Exercises for Seminar 1

MapReduce

1. What is the signature for `combine` given the signatures `map: (k1, v1) → list(k2, v2)` and `reduce: (k2, list(v2)) → list(k3, v3)`?
2. If we also use the reducer as combiner, what can we then say about k2, k3, v2, and v3?
3. Assume that for some MapReduce job *J* the runtimes for its successful mappers were $a_1, a_2, \ldots, a_m$ and the runtimes for its successful reducers were $b_1, b_2, \ldots, b_r$.
    a. What can be said about the runtime of *J*?
    b. Why is this important when we consider stragglers?
4. Sketch solutions in terms of MapReduce to the following problems:
    a. Given a collection of web pages, find the average number of HTML `<table>`s for the pages.
    b. Given a collection of football bets, find for each match the average number of goals the betters expect in the match
    c. Given a dictionary of all words in some language, find all anagrams (different words made of the same letters, for example *bus/sub*, *stop/post, car/arc,* and *orchestra/carthorse*)
5. Consider your answers to the previous exercise. Can you improve your solutions by using combiners? If yes, sketch how.
6. When Hadoop's TextInputFormat is used, the byte offset from the beginning of the file is used as key and the text line is used as value. Why didn't they just use the line number as key?

HBase

7. When HBase physically stores a cell version, it also stores the column name (the "qualifier").
    a. Why does it do that?
    b. Should you as a developer care about that?
8. When you create a new table in HBase, insertion speed can be pretty low at first. Why?
9. When HBase deletes a row *x*, it inserts a marker in its flush files telling that *x* has been deleted. Why doesn't it just set a flag in *x*?

Hive

10. In Hive, external tables can also be partitioned if the user explicitly uses ADD PARTITION. Partition locations can also be ALTERed. How is this useful if you have a lot of, say, log data where you most often only query the newest data?
11. When two data sets are bucketed on the same key and have *b* and *nb* buckets ($n, b \in \mathbb{Z}^+$), respectively, joins can be done efficiently.
    a. Why is this only possible when there are *b* and *nb* buckets, respectively?
    b. What if the buckets are stored in sorted order? Can we then join even more efficiently?

Pig

In the directory `/home/chr/data/baseball`, there are CSV files with baseball statistics[1]. You will need the file `Batting.csv` (you can read it from my directory or copy it to your own with the command `cp` *`source target`*).

Field 0 holds a player's ID, field 1 a year, and field 8 the number of runs by the player in that year (all fields are explained in the file `readme2014.txt`).

You can use `LOAD '`*`filename`*`' USING PigStorage(',')` to load a CSV file in Pig.

Your task is to use Pig to

12. find the number of runs for each year. The results will thus be something like (1871,3101), (1872,4487), …, (2016,42276)
13. find for each year, the maximum number of runs and the player(s) who did the runs. The results will thus be something like …, (1912,jacksjo01,226), (1912,cobbty01,226), …
14. Do something interesting with Pig if you have time enough ☺

The relevant Pig documentation is available from http://pig.apache.org/docs/r0.17.0/basic.html and http://pig.apache.org/docs/r0.17.0/func.html. It is possible to solve the exercises with the constructs you have seen in the lecture slides.

[1] The data comes from http://www.seanlahman.com/ and is available under the CC BY-SA 3.0 license