Ex. No: 6 Date:	Word Count Program using MapReduce

AIM

To count the number of words using JAVA for demonstrating the use of Map and Reduce tasks.

DESCRIPTION

MapReduce is a programming model for writing applications that can process Big Data in parallel on multiple nodes. MapReduce provides analytical capabilities for analysing huge volumes of complex data.

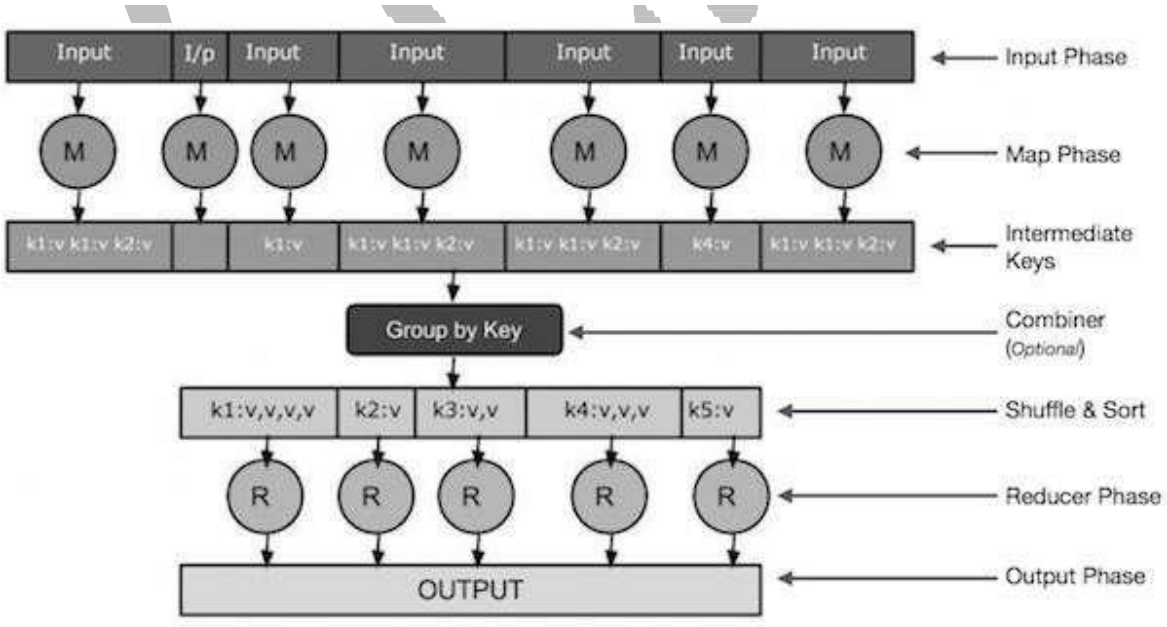
How MapReduce Works?

The MapReduce algorithm contains two important tasks, namely Map and Reduce.

- The Map task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key-value pairs).
- The Reduce task takes the output from the Map as an input and combines those data tuples (key-value pairs) into a smaller set of tuples.

The reduce task is always performed after the map job.

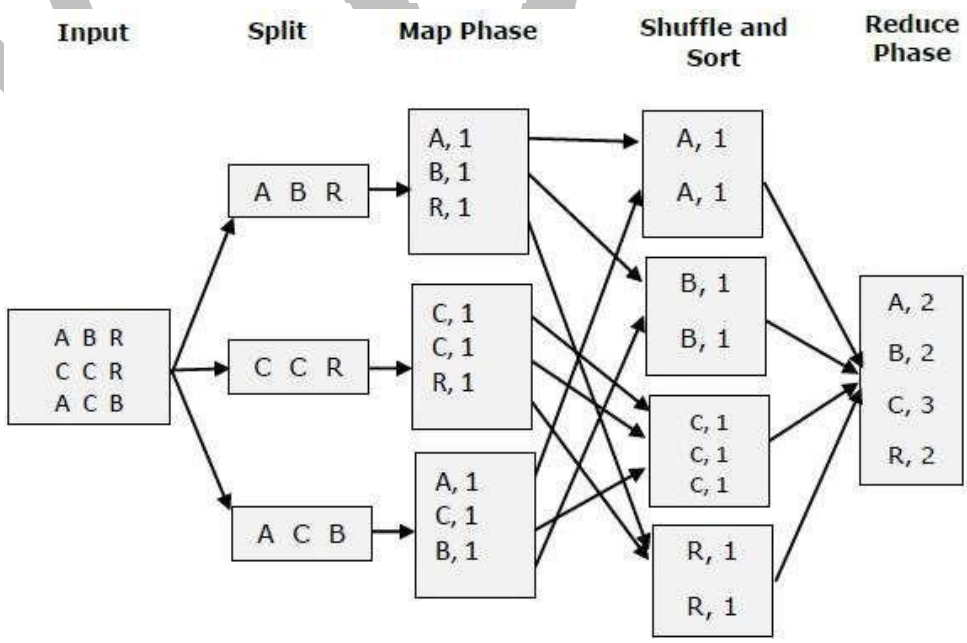
Let us now take a close look at each of the phases and try to understand their significance.



- **Input Phase:** A Record Reader that translates each record in an input file and sends the parsed data to the mapper in the form of key-value pairs.

- **Map:** Map is a user-defined function, which takes a series of key-value pairs and processes each one of them to generate zero or more key-value pairs.
- **Intermediate Keys:** They key-value pairs generated by the mapper are known as intermediate keys.
- **Combiner:** A combiner is a type of local Reducer that groups similar data from the map phase into identifiable sets. It takes the intermediate keys from the mapper as input and applies a user-defined code to aggregate the values in a small scope of one mapper. It is not a part of the main MapReduce algorithm; it is optional.
- **Shuffle and Sort:** The Reducer task starts with the Shuffle and Sort step. It downloads the grouped key-value pairs onto the local machine, where the Reducer is running. The individual key-value pairs are sorted by key into a larger data list. The data list groups the equivalent keys together so that their values can be iterated easily in the Reducer task.
- **Reducer:** The Reducer takes the grouped key-value paired data as input and runs a Reducer function on each one of them. Here, the data can be aggregated, filtered, and combined in a number of ways, and it requires a wide range of processing. Once the execution is over, it gives zero or more key-value pairs to the final step.
- **Output Phase:** In the output phase, an output formatter that translates the final key-value pairs from the Reducer function and writes them onto a file using a record writer.

Example



PROGRAM

WordCount.java

```
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();
```

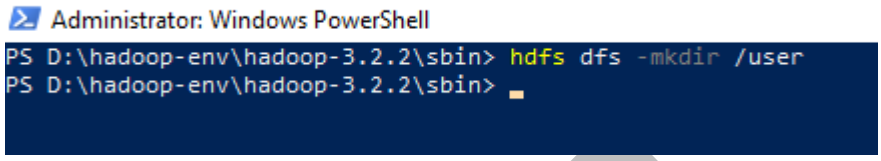
```
  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
 }

 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);
 }
}
```

PROCEDURE STEPS

Step 1: Create a text file **"D:/data.txt"**

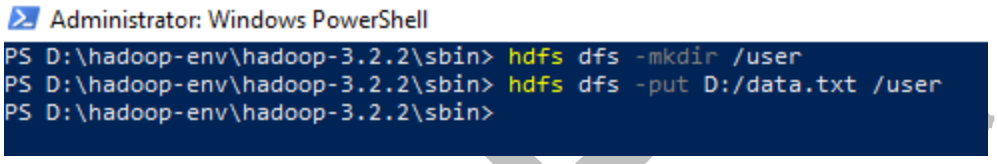
Step 2: Create a directory in HDFS, where to kept text file.

hdfs dfs -mkdir /user



Step 3: Upload the data.txt file on HDFS in the specific directory

hdfs dfs -put D:/data.txt /user

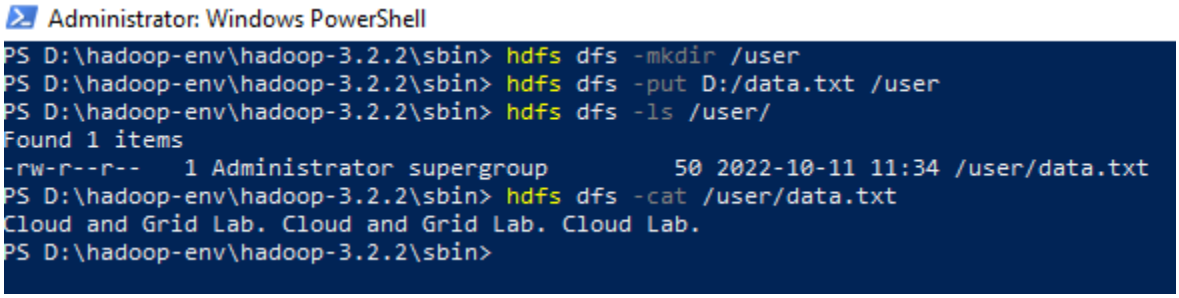


Step 4: List the files or directories in hdfs

hdfs dfs -ls /user/

Step 5: To view the content of the file **"/user/data.txt"**

hdfs dfs -cat /user/data.txt



Step 6: Run the jar file

hadoop jar D:/hadoop-env/hadoop-3.2.2/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar wordcount /user /out

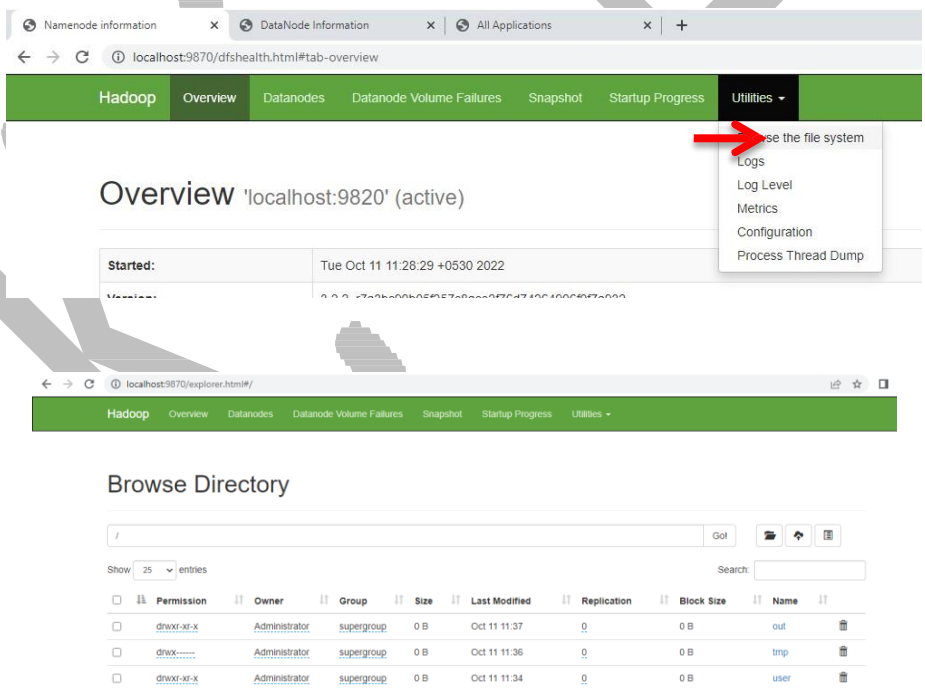
```
Administrator: Windows PowerShell
PS D:\hadoop-env\hadoop-3.2.2\sbin> hdfs dfs -mkdir /user
PS D:\hadoop-env\hadoop-3.2.2\sbin> hdfs dfs -put D:/data.txt /user
PS D:\hadoop-env\hadoop-3.2.2\sbin> hdfs dfs -ls /user/
Found 1 items
-rw-r--r--   1 Administrator supergroup         50 2022-10-11 11:34 /user/data.txt
PS D:\hadoop-env\hadoop-3.2.2\sbin> hdfs dfs -cat /user/data.txt
Cloud and Grid Lab. Cloud and Grid Lab. Cloud Lab.
PS D:\hadoop-env\hadoop-3.2.2\sbin> hadoop jar D:/hadoop-env/hadoop-3.2.2/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar wordcount /user /out
2022-10-11 11:36:37,212 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2022-10-11 11:36:38,133 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Administrator/.staging/job_1665467943708_0001
2022-10-11 11:36:38,559 INFO input.FileInputFormat: Total input files to process : 1
2022-10-11 11:36:38,864 INFO mapreduce.JobSubmitter: number of splits:1
2022-10-11 11:36:39,157 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1665467943708_0001
2022-10-11 11:36:39,159 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-10-11 11:36:39,483 INFO conf.Configuration: resource-types.xml not found
2022-10-11 11:36:39,484 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-10-11 11:36:40,208 INFO impl.YarnClientImpl: Submitted application application_1665467943708_0001
2022-10-11 11:36:40,320 INFO mapreduce.Job: The url to track the job: http://a4it55:8088/proxy/application_1665467943708_0001/
2022-10-11 11:36:40,322 INFO mapreduce.Job: Running job: job_1665467943708_0001
2022-10-11 11:36:54,677 INFO mapreduce.Job: Job job_1665467943708_0001 running in uber mode : false
2022-10-11 11:36:54,680 INFO mapreduce.Job:  map 0% reduce 0%
2022-10-11 11:37:00,861 INFO mapreduce.Job:  map 100% reduce 0%
2022-10-11 11:37:07,981 INFO mapreduce.Job:  map 100% reduce 100%
2022-10-11 11:37:09,016 INFO mapreduce.Job: Job job_1665467943708_0001 completed successfully
2022-10-11 11:37:09,239 INFO mapreduce.Job: Counters: 54
```

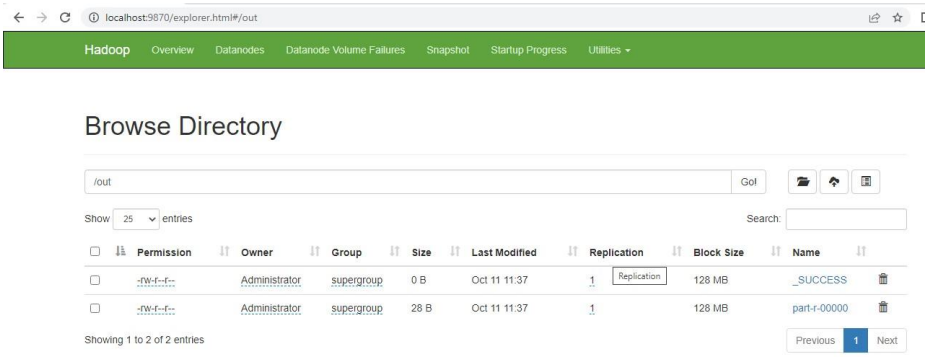
Step 7: To view the output in **"/out/*"**

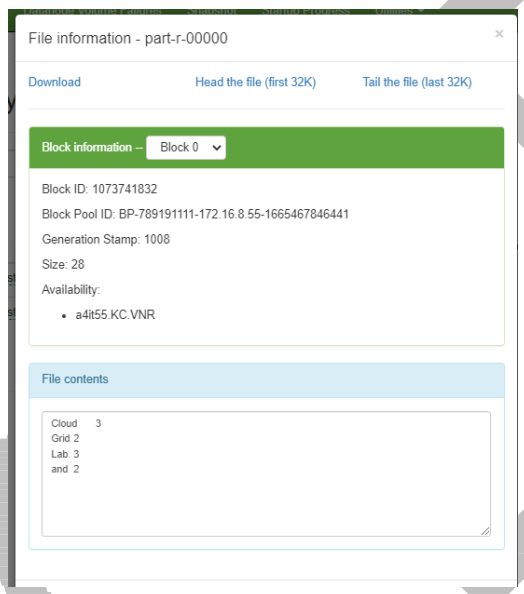
hadoop fs -cat /out/*

```
                Bytes Written=28
PS D:\hadoop-env\hadoop-3.2.2\sbin> hadoop fs -cat /out/*
Cloud   3
Grid    2
Lab.    3
and     2
PS D:\hadoop-env\hadoop-3.2.2\sbin>
```

Output







VIVA QUESTIONS

1. What is Hadoop MapReduce?

2. What are the operations performed in MapReduce for computation process?

3. State the benefits of MapReduce.