CSE 168: Lab 02: Distributed Data Analytics

Martin Urueta
LAB: Tuesday 7:30pm-10:20pm
Adam Weingram
October 11, 2022

Introduction

In this lab, we are learning how to use MapReduce using a program called Hadoop. The reason we would need to learn MapReduce is to be able to write scalable applications that can do parallel processing to pass a large amount of data. MapReduce has two essential functions; Mapper and Reducer. The Mapper is a function that is able to parcels out work to various nodes within the map. Then it will organize its results from the mapper and then the Reducer function reduces the result from each node into a cohesive answer. In this case we used Term Frequency to learn how MapReduce works.

Option 3: Term Frequency

In order to figure out how to get Term Frequency to work, first we have to know what Term Frequency is. Term Frequency is a way to know how frequently the word appears in that id line. For example if the word “my” is in an id line we would print out “ my {id, frequently}”. In order to solve this Term Frequently, we would use hadoop. In the mapper function, we would extract each id line from the txt file. For example the first row of that line would be “3,knitting is my hobby and my passion”. We then need to extract the id number of that line by converting the line from text to string and then split the string into two parts on an array. “3” and “knitting is my hobby and my passion” will be two parts in that array. Now we have the id to be “3” and the sentence of that line to be “knitting is my hobby and my passion”. Then we put our word as our key and the id as our value. We will emit (word, id) onto the reducer. In our example it will output “(knitting, 3), (is, 3), (my, 3), (hobby, 3), (and, 3), (my, 3), (passion, 3)”. From here the output will go through the shuffler and sort the same key to each of their own arrays. Our input would be “(knitting, [3]), (is, [3]), (my, [3, 3]), (hobby, [3]), (and, [3])”. Then the reduce would run 5 times simultaneously since there are 5 different keys. The way the reducer works is by pulling an input like (my, [3,3]). Then the reducer will make an array to hold the total number that the word appears in each id line. The index of the array is equal to # of time that word appears in that id line. The way we figure out the number of times that would appear is by adding 1 to that index of the array. Then after figuring out the total time the word appears on each id line, we would want to emmit (word, “index, #ofWordAppear”)

Our output from the reducer would be

```
knitting 3, 1
is       3, 1
my       3, 2
hobby    3, 1
and      3, 1
passion  3, 1
```

Figure 1

Expected output from the reducer example

In our test case we our input to the mapReducer in hadoop is

```
1,i love dogs
2,i hate dogs and knitting
3,knitting is my hobby and my passion
```

Figure 2

Input of our test case

We used the same logic as it was explained early, we got our output from the reducer to be

```
and      2, 1
and      3, 1
dogs     1, 1
dogs     2, 1
hate     2, 1
hobby    3, 1
i        1, 1
i        2, 1
is       3, 1
knitting         2, 1
knitting         3, 1
love     1, 1
my       3, 2
passion  3, 1
```

Figure 3

Output in alphabetical order

From our output we can see that the test case works and was able to print out the result that we are expected to print out but in a different order. Our output printed out in alphabetical order while the answer of the result printed out by id order. In the end we are still able to print the result out but in a different order. In our case order does not matter to us.

```
dogs 1, 1
i 1, 1
love 1, 1
and 2, 1
dogs 2, 1
hate 2, 1
i 2, 1
knitting 2, 1
and 3, 1
hobby 3, 1
is 3, 1
knitting 3, 1
my 3, 2
passion 3, 1
```

Figure 4

Output in id order