CMSC 724, Spring 2015: Homework 3
Due Friday March 13, 2015, 11:59pm.

The homework is to be done by yourself. You are free to make reasonable assumptions about the problem if something is not fully specified – make sure to state any assumptions. Try to be concise in your answers – most answers should be 1-2 paragraphs long. Total weight = 5%.

1. Consider a parallel database where the relation $R(A, B)$ is round-robin partitioned across a set of N machines. Our goal is to evaluate group-by aggregate queries of the type:

   ```
   select A, SUM(B)
   from R
   group by A;
   ```

   The result size (i.e., the number of distinct values of A) may range from small to very large (i.e., the result may be too large to fit in the memory of one machine). Suggest two approaches that would work well for the two scenarios and discuss the tradeoffs (in terms of communication cost and processing time). If you think a single appraoch would work well for both cases, that's fine – but you should provide a reasoning for the same.

2. Sketch a MapReduce program for calculating any *maximal matching* in an undirected graph. The simplest algorithm is a greedy algorithm that keeps on pairing up unmatched vertices till no two unmatched vertices exist that have an edge between them. The input data is provided in the adjacency list format: each input file contains the adjacency lists for a subset of the nodes one by one. For example, the input file below contains the adjacency lists (i.e., neighbors) of 1, 2, and so on.

   ```
   1: 2 3
   2: 1 3 10
   ...
   ```

   The final output should be pairs of matched nodes (e.g., (1, 2), (3, 4), ..).

3. List 3 of the limitations/criticisms of MapReduce that Stonebraker et al., point out, and Dean et al.'s response to them. Do those 3 limitations also apply to Spark (much of the discussion in Stonebraker et al., was specific to Hadoop MapReduce)? For the Spark question, you don't need to dig deeper into the implementation details if the answer is not obvious or more details are needed – answer to the extent you can.

4. Briefly explain how Spark does recovery through use of *lineage graphs*, including exactly how the lineage graphs are represented. What problems does this scheme have in the context of D-Streams computation model, and how does Spark Streaming solve it?

5. Briefly describe three sources of stragglers in Naiad, and how it deals with them. Do these issues also exist with Spark Streaming?