Distributed Data Management SS 2015
Prof. Dr.-Ing. Sebastian Michel
MSc. Evica Milchevski, MSc. Kiril Panev
TU Kaiserslautern, FB Informatik – Lehrgebiet Informationssysteme
Sheet 2: Handout 30.04.2015, Presentation 12.05.2015

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN
http://dbis.informatik.uni-kl.de

Assignment 1: SQL and Sort (1 P.)

Given the relation **City** (name, country, province, population, latitude, longitude)

(a) Write a pseudo code of a MapReduce job, which produces an equivalent output as the following SQL queries:

- `SELECT name, country, province, population FROM City WHERE population > 100000`
- `SELECT country, province, AVG(population) FROM City GROUP BY country, province`
- `SELECT country, SUM(population) FROM City GROUP BY country`

(b) Consider the following SQL query:

```
SELECT name, country, province, population FROM City WHERE population > 100000
ORDER BY population
```

What are the problems that can arise when implementing the ORDER BY clause in MapReduce. Discuss the possible solutions.

You can test your jobs using the City.dat file (a pipe (|) delimited file).

Assignment 2: Secondary Sorting in MapReduce (1 P.)

For this assignment, you will write a program that mines weather data. Weather sensors collect data every hour at many locations across Germany and gather a large volume of log data, this makes it a good candidate for analysis with MapReduce because it is semi-structured and record-oriented. The data stored in the files are of the following format: ***month/day/year;station;hour;temperature***.

Example:

1/1/2000;1;1;1.588586391772654
2/1/2000;2;3;1.981028401924819
2/1/2000;2;4;1.875632896548555
...

Your task is the following:

- Implement in Java a MapReduce job that will output the maximum temperature per day by using **secondary sorting**. The sorting should not be implemented in the reducer, but instead you should make use of the built-in sorting functionality of the MapReduce framework.

Assignment 3: Joins in MapReduce (1 P.)

As part of this task you should implement the Map-Side Join in Hadoop. Write a Java program that will output the inner join of the Orders and Customer relations from the TPC-H benchmark. The two relations have the following attributes, where → expresses a foreign key relationship:

- customer(c_custkey, c_name, c_address, c_nationkey, c_phone, c_acctbal, c_mktsegment, c_comment)
- orders(o_orderkey, o_custkey → customer.c_custkey, o_orderstatus, o_totalprice, o_orderdate, o_orderpriority, o_clerk, o_shippriority, o_comment)

You can obtain the customers and orders data, represented as a pipe (|) delimited file, from the link given below. Note: A Java program implementing another join algorithm will not be accepted as a correct solution.

Hints. *The given data is not sorted. For the implementation, you should use the classes provided with the new MapReduce API, contained in the following libraries, where x is the version of Hadoop that you are using (e.g. 2.6.0): hadoop-mapreduce-client-core-x.jar and hadoop-common-x.jar*

Remark. *For the implementation, you can either install Hadoop from scratch on your local machine or you can use the pre-installed virtual machine provided from HortonWorks or Cloudera.*

http://hortonworks.com/hdp/downloads/
http://www.cloudera.com/content/support/en/downloads/quickstart_vms.html

All the files required for the assignments can be downloaded from the following link:

http://dbis.informatik.uni-kl.de/files/teaching/ss15/ddm/protected/sheet2_files.tar.gz

Please note that the solutions for Assignment 2 and 3 need to be demonstrated using a laptop during the Exercise session. In order to make things easier it would be the best to use your own or a friend's laptop.