



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

---

Experiment No.4
Experiment on Hadoop Map-Reduce
Date of Performance:
Date of Submission:



**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.class);  
    Path outputPath = new Path(args[1]);  
  
    //Configuring the input/output path from the filesystem into the job  
    FileInputFormat.addInputPath(job, new Path(args[0]));  
    CSL702: Big Data Analytics Lab
```



```
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:**

The screenshot shows the Hadoop NameNode Overview page for 'localhost:9820' (active). The page has a green header with navigation tabs: Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The 'Overview' tab is selected.

#### Overview 'localhost:9820' (active)

Started:	Wed Sep 13 04:30:53 +0530 2023
Version:	3.2.4, r7e5d99830388e372fe640f21f048f22ae6e9eba
Compiled:	Tue Jul 12 17:28:00 +0530 2022 by ubuntu from branch-3.2.4
Cluster ID:	CID-146566e0-df7a-44ee-a644-041c94627871
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 Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	417.65 GB
Configured Remote Capacity:	0 B
DFS Used:	345 B (0%)



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

```
<?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>
```

```
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;

public class WC_Mapper extends MapReduceBase implements Mapper<LongWritable,Text,Text,IntWritable> {
  private final static IntWritable one = new IntWritable(1);
  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);
    }
  }
}
```



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

```
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;

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));
    }
}
```

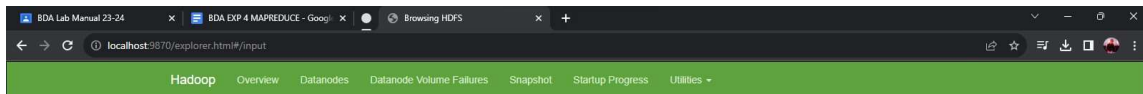
```
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;

public class WC_Runner {
    public static void main(String[] args) throws IOException{
        JobConf conf = new JobConf(WC_Runner.class);
        conf.setJobName("WordCount");
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(WC_Mapper.class);
        conf.setCombinerClass(WC_Reducer.class);
        conf.setReducerClass(WC_Reducer.class);
        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));
        JobClient.runJob(conf);
    }
}
```



```
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>
```



### Browse Directory

/input

Go!

Show

25

entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<input type="checkbox"/>	-rw-r--r--	admin	supergroup	36 B	Sep 13 04:53	1	128 MB	input.txt

Showing 1 to 1 of 1 entries

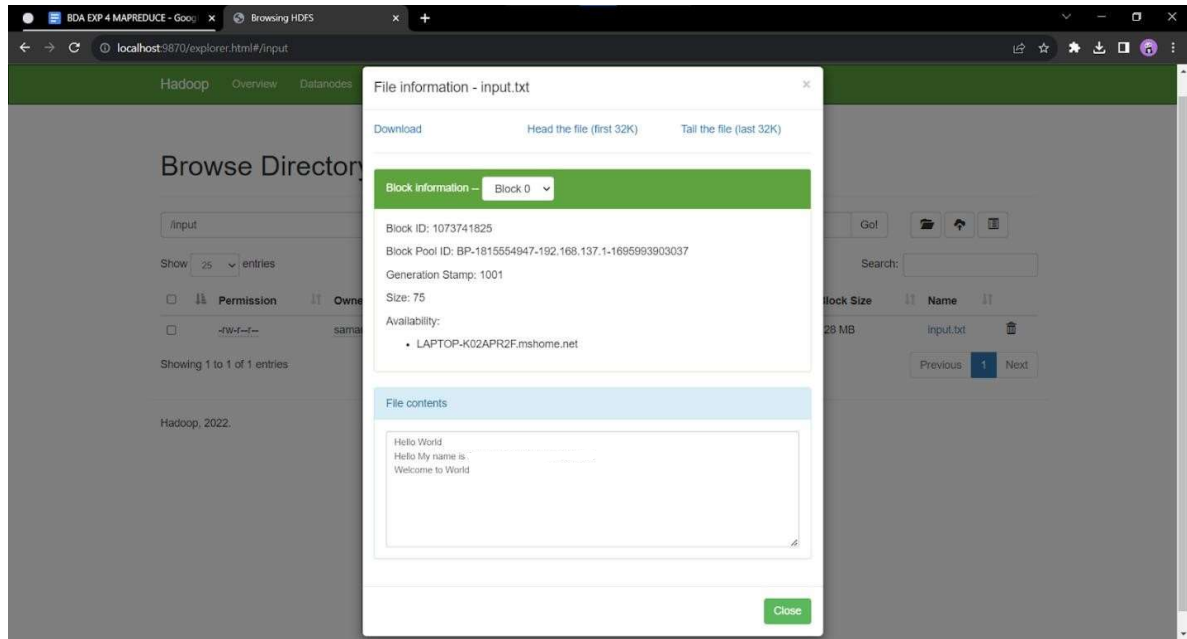
Previous

1

Next

Hadoop, 2022.



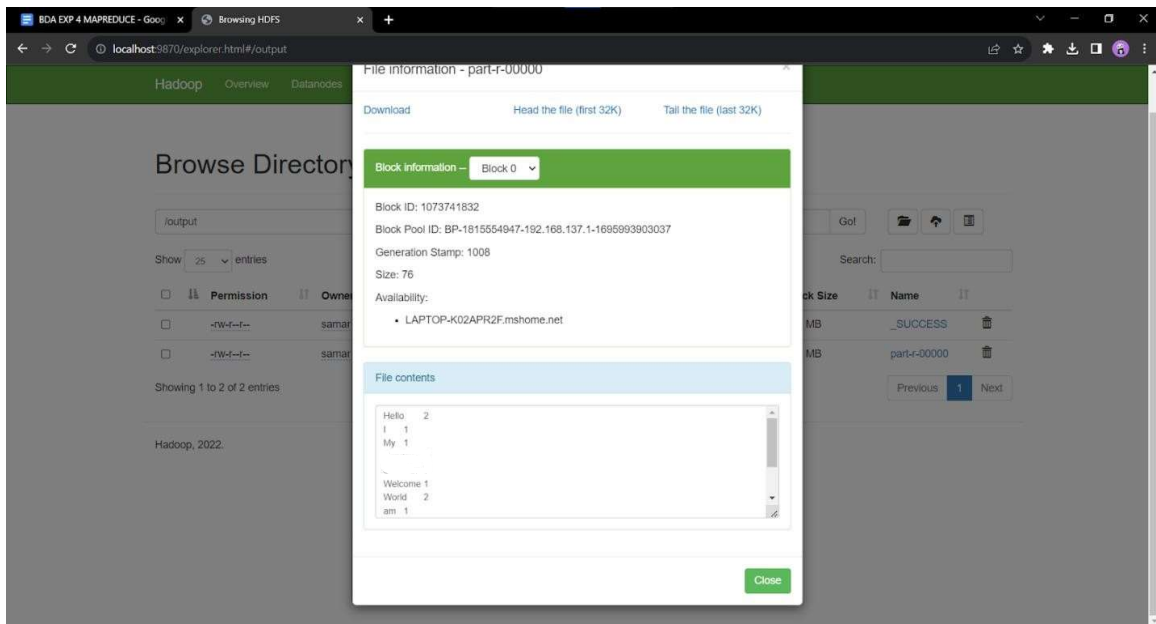
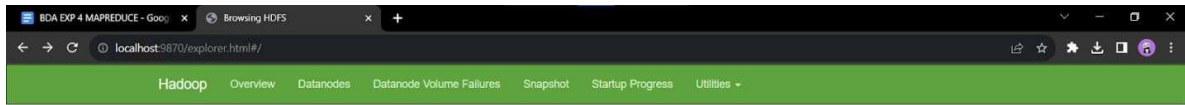


```
Command Prompt
C:\Users\samar\Desktop\hadoop fs -mkdir /input
C:\Users\samar\Desktop\hadoop fs -put input.txt /input
C:\Users\samar\Desktop\hadoop jar C:\Users\...IdeaProjects\WordCount\target\hadoop-mapreduce-3.2.4.jar wordcount /input/input.txt /output
2023-09-29 18:57:08,119 INFO client.RMProxy: 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
```



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering





---

### **CONCLUSION:**

Implementing a word count program using MapReduce showcases the power of distributed computing in handling large-scale data processing tasks.

This program is a fundamental example of how MapReduce, a core component of the Hadoop ecosystem, can efficiently analyze vast datasets by distributing workloads across multiple nodes.

The concept of Mapper and Reducer functions simplifies complex data analysis, making it accessible for developers to extract meaningful insights from extensive textual data.

By dividing the problem into smaller, parallelizable tasks, MapReduce enables high-speed, fault-tolerant data processing, making it an indispensable tool for big data applications and a testament to the capabilities of distributed computing paradigms.