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 3: Hand out 11.05.2017, Due 23.05.2017 - 16:00

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

Assignment 1: Analytics with PIG and Min-Hashing (1 P.)

In this assignment you need to install the Apache PIG platform on the machine with your Hadoop installation. Alternatively, you can use one of the pre-installed virtual machines provided in the Sheet 2.

(a) Reconsider the weather data CSV file from exercise sheet 2. The data stored in the file are of the following format: ***month/day/year;station;hour;temperature***, here is an example:

Example:

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

Implement a PIG script that is computing for each hour of the day the first and third quartile of the temperature. To do so, implement a user defined function (UDF) in Java that is computing the quartiles. Test it in your own Hadoop/PIG installation (you should have installed Hadoop by now, just download and add PIG). Note that you are not allowed to use any external libraries that already implement a UDF's for computing the quartiles. Submit the PIG script, that would run in the PIG shell.

(b) Consider the file docs.csv, provided on the course website, with the following structure:

document_id;term_id

The file contains pairs of documents and the terms that appear in them. In this task you will need to implement Min-Hashing using PIG. First define a UDF that will be applied on the input and computes for the given terms their hash outputs. To keep it simple, use only 2 hash functions. Then, use PIG to compute the documents' min-hash signatures and all pairs of documents that have at least one overlapping element in their min-hash signatures.

Assignment 2: Data Analytics and SQL Query Processing in Spark (1 P.)

As part of this task you should install Spark on your local or virtual machine and implement a solution for the tasks below. We recommend that you do this on Linux. Feel free to use Java, Scala, or Python as programming language.

(a) On the lecture website you find next to this sheet a file containing Twitter tweets in JSON format.

(i) Compute the pairs of #hashtags that are most frequently used. Report on the top 10 pairs.

(ii) Compute the #hashtags with the highest relative increase of their frequencies in 10-minute windows (starting at 10:50:00 in the file). Ignore #hashtags that have an increase of zero to non-zero occurrences.

(b) Consider the two tables `Customers` and `Orders` of the TCP-H benchmark as used already on sheet 1, with the following schema

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

The tables are represented as CSV files, as per sheet 2.

(i) Write an application in Java or Scala, that is using the Spark API to compute the following relational algebra expression. Do not use any SQL (DataFrame) features of Spark, just the operations on RDDs. Execute it five times and report the median runtime.

$$\pi_{c\_custkey,o\_clerk}(\sigma_{c\_nationkey='4'}((\sigma_{c\_mktsegment='AUTOMOBILE'}(\text{customer}))$$

$$\bowtie_{c\_custkey=o\_custkey}(\sigma_{o\_orderstatus='F'}(\text{orders}))))$$

(ii) Try to optimize the relational algebra statement of (i) using textbook query optimization techniques (i.e., the ones every DBS lecture should teach), implement it without using SQL features of Spark, measure the runtime (like above, take the median of five executions) and compare it to the median runtimes measured (i). Now, use the SQL features of Spark to compute the semantically equivalent query. Execute it five times and report the median runtime. Compare all obtained runtimes. Where you able to get (roughly) the same (or even better) runtime than the one of the Spark SQL engine, or what is even worse than the one of (i)? Also specify the optimized relational algebra expression.

Assignment 3: Potpourri (1 P.)

In order to mark this assignment solved, you have to submit solutions to at least 8 out of 12 questions.

1. Describe why the suffix-based n-gram counting technique proposed in the lecture is correct, that is, why it determines the correct count for each n-gram at one and only one reducer, for all returned n-grams, and does not miss any result.
2. Describe informally why the collission probability in min hashing resembles the Jaccard Coefficient between two sets.
3. Explain the difference between using no reducer and using the identity reducer in Hadoop and give one example application for each case.
4. Given two tables s $R(A, B, C, D)$ and $S(B, C, D, E)$. We want to compute the equi join between R and S on attributes B and C at the reduce side, with `m` reducers. Which of the following combinations of keys in the output of map together with a custom partitioner do work or will not work, and which functionalities of the reduce method is needed?

(i)	`emit((t.B,t.C), ...)`	with partitioner	`key.hashcode % m`
(ii)	`emit((t.B), ...)`	with partitioner	`key.hashcode % m`
(iii)	`emit((t.B,t.D), ...)`	with partitioner	`key[0].hashcode % m`

5. A set of 259,141 elements should be encoded in a Bloom filter. We demand a pfp of at most 1 per mill. How many hash functions and how many bits should the optimal bloom filter use?

6. What are the pros and cons of the naive n-gram counting method compared to the apriori-based technique?

7. Given the following document (each character is one term):

   abcdfbacdeefefdfe

   Specify all suffices necessary to compute all σ-grams up to $\sigma = 4$.

8. A reducer receives the following suffixes as input to the reduce function call: abc, adf, abb, ax, axy, aag, azq, abg. Is this possible to occur in the suffix-based n-gram technique discussed in the lecture? Explain why it is possible or why it is not possible.

9. Consider two tables $R(A, B)$ and $S(B, C)$. S contains 100000 tuples. We want to compute the natural join and encode the distinct values of the B column of table R in a bloom filter that has a pfp of 0.1, using a reduce side join with map-side filtering employing this Bloom filter. Discuss the implications of the false positives for result quality and network traffic compared to the join that is using the set of distinct B values of R directly.

10. Describe the difference between JOIN and COGROUP in PIG. Can one be expressed using the other? Give an example involving two tables.

11. Given the following three permutations π_i and four documents d_j. Compute the min-hash signatures for each documents, as well as the estimated and true Jaccard similarities between the documents.

π_1	π_2	π_3
4	3	4
7	1	6
5	7	5
2	4	2
6	5	3
3	2	1
1	6	7

d_1	d_2	d_3	d_4
1	1	1	0
1	0	0	0
0	1	1	0
1	0	0	1
0	1	1	1
0	0	1	0
0	0	1	0

12. Assume the output of 5000 map tasks together is 0.5 TB large and there are 50 nodes that act as reducers. Each such reducer node has 100MB main memory buffer available. How many merge steps need to be executed at each reducer node using m-way merge and what is m?