This exam is **closed-book** (no electronics allowed, except pocket calculators; no paper materials allowed).

Write your **name and student number** on the first page (below this box). In case any of the pages get detached from the rest, also write your student number in a header of each sheet of paper.

Each question is marked with a number of percentage points (for example, 6 %), and they add up to 90 % (10 % is given by default). Give your answers **on this paper** in the space provided. Handwrite neatly. Design your answer on **scratch paper before** you start writing here, or you may run out of space on this paper!

All questions require you to explain your answer. The explanation must be **correct and complete** in order to get the points for that question.

Name and student number: _____

Question 1 *(12 %)*

In the Google File System:

1. What type(s) of metadata are kept by the master server?
   Answer:

2. Briefly, how does the *data* flow happen in a *write* operation? (The main steps suffice.)
   Answer:

3. What will be the end result on the disks when multiple *write* operations are executed concurrently (at the same time) at roughly the same location in the same file?
   Answer:

Question 2 *(12 %)*

Consider two frameworks to store one very big dataset in: any Distributed File System, and Bigtable. Imagine that your big dataset consists of relatively small data records (such as emails or posts on social media). You're looking to configure and use one of these frameworks to store the data, and you have an application which will use this dataset. Answer briefly the following questions:

1. If your application mostly *reads* individual data records from across the entire dataset, which storage framework would be more *efficient*, and why?
   Answer:

2. What if your application mostly *adds* new records to this dataset?
   Answer:

3. What if your application mostly *reads* the latest records in the dataset (say, the records which arrived at most 1 hour ago)?
   Answer:

Question 3 *(18 %)*

In a Spark program for processing big data:

1. What are the differences between *actions* and *transformations*, and why does this distinction matter?
   Answer:

2. List what type (actions or transformations) are the following Spark methods: `collect`, `take`, `filter`, `reduceByKey`, `union`.
   Answer:

3. What is a *stage* in the execution of a Spark program?
   Answer:

4. List the most important differences between RDDs and DataFrames.
   Answer:

Question 4 *(16 %)*

Here is a quiz about the Bigtable framework. Answer briefly:

1. How is the right tablet server found in the computing cluster, when a client wants to read a row from it?
   Answer:

2. Once the correct tablet server was located, how is the correct row data found?
   Answer:

3. Are `write` operations into a Bigtable fast? Why?
   Answer:

4. Assuming that all tablets in the system are at most 128 MB large and each row of metadata is about 1 KB, what's the *maximum data size* (in bytes) that a Bigtable in this system can contain? Explain how you arrived at your answer.
   Answer:

Question 5 *(16 %)*

Answer the following questions on Spark Streaming (D-Streams):

1. What types of cluster problems would Spark Streaming tolerate better (more reliably) than a Storm application?
   Answer:

2. Briefly, what is a D-Stream object?
   Answer:

3. Can you estimate how many D-Stream objects will be created by a program which implements a classic map-reduce logic, but over streamed data? Say, schematically, the program reads streaming data, and then does a simple `.map().reduceByKey()`.
   Answer:

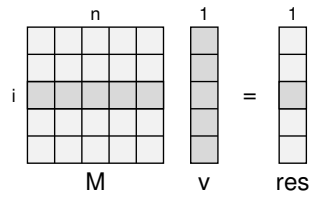
Question 6 *(16 %)*

At Google, the first application for MapReduce programming was computing the ranks of web pages – in other words, running the PageRank algorithm. This is simply a number of matrix-vector multiplications, but for big matrices.

Try to design a Spark program which takes:

- a big square matrix M of n rows and columns,
- a smaller vector (or column) v of n elements,

and computes the result $M \cdot v$, which is a new vector of n elements.



The matrix M comes from a (big) plain-text file on the Distributed File System, where a line contains an element M_{ij} in the form `i j element`. The vector v comes from another similar (smaller) file with lines of the form `i element`. The result should also be saved into a file. You are free to change these file formats as you like, if it helps your program design.

Try to solve the following two cases, in increasing order of difficulty, as far as you can. Points will be given for each case, separately:

1. The size n is small enough (say, 10 million or 10^7) so that you can consider vector v to be small data. M will be big data.
2. Now n is larger (say, 10 billion or 10^{10}), so even v is big data: it is too large to be loaded in memory on a server.

The solutions can be written in plain words, or PySpark code, or pseudocode. They will be assessed by their logic, not by whether you remember the PySpark API.

Answer:

Answer (continued):