Techniques for Large-Scale Data: Sample Exam Questions I

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 44 points in 5 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 [*18 points total*]

Suppose we have the following relations:

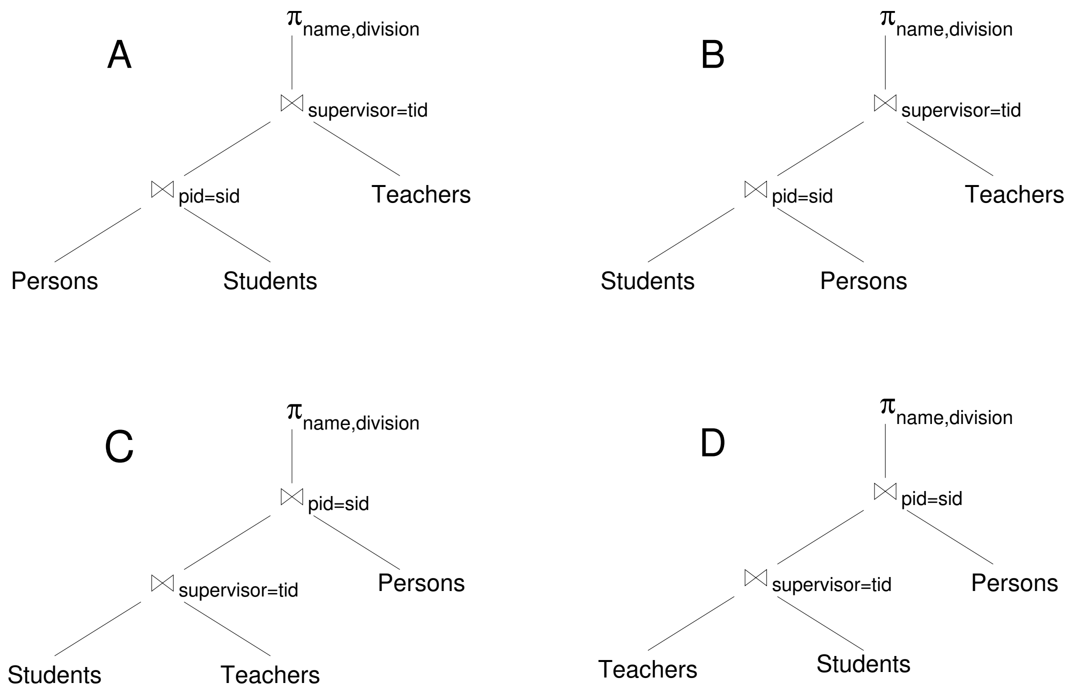
$Persons(\underline{pid}, name)$

$Teachers(\underline{tid}, division)$
$tid \rightarrow Persons.pid$

$Students(\underline{sid}, supervisor)$
$sid \rightarrow Persons.pid$
$supervisor \rightarrow Teachers.tid$

(a) [*6 pts*] A, B, C and D are four alternative logical query plans for the following SQL query:

```
SELECT   name, division
FROM     Persons, Students, Teachers
WHERE    pid=sid AND supervisor=tid
```



Assume that there are only indexes on the key attributes, and that relations Students and Teachers have approximately the same number of rows.

Discuss the efficiency of the logical query plans.

Solution:

- A is slow since we have to iterate over every Person in order to figure out which of the persons are students (but the lookup of each student is fast since *pid* is indexed).

- B and C are roughly equal in performance, we are iterating over Students and can look up persons and teachers by the indexed attributes *pid* and *tid*.
- D is far slower than the other plans, since we can't easily go from a teacher to the corresponding students (since supervisor isn't indexed).

(b) [*4 pts*] Suppose we want to model these relations using Semantic Web technology. Give RDFS or OWL 2 statements that model the metadata. Students and teachers should be modelled as subclasses of persons.

Solution:

```
:Person         rdf:type        owl:Class .

:hasName        rdfs:domain     :Person ;
                rdfs:range      xsd:string .

:hasPid         rdfs:domain     :Person ;
                rdfs:range      xsd:string .

:Teacher        rdfs:subClassOf :Person .

:hasDivision    rdfs:domain     :Teacher ;
                rdfs:range      xsd:string .

:Students       rdfs:subClassOf :Person .

:hasSupervisor  rdfs:domain     :Student ;
                rdfs:range      :Teacher .
```

Alternative syntaxes were accepted.

(c) [*2 pts*] Based on the metadata in your answer to part (b), what RDF triples would be needed to implement the following data:

Persons

pid	name
p1	Smith
p2	Jones

Students

sid	supervisor
p1	p2

Teachers

tid	division
p2	DS

Solution:

```
:p1     rdf:type      :Person ;
        rdf:type      :Student ;
        :hasPid       "p1" ;
```

```
            :hasName       "Smith" ;
            :hasSupervisor "p2" .

    :p2     rdf:type      :Person ;
            rdf:type      :Teacher ;
            :hasPid       "p2" ;
            :hasName      "Jones" ;
            :hasDivision "DS" .
```

We also allowed using :Smith and :Jones as identifiers, but note that this requires that the names are unique.

(d) [*3 pts*] Suppose that the triples in your answer to part (c) are stored in a Semantic Web database. Write a SPARQL query that computes the same result as the SQL query in part (a).

Solution:

```
SELECT ?name ?division
WHERE {
    ?student :hasName       ?name ;
             :hasSupervisor ?tid .
    ?teacher :hasPid        ?tid ;
             :hasDivision   ?division .
}
```

(e) [*3 pts*] Suppose that the triples in your answer to part (c) — and many other triples — are stored in a single relational database table with three columns. Translate the SPARQL query in your answer to part (d) to an SQL query that accesses this table to find the answer.

Solution:

Assuming the table is named Triples with (subject, predicate, object):

```
SELECT  Person.object AS name,
        Teacher.object AS division
FROM    Triples AS Person,
        Triples AS Teacher,
        Triples AS Student
WHERE   Person.predicate = "hasName"
        AND Person.subject = Student.subject
        AND Student.predicate = "hasSupervisor"
        AND Student.object = Teacher.subject
        AND Teacher.predicate = "hasDivision";
```

Question 2 [*4 points total*]

(a) [*4 pts*] Explain the main difference between document databases and key-value stores. In what circumstances are a document database and a key-value store essentially the same?

> **Solution:**
>
> The main difference is that documents are transparent while the values in a key value store are opaque. Thus we can "query into" the ducument structure, retrieve parts of a document, update parts of a document, and search based on fields of a document having particular values.
>
> Some key-value stores allow metadata sabout the value, (allowing indexing.
>
> Documents in a document database often have an ID field and we can ask "give me the thing with that ID"; the ID here is effectively the same as a key in a key-value store.

Question 3 [*3 points total*]

Suppose you are part of a university's IT team and have been asked to help improve campus safety by implementing automatic number plate recognition technology to monitor vehicles on campus.

Discuss whether there are ethical issues that need to be considered.

> **Solution:**
>
> Good solutions identify several stakeholders (e.g. the university, students driving on campus, others driving on campus, 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 4 [*11 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 computational resources are mostly used by the following workloads for three departments at Reynholm Industries.

- The CEO has a team running a software to predict customer demand for the next month. The software, combining partial differential equations and Bayesian statistics in a large-scale, parallel Monte Carlo (MC) simulation running about two days, takes a few historical summary statistics and key economy indicators and outputs predicted demand and a confidence interval for the prediction.
- 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.

- The marketing department runs advanced analytics (AA) aggregating primary data as used in the daily analysis over the span of several weeks to identify groups of customers and groups of website visitors. It also wants to identify factors which can help to predict when a website visitor becomes a customer. The output are models of groups and classifiers including features relevant to classification.

Provide a brief argument indicating key factors only when answering the following questions:

(a) [*3 pts*] If one were to reimplement software for the three workloads, which of the four paradigms for parallel computation we encountered—multi-threaded programming (MT), message passing (MP), map-reduce (MR), and Spark (SP) as an example of cluster computing—would you choose for the MC, DA, and AA workloads respectively.

Solution:

- MC can in principle be implemented in any of the four frameworks as the communication only occurs in the beginning and at the end of simulation run and only small amounts of data have to be transferred. MT or MP—depending whether all cores on one machine suffice or several machines are needed—is a natural answer, but MR and SP are certainly possible.
- DA is a data-intensive job. However, aggregating and computing summary statistics is sufficiently simple (and typically a one-pass operation) so that MR would be a natural model. As every MR job can also be expressed in SP, the latter would also be a legitimate answer. Note that SP would be unlikely to yield faster running times as caching in memory is irrelevant for one-pass jobs.

  MT/MP would be wrong due to handling big data.
- The likely iterative nature of the AA computations and the demands for data suggests SP as the natural framework.

  MT/MP would be wrong due to handling big data.

(b) [*3 pts*] If one were to acquire new hardware either in Reynholm Industries's data center or the cloud, which system architectures—distributed system with centralized disk storage (HPC), distributed system with local disk (DS), or a distributed system with local disks and large main memory (LM)—would be most appropriate for the workloads MC, DA, and AA respectively.

Solution:

- MC uses only small amounts of data so HPC is a natural answer. It would not benefit from DS, LM, but also run well on such system given equal number of cores and core speed.
- DA is not well-suited for HPC as bandwidth to the storage system is

a likely bottleneck in contrast to the parallel IO-speed on multiple DS nodes and total amount of storage.

- The likely iterative nature of the AA computations suggests LM as the natural platform. Iterative computations benefit from caching data in RAM.

(c) [*5 pts*] Describe the questions you would ask yourself and the formal analysis you would perform when deciding whether moving the workload DA to the cloud would result in a desired speedup of the computation. Use variables for all relevant quantities.

Solution: The questions to ask are:

- Can increasing the number of nodes accelerate the computation to a desired speedup? In other words: is a sufficient proportion amenable to parallelization?
- Is data collected locally or can it be collected in the cloud?
- In how far are computational improvements negated by upload speeds to the cloud?
- Measure f, the proportion of a program which can be accelerated by parallelization.
- Use Amdahl's law to compute the total speedup s_{total} and its limit as the number of cores d goes to infinity.
- For computations in the cloud using locally collected data the transfer time have to be included. Let t_u be the time for upload and t_r the running time, then $t = t_u + t_r$ is the total time and $f' = \frac{f\,t_r}{t_u + t_r}$ is the proportion amenable to speedups, which can be analyzed with Amdahl's law.

  Note that there are other ways to formalize it. Partial credit for insight that upload speed bound computational speedup.

Question 5 [*8 points total*]

Bloom filters are data structures for set membership queries. The two main operations for standard Bloom filters are `insert(x)` and `query(x)` which insert an item x into a Bloom filter, respectively answer whether an item x is present in a Bloom filter.

(a) [*8 pts*] Sketch an efficient parallel program for inserting all n items into a Bloom filter of size 2GB (two gigabytes). Assume that the nodes in your compute clusters are networked with very fast network connections, have at least 16GB RAM, and that the n items are stored in a distributed manner (e.g., in HDFS) on local disks of your cluster nodes. Your program should return the Bloom filter with all items inserted in one node (i.e, as if one would insert all items serially). *Hint: collecting all items in one node and inserting sequentially would not be efficient.*

Solution: The insertion of an item x can be viewed as computing the bitwise OR between the current Bloom filter of size b and an empty BF of equal size with only x inserted; i.e., containing a one at the k bits specified by $h_1(x), ..., h_k(x)$ and zero elsewhere. Note, bitwise OR is associative and commutative.

Consequently, partitioning items $x_1,...,x_n$ into d sets, inserting each of the d sets into their own Bloom filter, and computing the bitwise-OR of the d bloom filters yields the same result as inserting all items into one Bloom filter.

Hence, one can insert all items stored locally in a node in a BF and compute the OR of all Bloom filters with a reduce operation, which causes $d-1$ transfers of size 2GB and $d-1$ OR operations.

Extra points for observing that bitwise OR on contiguous memory has very good cache behavior and can be performed with 512bit operations on modern CPUs.