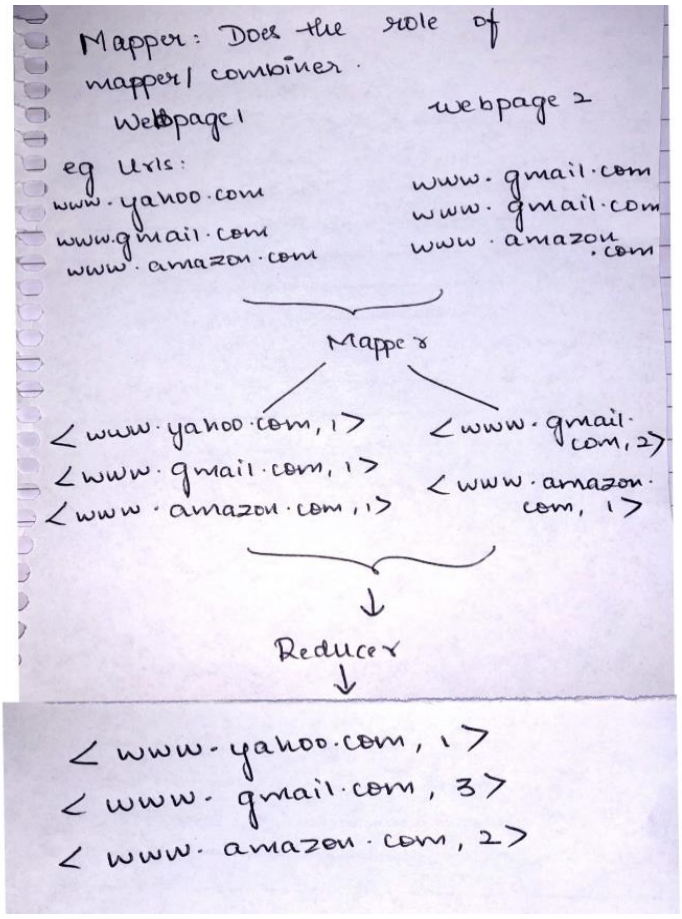
Data Center Scale Computing Lab2 – Modify wordcount to URL Count

Vandana Sridhar
SID: 109252783

Description:

The code works as follows: The UrlCount class contains 3 classes namely the LinkGetter class, the UrlMapper and the UrlReducer which contains the functions getLinks(), map() and reduce() respectively. The LinkGetter class takes in a String argument i.e. the URL of the webpage and outputs a list of URLs that match the regex expression. The regex expression essentially searches for strings that are enabled through the href tag of the html page. The getLinks() function additionally validates the extracted string through JavaScript and appends it to the final output string. Inside the UrlMapper class is the mapper function which utilizes the getLinks() function to obtains the links and uses the hash map data structure to maps the URL to its occurrence. So essentially the map() function does two jobs namely: map and combine. Consider the following example:



Here the Mapper takes in the URLs from each webpage, puts the URL into the hash map and checks if the current URL has an occurrence or not. If the current URL has occurred before, the mapper simply increments the occurrence by 1. If the current URL being checked is a brand-new URL then the mapper adds a 1 to its respective occurrence. This is done with the help of the ternary operator in java. Finally the

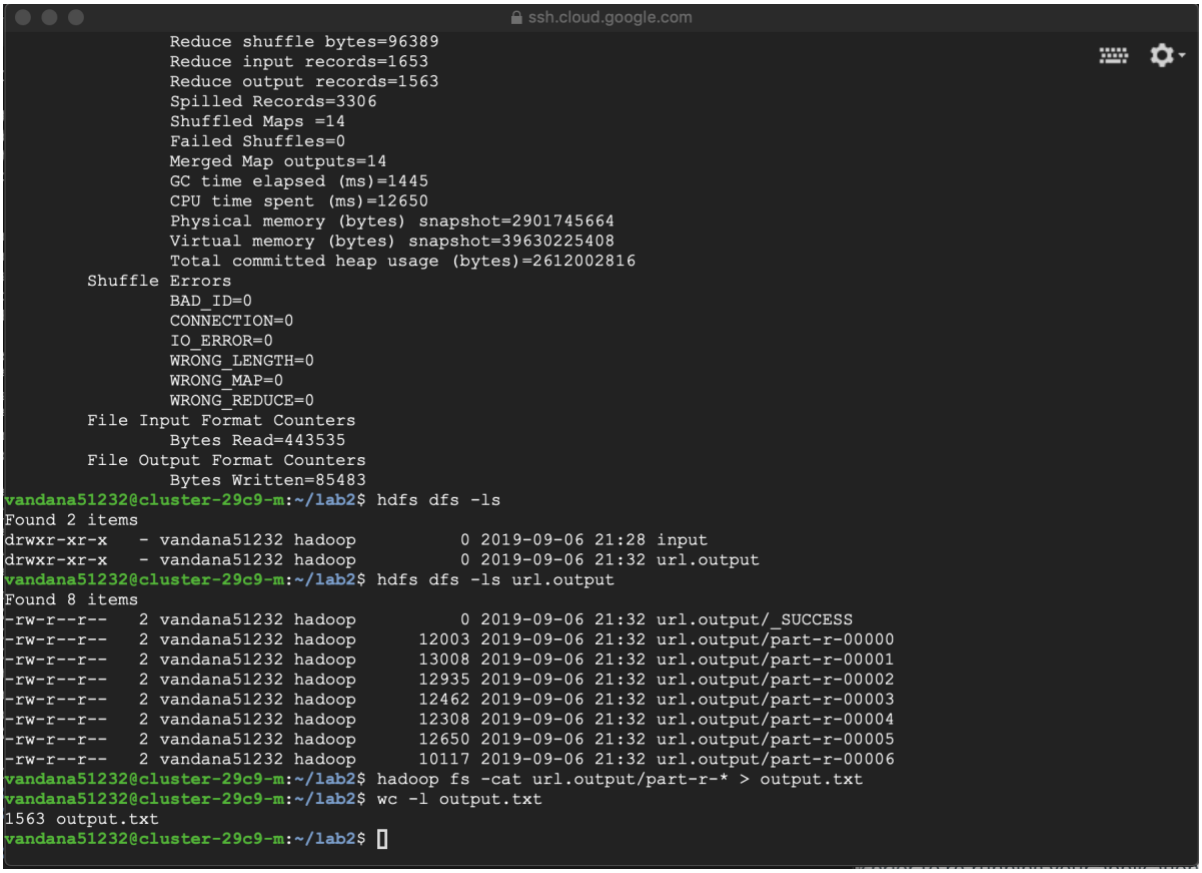
HashMap is iterated over and is written to the context. The reducer finally takes in this hash map from the context and reduces the occurrences of URLs as individual tuples. To summarize, the map function combines the unique occurrences of the URLs in each web page and then writes it to the context so that the reducer can reduce both webpage occurrences into one.

How is wordcount changed to URL Count?
The main change between the two programs is the Mapper function. In wordcount the String is tokenized into words and the tokens are iterated over and written along with the occurrence into context. In UrlCount, a new function that utilizes a specific regex value to search for valid URL links is utilized. Another change that I made to the map() function is the introduction of a hash map data structure that maps and combines URLs and their occurrences in individual web pages before it passes the work down to the reducer. This step simplifies the work of the reducer and is efficient.

How many Unique URLs?
I obtained 1563 unique URLs after the map-reduce task.

How did I run my code on the cluster?

1. First, I created a cluster using dataproc on the Google Cloud Platform.
2. Uploaded the zip file onto the VM. Unzipped the lab2 zip file using **unzip lab2.zip**
3. Created a filesystem through **make filesystem**
4. Created the jar file for the URL : **make UrlCount.jar**
5. To prepare the input files – **make prepare**
6. To run the map reduce task : **hadoop jar ./UrlCount.jar UrlCount input url.output**
7. To view the files located on the distributed file system: **hdfs dfs -ls**
8. To observe the output partitions: **hdfs dfs -ls url.output**, my reducer provided 7 partitions on the output file
9. To combine the output obtained from partitions into one output file : **hadoop fs -cat url.output/part-r-* > output.txt**
10. To obtain the number of unique URLs , I counted the number of lines in the output file : **wc -l output.txt**



Sources: https://www.vogella.com/tutorials/JavaRegularExpressions/article.html - Used the link given in Moodle to modify the LinkGetter class.
Colloboration policy:
Initial discussions with Pradyoth Srinivasan.