Technische Universität Berlin
Database Systems and Information Management Group
Prof. Dr. Volker Markl, Sebastian Schelter, Christoph Boden, Thomas Bodner

AIM3: Scalable Data Mining and Data Analysis
Summer term 2012

First Assignment

MapReduce and Hadoop

Due on May 2nd

Basics of MapReduce and Hadoop

1. **WordCount - "Hello World"of MapReduce**

   We'll start with the classic MapReduce example of counting words. Your task is to complete the code in *de.tuberlin.dima.aim3.assignment1.FilteringWordCount*. The output of this job should be a textfile holding the following data per line:

   word[TAB]count.

   An additional requirement here is that stop words like *to*, *and*, *in* or *the* should be removed from the input data and all words should be lowercased.

2. **A custom Writable**

   You will work on your first custom Writable object in this task. Have a look at the class *de.tuberlin.dima.aim3.assignment1.PrimeNumbersWritable*, which models a collection of prime numbers. Writable classes need to be able to serialize to and deserialize from a binary representation. Enable that for our custom Writable by implementing *write(DataOutput out)* and *readFields(DataInput in)*.

3. **Average temperature per month**

   Have a look at the file *src/test/resources/assignment1/temperatures.tsv*. It contains the output of a fictional temperature sensor, where each line denotes the year, the month and the temperature of a single recording. Additionally a quality parameter is included which expresses how "sure"the sensor was of a single measurement:

   year[TAB]month[TAB]temperature[TAB]quality

   Your task is to implement a MapReduce program that computes the average temperature per month of year. It should ignore all records that are below a given minimum quality. The output of your program will be a textfile holding the following data per line:

   year[TAB]month[TAB]average temperature

   Use *de.tuberlin.dima.aim3.assignment1.AverageTemperaturePerMonth* as a starting point.

Parallel Joins in MapReduce

Next we deal with bibliographic data about authors and books located in the folder *src/test/resources/assignment1/*. Author names and ids are contained in the file *authors.tsv*, the file *books.tsv* contains books, their year of publication and their author id. We will use MapReduce and Hadoop to sort and join this data.

4. **Sort the books with "Secondary Sort"**

   We want to transform the book data into a list of *(century, title)*-tuples, where century just denotes the first two digits of the year of publication. Each line in the output file should have the format

   century[TAB]book title.

   The output data must be sorted ascending by century and title. You must not sort the data yourself, but must use Hadoop's "Secondary Sort"capabilities to have the framework do the sorting for you in the shuffle phase.

5. **Join books and authors with a "Broadcast Join"**

   In this task we will perform an inner join of the books and authors on the author id. Use a "Broadcast Join": load the smaller dataset into memory in your mapper class and perform the join before sending the tuples to the reducer over the network. Each line in the output file must have the format:

   authorname[TAB]book title[TAB]year of publication

6. **Join books and authors with a "Reduce-side Join"**

   We will peform the same join here as in task 5, but we will use another technique called "Reduce-side Join". The join should be performed in the reducer class, an optimal solution should also avoid buffering more than one value in the reducer.

Deadline

Source code for the exercises is available at *https://github.com/dimalabs/scalable-datamining-class*.

Register in the ISIS information system at *https://www.isis.tu-berlin.de/course/view.php?id=6535* and upload your solution in the form of a patch file until May 2nd.