HW1: Big Data Analytics (COSC 6339)

Problem Description:

The solution of the problem is divided into three parts and two lists of books are provided namely shortlist and longlist for carrying out the task.

Part 1: Write a python MapReduce code that calculates the occurrence of all words in the provided books on a *per book*basis.

Part 2: Write a python MapReduce code that determines the occurrence of each word in how many books.
Part 3: Evaluate the performance of your code created in part 1 and part 2 separately for the large data set using 2, 5, and 10 reducers.

Output of these experiments are further analyzed using graphs.

Solution Strategy:

First read through the problem to understand the what is expected as a solution. Tools required are Map-Reduce framework, HDFS file system where the files are stored and python to write the required program to do the analysis.

Following are the skillset required to be able to complete the HW.

1. Acquire basic knowledge of Unix commands such as ls, cd, mkdir, rm and SSH are required.
2. Next, understand how Map-Reduce framework works and basic commands to HDFS for the ease of working with it.
3. Coding skills in python, should be able perform string parsing.

With the above approach, I could write 2 separate programs to find the frequency of each word in a text book (Part 1) and occurrence of each word in how many book (Part 2).

Description of how to run your code

My codes are available in the in the *whale.cs.uh.edu* server under the following directory.
/home2/cosc6339/bigd27

- First login to *whale.cs.uh.edu* server using bigd27 credentials. Once you are in the home directory you will find the following python code using *ls* command.

```
-rw-r--r-- 1 bigd27 hadoop      594 Mar  4 13:33 word_Count_Part1.py
-rw-r--r-- 1 bigd27 hadoop      613 Mar  4 13:33 word_Count_Part2.py
```

word_Count_Part1.py is for the first part of the problem definition.
word_Count_Part2.py is for the second part of the problem definition.

- Below is the command to run a MapReduce job.

pydoop submit --num-reducers 5 --upload-file-to-cache wc.py wc /cosc6339_s17/books-longlist /bigd27/output

- Below is the command to fetch the output form HDFS and combine multiples output files into one

hadoop fs -getmerge /bigd27/output/part-r ./longlist_part2_output.txt*

To run a MapReduce job on the cluster, the input data should be in HDFS, not in the local home directory, and the result in an output folder in HDFS.

Explanation of the Code:

For the **first part**, we have a mapper and reducer Class and a main Class.

In mapper Class, we are fetching the filename from the context object.

words = context.value.split() #Loads the file
filename = context.input_split.filename; #fetches the filename

We are using regular expression to filter any unwanted character (special character) in the words and emitting the key value pair to reducer. Key is filename+word and value is 1.

context.emit((filename+ " " + (re.sub('[^A-Za-z]+', '', w))).lower(), 1)

The Reducer class is summing up all the values for a key and finally emitting the total count for each word present in one file.

```
s = sum(context.values)
context.emit(context.key, s)
```

For the **second part**, we also have a mapper and reducer Class and a main Class.

In this part we are fetching the filename and word like before.

```
for w in words:
        word = re.sub('[^A-Za-z]+', '', w);
        if(word):
           context.emit(word.lower(), filename)
```

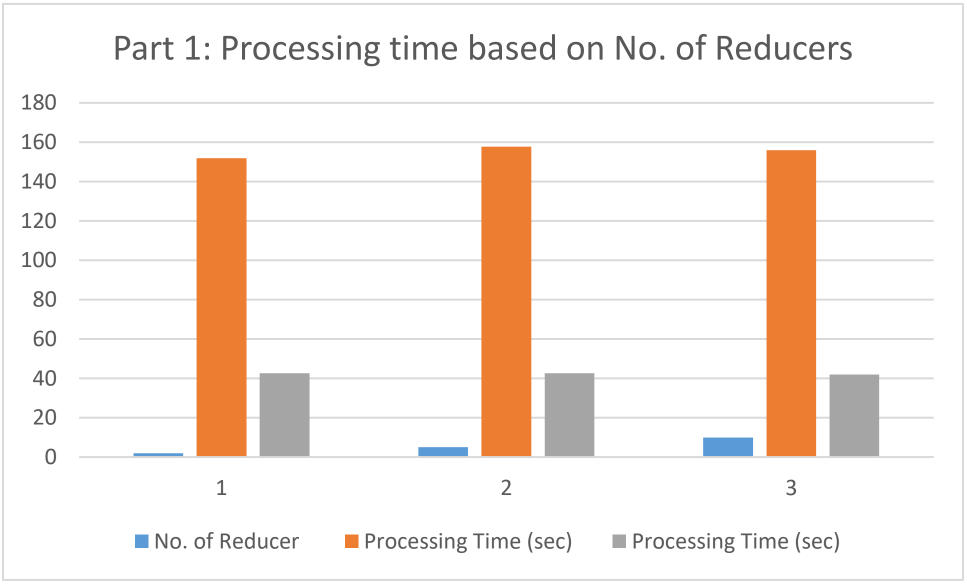
In this we are filtering unwanted word before emitting.

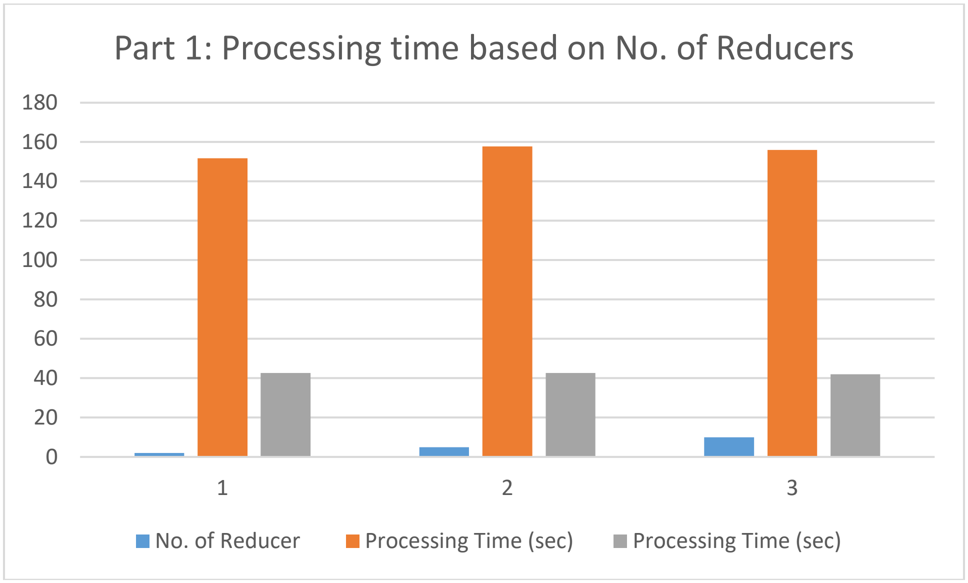
The Reducer class counts the total number of times a file name is appearing in the list using the function len() and finally emitting the word as key and the number of files it appeared as its value.

Results:

Part 1	Large files	Short files
No. of Reducer	**Processing Time (sec)**	**Processing Time (sec)**
2	151.8	42.6
5	157.8	42.6
10	156	42

Part 2	Large files	Short files
No. of Reducer	**Processing Time (sec)**	**Processing Time (sec)**
2	161.4	42.6
5	132.6	42
10	132	42.6





From the above graphs, it can be observed that with the increase in the number of reducers, the execution time is almost constant for the short list of books but for the long list of books the execution time reduced with increasing no. of reducer for part 2. Wherein part 1 we could observe a increasing trend.