Distributed Data Management SS 2017
Prof. Dr.-Ing. Sebastian Michel
MSc. Kiril Panev
TU Kaiserslautern, FB Informatik – Lehrgebiet Informationssysteme
http://dbis.informatik.uni-kl.de
Exercise Sheet 2: Hand out 27.04.2017, Due 09.05.2017 - 16:00

Note: You need to upload a single zip containing all files - the Java code and a pdf for the non-coding parts.

Assignment 1: 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.

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. 2.8.0): hadoop-mapreduce-client-core-x.jar and hadoop-common-x.jar

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. If you are installing Hadoop by yourself, we recommend you do it on Linux.

`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/ss17/ddm/protected/sheet2_files.tar.gz`

Assignment 2: Secondary Sort (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: Total Sort (1 P.)

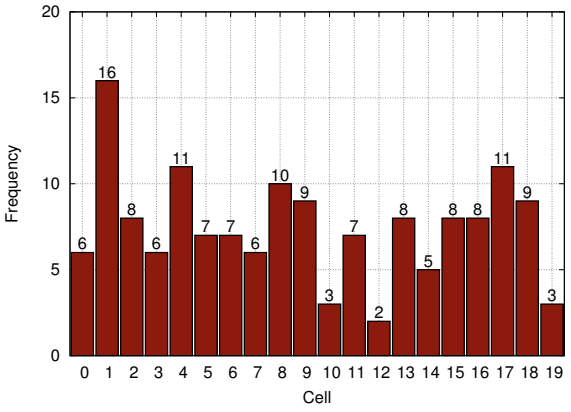
We have seen in the lecture that creating a sorted output is only possible within the output of a reducer, but not globally across the individual output files. The naive solution to perform a total sort is to use only one reducer but this is apparently for large data not advisable. Instead, one way to split work across multiple reducers is to let each reducer handle a specific key range, such that the reducers' output files need just to be "concatenated" to have a fully sorted output.

(a) **MapReduce Job for Total Sort:** Write pseudo code of a MapReduce job that is doing this total sort based on key ranges. For this, assume input in form of pairs of integer values of `employee_id` and `salary`. We know the salary is between 0 and 100,000 and want to sort by salary in descending order.

(b) **Load Balancing using Histograms:** Without knowledge of the distribution of the salary values (i.e., how many employees have a specific salary) it is not possible to group by partitions that impose roughly the same load to each reducer, in terms of number of tuples to process. Luckily, we have the following histogram at hand that shows 20 equi-width cells in the range 0 to 100,000, showing how many employees have a salary that falls in that cell. Assume that data is uniformly distributed within each cell.

(i) Compute given this histogram the ranges in total sort that would lead to 5 equally sized partitions; do this as accurate as possible.

(ii) How can you actually create such an equi-width histogram with an a priori fixed number of cells in MapReduce? Write pseudo code of the appropriate Map and Reduce functions.



(c) **Load Balancing using a given Distribution Function:** Instead of having a histogram as in part(b) assume that we know that the salaries are distributed following a normal (Gaussian) distribution with mean $\mu = 50000$ and standard deviation $\sigma = 11000$. Compute the ranges that would lead to 5 equally sized partitions. You also need to explain your approach. **Hint:** interpret the cumulative distribution function (cdf) of the normal distribution or directly use its quantile function.