CS6240 **Manjit Ullal** **Homework 1**

Map Reduce Implementation

Main Idea of the Map Reduce program:

The program reads the twitter data from the input edges.csv file. The dataset is made up of 85 million rows, where each row has “id, follower id” structure. The program counts the number of followers each user has. The follower count is found by grouping the “follower ids” and then summing their count.

Mapper function emits each “id” plus an associated count of occurrences (just “1” in this case).
For example:

Below case 1,2,3 follow 4 . We group by 4.

1→4
2→4
3→4

After Map phase, we will have

4→1
4→1
4→1

We use a default Combiner, which will group all the same keys together, and then sort the keys internally before passing the data to reducer.

Reducer, takes input from the combiner. Reducer will sum the similar keys thereby getting the follower count.

Algorithm

```
map(String key, String value)
 {
      // key: document name
      // value: document contents
      // follower id is the second column

      for ( each follower id in value )
      {
            emit(id , “1”)

      }
}
```

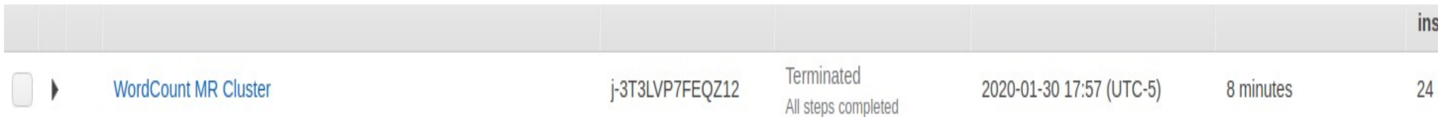
```
reduce( String key, Iterator values )
{

      // key: an id
      // values: a list of counts for that id

      int sum = 0;

      for (each val in values) {
            sum += 1;
      }

      // sum will be the sum of followers each user has
      emit(key, sum)
}
```

Running Time Measurements

Run 1 and Run 2 are two independent runs on AWS and their relevant measurements

Run 1

Running time of program



						ins
WordCount MR Cluster	j-3T3LVP7FEQZ12	Terminated All steps completed	2020-01-30 17:57 (UTC-5)	8 minutes		24

```
Total time spent by all map tasks (ms)=1246800
Total time spent by all reduce tasks (ms)=15572

CPU time spent (ms)=492460
```

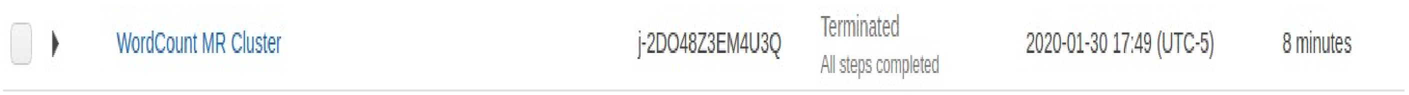
We see that the total time spend by the Map-Reduce program is equal to the CPU time, which is 8 minutes in this case.

Amount of Data Transferred

```
FILE: Number of bytes read=62489917
FILE: Number of bytes written=160314460

S3: Number of bytes read=1319451770
S3: Number of bytes written=67641452

Map input records=85331845
Map output records=85331845

Reduce input records=15362582
Reduce output records=6626985
```

Run 2

<u>Running time of program</u>


WordCount MR Cluster j-2DO48Z3EM4U3Q Terminated All steps completed 2020-01-30 17:49 (UTC-5) 8 minutes

```
Total time spent by all map tasks (ms)=1237047
Total time spent by all reduce tasks (ms)=151612

CPU time spent (ms)=497270
```

We see that the total time spend by the Map-Reduce program is equal to the CPU time, which is 8 minutes in this case as well.

<u>Amount of Data Transferred</u>

We notice that the amount of data transferred is same for both the runs.

```
FILE: Number of bytes read=62489917
FILE: Number of bytes written=160314460

S3: Number of bytes read=1319479722
S3: Number of bytes written=67641452

Map input records=85331845
Map output records=85331845

Reduce input records=15362582
Reduce output records=6626985
```

<u>Argue</u> briefly, why or why not your MapReduce program is expected to have good speedup. Make sure you discuss (i) *how many tasks* were executed in each stage and (ii) if there is a part of your program that is *inherently sequential* (see discussion of Amdahl's Law in the module.)

i) The Map-Reduce program is expected have good speedup because of couple of reasons.

One, the program is solving a typical group and count problem which can be programmed to be computed using Map-Reduce, which is proven to run fast due to the Map-Reduce programming Model and HDFS (Hadoop File System).

Secondly, we are using a single Master and 5 data nodes, which splits the tasks efficiently between the mapper and then from mapper to reducers. In this case there are 20 mapper tasks and 10 reducer tasks. Each of the Mapper and Reducer tasks run in parallel. Majority part of the computation is done in parallel so it achieves very good speedup.

ii) There are at-least two parts in the program which run sequentially and cannot be made to run in parallel. One is the **User program** and the other is the **Master node**. The user program can be any client that runs the Map-Reduce program, which in most cases cannot fork/handle multiple requests in parallel. Master node is internally managed by Hadoop and also runs sequentially. Therefore the speedup that can be achieved out of this program will be bounded by O(time taken to run the sequential processes)

Deliverable s

Log files

The syslogs are named as syslog1 and syslog2 referring to, two different independent runs

Twitter-FollowerCount/logs/syslog1.txt
Twitter-FollowerCount/logs/syslog2.txt

Input Files

Since the original input file is 1.3 GB, we are using only the top 2000 lines of input as discussed with professor to reduce size of the tar file

Twitter-FollowerCount/input

Output files

All the output files are in below folder
Twitter-FollowerCount/output

```
manjit@ubuntu:~/Downloads/Twitter-FollowerCount/output$ ls -ltr
total 108
-rw-r--r-- 1 manjit manjit 10932 Jan 31 18:19 output_1000_0
-rw-r--r-- 1 manjit manjit 10937 Jan 31 18:19 output_1000_1
-rw-r--r-- 1 manjit manjit 10939 Jan 31 18:20 output_1000_2
-rw-r--r-- 1 manjit manjit 10938 Jan 31 18:20 output_1000_3
-rw-r--r-- 1 manjit manjit 10934 Jan 31 18:20 output_1000_4
-rw-r--r-- 1 manjit manjit 10937 Jan 31 18:20 output_1000_5
-rw-r--r-- 1 manjit manjit 10929 Jan 31 18:20 output_1000_6
-rw-r--r-- 1 manjit manjit 10933 Jan 31 18:20 output_1000_7
-rw-r--r-- 1 manjit manjit 10926 Jan 31 18:20 output_1000_8
```

Since it is mentioned in the homework, that if the log file is greater than 1 MB we need to output only the top 1000 lines . all the output files have 1000 lines only copied from original output