Correction Report: *Big Data Analytics* Exam 20/8/2020

Christoph Kessler, IDA

Part 1 Questions 6-8

This was (intended to be) the easy part, and most students indeed got many points here.

Question 6 (1.5p)

The execution of a MapReduce operation involves seven phases. Which of these phases of MapReduce may involve *network* I/O, and for what purpose?
To answer this question write a maximum of 200 words.

Correction notes:
Note that the question is about *network* I/O, not *disk* I/O. The three phases that may involve network I/O are: record reader (read from HDFS, might be from a remote disk), shuffle-and-sort (download data with relevant key from mapper nodes' local disks, I accepted if this was named as part of the reducer with the right explanation), and output formatter (write to HDFS, at least two of the three replicas will go to remote disks). 0.5p for each right phase (0.25 name + 0.25 explanation); deductions for missing, wrong or incomplete explanations, and -0.5p for wrong phases (e.g., for the *partitioner* which does only local disk I/O).

Question 7 (3p)

Write pseudocode for a MapReduce program that reads floating point numbers x_i from a HDFS input file (in a text format of your choice, such as one value per line) and computes their geometric mean (sqrt($\Sigma_i x^2_i$)). Explain your code!
Hint: Make sure to follow the MapReduce programming model and clearly identify its different program parts in your code. — If you are unable to write MapReduce pseudocode, you may, at reduced points, write Spark pseudocode instead, or just try and describe the MapReduce program in plain English as precisely as you possibly can.
To answer this question write a maximum of 50 lines of commented pseudocode and a maximum of 200 words of explanation.

Correction notes: 1p each for identifying the right map and reduce computations (also for English and Spark pseudocode) and the correct overall structure (this was easy), the third point is given for proper MapReduce pseudocode including properly identified key and value types. +0.5 for those who use a combiner (with explanation). -0.5 point is deducted for inefficient programs first temporarily storing the entire sequence of squares in a list or similar data structure on the reducer node (as this leads to a huge working-set, resulting in a space/performance issue), a scalar accumulator variable should be used in the reducer.

Question 8 (0.5p)

Where are the data elements of a Spark RDD stored when evaluating a lineage of RDDs?

Correction notes: Right answer: each partition's elements are stored in (main) *memory* (i.e., *not* on disk, unless explicitly requested by the program or the node runs out of physical memory, which is unlikely if the Spark implementation is properly optimized) on each node computing that partition of the RDD. Note that the question is about the *data elements* of the RDD, not about the RDD itself.