

Exercise 2: Hadoop MapReduce



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Concepts and Technologies for Distributed Systems and Big Data Processing – SS 2016

Solution 2 Implementation

You can download the code for the solution for this task from the course website.

Solution 3 Completion

Complete the following code for `WordLength`, which should count how many words belong to each of the following four length categories:

tiny: 1 letter — *small*: 2–4 letters — *medium*: 5–9 letters — *big*: more than 10 letters

```
1 public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
2     private final static IntWritable one = new IntWritable(1);
3     private Text category = new Text();
4
5     @Override
6     protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {
7         StringTokenizer tokenizer = new StringTokenizer(value.toString(), ",;\\. \\t\\n\\r\\f");
8         while (tokenizer.hasMoreTokens()) {
9             String word = tokenizer.nextToken();
10
11             int length = word.length();
12             String c = ((length == 1) ? "tiny" :
13                 (length >= 2 && length <= 4) ? "small" :
14                 (length >= 5 && length <= 9) ? "medium": "big");
15             category.set(c);
16             context.write(category, one);
17         }
18     }
19 }
20 }
21
22 public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
23     private IntWritable result = new IntWritable();
24
25     @Override
26     protected void reduce(Text key, Iterable<IntWritable> values, Context context)
27         throws IOException, InterruptedException {
28
29         int sum = 0;
30         for (IntWritable val: values) {
31             sum += val.get();
32         }
33         result.set(sum);
34         context.write(key, result);
35     }
36 }
37 }
```

Solution 4 Comprehension

Understand and explain what the following code does. What is the output of the program for the following input?

file1.txt: Hello World Bye World

file2.txt: Hello Hadoop Goodbye Hadoop

```
1 public static class TokenizerMapper extends Mapper<LongWritable, Text, Text, Text> {
2     private Text word = new Text();
3     private Text file = new Text();
4
5     @Override
6     protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
7         FileSplit fileSplit = (FileSplit)context.getInputSplit();
8         String fileName = fileSplit.getPath().getName();
9         file.set(fileName);
10
11         StringTokenizer tokenizer = new StringTokenizer(value.toString());
12         while (tokenizer.hasMoreTokens()) {
13             word.set(tokenizer.nextToken());
14             context.write(word, file);
15         }
16     }
17 }
18
19 public static class InvertedReducer extends Reducer<Text, Text, Text, Text> {
20     private Text result = new Text();
21
22     @Override
23     protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
24         StringBuilder sb = new StringBuilder();
25         for (Text val: values) {
26             sb.append(val);
27
28             if(values.iterator().hasNext()) {
29                 sb.append(",");
30             }
31         }
32         result.set(sb.toString());
33         context.write(key, result);
34     }
35 }
```

The code computes the inverted index for the given documents, i.e., a list of references to documents for each word. It produces the following output:

```
Bye file01
Goodbye file02
Hadoop file02,file02
Hello file02,file01
World file01,file01
```