CS60021: Scalable Data Mining

Practice Questions: Hadoop and Spark

State whether following statements are true or false with max 2-sentences of explanations:

1. Join operation in Spark always induces ”fat” dependencies.

2. Total memory consumed by all mapper output records should fit into the combined main memory of all the computers in the network.

3. Once an RDD on which an action has been called, is materialized, the RDDs on which it depends can be de-materialized (memory unallocated).

4. HDFS replication improves both reading and writing speed for large files in hadoop applications.

5. HDFS is a best suited for fast storage of many small files in a big data cluster.

6. In Spark, Directed Acyclic Graph Scheduler gets initiated after the transformation operation on RDD.

7. For fault-tolerance, execution engine of Spark constructs a dependency graph, where each partition of each RDD is a node.

8. Backup tasks in Hadoop not only improve job completion time, but also improve fault-tolerance.

Long-answer questions:

Q1. What does the Hadoop MapReduce framework ensure in terms of in terms of mapper output records and reducer input records ? Explain how it is ensured by the shuffle process with a step by step description.

Q2. Write the formulae for calculating the class conditional probabilities $P(x_i|y)$ and class probabilities $P(y)$ for binary Naïve Bayes classification, where y is the class label and $x_i, i = 1, \dots, d$ are the Based on the above formulae, write a spark program for implementing Naïve Bayes classifier for a binary classification problem, given the following record format:
$<y, x_1, \dots, x_d>$

Q3. Write a Spark program that takes an RDD points as input, where each record is an image (256×256 matrix), and another image testpoint. The program outputs list of k-nearest images to testpoint. Note that here k is small and hence you can store a list k

nearest points as a local variable. You are not allowed to use the Spark built-in sort function, or any other built-in top-k functions.

Q4. State 2 design objectives of HDFS which make it different from other distributed file systems like NFS. Name the components of HDFS. Which component is responsible for initiating self-healing? Explain the self- healing process.

Q5. Write a program in Apache Spark which computes the histogram of a list of real numbers in a distributed manner, by first determining the range of numbers between min and max, and then splitting the range into the required number of equal sized bins. The program takes as input an RDD X containing a list of numbers, and an integer m denoting the number of number of bins for computation of a histogram. The program should output the counts of numbers of datapoints in each bin. At no point should you program store a O(length of list) many numbers in a local data structure.

Q6. Explain how a large number of mapper output records are sorted into the correct reducer bins, even though there may not be enough memory to hold all the records in a given machine. What is this process called. Is this possible even if the number of records is larger than the disk space in a given node ?

Q7. Consider the following program which prints the candidates for maximum and minimum from a given input file of numbers in mapreduce.

Mapper:

```
# Mapper.py

for line in sys.stdin:
    line.rstrip()
    value = line.split('\t')
    r = random.randint(1,5)
    print('min'+'##'+str(r)+'\t'+str(value)+'\n')
    print('max'+'##'+str(r)+'\t'+str(value)+'\n')
```

Reducer:

```
# Reducer.py

currmin = -99999999
currmax = 99999999

for line in sys.stdin:
    line.rstrip()
    [key1, val] = line.split('\t')
    [s, r] = key1.split('##')
    if s == 'min' and currmin > val:
        currmin=val
    if s == 'max' and currmax < val:
        currmax=val

if currmin != -99999999:
    print('min'+currmin)
if currmax != 99999999:
    print('max'+currmax)
```

Given a file with input numbers from 1 to 1000,000, and number of reducers is 6, answer the following questions:

- How many reducer input key buckets will be there?
- Considering that the mapper input split size is 10000 records, how many mappers will be there?
- How many mapper output key buckets (files on the local disk) will be there?
- How many mapper output key buckets will be received by each reducer?
- On an average, how many mapper output records will be there in each mapper output key bucket?

Q8. Write a spark program in Scala / pseudocode for computing k nearest neighbors for each data point in an input dataset. Assume that the datapoints are given as records of an input RDD. Note that you cannot store O(n) datapoints in one record of any intermediate RDD, where n is the total number of datapoints. For example, for k = 2 the following input should have the given output.

Input:
(1,1)
(1,2)
(1,3)
(1,4)
(1,5)

Output:
(1,1), ((1,2),(1,3))
(1,2), ((1,1),(1,3))
(1,3), ((1,2),(1,4))
(1,4), ((1,3),(1,5))
(1,5), ((1,3),(1,4))