CS580K: Adv. Topics In Cloud Computing

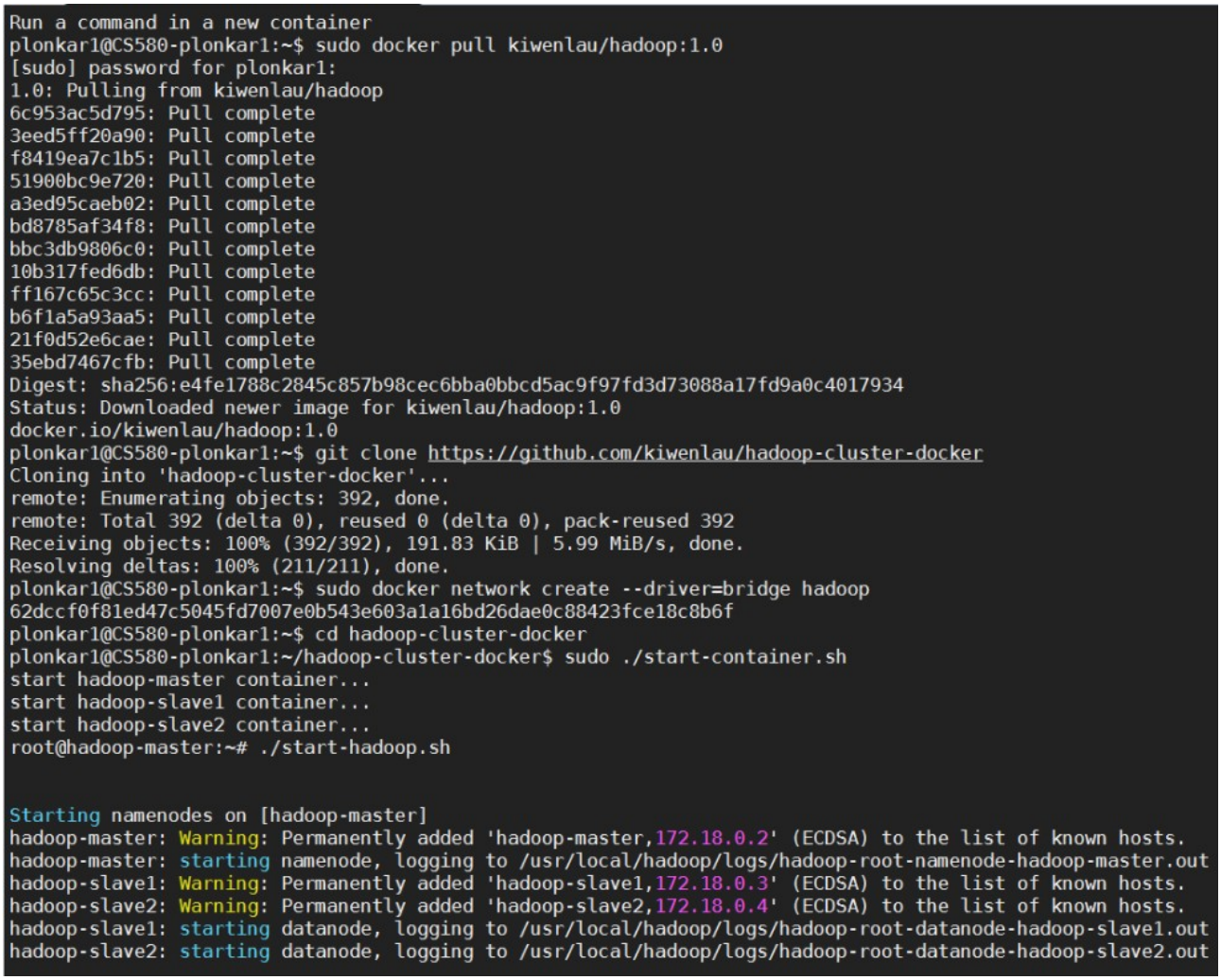
Mini Project-2

Name : Prathamesh N Lonkar

Bnumber: B00811727

Installation

The installation was done by the commands provided in the doc file as the following image depicts:



Task I

Q1.)Can you figure out what are the main steps do we need to run a hadoop mapreduce task (i.e., wordcount here)?
Answer:

The main steps to run the wordcount program here are as follows:

- Firstly we are running a sample wordcount code which is present in the hadoop ecosystem, hence we will create our own input files, which will be passed to the hadoop program. Which is done in the run-wordcount script as follows:

```
# create input files
mkdir input
echo "Hello Docker" >input/file2.txt
echo "Hello Hadoop" >input/file1.txt
```

- After this we will transfer all the input we created to the hdfs input folder for the jar wordcount file to read from it after running the run-wordcount script.
- After we run the ./run-wordcount.sh script it will take all the input from the hdfs input folder and pass it to the slaves which in turn create different jobs for the master to handle. The master will then perform the mapping and reducing operations on our input and count each occurrences of a word.
- After the counting is done it will write the data to the output file which is present in the hdfs folder.

Q2.)What does this command mean — “hdfs dfs -put ./input/* input”? (Hint, HDFS is Hadoop’s distributed file system. Please refer to [4].)

Answer:

The above command will simply take all the input files which we have created in the input folder which is done by ./input/* where * means all files , the -put command will simply put these files in a folder present in the dfs of the hdfs file system in this case “input” folder for the program to read input from it.

The above command can also be modified to delete folders i.e. input and output folders from hdfs system as if we run the same program with different input , there is a chance that the system might take the previous input. To do so the following commands can be used:
hdfs dfs -rm -r output (to delete output folder)
hdfs dfs -rm -r input (to delete the input folder)

Q3.)How many mappers and reducers are launched for executing the above wordcount program?

Answer:

The number of mappers used were 2
The number of Reducers used were 1 as shown in the output below:

```
Job Counters
        Launched map tasks=2
        Launched reduce tasks=1
        Data-local map tasks=2
        Total time spent by all maps in occupied slots (ms)=11566
        Total time spent by all reduces in occupied slots (ms)=4829
        Total time spent by all map tasks (ms)=11566
        Total time spent by all reduce tasks (ms)=4829
        Total vcore-milliseconds taken by all map tasks=11566
        Total vcore-milliseconds taken by all reduce tasks=4829
        Total megabyte-milliseconds taken by all map tasks=11843584
        Total megabyte-milliseconds taken by all reduce tasks=4944896
```

Q4.)How much time do mappers and reducers spend for the above tasks, separately?
The Time spend by the mapper was 11566 ms
The time spend by the reducer was 4829 ms as shown in the following output:

```
Job Counters
        Launched map tasks=2
        Launched reduce tasks=1
        Data-local map tasks=2
        Total time spent by all maps in occupied slots (ms)=11566
        Total time spent by all reduces in occupied slots (ms)=4829
        Total time spent by all map tasks (ms)=11566
        Total time spent by all reduce tasks (ms)=4829
        Total vcore-milliseconds taken by all map tasks=11566
        Total vcore-milliseconds taken by all reduce tasks=4829
        Total megabyte-milliseconds taken by all map tasks=11843584
        Total megabyte-milliseconds taken by all reduce tasks=4944896
```

Q5.)After execution, what are the files in the output folder in HDFS, and what content do they contain?

Answer:

The output folder in the hdfs contain the following two files:

```
root@hadoop-master:~# hdfs dfs -ls output
Found 2 items
-rw-r--r--   2 root supergroup          0 2020-11-15 21:47 output/_SUCCESS
-rw-r--r--   2 root supergroup         26 2020-11-15 21:47 output/part-r-00000
root@hadoop-master:~#
```

These files contain the following content:

```
root@hadoop-master:~# hdfs dfs -cat output/part-r-00000
Docker  1
Hadoop  1
Hello   2
root@hadoop-master:~# hdfs dfs -cat output/_SUCCESS
root@hadoop-master:~#
```

Task II

Q6.)How many master and slave containers do you launch separately this time?

After resizing the cluster and changing the start-container.sh we get 1 master and 4 slaves as shown in the output below:

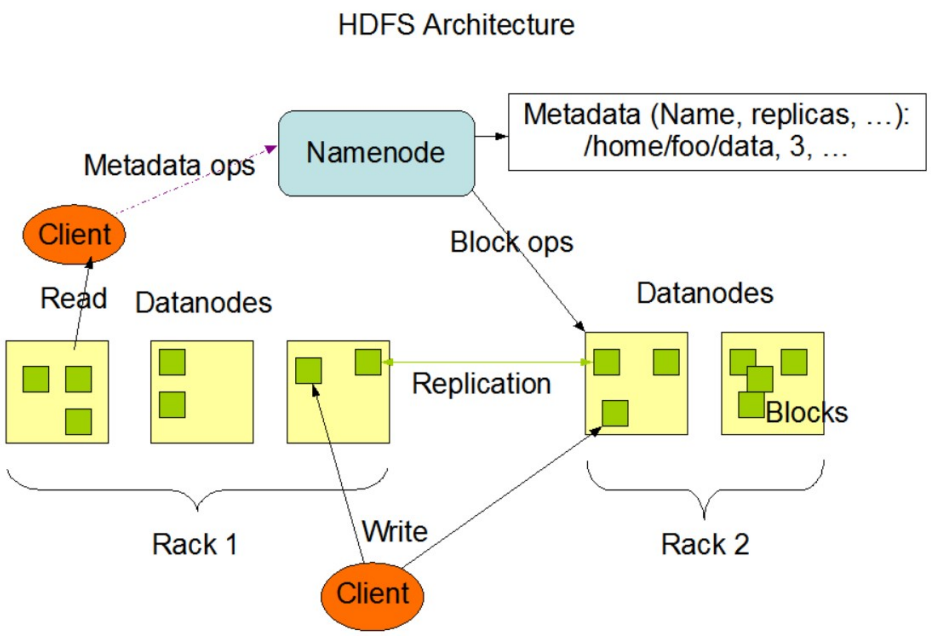
```
plonkar1@CS580-plonkar1:~/hadoop-cluster-docker$ sudo ./start-container.sh
[sudo] password for plonkar1:
start hadoop-master container...
start hadoop-slave1 container...
start hadoop-slave2 container...
start hadoop-slave3 container...
start hadoop-slave4 container...
```

Q7.)Please figure out what a master container/node and a slave container/node are used for?

Answer:

The Master Node is also called as a Namenode, an HDFS architecture only consist of a single master node and multiple Datanodes or also called as slave nodes. The main function of the master Node is to manage the name space of the file system and provide the access of the data to the client. It also manages the mapping of blocks to slave nodes.

On the other hand there can be multiple data nodes or slaves present in an HDFS architecture. The functionality of these data nodes is to store the info or data of the nodes that they are running on. These are then managed by the master node if a client needs that data. These slave nodes are responsible for managing read write requests from the client or the master node.
(Reference: https://hadoop.apache.org/docs/r1.2.1/hdfs_design)
The depiction of the Master Slave node is shown as follows:



Q8.)How many mappers and reducers are launched for executing the above wordcount program?

Answer:
The number of mappers were 3
The number of reducers were 1 as shown in the following output:

```
Job Counters
        Launched map tasks=3
        Launched reduce tasks=1
        Data-local map tasks=3
        Total time spent by all maps in occupied slots (ms)=27737
        Total time spent by all reduces in occupied slots (ms)=4694
        Total time spent by all map tasks (ms)=27737
        Total time spent by all reduce tasks (ms)=4694
        Total vcore-milliseconds taken by all map tasks=27737
        Total vcore-milliseconds taken by all reduce tasks=4694
        Total megabyte-milliseconds taken by all map tasks=28402688
        Total megabyte-milliseconds taken by all reduce tasks=4806656
```

Q9.)How much time do mappers and reducers spend for the above tasks, separately?

Answer:

The time taken by mapper for the task was 27737 ms
The time taken by the reducer for the task was 4694 ms as shown in the output:

```
Job Counters
        Launched map tasks=3
        Launched reduce tasks=1
        Data-local map tasks=3
        Total time spent by all maps in occupied slots (ms)=27737
        Total time spent by all reduces in occupied slots (ms)=4694
        Total time spent by all map tasks (ms)=27737
        Total time spent by all reduce tasks (ms)=4694
        Total vcore-milliseconds taken by all map tasks=27737
        Total vcore-milliseconds taken by all reduce tasks=4694
        Total megabyte-milliseconds taken by all map tasks=28402688
        Total megabyte-milliseconds taken by all reduce tasks=4806656
```

Q10.)What are the two most frequently occurring words, and how many times do they occur?

Answer :
The two most frequently occurring words are “of” which occurs 27 times and “the” which occurs 42 times as shown in the output snippet as follows:

```
noticed 2
now.    2
of      27
often   1
overspreads     1
own     1
paper   1
```

```
that    7
the     42
then    2
```

Task III

Q11.)Please describe the basic steps in the map function of WordCount.java.

Answer:

The map functions provided in the WordCount.java file follows these basic steps:

- Firstly we are accepting the whole text in input.txt as value with data type text.
- Then this value is converted to string and passed to the StringTokenizer function as this function only accepts string as an input.
- The above function will then tokenize each word and store it in the variable itr with the StringTokenizer data type.
- After this the while loop will run until all the word in the itr are read, this is done by the function hasMoreTokens(), which will check if itr has any tokens left.
- In the while loop we will just group the words and their occurrences using the context.write() function where word is each word and one is the IntWritable datatype.

Q12.) Please describe the basic steps in the reduce function of WordCount.java.

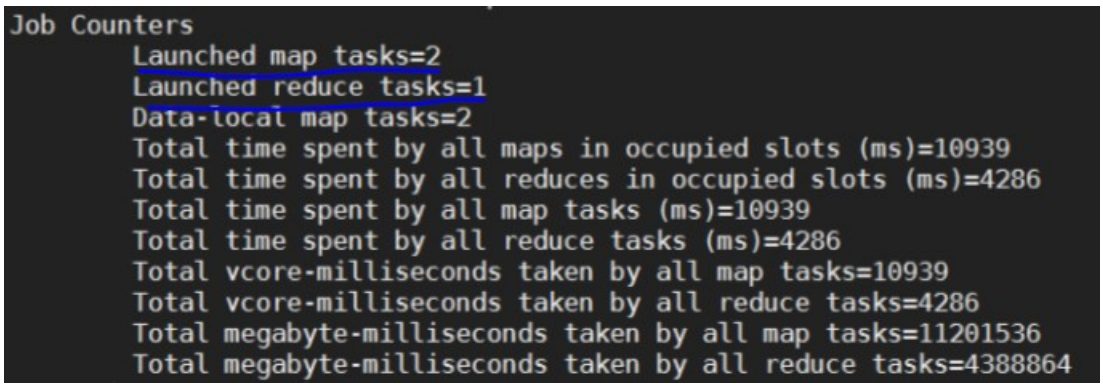
Answer:

The Reduce function provided in the WordCount.java file follows these basic steps:

- The reduce function gets two values as input, one is the key which is in the form of text, and other is the values which is the list of occurrence for that key.
- Here the key is a word and values contains the IntWritable list of values in this case one (1).
- The for loop then thus itterate over the values and stores the value in val, the val.get() function then get the value and add's it with the value in sum and stores it in sum.
- After this is done for one word the key(word), and the result(sum) is written in the context and is stored in the output file.
- The above steps are then done for each word.

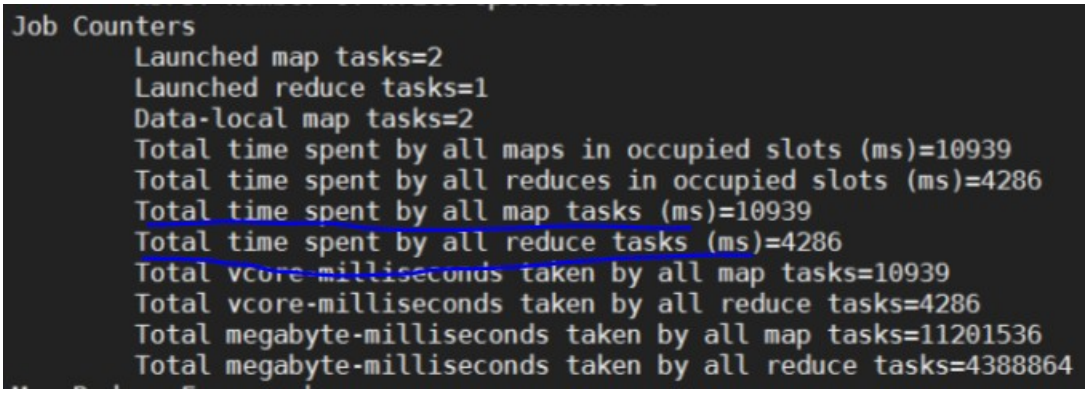
Q.13 How many mappers and reducers are launched for executing the above wordcount program?

The number of mappers were 2
The number of reducers were 1 as shown in the following output:



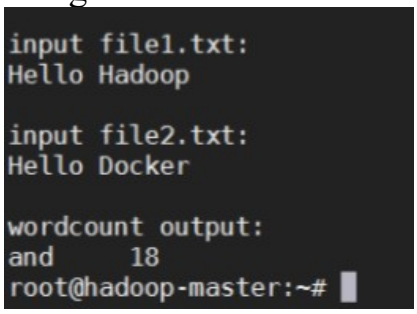
Q14.)How much time do mappers and reducers spend for the above tasks, separately?

The time taken by the mappers is 10939 ms
The time taken by the reducers is 4286 ms as shown in the output below.

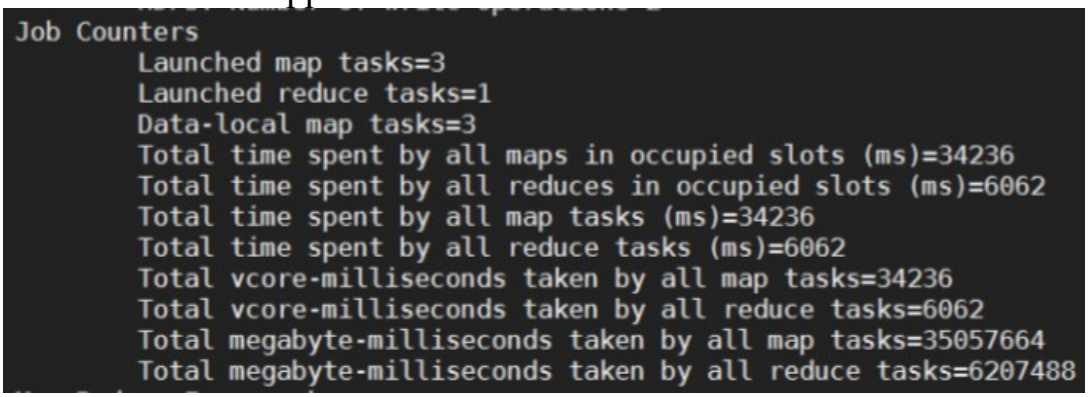


Task IV

- For this task I have made changes in the existing file provided by the professor. Can check the WordCount2.java file
- The output for the following task was as follows:



- The Number of Mappers and Reducers and the Time taken were as follows:



Task V

For this task I have created a program to perform logical operations such as and, or ,xor, mul, sub, add etc.
I am providing the input file as :
and 2
and 3
or 4
or 5
xor 6
xor 7

in this the .java file will map the and values and put them in a list and will pass this value to reducer where they will be processed using the key as ‘and’.

The Output is as follows:

```
input operations.txt:
and 2
and 3
or 4
or 5
xor 6
xor 7
mul 4
mul 5
sub 3
sub 2

wordcount output:
and     2
mul     20
or      5
sub     -1
xor     1
root@hadoop-master:~#
```

The Number of mappers and Reducers and their time taken is as follows:

```
Job Counters
        Launched map tasks=1
        Launched reduce tasks=1
        Data-local map tasks=1
        Total time spent by all maps in occupied slots (ms)=3470
        Total time spent by all reduces in occupied slots (ms)=3741
        Total time spent by all map tasks (ms)=3470
        Total time spent by all reduce tasks (ms)=3741
        Total vcore-milliseconds taken by all map tasks=3470
        Total vcore-milliseconds taken by all reduce tasks=3741
        Total megabyte-milliseconds taken by all map tasks=3553280
        Total megabyte-milliseconds taken by all reduce tasks=3830784
```