WordCount.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.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

import java.util.StringTokenizer;

public class WordCount {

// Mapper Class

public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

// The map method is called once for each line of text input.

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

// Tokenizing the line into words using a StringTokenizer

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

context.write(word, one); // Emit the word with a count of 1

}

}

}

// Reducer Class

public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

private IntWritable result = new IntWritable();

// The reduce method is called for each unique key.

public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get(); // Summing the counts

}

result.set(sum);

context.write(key, result); // Emit the word and its total count

}

}

public static void main(String[] args) throws Exception {

// Configuration and Job setup

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(WordCount.class);

// Setting Mapper and Reducer classes

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class); // Optional combiner to optimize network traffic

job.setReducerClass(IntSumReducer.class);

// Setting output key and value classes

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

// Setting input and output file paths

FileInputFormat.addInputPath(job, new Path(args[0])); // Input path

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

// Submitting the job and waiting for completion

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

Jar

javac -classpath $(hadoop classpath) -d wordcount\_classes WordCount.java

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

run

hadoop jar wordcount.jar WordCount input.txt output\_dir

Input File Example (input.txt)

Hello World

Hello Hadoop

Hello World Hadoop

Expected Output (part-r-00000)

Hadoop 2

Hello 3

World 2