COSC 3365 – Distributed Databases Using Hadoop
Project 5 – MapReduce Combiner Application

In Project 4, your Mapper read data from the input file and generated output in a key-value pairs as <name, 1>. The output of all the Mappers was then transmitted over the network to the Reducer phase where a Partitioner generated keys and list of values and the Reducer added the list to get the word count.

The process worked successfully, but it was not optimized. The file size was very small. In real life scenario, the input files could be in gigabytes, terabytes, and even petabytes. The Mapper will generate millions of key-value pairs which will be transmitted over the network. This will cause network congestion and slows the performance of the job.

The solution to this problem in the MapReduce Framework is using Combiners. The primary goal of Combiners is to minimize the number of key-value pairs sent over the network. A Combiner class is used in between the Map class and the Reduce class to reduce the volume of data transferred between Map and Reduce.

In the WordCount application, the combiner receives N number of <word,1> pairs as input and outputs a single <word, N> pair.

For example, if an input processed by a Map task had 1,000 occurrences of the name "John", the Mapper will generate 1,000 <John,1> pairs, while the Combiner will generate one <John,1000> pair, thus reducing the amount of data needed to be transmitted to the Reduce tasks over the network. See **Figure 5-1** and **Figure 5-2**.

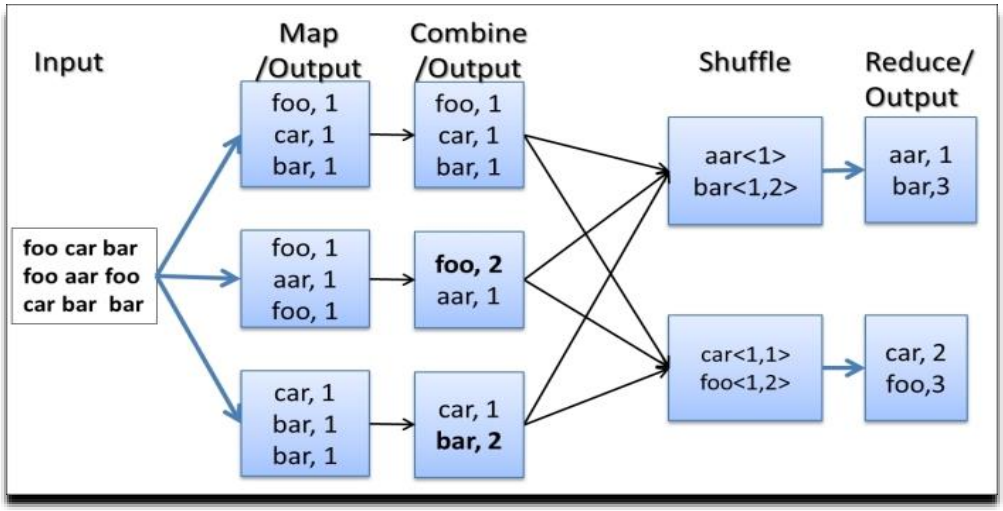

Figure 5-1

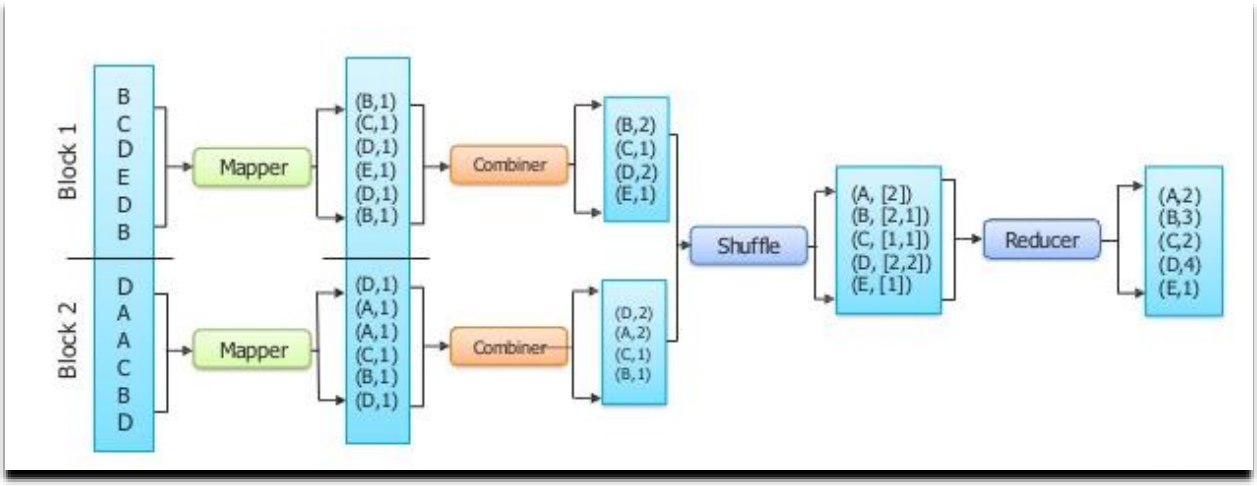

Figure 5-2

Sometimes the Combiner is referred to as Mini Reducer because the code used by both is very similar. The Combiner can run many times on the Mapper output. See ***Figure 5-3.***

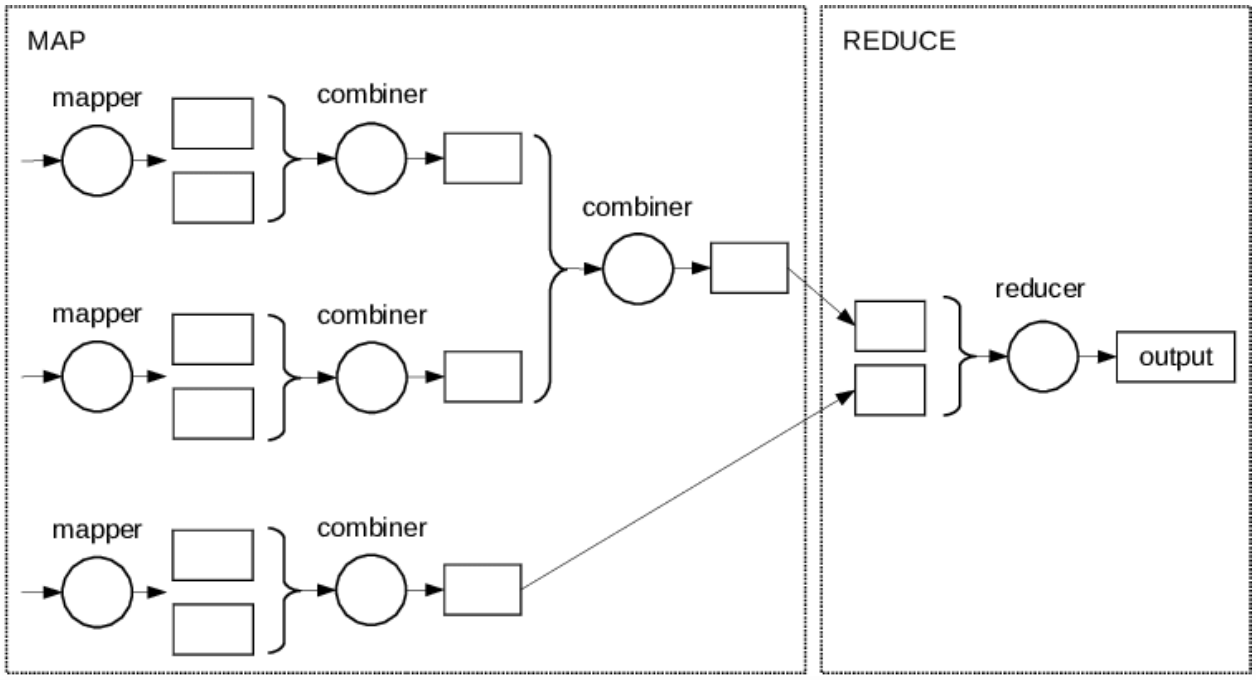

Figure 5-3

The Combiner does not work on very problem. It works only when the problem is commutative (A + B = B + A) and associative ((A + B) + C = A + (B + C)).

Rewrite Project 4 using a Combiner class. **All classes in this project must be public, non-static and not nested in other classes.**

Name the Driver class as Project5.java.

Every method in your program should be limited to performing a single, well-defined task, and the name of the method should express that task effectively. All methods should be non-static unless it is absolutely necessary for it to be static.

Compile the program into a .jar file for Hadoop.

Create a directory in HDFS and move the input text file, **Project5.txt**, there.
Take a screenshot of this process showing the HDFS command and save the image as JPG or PNG in a file named Project5-1.xxx (where "xxx" should indicate the file format).

Execute the Hadoop jar file.
Take a screenshot of this process showing the Hadoop command and save the image as JPG or PNG in a file named Project5-2.xxx (where "xxx" should indicate the file format).

Display the output file in HDFS and verify that it is correct.
Take a screenshot of this process showing the HDFS command and save the image as JPG or PNG in a file named Project5-3.xxx (where "xxx" should indicate the file format).

Take a screenshot of your output and save the image as JPG or PNG file named Project5-4.xxx (where "xxx" should indicate the file format).

Create a folder named, **YourFullName_Project5**. Copy your image files, the Java source codes, and the jar file to the folder. **Zip the folder, as a ".zip" file, and upload it to Blackboard**.

Before you upload your project to Blackboard:

- Ensure that your code conforms to the style expectations set out in class and briefly discussed below.

- Make sure your variable names and methods are descriptive and follow standard capitalization conventions.

- Put comments wherever necessary. Comments at the top of each module should include your name, file name, and a description of the module. Comments at the beginning of methods describe what the method does, what the parameters are, and what the return value is. Use comments elsewhere to help your reader follow the flow of your code.

- *Program readability and elegance are as important as correctness.* After you have written your method, read and re-read it to eliminate any redundant lines of code, and to make sure variables and methods names are intuitive and relevant.

Read the assignment very carefully to ensure that you have followed all instructions and satisfied all requirements. **You will not get full credit for this project if it is not written as instructed even if it works as expected.**