Trier University
Department IV – Computer Sciences
Prof. Dr.-Ing. Ralf Schenkel (H 539, schenkel@uni-trier.de)

Trier, April 27, 2022

Assignment 3 of the course Big Data Analytics

Summer semester 2022

Deadline 10:15h on Wednesday, May 4, 2022

General remarks on programming for Hadoop

In StudIP you can find an Eclipse project for programming Hadoop applications. This project includes the WordCount example from the lecture, together with example data and launch configurations. You can test your solutions in standalone mode using the data provided for each task. For new launch configurations, it is important to include an environment setting that sets `HADOOP_HOME` to `$project_loc:BDA/hadoop-3.2.0`, where you need to replace `BDA` with your project name; see the example launch configurations in the project. This is important especially on Windows to include the required binary and DLL.

If you want to use other IDEs (which is not a recommended setup), include the jar files from the Eclipse project in your application's class path. Running the application in stand-alone mode may be difficult, though, especially under Windows. It is important to set the environment variable `HADOOP_HOME` to refer to the directory `hadoop-3.2.0`. Under Windows, Java needs to be able to find the Hadoop DLL.

You do not have to run your solutions on a real Hadoop cluster. It is sufficient if you run your programs in the stand-alone mode.

Task 1: WordCount++ 15 points

Modify the `WordCount` example such that not the number of overall occurrences is counted, but the number of lines in which a term occurs.

- You only need to change the Mapper implementation. Instead of directly writing the tokens retrieved from the `StringTokenizer`, store them in an appropriate data structure like a `Set`. After all tokens were consumed, output the tokens included in the set. You can use a for-each loop for this, for example.

Hand in your code and the output of the reducer for the example input file `small_corpus.txt` that can be found in Moodle.

Task 2: Grep 15 points

Implement the grep example from slide 3-30 as an Hadoop MapReduce application.

- You can use the `WordCount` example as a template.
- The input to this task should be the file `corpus_with_line_numbers.txt` provided in Moodle (you need to decompress it first). In this file, every line corresponds to one document. A line has the form
  `docid: text...`
  where `docid` is a unique integer greater than 0, and `text` is the text of the document.

- You can either read the expression to search for from the command line or define it in your source code.
- Write a corresponding implementation of the `Mapper` interface (like in the `WordCount` example). Extract the docid and the text from the line read. If the text includes the searched expression, emit the docid as a result of the wrapper; otherwise, emit nothing. Use an appropriate Java function for identifying matching text.
- Since this is a Map-only job, there is no need for a reducer. You need to set this in the Job configuration in the `run()` method, using
  `job.setNumReduceTasks(0);`

Hand in your code and the result of grepping for the term `"MapReduce"` (without the quotes) in the provided file.

Task 3: Inverted Index 15 points

Write an Hadoop MapReduce application that generates an inverted index for a document collection, following slides 3-36 and 3-37 from the lecture.

- You can use the `WordCount` example as a template.
- The input to this task should be the file `small_corpus_with_line_numbers.txt` provided in Moodle. In this file, every line corresponds to one document. A line has the form
  `docid: text...`
  where `docid` is a unique integer greater than 0, and `text` is the text of the document.
- Write a corresponding implementation of the `Mapper` interface (like in the `WordCount` example). Extract the docid and the text from the line read. Split the text into its tokens using a `StringTokenizer`. For each word, emit the word as a key and the docid as a value (converted to an `int`). Write a corresponding implementation of the `Reducer` interface (like in the `WordCount` example). Your reducer will emit `Text` as key (for the word) and `Text` as value (for the list), so you need to adapt the generics:
  ```
  public static class Reduce
  extends Reducer<Text,IntWritable,Text,Text>
  ```
  In the reducer, build a String representation of the corresponding list. For example, if you read the numbers 1,2, and 5 from the `Iterable`, the String should look as follows:
  `1, 2, 5`
  Write that String as a result, using the `write` method of the `context` object. Instead of an `IntWritable`, you need a `Text` object since we are writing a String, not a number.
- In order to reduce the number of written lists, **emit the list only if it contains more than three entries**. For shorter lists, emit nothing.
- Adapt the job configuration such that the result value of the reducer is of type `Text`, not `IntWritable`:
  `job.setOutputValueClass(Text.class);`
  in the `run()` method. Since our job now has different keys and values for Map and Reduce, we also need to specify the corresponding classes for the Map task:
  ```
  job.setMapOutputKeyClass(Text.class);
  job.setMapOutputValueClass(IntWritable.class);
  ```

Hand in your code and the output of the reducer for the example input file.

These tasks will be discussed on May 18, 2022.

General remarks:

- The tutorial group takes place on Wednesdays in the regular Zoom meeting at 10:15, see StudIP (slides with general information on the lecture) for the registration link.
- To be admitted to the final exam, you need to acquire at least 50% of the points in the assignments.
- It is preferred to submit in groups of size 3; only one submission is sufficient for the whole group. Groups must be chosen in Moodle (see link on the course page in Moodle). Write the names of all group members on your solutions.
- Solutions must be handed in before the deadline in Moodle (`https://moodle.uni-trier.de/`, course `BDA-22`) as as a PDF or, if submitting multiple files, as an archive (.zip or comparable). Submissions that arrive after the deadline will not be considered.
- Graded versions of your submissions will be returned in Moodle until the following tutorial.
- Announcements regarding the lecture and the tutorial group will be done in the area of the lecture in StudIP.