PRACTICAL - 04

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

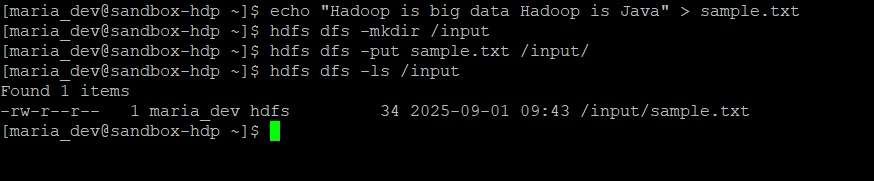
**[ Hadoop WordCount Execution Steps (Using PuTTY) ]**

**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



# 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);

}

}



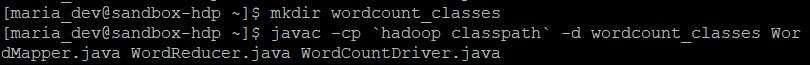
# Step 3: Compile Java Files

mkdir wordcount\_classes

hadoop com.sun.tools.javac.Main -d wordcount\_classes WordMapper.java WordReducer.java WordCountDriver.java

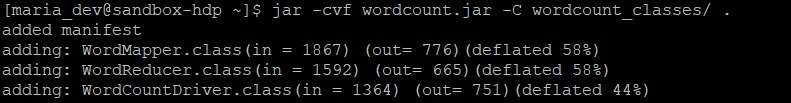
(If error, use:)

javac -cp `hadoop classpath` -d wordcount\_classes WordMapper.java WordReducer.java WordCountDriver.java



# Step 4: Create JAR

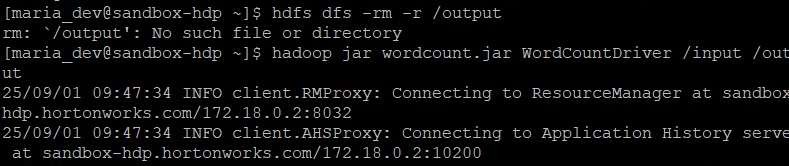
jar -cvf wordcount.jar -C wordcount\_classes/ .



# Step 5: Run MapReduce Job

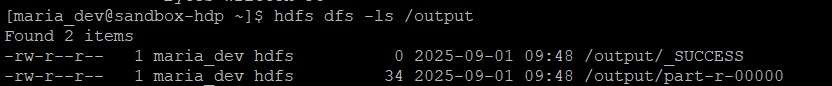
hdfs dfs -rm -r /output

hadoop jar wordcount.jar WordCountDriver /input /output



# Step 6: View Output

hdfs dfs -ls /output



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

