Correction Report
on Part 2 of BDA Exam 20/08/2020

Christoph Kessler, IDA

Question 13 (1.5p)

In MapReduce, sometimes workers might be temporarily slowed down (e.g. due to repeated disk read errors) without being broken. Such workers could delay the completion of an entire MapReduce computation considerably. How could the master process speed up the overall MapReduce processing if it observes (how?) that some worker is late?
To answer this question write a maximum of 200 words.

Correction remarks: Observation of worker liveness is by polling (0.5p), which will also spot temporarily unresponsive workers, and the master will redistribute a late task to a different worker to not delay the entire computation indefinitely (1p). This question was correctly answered by almost all students.

Question 14 (1.5p)

Reconsider the geometric mean MapReduce program of Part 1 of this exam, applied to an overall (HDFS) input of N numbers. What is (i) the work (as defined in the lecture) performed by this MapReduce program on already distributed input of size N numbers, and what is (ii) the (parallel) time if executed as M mapper and R reducer tasks on P > 1 cluster nodes? Derive parametric formulas for work and time (use big-O notation where applicable) and justify your answer.
Assume for simplicity that a cluster node runs one task at a time, that reading/writing a single number from/to HDFS takes constant time, communicating K numbers costs time a K + b for constants a; b > 0, and that additions and multiplications of two numbers take constant time too. If you need to make any further assumptions, state them carefully.
To answer this question write a maximum of 200 words.

Correction remarks: 1p for a correct work formula including proper explanation, 0.5p for time.

This question was not even tried by most participants, and most answers given were wrong.

Question 15 (1p)

From a performance point of view, is it better to have long lineages or short ones in Spark programs? Motivate your answer (technical explanation).
To answer this question write a maximum of 100 words.

Correction remarks: The right answer is “long lineages” and the justification needs to argue over the amount of disk I/O accesses saved at every data flow transition after transformations within the lineage which Spark can perform in memory instead. The longer the lineage (i.e., the more transformations it does), the more we save by using Spark’s lazy execution of lineages with in-memory buffering of intermediate results.
I also gave partial points (up to 0.5p depending on explanation quality) for the argument that

excessively long lineages can increase the impact of failures and that in such cases an upper limit on lineage length could be appropriate after which the data is saved on disk (i.e. a kind of checkpointing), thus sacrificing some efficiency in the fault-free case for lower recovery cost in the fault case. However, because the fault-free case is much more common than the fault case, long lineages are overall better on average.

Question 16 (1p)

We know that Spark offers support for stream computing, i.e., computing on very long or even infinite data streams. Which fundamental property of stream computations makes it possible to overlap computation with data transfer?
To answer this question write a maximum of 100 words.

Correction remarks: In streaming / stream computing, we do not need to have all input data in place to start the computation. Because there are no data dependences to "previous" elements in the stream, we can work forward on the stream as data comes in, in portions of arbitrary size. This allows to process one block of data (through Spark processing, for example) while already aggregating and loading the next block of the stream in parallel.

Many answers skipped the fundamental property (dependence structure, see above) asked for and instead wrote (only) about "windowing" (which had been asked about in a previous exam), which was not asked for in this exam and thus got no partial points. Windowing, as well as pipelining, are not the fundamental property itself, they are just techniques that exploit it. Some simply claimed that streaming and data transfer be independent tasks, which is also wrong as a general statement – computing (!) and transfer are independent for different elements/blocks, but dependent for the same element/block. See the lecture notes.