Techniques for Large-Scale Data: Sample Exam Questions II

University of Gothenburg | Chalmers University of Technology
Department of Computer Science and Engineering
Period 4, 2020 (DIT873 (and re-exam in DIT871) / DAT346)
Instructors: Dr. Alexander Schliep (Examiner) with Dr. Graham Kemp

Information:

- This is a collection of sample problems reflecting online exams in 2020. A real exam would have about 60 points. This collection has 45 points in 6 questions.
- Grades for GU students (DIT873) are normally determined as follows: $\geq$ 70 % for grade VG; $\geq$ 40 % for grade G. Grades for Chalmers students (DAT346) are normally determined as follows: $\geq$ 80 % for grade 5; $\geq$ 60 % for grade 4; $\geq$ 40 % for grade 3.

Instructions:

- Begin the answer to each question on a new page. Write page number and question number on **every** page.
- Write clearly; unreadable = wrong!
- Fewer points are given for unnecessarily complicated solutions.
- Indicate clearly if you make any assumptions that are not given explicitly in the question.
- Show **ALL** your work. You will get little or no credit for an unexplained answer. Please indicate why a specific computation or transformation is appropriate. The points of each question appear in parentheses; use this for guiding your time.
- There is no need to compute numerical answers; you may leave Binomials, factorials and fractions, should they arise, as is.

Question 1 [*5 points total*]

A media article discusses data gathered about public transport usage in London:

> Transport for London (TfL) now has a bird's eye view of the estimated 4.1 million journeys taken on its network each day.
> It knows where people get on and off and it can start to see patterns in the data — for instance, someone who uses the system during the day but not at peak times is likely to be a student or a retired person, someone who has one day a week when they don't use the network may work from home that day, someone who takes a brief diversion along their usual route may be dropping off a child at nursery.
> "The data can be used to inform future expansion, whether we need to add a bus route or increase the frequency of trains, to alleviate capacity issues by informing people about the most crowded times and places and generally helps us to understand customers better," said . . . a data scientist at TfL.

Discuss whether the data gathered by Transport for London raises potential ethical issues.

Solution:

Good solutions identify several stakeholders (Transport for London, and customers are obvious ones mentioned in the question; others might include society in general, businesses in London, subdivisions of customers, etc.), list some examples of possible benefits and harm (if any) for each, then argue whether any harm seems sufficiently serious to give concern about implementing the system.

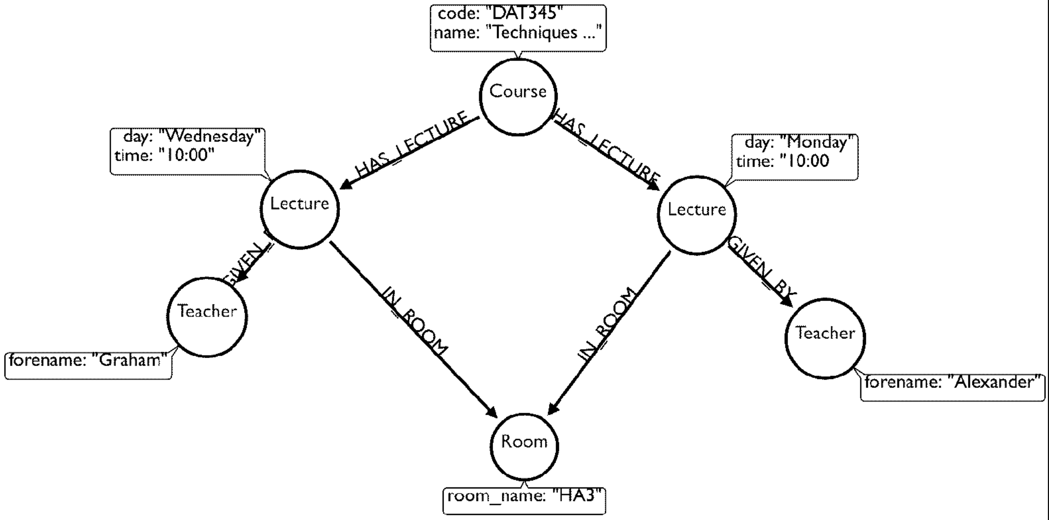
Question 2 [*13 points total*]

The following CYPHER statement was generated using the "Arrow" tool.

```
CREATE
  (‘0‘ :Course {code:"DAT345",name:"Techniques for large-scale data"}) ,
  (‘1‘ :Lecture {day:"Wednesday",time:"10:00"}) ,
  (‘2‘ :Room {room_name:"HA3"}) ,
  (‘3‘ :Teacher {forename:"Graham"}) ,
  (‘4‘ :Lecture {day:"Monday",time:"10:00}) ,
  (‘5‘ :Teacher {forename:"Alexander"}) ,
  (‘0‘)-[:‘HAS_LECTURE‘ ]->(‘1‘),
  (‘1‘)-[:‘IN_ROOM‘ ]->(‘2‘),
  (‘1‘)-[:‘GIVEN_BY‘ ]->(‘3‘),
  (‘0‘)-[:‘HAS_LECTURE‘ ]->(‘4‘),
  (‘4‘)-[:‘IN_ROOM‘ ]->(‘2‘),
  (‘4‘)-[:‘GIVEN_BY‘ ]->(‘5‘)
```

(a) [*3 pts*] Sketch how this information would appear in the "Arrow" tool.

Solution:



(b) [*4 pts*] Suppose we want to store the same information as triples in a Semantic Web database. What triples would we have?

Solution:

```
:r1     rdf:type       :Room ;
        :room_name     "HA3" .

:t1     rdf:type       :Teacher ;
        :forename      "Graham" .

:t2     rdf:type       :Teacher ;
        :forename      "Alexander" .

:l1     rdf:type       :Lecture ;
        :day           "Wednesday" ;
        :time          "10:00" ;
        :given_by      :t1 ;
        :in_room       :r1 .

:l1     rdf:type       :Lecture ;
        :day           "Monday" ;
        :time          "10:00" ;
        :given_by      :t2 ;
        :in_room       :r1 .

:c1     rdf:type       :Course ;
        :code          "DAT345" ;
        :name          "Techniques for large-scale data" ;
        :has_lecture   :l1 , :l2 .
```

(c) [*3 pts*] Give RDFS or OWL 2 statements that model the metadata describing the *relationships* in this example.

Solution:

Assuming class declarations:

```
:Course   rdf:type  owl:Class .
:Lecture  rdf:type  owl:Class .
:Room     rdf:type  owl:Class .
:Teacher  rdf:type  owl:Class .
```

Relationships are:

```
:given_by rdf:type owl:ObjectProperty ;
          rdfs:domain :Lecture ;
          rdfs:range :Teacher .

:has_lecture rdf:type owl:ObjectProperty ;
             rdfs:domain :Course ;
             rdfs:range :Lecture .

:in_room rdf:type owl:ObjectProperty ;
         rdfs:domain :Lecture ;
         rdfs:range :Room .
```

(d) [*3 pts*] Based on the triples in your answer to part (b), and assuming many other similar triples, write a SPARQL query that finds the forenames of teachers who give a lecture in room HA3 on Wednesday.

Solution:

```
SELECT ?f
WHERE {
    ?lecture :day            "Wednesday ;
             :given_by       ?teacher ;
             :in_room        ?room .
    ?room    :room_name      "HA3" .
    ?teacher :forename       ?f .
}
```

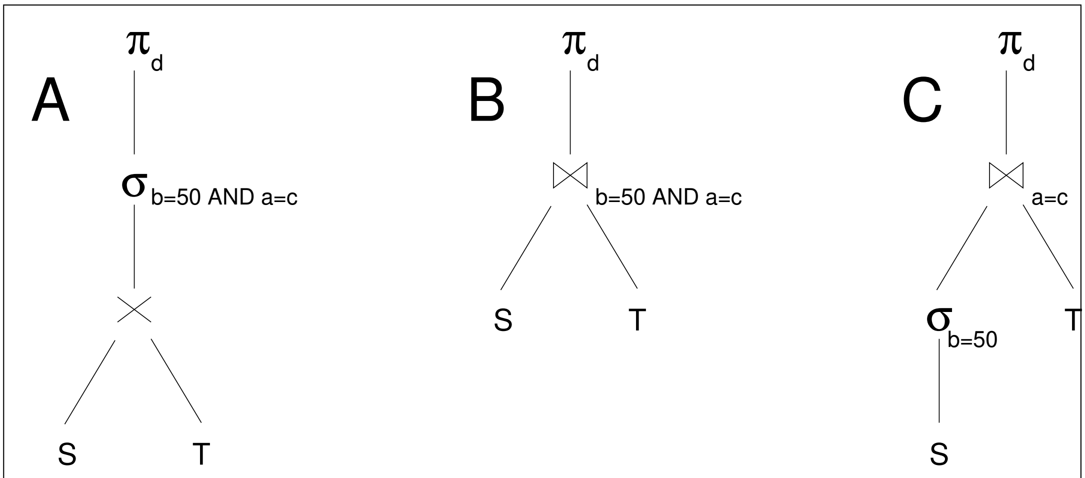
Question 3 [*6 points total*]

(a) [*6 pts*] Suppose we have relations $S(a, b)$ and $T(c, d)$ and query:

```
SELECT d
FROM   S, T
WHERE  b=50 and a=c
```

Draw *three* alternative logical query plans for this query.
Discuss the efficiency of the logical query plans.

Solution:



A: Inefficient; Carteasian product could be very large, then no possibility to use indexes when performing selection.

B: Much better than A.

C: Performing selection early on S can result in a very small intermediate relation, making join operation more efficient.

(Whether B or C is most efficient will depend on size of relations, approximate frequency of different values for an attribute, availability of indexes, layout of data on disk.)

Question 4 [*5 points total*]

(a) [*5 pts*] Suppose a relational database contains relation *Employees(empId, name, salary, dept, city)*, that this relation is stored in 30 disc blocks, and that each city has employee records stored (on average) in 5 disc blocks.

Suppose that two kinds of task are performed on this relation:

- task 1: inserting a new row;
- task 2: finding the employee identifiers for a given *city*.

For each of these tasks, state how many disc block transfers will be needed if:

i) there are no indexes;

ii) there is an index on *city* (assume that this index fits into a single disc block).

Suppose that 90% of operations performed on this relation are task 1 (inserting new rows) and 10% are task 2 (finding the employee identifiers for given cities).

iii) Discuss whether it would be better to have an index on *city*, or to have no indexes.

Solution:

i) task 1: 2
task 2: 30

ii) task 1: 4
task 2: 6

iii) It would be better to have an index on $city$ (cost: 420 vs. 480).

Question 5 [*8 points total*]

During the course we learned about four main paradigms for writing parallel code for computations: multi-threaded programming (MT), message passing (MP), map-reduce (MR), and Spark (SP) as an example of cluster computing.

Consider the task of computing histograms and summary statistics (mean, standard deviation, min, max; not median) for a given array of numbers read from a file.

Sketch (either graphically, with bullet points, or commented pseudo-code) and explain how the communication between parallel tasks is handled in two different out of the four frameworks (you may choose which two) using the task of computing histograms and summary statistics for univariate observations as an example.

You do not have to explain File I/O. Do explain the data flow and how the execution progresses from an array of numbers (MT, MP) respectively tuples (MR, SP) to the final result of a histogram and summary statistics.

Solution: Consider MT and MR. In MP an efficient code to solve the problem, will split the data into a number of parts equal to the number of threads (i.e. a scatter operation). Each thread will independently compute min, max, n, sum of values and sum of squares of values for its part of the data. Equally for a vector of bin counts for the histogram, which can be computed in one pass if histogram boundaries are fixed. Here is is important that each thread computes the partial statistics in thread-local variables independently to avoid locks on shared global variables. After completion, the partial statistics are returned to the manager thread (a gather operation) and the partial counts (one per thread) are combined to yield the final output.

In MR, assuming global histogram boundaries, the fundamental idea is to compute partial statistics per bin of the histogram. Mapper yields values as (key, value)-pairs where the bin index (which can be efficiently computed as floor of (value - min) divided by bin width) is the key. The combiner aggregates such tuples and yields partial (all values on the same node in one bin) per bin information (key, n, sum, sum of squares, min, max) for the reducer. Reducer produces complete per-bin statistics, e.g. ("stats", bin, n, sum, sum of squares, min, max) for a final reducer which outputs the bin counts and dataset statistics.

Question 6 [*8 points total*]

As a junior data scientist recently hired to Reynholm Industries you are tasked with overhauling and redesigning the IT infrastructure and accelerating existing computational workloads. The COO’s team runs daily analysis (DA) on orders, shopping baskets, online advertising, and website visits. The total amount of data is about 200GB per day collected in Reynholm Industries’ data center. The output are summary statistics (histograms, averages, etc.) and nice plots enabling monitoring the state of affairs.

It is desirable to collect historical data, but the size of the data makes it impossible to keep all the data in one database on one machine.

Before your arrival the IT people split up the database over fifty machines and demand keeps growing. The way the data is split up, it is necessary to query all the databases if one wants to summarize data, say compute the total of sales, for one specific customer over Reynholm Industries' lifetime. The querying is done in parallel on the fifty machines, so it is reasonably time-efficient for one query. However, as all the servers are busy for each query, this approach is neither energy-efficient nor time-efficient if many queries are run concurrently. Obviously, for customers who rarely shop at Reynholm Industries queries for sales to that customer would return no result on most of the fifty servers which hold the historical sales data in a distributed manner.

(a) [*4 pts*] How do you impress your new colleagues by reducing the total computational effort needed for computing the total of sales for one specific customer?

Solution: One simple solution is to introduce Bloom-filters (BF), one per server, which have the customer ID inserted into them, if there are sales records for that customer on that server. The BFs can be computed in parallel and easily transmitted and stored in memory on a manager node for queries. Queries aggregating sales per customer will hit the BFs and only query servers where the BF indicates relevant data. The false positive rate of the BFs has no ill effect as it only causes an avoidable computation.

Computing the, likely small, BF can be done in parallel based on all the customer IDs in a table.

More ambitious solutions might make use of other index data structures; then the analysis of costs and benefits might be more difficult.

(b) [*2 pts*] To assure Reynholm Industries' competitiveness, management suggests to investigate deployment of machine learning models trained with the large amount of customer data. Propose an appropriate hard- and parallel software environment supporting such advanced computations and argue which specific aspects influence your decision.

Solution: Generally, machine learning models require iterative training (cf. lecture slides comparing Spark and Map-Reduce), which benefits from keeping data in RAM. While machine learning models on moderate amounts of data can be effectively parallelized using multi-threading programming, large data sets can easily be too large for the total amount of RAM available on one SMP-computer (typical: up to 1-2TB RAM). In this case, a compute cluster consisting of high memory nodes (where the total amount of RAM across the cluster can easily exceeds tens of TB even with moderate amounts, e.g. 128GB, per node) with local disks is an appropriate platform for iterative computations using SPARK as a software platform. Such a setup also increases disk read throughput by parallelizing IO if the data is distributed over the cluster nodes and each node just has to read a part.

(c) [*2 pts*] Your boss wants to accelerate computations for one specific analysis task. The two options are (N) to buy more of the existing type of cluster nodes (e.g., Intel-based SMP general purpose computers with 128GB RAM and disks) or (H) buy tensor-unit hardware accelerators, which can execute specific types of operations very rapidly. Describe your decision process including the variables you need to determine.

Solution: The fundamental question is how much can the code be parallelized, respectively accelerated, and what is the cost of reaching a specific total speedup in the two scenarios. Through measurements the proportions amenable to improvements have to be established for each scenario (note: it is unlikely a tensor unit or additional cores will be able to accelerate the same parts of the code). Amdahl's law then indicates both a limit to the total speedup and upper bounds on the speedup for specific number of cores, respectively higher throughput with the accelerators. This allows to compute acceleration at a specific price point.