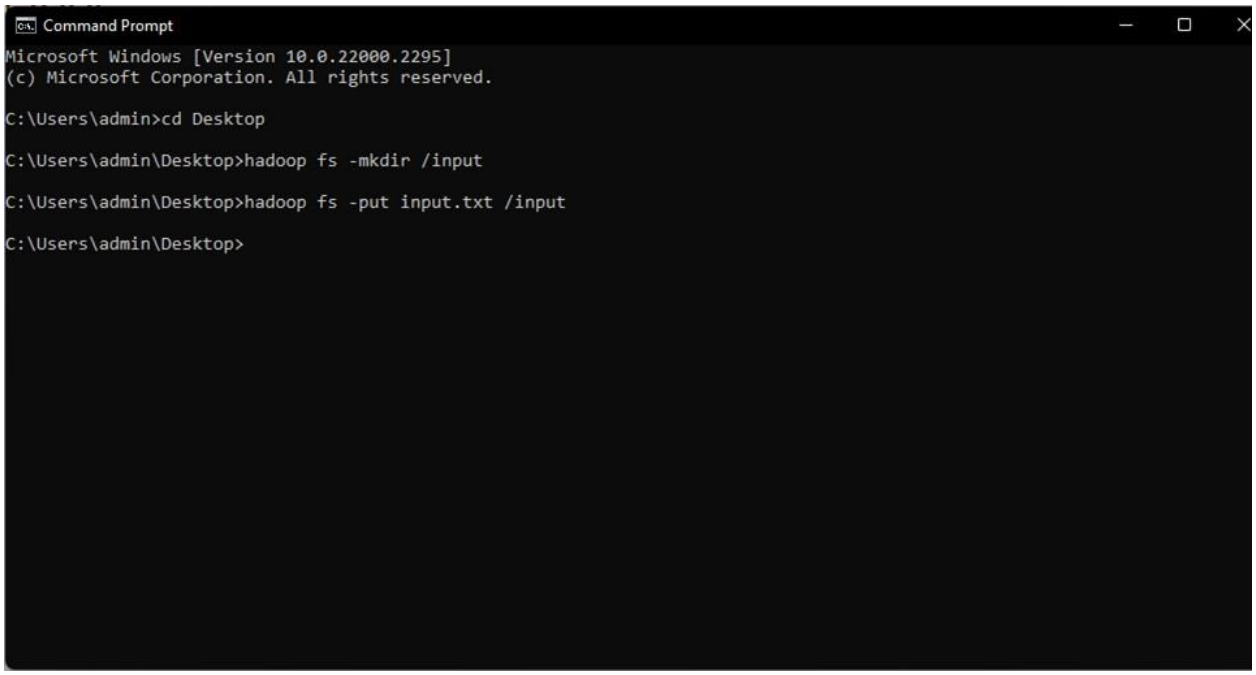

Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Overview 'localhost:9820' (active)
Started: Wed Sep 13 04:30:53 +0530 2023
Version: 3.2.4, r7e5d9983b388e372fe640f21f048f2f2ae6e9eba
Compiled: Tue Jul 12 17:28:00 +0530 2022 by ubuntu from branch-3.2.4
Cluster ID: CID-146566e0-df7a-44ee-a644-d41c94627871
Block Pool ID: BP-1532262397-192.168.12.89-1692767105768
Summary
Security is off.
Safemode is off.
3 files and directories, 1 blocks (1 replicated blocks, 0 erasure coded block groups) = 4 total filesystem object(s).
Heap Memory used 93.19 MB of 204.5 MB Heap Memory. Max Heap Memory is 889 MB.
Non Heap Memory used 51.98 MB of 53.3 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.
Configured Capacity: 417.65 GB
Configured Remote Capacity: 0 B
DFS Used: 345 B (0%)

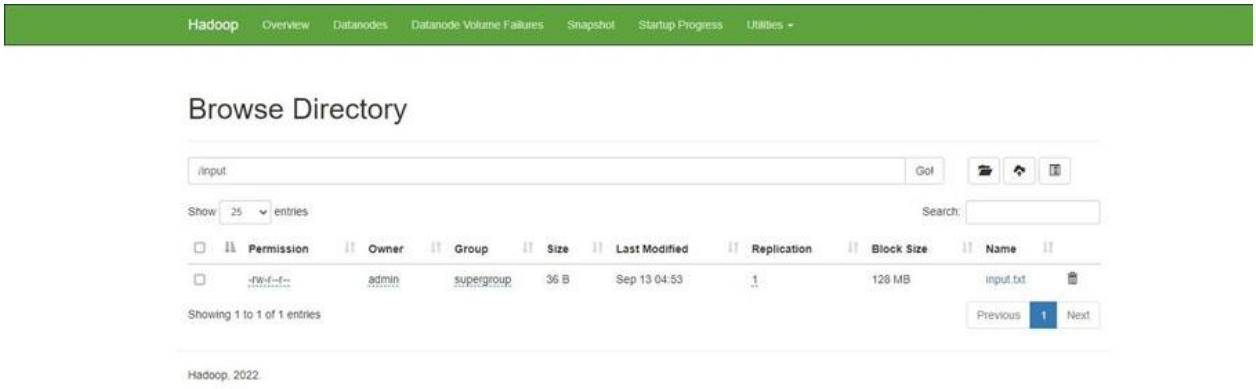

Project
WordCount C:\Users\admin\IdeaProjects\WordCount
.idea
src
main
java
org.samarth
resources
test
pom.xml
External Libraries
Scratches and Consoles
pom.xml (WordCount)
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>org.samarth</groupId>
<artifactId>WordCount</artifactId>
<version>1.0-SNAPSHOT</version>
<properties>
<maven.compiler.source>11</maven.compiler.source>
<maven.compiler.target>11</maven.compiler.target>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-common</artifactId>
<version>3.3.3</version>
</dependency>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-core</artifactId>
<version>3.3.3</version>
</dependency>
</dependencies>
</project>
project
dependencies
dependency

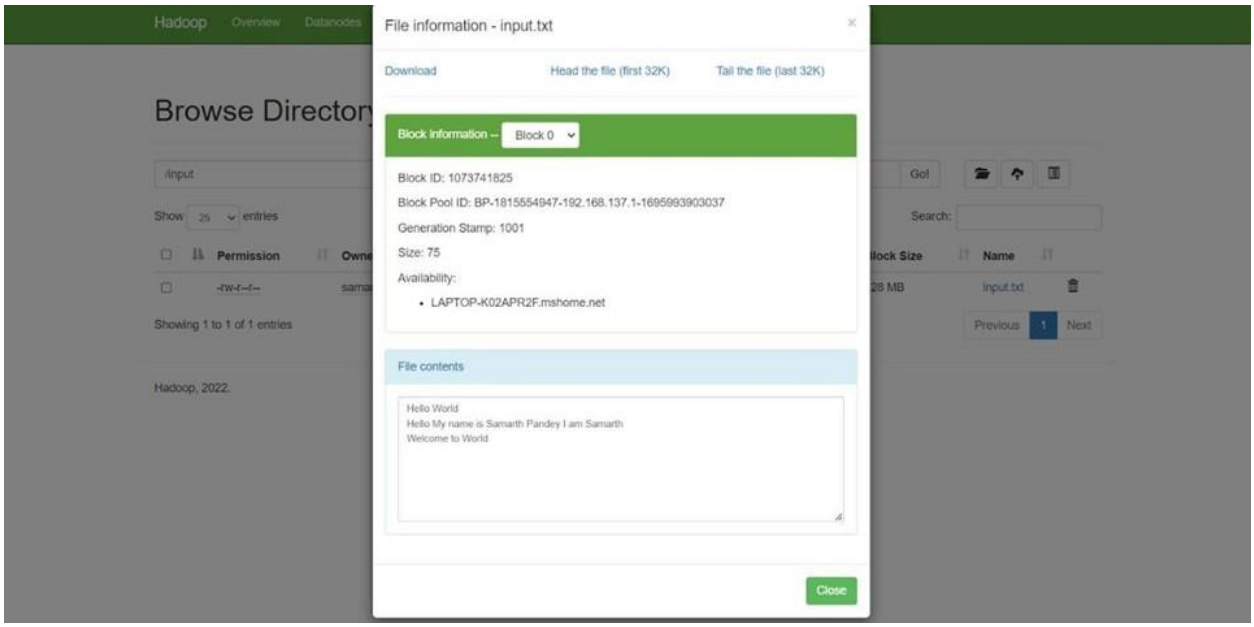

Project
WordCount C:\Users\admin\IdeaProjects\WordCount
.idea
src
main
java
org.samarth
WC_Mapper
WC_Reducer
WC_Runner
resources
test
pom.xml
External Libraries
Scratches and Consoles
pom.xml (WordCount)
WC_Mapper.java
WC_Runner.java
WC_Reducer.java
package org.samarth;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
1 usage
public class WC_Mapper extends MapReduceBase implements Mapper<LongWritable,Text,Text,IntWritable> {
1 usage
private final static IntWritable one = new IntWritable(value: 1);
2 usages
private Text word = new Text();
public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
Reporter reporter) throws IOException {
String line = value.toString();
StringTokenizer tokenizer = new StringTokenizer(line);
while (tokenizer.hasMoreTokens()) {
word.set(tokenizer.nextToken());
output.collect(word, one);
}
}
}


Project
WordCount C:\Users\admin\IdeaProjects\WordCount
.idea
src
main
java
org.samarth
WC_Mapper
WC_Reducer
WC_Runner
resources
test
pom.xml
External Libraries
Scratches and Consoles
pom.xml (WordCount)
WC_Mapper.java
WC_Runner.java
WC_Reducer.java
package org.samarth;
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
2 usages
public class WC_Reducer extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable> {
public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable> output,
Reporter reporter) throws IOException {
int sum=0;
while (values.hasNext()) {
sum+=values.next().get();
}
output.collect(key,new IntWritable(sum));
}
}

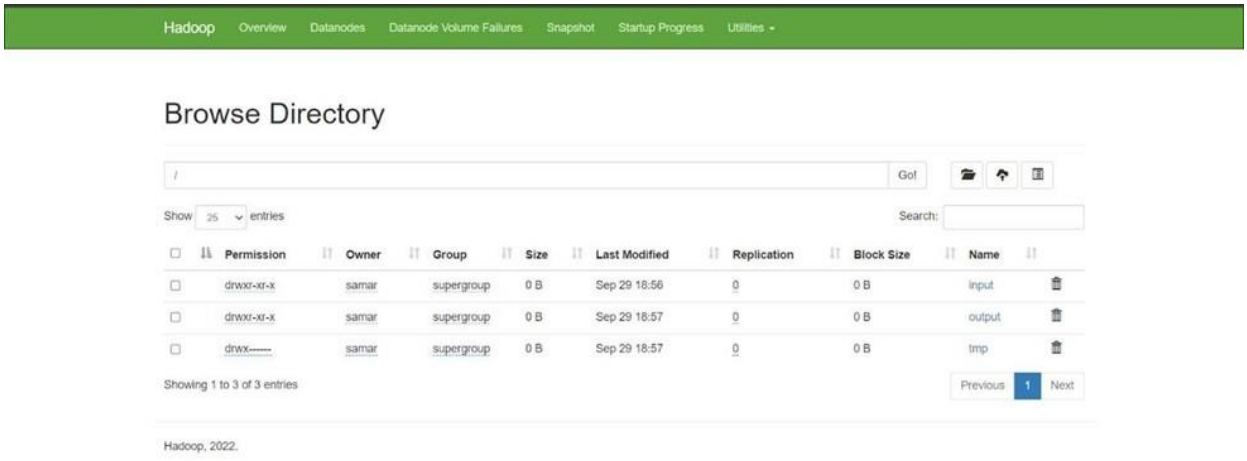

Project
WordCount C:\Users\admin\IdeaProjects\WordCount
.idea
src
main
java
org.samarth
WC_Mapper
WC_Reducer
WC_Runner
resources
test
pom.xml
External Libraries
Scratches and Consoles
pom.xml (WordCount)
WC_Mapper.java
WC_Runner.java
WC_Reducer.java
package org.samarth;
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
2 usages
public class WC_Reducer extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable> {
public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable> output,
Reporter reporter) throws IOException {
int sum=0;
while (values.hasNext()) {
sum+=values.next().get();
}
output.collect(key,new IntWritable(sum));
}
}

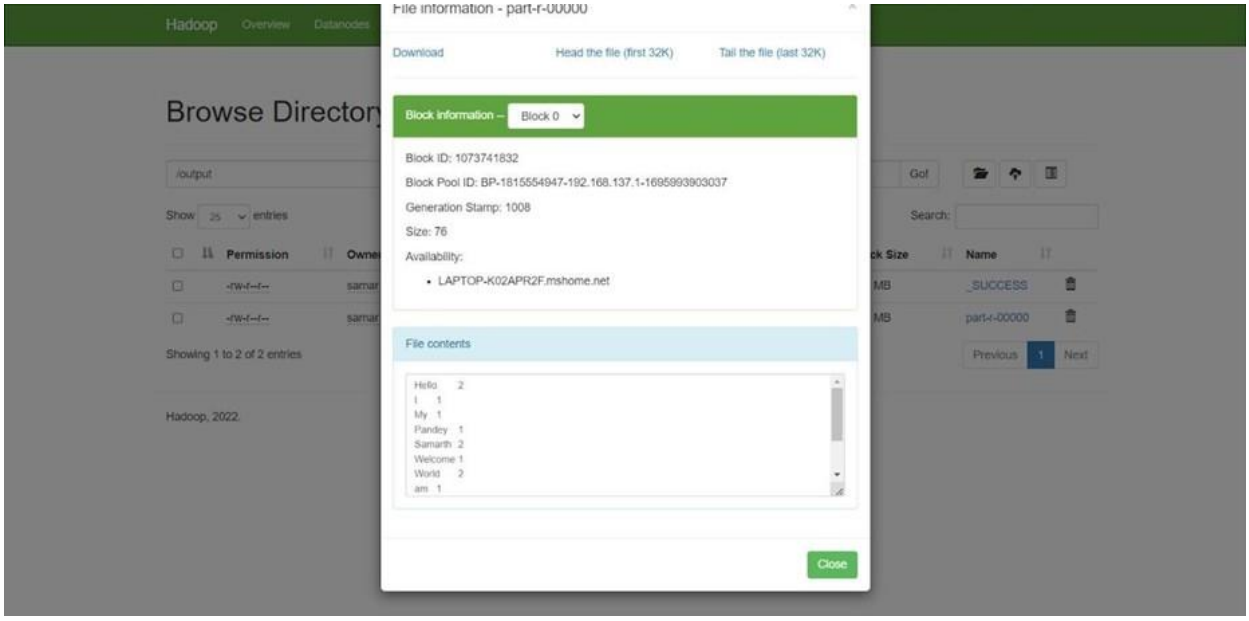

Command Prompt
Microsoft Windows [Version 10.0.22000.2295]
(c) Microsoft Corporation. All rights reserved.
C:\Users\admin>cd Desktop
C:\Users\admin\Desktop>hadoop fs -mkdir /input
C:\Users\admin\Desktop>hadoop fs -put input.txt /input
C:\Users\admin\Desktop>


Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Browse Directory
/input
Go!
Show 25 entries
Search:
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
-rw-r--r--
admin
supergroup
36 B
Sep 13 04:53
1
128 MB
input.txt
Showing 1 to 1 of 1 entries
Previous
1
Next
Hadoop, 2022.


File information - input.txt
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information -- Block 0
Block ID: 1073741825
Block Pool ID: BP-1815554947-192.168.137.1-1695993903037
Generation Stamp: 1001
Size: 75
Availability:
LAPTOP-K02APR2F.mshome.net
File contents
Hello World
Hello My name is Samarth Pandey I am Samarth
Welcome to World
Close

```
Command Prompt
**********************************************************/

C:\Users\samar\Desktop>hadoop fs -mkdir /input

C:\Users\samar\Desktop>hadoop fs -put input.txt /input

C:\Users\samar\Desktop>hadoop jar C:\Users\samar\IdeaProjects\WordCount\target\hadoop-mapreduce--3.2.4.jar wordcount /input/input.txt /output
2023-09-29 18:57:08,319 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2023-09-29 18:57:09,763 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/samar/.staging/job_1695993949979_0001
2023-09-29 18:57:10,326 INFO input.FileInputFormat: Total input files to process : 1
2023-09-29 18:57:10,697 INFO mapreduce.JobSubmitter: number of splits:1
2023-09-29 18:57:11,007 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1695993949979_0001
2023-09-29 18:57:11,010 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-09-29 18:57:11,299 INFO conf.Configuration: resource-types.xml not found
2023-09-29 18:57:11,300 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2023-09-29 18:57:11,723 INFO impl.YarnClientImpl: Submitted application application_1695993949979_0001
2023-09-29 18:57:11,814 INFO mapreduce.Job: The url to track the job: http://LAPTOP-K02APR2F:8088/proxy/application_1695993949979_0001/
2023-09-29 18:57:11,816 INFO mapreduce.Job: Running job: job_1695993949979_0001
2023-09-29 18:57:27,135 INFO mapreduce.Job: Job job_1695993949979_0001 running in uber mode : false
2023-09-29 18:57:27,136 INFO mapreduce.Job:  map 0% reduce 0%
2023-09-29 18:57:35,308 INFO mapreduce.Job:  map 100% reduce 0%
2023-09-29 18:57:43,413 INFO mapreduce.Job:  map 100% reduce 100%
2023-09-29 18:57:44,434 INFO mapreduce.Job: Job job_1695993949979_0001 completed successfully
2023-09-29 18:57:45,177 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=126
                FILE: Number of bytes written=478089
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=177
                HDFS: Number of bytes written=76
                HDFS: Number of read operations=8
                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=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=5488
                Total time spent by all reduces in occupied slots (ms)=5838
                Total time spent by all map tasks (ms)=5488
                Total time spent by all reduce tasks (ms)=5838
                Total vcore-milliseconds taken by all map tasks=5488
                Total vcore-milliseconds taken by all reduce tasks=5838
                Total megabyte-milliseconds taken by all map tasks=5619712
                Total megabyte-milliseconds taken by all reduce tasks=5978112
        Map-Reduce Framework
                Map input records=3
```





Conclusion:

This exercise demonstrated a basic illustration of Hadoop's data processing capabilities by developing a Word Count program utilizing the MapReduce paradigm. The Mapper, Reducer, and Driver are the three main parts of the Word Count application. After processing input data, the Mapper separates it into words and outputs <word, 1> pairs. To get the final count for each word, the Reducer adds up and totals these pairings. Through the configuration of input/output pathways and task execution, the Driver manages the MapReduce job.