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

Master Project
Data Analysis & Querying on Hadoop
Winterterm 2016/2017

19.10.2016

Exercise 1. 1 (Introduction)
First of all, take a look at some introductions about MapReduce and Hadoop. The following literature is recommended for the first steps.
Articles:

- *The Google file system*. Ghemawat, S. and Gobioff, H. and Leung, S.T., 2003.
- *MapReduce: Simplified Data Processing on Large Clusters*. Dean, J. and Ghemawat, S., ACM 2008.
- *Impala: A Modern, Open-Source SQL Engine for Hadoop*. Kornacker, M. et al., 2015.
- *Spark SQL: Relational Data Processing in Spark*. Armbrust, M. et al., 2015.

Books:

- *Hadoop: The Definitive Guide*. Third Edition, Tom White, O'Reilly Media 2010.
- *Data-Intensive Text Processing with MapReduce*. Lin, J. et al, Morgan and Claypool Publishers 2010.

Exercise 1. 2 (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.

Exercise 1. 3 (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 can find the file on ILIAS.

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} $\longrightarrow$ {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} $\longrightarrow$ {(be,2) (not,1) (or,1) (to,2)}

c) 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. Use the content of the file '`english.stop.txt`' (available on ILIAS) that contains a collection of typical english stop words.
Example: science (1024, 6824) ⟶ 'science' is contained in two rows with offsets 1024 and 6824

Exercise 1. 4 (SQL on Hadoop)
Run the following SQL queries on Impala and also on Spark (using Spark SQL).
Use the files '`Country.dat`' and '`City.dat`' (available on ILIAS) as input.

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

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

c)
```
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
```