Lab 4) Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm.

MapReduce is the heart of Hadoop. It is the programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. The MapReduce concept is fairly simple to understand for those who are familiar with clustered scale-out data processing solutions. The term MapReduce actually refers to two separate and distinct tasks that Hadoop program perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

ALGORITHM MAPREDUCE PROGRAM

WordCount is a simple program which counts the number of occurrences of each word in a given text input data set. WordCount fits very well with the MapReduce programming model making it a great example to understand the Hadoop Map/Reduce programming style. Our implementation consists of three main parts:

1. Mapper
2. Reducer
3. Driver

Step-1. Write a Mapper

A Mapper overrides the –mapll function from the Class "org.apache.hadoop.mapreduce.Mapper" which provides <key, value> pairs as the input. A Mapper implementation may output
<key,value> pairs using the provided Context .

Input value of the WordCount Map task will be a line of text from the input data file and the keywould be the line number <line_number, line_of_text> . Map task outputs <word, one> for eachword in the line of text.

Pseudo-code

```
void Map (key, value){
    for each word x in
        value:
        output.collect(x,
        1);
}
```

Step-2. Write a Reducer

A Reducer collects the intermediate <key,value> output from multiple map tasks and assemble asingle result. Here, the WordCount program will sum up the occurrence of each word to pairs as
<word, occurrence>.

Pseudo-code

```
void Reduce (keyword, <list of
value>){for each x in <list of
value>:
sum+=x;
final_output.collect(keyword, sum);
}
```

Step-3. Write Driver

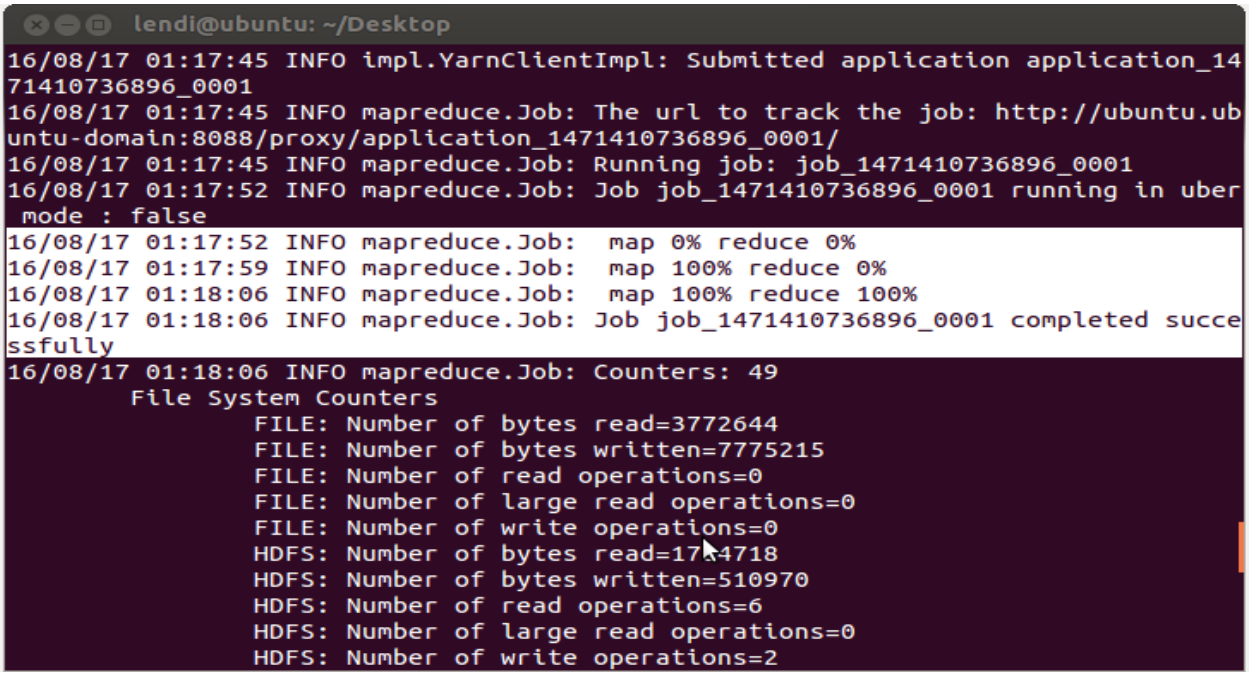
The Driver program configures and run the MapReduce job. We use the main program to perform basic configurations such as:

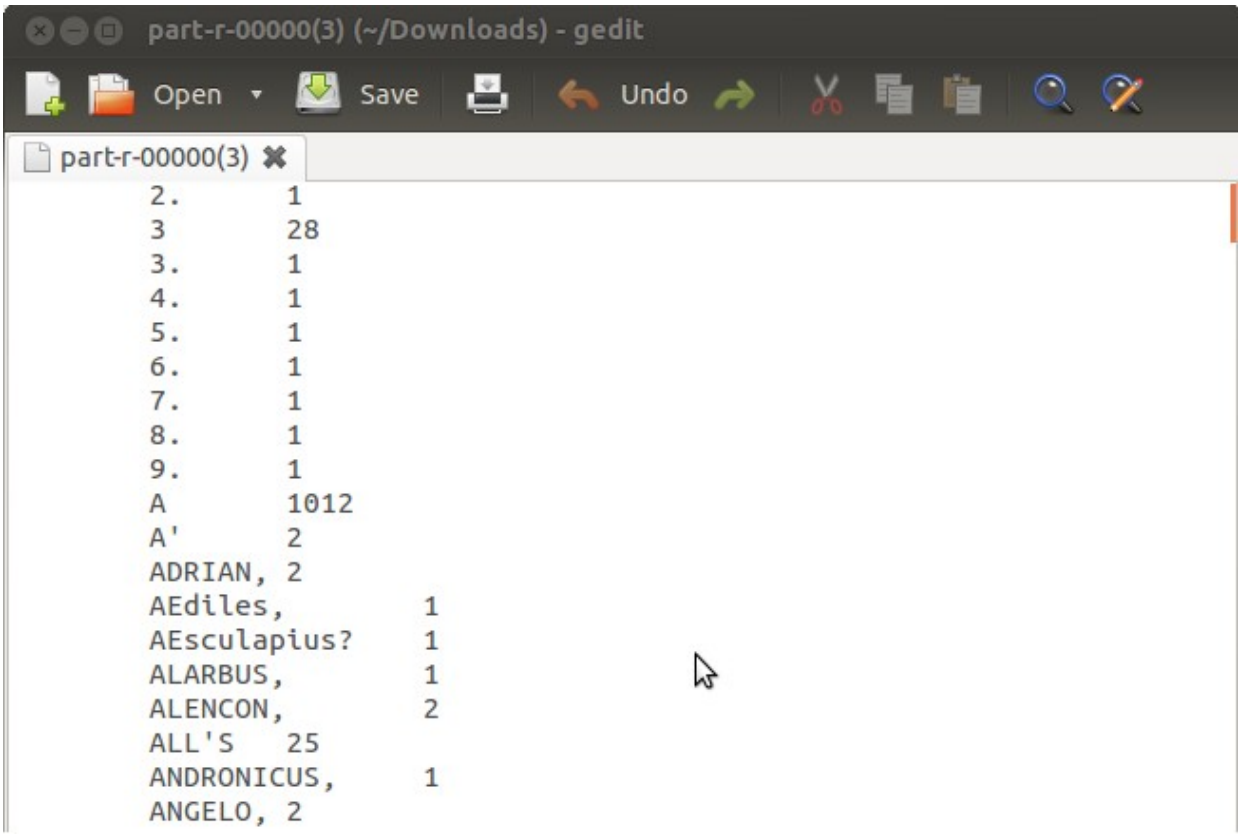
- Job Name : name of this Job
- Executable (Jar) Class: the main executable class. For here, WordCount.
- Mapper Class: class which overrides the "map" function. For here, Map.
- Reducer: class which override the "reduce" function. For here , Reduce.
- Output Key: type of output key. For here, Text.
- Output Value: type of output value. For here, IntWritable.
- File Input Path
- File Output Path

INPUT: -

Set of Data Related Shakespeare Comedies, Glossary, Poems

OUTPUT: -


lendi@ubuntu: ~/Desktop
16/08/17 01:17:45 INFO impl.YarnClientImpl: Submitted application application_1471410736896_0001
16/08/17 01:17:45 INFO mapreduce.Job: The url to track the job: http://ubuntu.ubuntu-domain:8088/proxy/application_1471410736896_0001/
16/08/17 01:17:45 INFO mapreduce.Job: Running job: job_1471410736896_0001
16/08/17 01:17:52 INFO mapreduce.Job: Job job_1471410736896_0001 running in uber mode : false
16/08/17 01:17:52 INFO mapreduce.Job: map 0% reduce 0%
16/08/17 01:17:59 INFO mapreduce.Job: map 100% reduce 0%
16/08/17 01:18:06 INFO mapreduce.Job: map 100% reduce 100%
16/08/17 01:18:06 INFO mapreduce.Job: Job job_1471410736896_0001 completed successfully
16/08/17 01:18:06 INFO mapreduce.Job: Counters: 49
File System Counters
FILE: Number of bytes read=3772644
FILE: Number of bytes written=7775215
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=1774718
HDFS: Number of bytes written=510970
HDFS: Number of read operations=6
HDFS: Number of large read operations=0
HDFS: Number of write operations=2


part-r-00000(3) (~/Downloads) - gedit
Open
Save
Undo
part-r-00000(3)
2. 1
3 28
3. 1
4. 1
5. 1
6. 1
7. 1
8. 1
9. 1
A 1012
A' 2
ADRIAN, 2
AEdiles, 1
AEsculapius? 1
ALARBUS, 1
ALENCON, 2
ALL'S 25
ANDRONICUS, 1
ANGELO, 2