Name: Adamya gowri H K

USN: 1NT19IS005

C1 BATCH

Exercise-3: MAPREDUCE

Use the Hadoop framework to write a custom MapReduce program to perform word count operation on a custom data set.

Initially create a new project, package and class in eclipse to run a java code.

To install jar files:

Right click on project (Mapreduce)

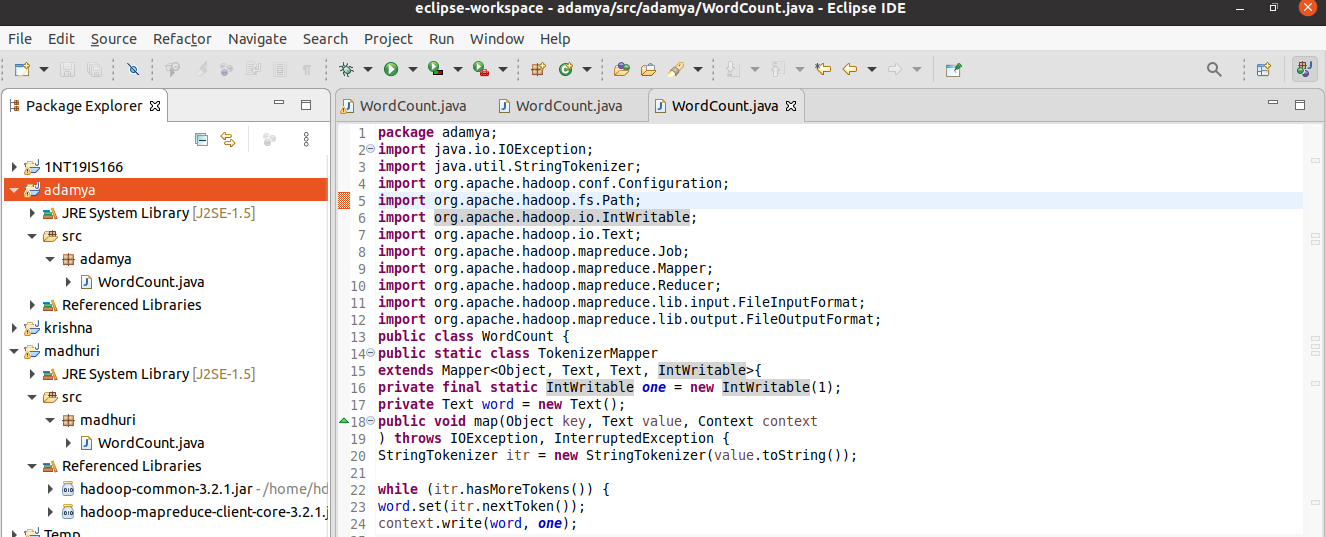
Click on -> build a path -> add external archives -> Hadoop 3.2.1 -> share

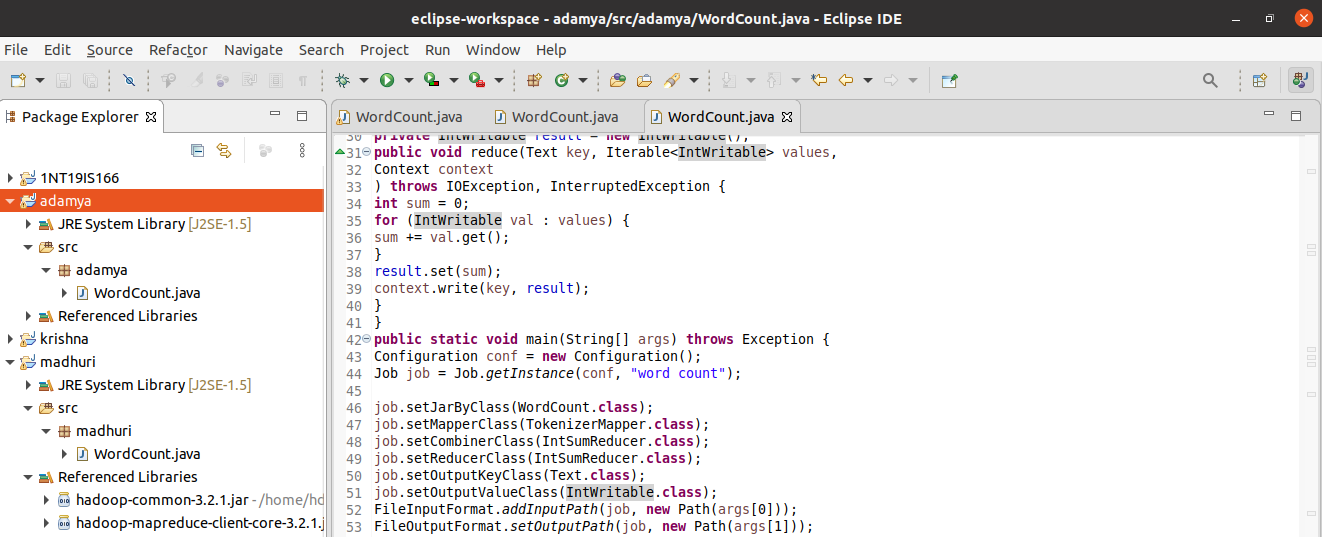
In share 1. Click on common ->open hadoop-common-3.2.1.jar

2. Click on mapreduce -> open hadoop-mapreduce-client-core-3.2.1.jar

Right click on project -> export -> java -> jar file -> next

Browse the address of the java file and save it in desktop/document/downloads and name it (adamya.jar)





In Terminal:

cd $HADOOP\_HOME

cd sbin

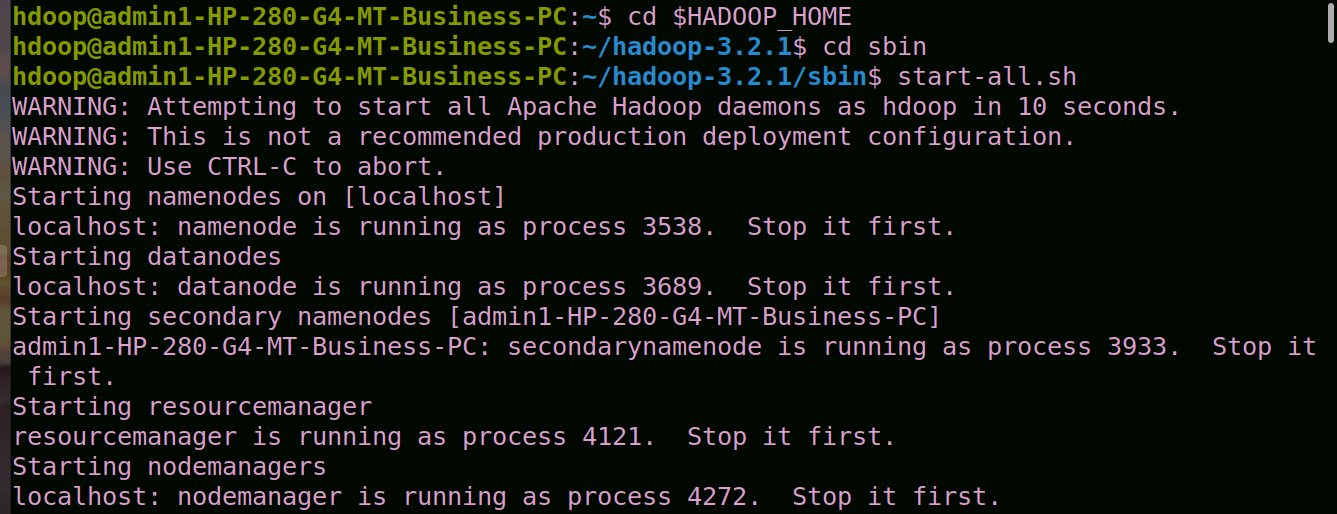
jps

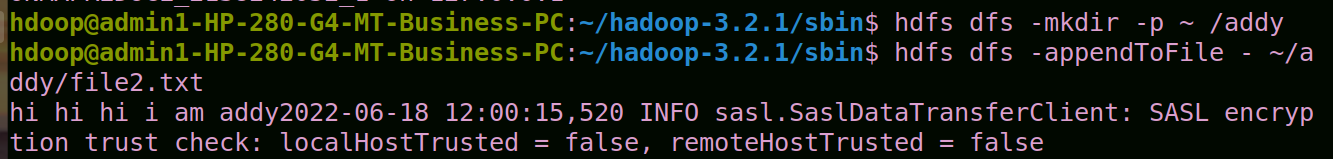
Start-all.sh

hdfs dfs -mkdir -p ~ /input

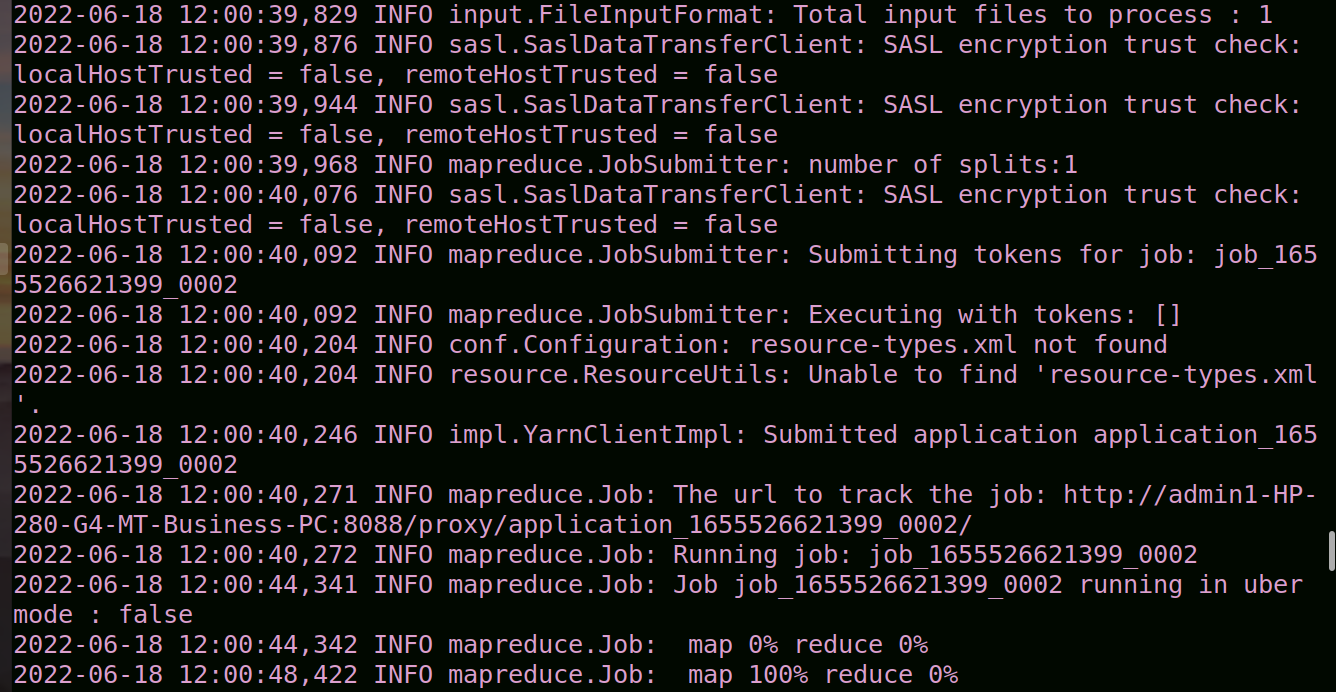
hdfs dfs -appendToFile - ~/input/text.txt

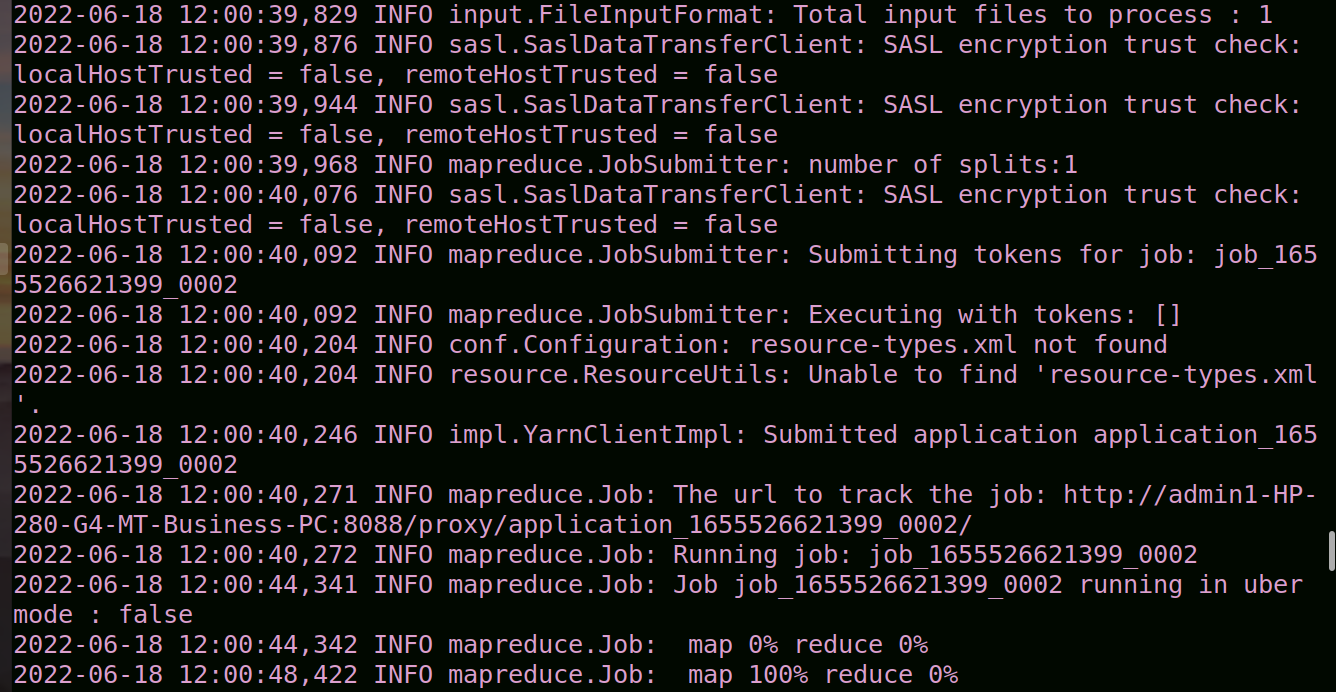
Create a file and add content to it. ->(ctrl D two times)

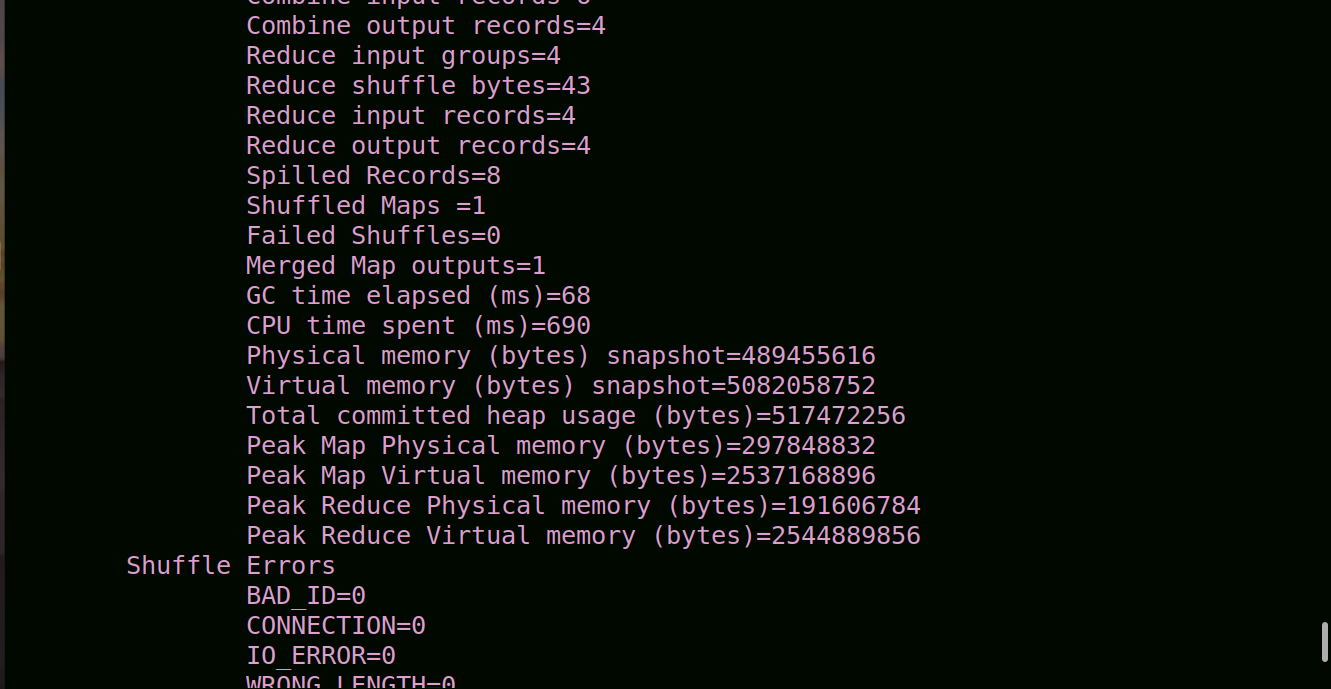


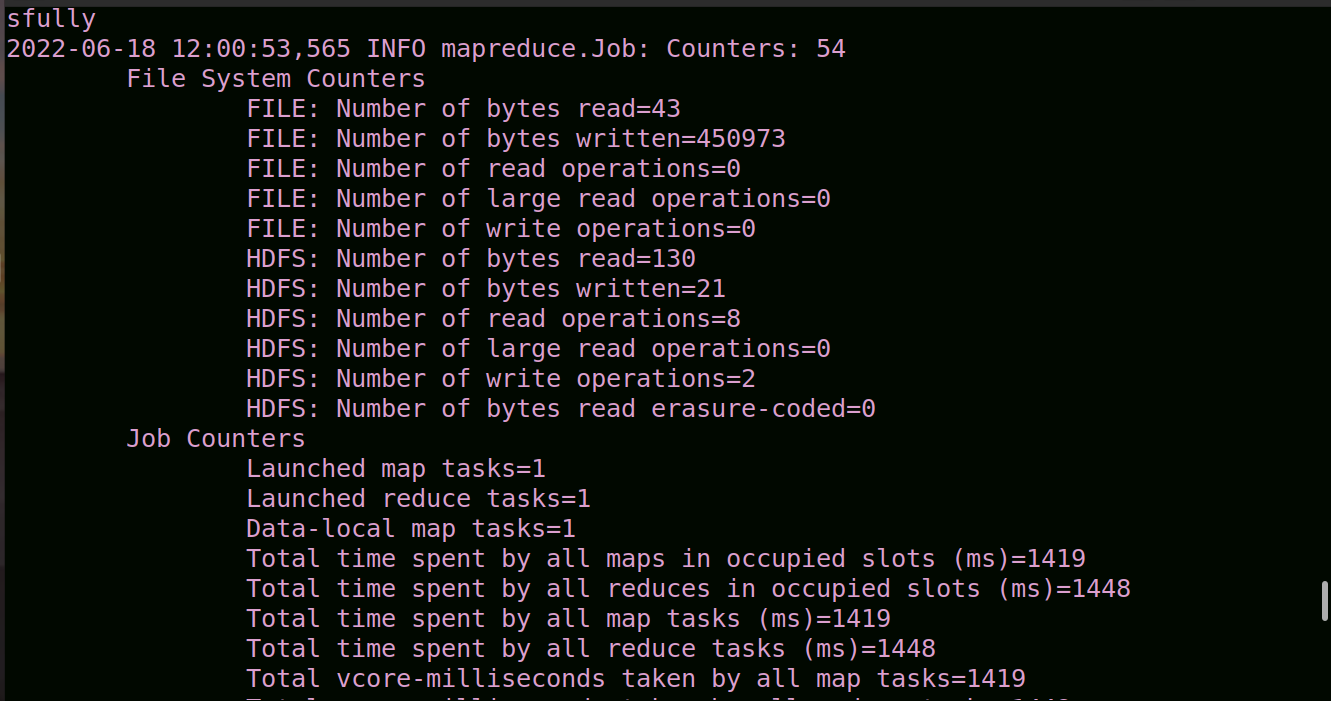






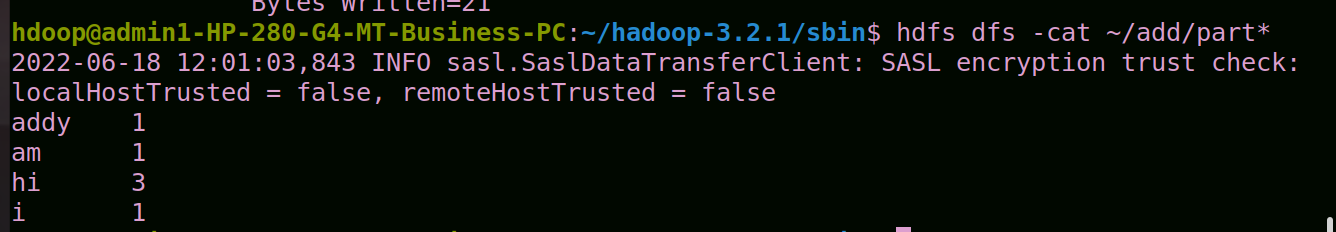






hdfs dfs -cat ~/output/part\*

We can see the wordcount of the words in file



Code:

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 Mapreduce {

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(Mapreduce.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);

}

}