CME 323: Distributed Algorithms and Optimization

Instructor: Reza Zadeh (rezab@stanford.edu)
TA: Anuj Nagpal (anujnag@stanford.edu)
HW#3 – Due May 19
Total Score – 75 points

1. **(10 points) Intro to Spark** Download the following materials.

   - Slides
   - Spark and Data

   You don't need to install Spark from scratch but use the REPL prompt as described in slides. Make sure to install Java JDK 6/7.

   Now, answer the following questions.

   (a) Checkpoint on slide 11.
   (b) Checkpoint on slide 55.
   (c) Checkpoint on slide 60.
   Note: slide 59 references the file `CONTRIBUTING.md` which is not in the provided zip file. Instead, use the file `website/getting-started.md`

   Submit your code and answers.

2. **(10 points) Least Squares Fit:** Write a Spark program to find the *least squares fit* on the following 10 data points. The variable `y` is the response variable, and `X1`, `X2` are the independent variables.

```
              X1         X2           y
 [1,] -0.5529181 -0.5465480 0.009519836
 [2,] -0.5428579 -1.5623879 0.982464609
 [3,] -1.3038629  0.5715549 0.499441144
 [4,]  0.6564096  1.1806877 0.495705999
 [5,] -1.2061171  1.3430651 0.153477135
 [6,]  0.2938439 -1.7966043 0.914381381
 [7,] -0.2578953  0.2596407 0.815623895
 [8,]  0.9659582  2.3697927 0.320880634
 [9,] -0.4038109  0.9846071 0.488856619
[10,]  0.6029003 -0.3202214 0.380347546
```

   More precisely, find w_1, w_2, such that $\sum_{i=1}^{10}(w_1 X1_i + w_2 X2_i - y_i)^2$ is minimized. Report w_1, w_2, and the Root Mean Square Error and submit code in Spark. Analyze the resulting algorithm in terms of all-to-all, one-to-all, and all-to-one communication patterns.

3. **(8 points) Intro to Map Reduce** Assume you are given a typical MapReduce implementation where you only have to write the Map and Reduce functions. The Map function you will write takes as input a (key, value) record and returns either a (key, value) record or nothing. The Reduce function you will write takes as input (key, list of all values for that key) and returns either a record or nothing. The framework already takes care of iterating the Map function over all the records in the input file, key-based intermediate data transfer between Map and Reduce, and storing the returned value of Reduce. For all the following questions, provide algorithms at the level of pseudocode.

   (a) Given as set of records (for example, movie names and ranking), provide a MapReduce algorithm to output the top K movies of the set.

   (b) Suppose you are given an input file which contains comprehensive information about a social network that has asymmetrical (directed) links, i.e., a network where users follow other users but not necessarily vice-versa (e.g., Twitter). Each record in this input file is (userid-a, userid-b), where userid-a follows userid-b (i.e., points to it). Note that this record tells you nothing about whether or not userid-b follows userid-a. Write a MapReduce program (i.e., Map function and Reduce function) that outputs all pairs of userids who follow each other.

4. **(8 points) Product Inventory:** Consider the following product inventory table as an example:

Product Id	Supplier	Delivery Time	Price	Rating
1	Josh	4	30	4
2	Josh	1	40	4.5
3	Brian	2	10	3
4	Brian	2	10	5
5	Brian	3	20	4

   The actual table has a large number of entries and there are certain operations that you need to perform on it frequently. Provide a MapReduce algorithm with pseudocode for each of these operations:

   (a) UNIQUE: Find the distinct (or unique) suppliers in your inventory table.

   (b) SHUFFLE: Randomly re-order the records in your table.

   (c) RATING: Output a list of suppliers along with the average rating of products provided by the supplier.

5. **(8 points) Twitter Analytics:** You have a table that has some analytics data recorded for tweets on Twitter. An example table is shown below:

Id	Tweet	Date	Likes
14126574	Puppies are so cute!	04-22-2022	34
85631462	Murphy's Law also holds for PhD Defense	04-25-2022	42
36908221	RT Puppies are so cute!	04-25-2022	11
14126574	Puppies are so cute!	04-28-2022	18
79109305	RT Puppies are so cute!	04-28-2022	14
79109305	RT Puppies are so cute!	05-02-2022	5
48109305	Giving CME323 Midterm, wish me luck	05-02-2022	26

(a) If a tweet has more than 10000 likes and more than 99% of them came from a single month, we suspect that some user paid for those surge in likes. Provide a MapReduce algorithm with pseudocode to find such suspicious Tweet Ids.

(b) In Twitter, some status-messages are repeats of earlier status messages and are called 'Retweets'. These repeated messages in our table are preceded by "RT" followed by the original status message. Provide a MapReduce algorithm to output a list of Tweet messages that were retweeted along with the total number of likes their retweets received.

6. **(8 points) Connected Components with MapReduce** Finding out the number of connected components in a graph is a key subroutine in many graph algorithms. Provide and prove the correctness of a MapReduce algorithm to count the number of connected components in a graph (represented as an edge list).

7. **(7 points) Sampling from multiple streams** Suppose we have numerous substreams of data (say $S_1, \ldots, S_n$), provide and prove the correctness of an algorithm to generate k random samples from the aggregate stream.

8. **(6 points) Word Count Shuffle** Consider counting the number of occurrences of words in a collection of documents, where there are only k possible words. Write a MapReduce to achieve this, and analyze the shuffle size with and without combiners being used (assuming B mappers are used).

9. **(10 points) Prefix Sum** The *prefix-sum* operator takes an array $a_1, \ldots, a_n$ and returns an array $s_1, \ldots, s_n$ where $s_i = \sum_{j \leq i} a_j$. For example, starting with an array `[17 0 5 32]` it returns `[17 17 22 54]`. Describe (in detail) how to implement *prefix-sum* in MapReduce, where the input is stored as $\langle i, a_i \rangle$. That is, the key is the position in the array, and the value is the value at that position. Analyze the shuffle size and the reduce-key space and time complexity.