CPRE/SE 419 Software Tools for Large Scale Data Analytics

Spring 2022

Quiz #1

Yuichi Hamamoto

(due: Tuesday, March 22, at 11:59PM)

Preamble:

As explained in discussions/preparation tips, your quiz has three categories of questions – or a total of 7 questions, split in three parts. As discussed,

- The first category consists of quick-answer-problems; call them "warm up problems".
- The second category consists of "concept-style" problems, with an intent to demonstrate an understanding (and capability of) explaining/using certain concepts and abstractions; call them – "first-round interview" questions.
- The third category consists of problems for which you need to provide a pseudo-code for the solution (and whatever else may be explicitly requested in the problem statement).

The points total to 105 (5 extra credits, randomly scattered).
Please make sure to read each problem carefully, and please make++ sure++ that you explicitly state any assumption that you deem needed when providing the solution(s).

NOTE: you can work in teams of two students for this assignment. You are required to upload the solutions on Canvas – and, in case that the work was done by a team, make sure that both names are printed at the header of the file. We expect typed solution – and if there are cases where handwriting is deemed necessary (e.g., a formula), there is the requirement of legibility.

Part I – Provide brief/concise answer:

1. (6 pts.) Can you modify (i.e., update/insert/delete data from) an existing file in HDFS? Briefly justify your answer.

   We cannot modify an existing file in HDFS because it only allows us to write. However, we can create copy of the existing file and modify it, then store it as an updated version.

2. (7 pts.) Although physically distributed in blocks, when processing an MR job, the HDFS files are divided in (logical) Splits. What is it that determines the number of Splits?

   The input path and the input size for the job determines the number of Splits in the block.

3. (7 pts.) What are the types of long-running daemons in Yarn?

   Resource manager which manages the use of resources across cluster and node manager which is running on the nodes in the cluster to lunch and monitor container.

4. (7 pts.) Describe briefly the notion of RDDs in Spark.

   RDD is a dataset which is distributed across multiple nodes and survives in node failure. In short, it is a partition collection of records.

Part II

1. (12 pts.) It is well known that *if the probability of a disk failure (within a given time-frame) is p, then the probability of a disk failing from among a collection of M of them is surely greater than p*. Then, why is it that Hadoop uses multiple nodes to store the data/files?

   Let P_1 be the possibility which at least,one of disk fails in M within a given timeframe, then
   $P_1 = 1-(1-p)^M$

   Assume the data is duplicated and stored in M nodes to backup. Let P_2 be the possibility which we lose the data (all the disk fail), then
   $P_2 = p^M$

   Hadoop does not care P_1 but P_2 because even if one of disk fails, it will not lose the data and P_2 is every small. That is why Hadoop uses multiple nodes to store the data/files.

2. (13 pts.) Suppose that a given file has k blocks, and each block is stored on a different machine (no replication – single copy of each block). Assume that the probability of a machine failure is p, and that failures of different machines are independent of each other. What is the probability that the file is lost due to machine failure?

   Possibility of all of them do not fail is $(1-p)^k$
   Therefore the possibility of at least one of machine fails is

   $1-(1-p)^k$

3. (11 pts) Explain the concept of a *Partitioner* (i.e., what is its use/purpose) and give an example of types of Partitioners readily available for MR jobs in HDFS.

   Partitioner is used to decide where to set partition so that each reducer can have fair amount of work to process efficiently. Hadoop has HashPartitioner, TotalOrderPartitioner, BinaryPartitioner, and KeyFieldBasedPartitioner. We used TotalOrderPartitioner to sort the data with the key.

Part III – Coding questions

1. (22 pts.) A *directed graph* G= (V,E) consists of a set of vertices V, and a set of edges E such that each element e in E is an ordered pair (u,v), denoting an edge directed from u to v. In a directed graph, a *directed cycle* of length three is a triple of vertices (x,y,z) such that each of (x,y) (y,z) and (z,x) is an edge in E. Write a Mapreduce algorithm whose input is a directed graph presented as a list of edges (on a file in HDFS), and whose output is the list of all directed cycles of length three in G.

   Write the pseudocode for the mappers/reducers methods. Also, assuming that there are M mappers, R reducers, m edges and n vertices -- analyze the (upper-bound of the) communication cost(s).

```
map(String key, String edge){
        v1, v2 = vertices in edge;
        emit(1, tuple<v1, v2>)
}

reduce(String id, Iterator values){
        count = 0;
        for each edge in values{
                e1 = edge
                for other edge in values{
                        e2 = edge
                        if(e1.v2 ==e2.v1){
                                for other edge in values{
                                        e3 = edge
                                        if(e2.v2 ==e3.v1){
                                                if(e3.v2 ==e1.v1){
                                                        count++;
                                                }
                                        }

                                }
                        }
                }
        }
        --divide it by 3 because it will not be able to distinguish the same triangles starts from different vertices
        count /= 3;
}
```

   Total map cost = O(m), per map cost = O(m/M), total reduce cost = $O(m^3)$, per reduce cost = $O(m^3/R)$, communication cost = O(m)

2. (20 pts.) Recall the problem from your homework: Assume that you are given a large file named *sales_data.txt* in which the lines are of the format: *(store, item, total_sales)*. For this problem, you are expected to write a *PigLatin code* that will generate the *"trimmed-average"* of the sales per store (for the items stored by a particular store). The fundamental difference is that the trimmed-average should exclude the tuples for which the item sold is *milk* or *beer*. In other words, you need the average of the sales per store for all the other items, excluding the sales for milk and for beer.

   data = LOAD '*sales_data.txt*' USING PigStorage(', ') AS (store: chararray, item: chararray, total_sales: int);

   temp = FILTER data BY NOT (item MATCHES '.*milk.*' AND item MATCHES '.*beer.*');
   group = GROUP temp BY store;

   output = FOREACH group GENERATE group, SUM(total_sales)/COUNT(item) AS trimmed_average;

   STORE ordered INTO '/output' USING PigStorage('\t');