Homework 3

Word counting part:

preparing part

Text

Description automatically generated with medium confidence

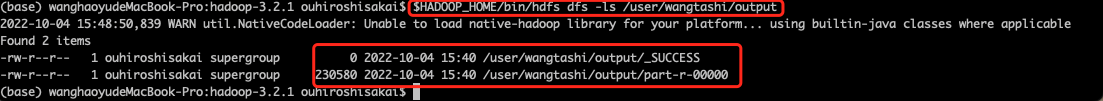
hadoop part:Text

Description automatically generated

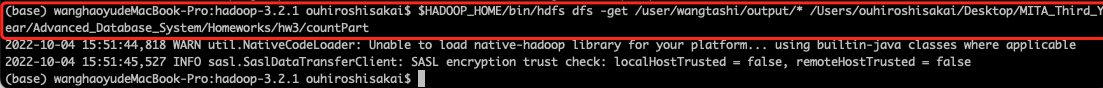
Text

Description automatically generated

output part:



paste the result to local



sortPart:

Text

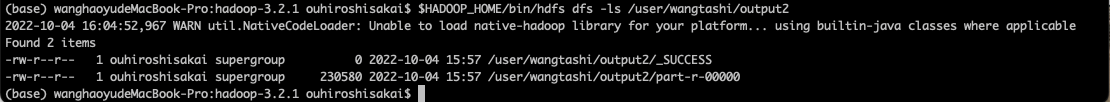
Description automatically generated with medium confidence

Graphical user interface, text

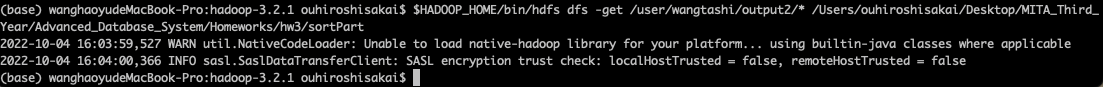
Description automatically generated

Text

Description automatically generated

running status: 

paste to local:



result:

countpart:

Graphical user interface, application

Description automatically generated

sortpart:

Text

Description automatically generated with medium confidence

Aws part:

countPart:

Graphical user interface, application, Teams

Description automatically generated

countPartMR gives me 6 file so I run countPartMR1 again, it still have 6 file -- just need to verify I am not doing it wrong. countPartSteamingMerge merge the 6 files to one, notice that the file here is all reordered since the intermediate part reordering it using the KEY.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, website

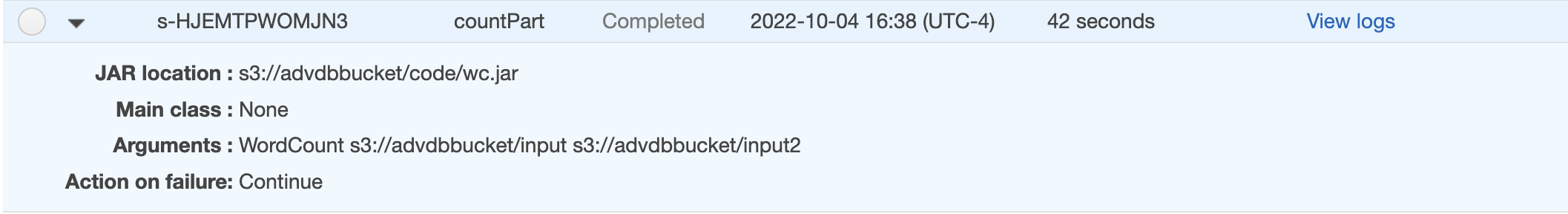
Description automatically generated

Shape

Description automatically generated with low confidence

sortPart:

Graphical user interface, application, Teams

Description automatically generated

I just change my code to force the MR job flow to have only one reducer by "job.setNumReduceTasks(1);"

Then it give me only one file.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated with medium confidence

code:

wc.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;  **public class** WordCount {  **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[] list = value.toString().split(**"\\W"** );  **for** (**int** i = 0; i < list.length; i ++ ) {  word.set(list[i].toLowerCase());  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, **"WORDCOUNT"**);  job.setJarByClass(WordCount.**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);  }   } |

wcr.java

|  |
| --- |
| **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.mapred.OutputCollector; **import** org.apache.hadoop.mapred.Reporter; **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** wcr {  **public static class** TokenizerMapper  **extends** Mapper<Object, Text, LongWritable, Text> {  **private** Text **word** = **new** Text();  **private** LongWritable **count** = **new** LongWritable();  **public void** map(Object key, Text value, Context context  ) **throws** IOException, InterruptedException { *// String[] list = value.toString().split(System.lineSeparator()); // for (int i = 0; i < list.length; i ++ ) { // String[] nodes = list[i].split("\\s");* String[] nodes = value.toString().split(**"\\t"**);  *//.replaceAll("\\s+","")* **word**.set(nodes[0]);  **count**.set(-1\*Integer.*parseInt*(nodes[1]));  context.write(**count**, **word**); *// }* }  }   **public static class** IntSumReducer  **extends** Reducer<LongWritable,Text,Text,LongWritable> {  **private** LongWritable **result** = **new** LongWritable();  **private** Text **finalWorld** = **new** Text();  **public void** reduce(LongWritable count,Iterable<Text> words,  Context context  ) **throws** IOException, InterruptedException {  **long** intCount = count.get();  intCount \*= -1;  **result**.set(intCount);  **for**(Text word : words){  **finalWorld**.set(word);  context.write(**finalWorld**,**result**);  }   }  }   **public static void** main(String[] args) **throws** Exception {  Configuration conf = **new** Configuration();  Job job = Job.*getInstance*(conf, **"WORDCOUNT"**);  job.setJarByClass(wcr.**class**);  job.setMapperClass(TokenizerMapper.**class**);  job.setReducerClass(IntSumReducer.**class**);  *//* job.setMapOutputKeyClass(LongWritable.**class**);  job.setMapOutputValueClass(Text.**class**);  *//* job.setOutputKeyClass(Text.**class**);  job.setOutputValueClass(LongWritable.**class**);   job.setNumReduceTasks(1);  FileInputFormat.*addInputPath*(job, **new** Path(args[0]));  FileOutputFormat.*setOutputPath*(job, **new** Path(args[1]));  System.*exit*(job.waitForCompletion(**true**) ? 0 : 1);  }  } |