

## PRACTICAL - 04

**AIM : Run a java program based on parallel programming to implement the concept of Map Reduce.**

**[ Hadoop WordCount Execution Steps (Using PuTTY) ]**

### PROCEDURE :

#### Step 1: Prepare Input Data

```
echo "Hadoop is big data Hadoop is Java" > sample.txt
hdfs dfs -mkdir /input
hdfs dfs -put sample.txt /input/
hdfs dfs -ls /input
```

```
[maria_dev@sandbox-hdp ~]$ echo "Hadoop is big data Hadoop is Java" > sample.txt
[maria_dev@sandbox-hdp ~]$ hdfs dfs -mkdir /input
[maria_dev@sandbox-hdp ~]$ hdfs dfs -put sample.txt /input/
[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls /input
Found 1 items
-rw-r--r--  1 maria_dev hdfs          34 2025-09-01 09:43 /input/sample.txt
[maria_dev@sandbox-hdp ~]$
```

#### Step 2: Create Java Files :

Use commands :

1. vi WordMapper.java
2. vi WordReducer.java
3. vi WordCountDriver.java

#### WordMapper.java

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

public class WordMapper extends 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, Context context) throws IOException,
    InterruptedException {
        String line = value.toString();
        for (String token : line.split("\\s+")) { // Corrected escape sequence
            word.set(token);
            context.write(word, one);
        }
    }
}
```

**WordReducer.java**

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class WordReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
}
```

**WordCountDriver.java**

```
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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCountDriver {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: WordCountDriver <input path> <output path>");
            System.exit(-1);
        }
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCountDriver.class);
        job.setMapperClass(WordMapper.class);
        job.setReducerClass(WordReducer.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);
    }
}
```

```
[maria_dev@sandbox-hdp ~]$ vi WordMapper.java
[maria_dev@sandbox-hdp ~]$ vi WordReducer.java
[maria_dev@sandbox-hdp ~]$ vi WordCountDriver.java
```

### Step 3: Compile Java Files

```
mkdir wordcount_classes
```

```
hadoop com.sun.tools.javac.Main -d wordcount_classes WordMapper.java
```

```
WordReducer.java WordCountDriver.java
```

(If error, use:)

```
javac -cp `hadoop classpath` -d wordcount_classes WordMapper.java WordReducer.java  
WordCountDriver.java
```

```
[maria_dev@sandbox-hdp ~]$ mkdir wordcount_classes  
[maria_dev@sandbox-hdp ~]$ javac -cp `hadoop classpath` -d wordcount_classes WordMapper.java WordReducer.java WordCountDriver.java
```

### Step 4: Create JAR

```
jar -cvf wordcount.jar -C wordcount_classes/ .
```

```
[maria_dev@sandbox-hdp ~]$ jar -cvf wordcount.jar -C wordcount_classes/ .  
added manifest  
adding: WordMapper.class(in = 1867) (out= 776) (deflated 58%)  
adding: WordReducer.class(in = 1592) (out= 665) (deflated 58%)  
adding: WordCountDriver.class(in = 1364) (out= 751) (deflated 44%)
```

### Step 5: Run MapReduce Job

```
hdfs dfs -rm -r /output
```

```
hadoop jar wordcount.jar WordCountDriver /input /output
```

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -rm -r /output  
rm: `/output': No such file or directory  
[maria_dev@sandbox-hdp ~]$ hadoop jar wordcount.jar WordCountDriver /input /output  
25/09/01 09:47:34 INFO client.RMPProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032  
25/09/01 09:47:34 INFO client.AHSPProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
```

### Step 6: View Output

```
hdfs dfs -ls /output
```

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls /output  
Found 2 items  
-rw-r--r-- 1 maria_dev hdfs 0 2025-09-01 09:48 /output/_SUCCESS  
-rw-r--r-- 1 maria_dev hdfs 34 2025-09-01 09:48 /output/part-r-00000
```

hdfs dfs -cat /output/part-r-00000

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -cat /output/part-r-00000
Hadoop 2
Java 1
big 1
data 1
is 2
[maria_dev@sandbox-hdp ~]$
```