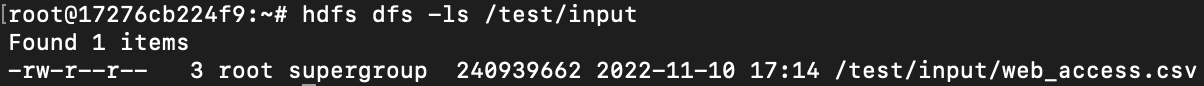
Practical 6

Dawid Skraba 19433692

***Exercise 1.***

Task 1:







I cloned the specified git repository. I then modified the mentioned components in the docker compose so our docker build command triggers these services. I then modified each component dockerfile so that we will download python in each container. I used the docker exec command with bash to enter each modified component and made sure python was installed correctly with the python –version command. I then used the Hadoop fs

-copyFromLocal to transfer the file from local storage to the hdfs file system.

Task 2:

*Function*: Total number of Bytes by each Host.

MapReduce works with <key, value> pairs.

The input for the mapper function would be the web\_access file which it will take it as input line by line and sort it to key value pairs so it can give it off to the reducer function.

The output of the mapper function will be multiple key value pairs with the host as the key and the number of bytes as the value. The output will produce each key value pair for each line in the file. These key value pairs include multiple pairs with the same host E.g., <\*\*\*.novok.dk, 363>, <\*\*\*.novok.dk, 5665>

The input for the Reducer function is <host, bytes> E.g., <\*\*\*.novok.dk, 363>

This is the input as we must sum up all the bytes “belonging” to one host.

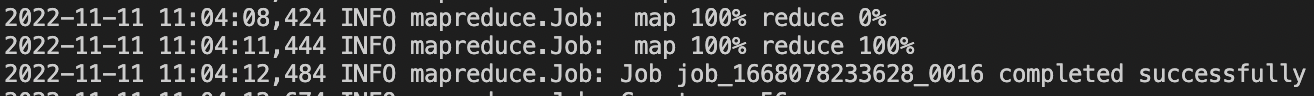
The output of the function is also going to be <key, value> pairs. These key, value pairs will hold each host as the key and the total summed bytes in the value. The output will hold the information how many bytes each host has in total.

E.g., <\*\*\*.nonk.dk, 4356355>

<007.thegap.com, 50000>

Task 3:





I wrote a mapper and reducer program. In the mapper program we firstly define a hostList dictionary where we will store our key value pairs. The key value pairs that the mapper will output to the reducer function are: <google.com 2> <google.com 3> (I used a tab as a delimiter). We then read line by line the file from stdin. I then use the strip command which gets rid of any whitespace which might be present in the line. The reader array holds all the lines that have been modified by the split command. This command using the ‘,’ delimiter will store just the values from the file of each line. After we go through the whole file, we loop through the reader array. We use l[1] for the host and l[6] for the byte. We the store these key value pairs in the dictionary mentioned above. We then output these pairs for the reducer.

The reducer program uses a collections counter to store the byte count so that we can easily query the five most popular hosts. We also read the inputs in from stdin and get rid of any whitespace. We then use the split command to split up the input into the host and byte. In order to skip the first line (since that’s the line with the header) we check if the host is a string, and a byte is an int. Then if the host isn’t in the collection, we store the byte count corresponding to the host and if it already exists, we add it on. Then we simply use the ‘most\_common’ command to query the hosts with the highest byte counts.

Here is the output of the 5 most accessed hosts and their bytes.

Text

Description automatically generated