CHAPTER 10

Big Data

Practice Exercises

10.1 Suppose you need to store a very large number of small files, each of size say 2 kilobytes. If your choice is between a distributed file system and a distributed key-value store, which would you prefer, and explain why.

Answer:
The key-value store, since the distributed file system is designed to store a moderate number of large files. With each file block being multiple megabytes, kilobyte-sized files would result in a lot of wasted space in each block and poor storage performance.

10.2 Suppose you need to store data for a very large number of students in a distributed document store such as MongoDB. Suppose also that the data for each student correspond to the data in the *student* and the *takes* relations. How would you represent the above data about students, ensuring that all the data for a particular student can be accessed efficiently? Give an example of the data representation for one student.

Answer:
We would store the student data as a JSON object, with the takes tuples for the student stored as a JSON array of objects, each object corresponding to a single takes tuple. Give example ...

10.3 Suppose you wish to store utility bills for a large number of users, where each bill is identified by a customer ID and a date. How would you store the bills in a key-value store that supports range queries, if queries request the bills of a specified customer for a specified date range.

Answer:
Create a key by concatenating the customer ID and date (with date represented in the form year/month/date, e.g., 2018/02/28) and store the records indexed on this key. Now the required records can be retrieved by a range query.

10.4 Give pseudocode for computing a join $r \bowtie_{r.A=s.A} s$ using a single MapReduce step, assuming that the map() function is invoked on each tuple of r and s. Assume that the map() function can find the name of the relation using context.relname().

Answer:
With the map function, output records from both the input relations, using the join attribute value as the reduce key. The reduce function gets records from both relations with matching join attribute values and outputs all matching pairs.

10.5 What is the conceptual problem with the following snippet of Apache Spark code meant to work on very large data. Note that the collect() function returns a Java collection, and Java collections (from Java 8 onwards) support map and reduce functions.

```
JavaRDD<String< lines = sc.textFile("logDirectory");
int totalLength = lines.collect().map(s -> s.length())
                    .reduce(0,(a,b) -> a+b);
```

Answer:
The problem with the code is that the collect() function gathers the RDD data at a single node, and the map and reduce functions are then executed on that single node, not in parallel as intended.

10.6 Apache Spark:

a. How does Apache Spark perform computations in parallel?

b. Explain the statement: "Apache Spark performs transformations on RDDs in a lazy manner."

c. What are some of the benefits of lazy evaluation of operations in Apache Spark?

Answer:

a. RDDs are stored partitioned across multiple nodes. Each of the transformation operations on an RDD are executed in parallel on multiple nodes.

b. Transformations are not executed immediately but postponed until the result is required for functions such as collect( ) or saveAsTextFile( ).

c. The operations are organized into a tree, and query optimization can be applied to the tree to speed up computation. Also, answers can be pipelined from one operation to another, without being written to disk, to reduce time overheads of disk storage.

10.7 Given a collection of documents, for each word w_i, let n_i denote the number of times the word occurs in the collection. Let N be the total number of word occurrences across all documents. Next, consider all pairs of consecutive words (w_i, w_j) in the document; let $n_{i,j}$ denote the number of occurrences of the word pair (w_i, w_j) across all documents.

Write an Apache Spark program that, given a collection of documents in a directory, computes N, all pairs (w_i, n_i), and all pairs $((w_i, w_j), n_{i,j})$. Then output all word pairs such that $n_{i,j}/N \geq 10 * (n_i/N) * (n_j/N)$. These are word pairs that occur 10 times or more as frequently as they would be expected to occur if the two words occurred independently of each other.

You will find the join operation on RDDs useful for the last step, to bring related counts together. For simplicity, do not bother about word pairs that cross lines. Also assume for simplicity that words only occur in lowercase and that there are no punctuation marks.

Answer:
FILL IN ANSWER (available with SS)

10.8 Consider the following query using the tumbling window operator:

select *item*, *System.Timestamp* **as** *window_end*, **sum**(*amount*)
from *order* **timestamp by** *datetime*
group by *itemid*, **tumblingwindow**(*hour*, 1)

Give an equivalent query using normal SQL constructs, without using the tumbling window operator. You can assume that the timestamp can be converted to an integer value that represents the number of seconds elapsed since (say) midnight, January 1, 1970, using the function *to_seconds*(*timestamp*). You can also assume that the usual arithmetic functions are available, along with the function *floor*(a) which returns the largest integer $\leq a$.

Answer:
Divide by 3600, and take floor, group by that. To output the timestamp of the window end, add 1 to hour and multiply by 3600

10.9 Suppose you wish to model the university schema as a graph. For each of the following relations, explain whether the relation would be modeled as a node or as an edge:

(i) *student*, (ii) *instructor*, (iii) *course*, (iv) *section*, (v) *takes*, (vi) *teaches*.

Does the model capture connections between sections and courses?

Answer:
Each relation corresponding to an entity (student, instructor, course, and section) would be modeled as a node. *Takes* and *teaches* would be modeled as edges. There is a further edge between *course* and *section*, which has been

merged into the *section* relation and cannot be captured with the above schema. It can be modeled if we create a separate relation that links sections to courses.