Assignment 2 - Hadoop MapReduce Word Counting

In this assignment you will implement a simple Hadoop MapReduce (M/R) job.

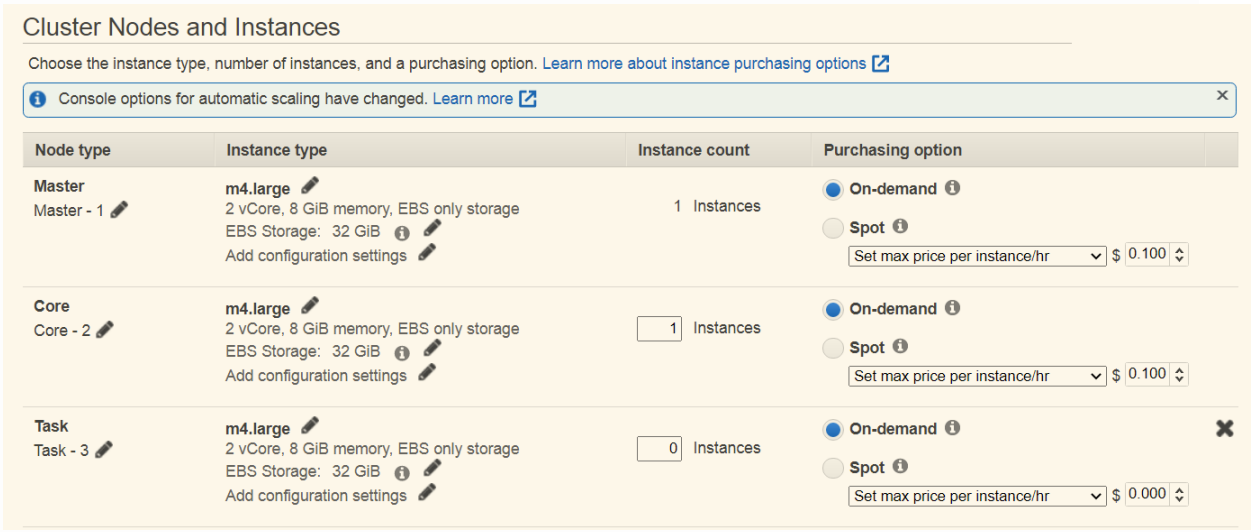
Our M/R program is WordCount. It should read text files and count how many times each word appears in the files. The input is a list of text files. The output is a text file, each line of which contains a word and its total count, separated by a tab.

Details:

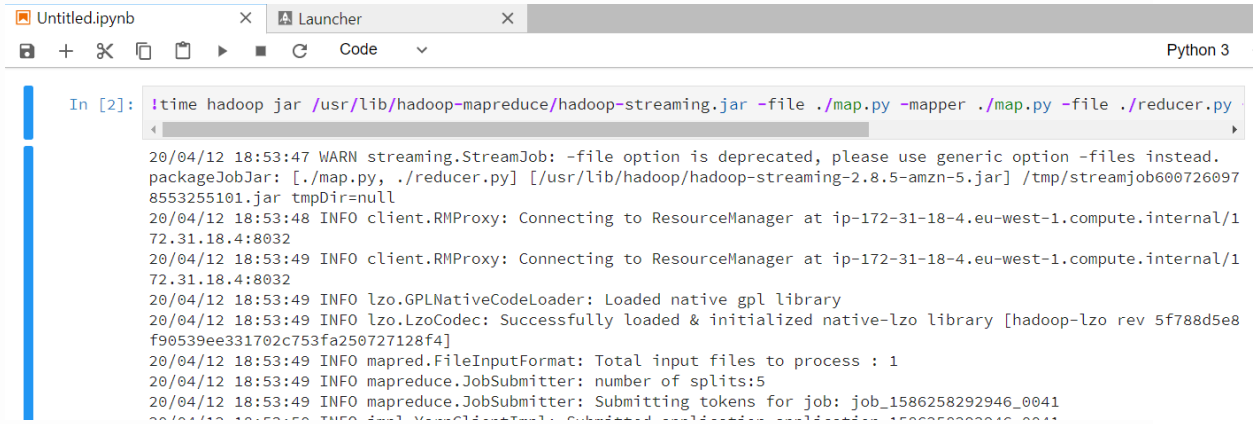
1. You will run your code on AWS on an EMR (Hadoop) cluster. It is recommended you use **m4.large** instances for cost control, and you can also request **spot instances** to reduce the cost by about 50-60%. See tips below.
2. Shut down the cluster when you are not using it.
3. The input text files are located in a shared read-only S3 bucket **here**. Under Data there are 3 folders. Gutenberg is the smallest, so it is recommended you develop and test your code on it. Encyclopaedia Britannica is medium size. Wikipedia is the largest. You should submit a table of results for all 3 datasets as explained below.
4. You should read the text files of each data set to the EMR cluster, save them as HDFS, process them using your M/R program, and write the output file back to an S3 bucket that you create for this purpose - pay attention to make your bucket public so we can check your assignment.
5. Run your code first with one worker, then 5 workers and finally 8 workers. Make sure the output file is identical in all cases! Print how many times your Map and Reduce tasks ran, and how long the entire iteration took.
6. Your submission should include 5 distinct parts (in one document): (a) reading data from S3 to HDFS, (b) the mapper function, (c) the reducer function, (d) the main M/R job, (e) the table below.
7. * If you want to learn more, try writing the same program in Pig and/or Hive. Note the resulting size of code and execution time(s).

Tips:

1. To use spot instances create the EMR cluster and click on Advanced Options. Then choose Spot instances and configure the bid price as shown below. Click on the ℹ icons to see current prices. You can typically bid for a price that is < 50% of the on-demand price for that instance type. However you may have problems resizing your cluster (it will say "**suspended**"). In that case increase your bid price or use on-demand instances.



2. The M/R job should run on the master node of the EMR cluster. The steps to open ssh and run Jupyter Lab on the master node are outlined here.
3. Once you have Jupyter Lab running on the EMR master node continue as follows:
   - Open a Jupyter notebook in your Jupyterlab.
     You can run shell commands above from within the notebook with "!"



     i. Copy the files from S3 to HDFS.
        1. You can do this in multiple ways. One way is to run the **`distcp`** command (distributed copy, a tool used for large inter/intra-cluster copying that uses M/R internally) from a terminal (CLI), as follows.

           Note: using the hadoop commands you should specify the S3 folder address as:

           `S3://bucket-name/key-name` (the URL address **`https://...`** will not work)
        2. First check you can see the source S3 data. E.g.:

           `> hadoop fs -ls s3://afeka-big-data-course/Data`

           You can get more data such as total size using

```
> aws s3 ls --summarize --human-readable --recursive
s3://afeka-big-data-course/Data
```

3. Copy the data:
```
> hadoop distcp s3://afeka-big-data-course/Data .
```
4. Check the data is here
```
> hadoop fs -ls .
```
Found 1 items

drwxr-xr-x - hadoop hadoop 0 2020-04-18 11:40 Gutenberg

Note: the regular `ls` command will not show HDFS files and directories!

ii. Use jupyter lab to write your mapper and reducer functions, and place them in files **mapper.py** and **reducer.py** in your home directory. You can also upload these files from your computer. Give them execute permissions:
```
> chmod ugo+x *.py
```

iii. Test your mapper.py and reducer.py code locally or on an EMR cluster notebook before running them as part of the M/R job, as follows:

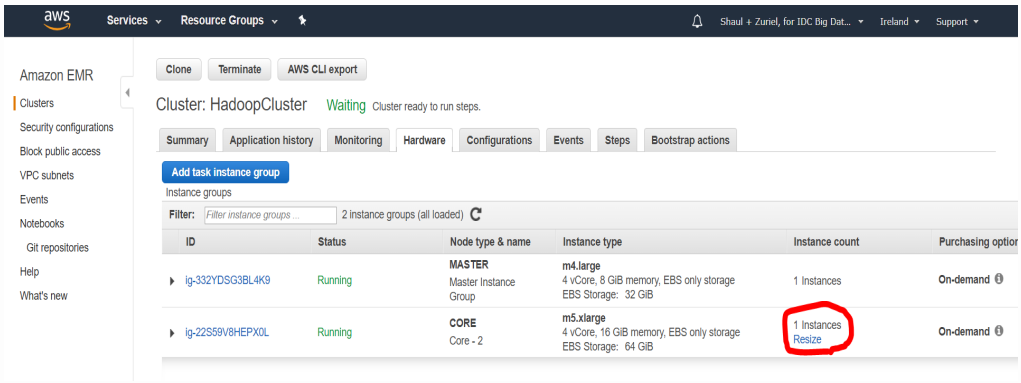
Mapper:
```
> cd ~
```
Test 1: `> echo 'foo foo quux labs foo bar quux' | ./mapper.py`

Test 2:
```
> cd data
data> hadoop fs -cat Gutenberg/* | /home/hadoop/mapper.py
```

Reducer:
```
> cd ~
```
Test 1: `> echo 'foo foo quux labs foo bar quux' | ./mapper.py | sort -k1,1 | ./reducer.py`

Test 2:
```
> cd data
data> hadoop fs -cat Gutenberg/* | /home/hadoop/mapper.py |
sort -k1,1 | /home/hadoop/reducer.py | sort -k2 -g -r >
Gutenberg.out
```

iv. Run the MapReduce job. There are multiple ways to do this, such as using Hadoop Streaming. Check the time of command for each iteration (1, 5 or 8 workers - you can change the number of workers (cores) at the hardware tab:



v. Run experiments and fill up the results in the table below. Copy the maps tasks, reduce tasks and CPU time from the M/R output (Launched map tasks, Launched Reduce tasks, CPU time spent). Run time is the clock time (e.g. add "time" at the beginning of the command. Note there are 10 configurations: you should run each dataset with 1 master node + 1, 5, and 8 workers. For the 10th experiment use the Wikipedia dataset and set the number of reduces to 30.

Input:	Gutenberg (3.5MB)	Britannica (27.8 MB)	Wikipedia (2.2 GB)
1 master + 1 slave	map tasks= reduce tasks= Run time = mm:ss CPU time = ms	map tasks= reduce tasks= Run time = mm:ss CPU time = ms	map tasks= reduce tasks= Run time = mm:ss CPU time = ms
1 master + 5 slaves	map tasks= reduce tasks= Run time = mm:ss CPU time = ms	map tasks= reduce tasks= Run time = mm:ss CPU time = ms	map tasks= reduce tasks= Run time = mm:ss CPU time = ms
1 master + 8 slaves	map tasks= reduce tasks= Run time = mm:ss CPU time = ms	map tasks= reduce tasks= Run time = mm:ss CPU time = ms	map tasks= reduce tasks= Run time = mm:ss CPU time = ms
1 master + 8 slaves 30 reducers			map tasks= reduce tasks= Run time = mm:ss CPU time = ms

- **Submit to Moodle your document as one Word or PDF file.**