

Assignment 06

Handed out: 03/08/2014

Due by 11:59PM on Friday, 03/14/2014

Problem 1) Attached file “Using String Tokenizer in Java.docx” contains a small tutorial article on properties of `jav.util.StringTokenizer` class. You should also consult Java docs for that class. You can find those at:

<http://docs.oracle.com/javase/6/docs/api/java/util/StringTokenizer.html>

Both from the article and from the class description you realize that the String Tokenizer used in our class example `WordCount.java`, file is attached, could do more. If not told otherwise, it breaks strings into tokens on empty spaces. That left a lot of punctuations, parenthesis and the like on the result of our `wordcount.jar` MapReduce program.

Modify `WordCount.java` class so that when you run `wordcount.jar` on all-bible text, you get the list of words and without punctuations and other non-word characters.

When developing `map()` method, please extract the `map()` method into a standalone Java class which has its own `main()` method and which you can run on the command prompt. In that standalone class, instead of writing to the `context` object, use

`System.out.println` to write to the console. When you are sure that your `map()` method does what you expect it to do, only then place it back into the class

`WordCount.java`. If you are new to Java, use method `replace()` of class `java.lang.String` to remove undesired characters. If you are a Java pro, please use regular expressions.

Similarly, modify the `reduce()` method, so that it emits only those words which appear less than 1000 times. Let `reduce()` also get rid of verse counts in the form 01:004:010.

Run your new version of `wordcount.jar` on your favorite VM. Use the attached `all-bible.txt` file for your test.

Problem 2) Demonstrate that your modified `WordCount.java` will produce the same result whether you compile it and `jar` it on your PC (MAC) or on your Linux VM.

Problem 3) The result of `wordcount.jar` MapReduce job is an (unordered) list of words and their frequencies, i.e. `zeal 13`, `youths 1`, etc. Write a simple MapReduce program that would take the output file of your `WordCount` program and flip the words and frequencies and produce a list of output frequencies vs. words, like: `13 zeal`, `1 youths`, etc. Could you make that output ordered so that it starts with the lowest frequency and ends with the highest.

Problem 4) On a slide towards the end of the lecture notes we made a claim that the original `WordCount.java` was too complex and that we could have achieved the same result using built in Mappers and Reducers in the manner contained in class

`WordCount2.java`.

```

package org.apache.hadoop.examples;
public class WordCount2 {
public static void main(String[] args) {
    JobClient client = new JobClient();
    JobConf conf = new JobConf(WordCount2.class);
    FileInputFormat.addInputPath(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(LongWritable.class);
    conf.setMapperClass(TokenCountMapper.class);
    conf.setCombinerClass(LongSumReducer.class);
    conf.setReducerClass(LongSumReducer.class);
    client.setConf(conf);
    try {
        JobClient.runJob(conf);
    } catch (Exception e) { e.printStackTrace(); }
}
}

```

Please, make the above class work and verify the claim.

Problem 5) Hadoop's HDFS API allows you to manipulate files and date programmatically. When running your MapReduce jobs Hadoop prefers to work with one file rather than many. For whatever reasons, there appears to be no utility that merges files. The attached simple utility class `PutMerger.java` attempts to make up for that deficiency. Please try to fix the class if anything is wrong with it and then examine whether it could truly merge two files into one. Please note, you should run the class using **hadoop jar** command and the standard `hadoop classpath` command.

Capture all steps of your implementation with comments indicating what is it you are accomplishing with every step in an MS Word document.

Please place all files you want to submit in a folder named: HW06. Compress that folder into an archive named E63_LastNameFirstNameHW06. ZIP. Upload the archive to the course drop box on the class web site. Please send comments and questions to cscie63@fas.harvard.edu