Universität Freiburg
Institut für Informatik
Prof. Dr. G. Lausen
Alexander Schätzle
Martin Przyjaciel-Zablocki

Georges-Köhler Allee, Geb. 51
D-79110 Freiburg
lausen@informatik.uni-freiburg.de
schaetzl@informatik.uni-freiburg.de
zablocki@informatik.uni-freiburg.de

Advanced Information Systems
Summerterm 2011
07.07.2011

5. Exercise Sheet: MapReduce

Discussion: 21.07.2011

Submission Guidlines: Source code + executable program.
If you want your solutions to be reviewed you have to send them by email until 19.07.2011. Use comments to explain your solution. Not commented solutions will not be reviewed!

Exercise 1 (Hadoop Installation)
Install the Hadoop Distribution of Cloudera (`http://www.cloudera.com/hadoop/`) in Pseudo-Distributed Mode or use the VMWare Image provided by Cloudera to familiarize yourself with Hadoop, especially with the distributed file system HDFS and the implementation of MapReduce programs in Java. You can find a good introduction to MapReduce and Hadoop in the given literature at the end of this exercise sheet, for example.

Exercise 2 (Basic Text Operations)
For the following tasks use the file '`twain.txt`' as input which contains a collection of the works of Mark Twain. You will find the file on the course website.

a) Implement a MapReduce program that outputs all words of the input in a sorted order. Your program should not distinguish between upper and lower case and duplicates should be preserved.
Example: {To be or not to be} ⟶ {be be not or to to}

b) Extend your program from part (a) such that every word occurs only once in the output together with the corresponding frequency of the word. Your program should not distinguish between upper and lower case.
Example: {To be or not to be} ⟶ {(be,2) (not,1) (or,1) (to,2)}

c) Extend your word count implementation from part (b) with an additional *Combiner*. Therefor you should familiarize yourself with the function of a Combiner and think about how to usefully integrate a Combiner into your implementation. Characterize advantages and disadvantages of a Combiner.

d) Implement a MapReduce program that computes the *inverted index* for the given input, i.e. for every word in the input it should output a list of (byte) offsets. The offset should be the byte offset of the row that contains the word. However, typical *stop words* should not be part of the index. Stop words are frequently occuring words like '*and*' that do not have a substantial relevance. You can find a list of typical english stop words in the file '`english.stop.txt`' from the course website.
Example: science (1024, 6824) ⟶ 'science' is contained in two rows with offsets 1024 and 6824

Exercise 3 (SQL, 10+15+15+15)
`City(name, country, province, population, latitude, longitude)`
`Country(name, code, capital, province, population, area)`

Consider relations `City` and `Country`. Implement for every of the following SQL queries an equivalent MapReduce program. Use the files '`Country.dat`' and '`City.dat`' from the course website.

a)
```
SELECT name, country, province, population
FROM City
WHERE population > 100000
```

b)
```
SELECT country, province, SUM(population)
FROM City
GROUP BY country, province
```

c) In MapReduce a JOIN can be implemented as *Map-Side-Join* or *Reduce-Side-Join*. Describe both approaches and characterize advantages and disadvantages. Choose the approach which seems appropriate to you to implement the following SQL query in MapReduce.

```
SELECT Country.name, City.name, City.population
FROM Country JOIN City ON Country.capital = City.name
     AND Country.province = City.province AND Country.code = City.country
```

Literature:
1. Tom White: *Hadoop: The Definitive Guide*, Second Edition, 2010. ISBN 978-1-449-38973-4.