* 1. hadoop version

start-all.sh

jps

sudo apt install net-tools

ifconfig

URL [http://your-server-ip:9870.](http://your-server-ip:9870./) Or localhost:9870

URL <http://your-server-ip:8088.or> localhost:8088

Hadoop fs -mkdir /input

DOC->CREATE FOLDER NAME FILES->page1.txt(add content)->copy its path

Hadoop fs -put filepath /input

fs -ls /input

eclipuse->java project->MapReduceWordCount

com.mapreduce.wc

WordCount

WordCountMapReduce.jar

Hadoop jar jarfilepath com.mapreduce.wc/WordCount /input/page1.txt /output

Hadoop dfs -cat /output/\*

package com.mapreduce.wc;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

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 org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

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

Configuration c = new Configuration();

String[] files = new GenericOptionsParser(c, args).getRemainingArgs();

// Ensure correct input arguments

if (files.length < 2) {

System.err.println("Usage: WordCount <input path> <output path>");

System.exit(-1);

}

Path input = new Path(files[0]);

Path output = new Path(files[1]);

Job j = Job.getInstance(c, "wordcount");

j.setJarByClass(WordCount.class);

j.setMapperClass(MapForWordCount.class);

j.setReducerClass(ReduceForWordCount.class);

j.setOutputKeyClass(Text.class);

j.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(j, input);

FileOutputFormat.setOutputPath(j, output);

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

}

// Mapper Class

public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable> {

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

private Text wordText = new Text();

public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException {

String line = value.toString().trim();

String[] words = line.split("\\s+"); // Handles multiple spaces

for (String word : words) {

if (!word.isEmpty()) { // Avoid empty strings

wordText.set(word.trim().toUpperCase());

con.write(wordText, one);

}

}

}

}

// Reducer Class

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

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

int sum = 0;

for (IntWritable value : values) {

sum += value.get();

}

con.write(word, new IntWritable(sum));

}

}

}

2.

import java.io.IOException;

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 {

// 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();

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

String[] words = value.toString().split("\\s+");

for (String w : words) {

word.set(w);

context.write(word, one);

}

}

}

// Reducer Class

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);

}

}

// Main Driver Method

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.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);

}

}

3.

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;

public class LogCount {

// Mapper Class

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

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

private Text word = new Text();

@Override

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

String[] parts = value.toString().split(" ");

if (parts.length > 0) {

word.set(parts[0]); // Take the first word (like ERROR, INFO)

context.write(word, one); // Emit (word, 1)

}

}

}

// Reducer Class

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

@Override

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

int sum = 0;

for (IntWritable val : values) {

sum += val.get(); // Sum all counts

}

context.write(key, new IntWritable(sum)); // Emit (word, total count)

}

}

// Driver (Main Function)

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Log Count");

job.setJarByClass(LogCount.class);

job.setMapperClass(LogCountMapper.class);

job.setReducerClass(LogCountReducer.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);

}

}

4.

package com.mapreduce.lf;

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 Process{

public static class IPMapper

extends Mapper<Object, Text, Text, IntWritable>{

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

private Text ip = new Text();

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

// Assuming the IP address is the first token in each line

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

if (itr.hasMoreTokens()) {

ip.set(itr.nextToken());

context.write(ip, 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, "IP address count");

job.setJarByClass(Process.class);

job.setMapperClass(IPMapper.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);

}

}