

## MGS 655: Project 1

**Submitted by:** Aqib Junaid - 50412947, Akhilesh Anand Undralla - 50365549

### Hadoop DFS Setup and Your Environment:

a) Hadoop is installed on virtual box with 8GB memory available and 8VCores

Data Node Configuration :

Processors: 2 Quad Core CPU running @ 2 GHz

RAM: 8 GB

Disk: 1TB

SATA Network: 10 Gigabit Ethernet

b) Hadoop 2.6.4

c) `hadoop fs -copyFromLocal /home/Download/test.dat hdfs://localhost/user/dir/test.dat` used to move the file from the local system to the HDFS.

d) `hadoop fs -rm -r -f /user/the/path/to/your/dir` is used to delete the file from HDFS.

e) We will update the number of reducers by using the command `job.setNumReduceTasks(int)` where 'int' is the number of reducers.

### Hands-On Exercise:

a) We made a directory named "Alpha" using the command (Figure 1.1)

`hadoop fs -mkdir /Alpha`

After creating the directory, the following command is used to transfer the text file from the local `hdfs dfs -copyFromLocal /home/Hadoop/Desktop/WeddingOfFish.txt /Alpha`

To check if the file got stored in the right location, command `hadoop fs -ls /Alpha` is used.

b) We split the files manually using the text editor and uploaded the file in a single command (Figure 1.2): `hdfs dfs -copyFromLocal /path1/txt1 /path2/txt2 pathx/Alpha.`

c) Respectively, above commands were used to create and split the files.

d) Since these files are not large we did not give any specific algorithm for load balancing. Successful partitioning is achieved by blocks of 128MB each after uploading all the files (Figure 1.3).

### Case Study:

'Unacademy' has an environment to learn an array of topics with an algorithm that tracks user activity data. Data collected is used to provide recommendations guided by the user's relevant tracked data. Recommendation engine fed by the algorithm is based on a stream of structured and unstructured data containing video code catalogs, mouse tracking information, course status

information, Review, discussions forums, metadata regarding learners personal details, location used by mobile devices, user behavior containing browsing and scrolling behavior.

One of the Hadoop features, high availability is provided on the Hadoop cluster in case if any of the DataNode goes down the same data can be retrieved from any other node where the data is replicated.

Another hadoop feature i.e., flexibility is achieved from structured(MySql Data), Semi-Structured(XML, JSON), Un-structured (Images and Videos) efficiently and analyzing valuable insights of data from sources like social media, email, etc. With this flexibility, Hadoop can be used with log processing. MapReduce will help in processing huge amount of data due to parallel architecture and also by reducing its size.

### Practical Experiences:

Few of the challenges faced during the project are the limited size of the data taken which could not be tested for the load balancing mechanism of namenodes. It was difficult to upload multiple files at the same time to the HDFS system.

### Future Work:

- Improve the Configuration of the hardware used.
- User-friendly interface and setup for easy processing.

### Appendix:

```
hadoop@hadoop-VirtualBox:~$ hadoop fs -mkdir/Project1
-mkdir/Project1: Unknown command
hadoop@hadoop-VirtualBox:~$ hadoop fs -mkdir/ Project1
-mkdir/: Unknown command
hadoop@hadoop-VirtualBox:~$ hadoop fs -mkdir /Alpha
21/11/01 19:20:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
hadoop@hadoop-VirtualBox:~$ hdfs dfs -copyFromLocal /home/hadoop/Desktop/WeddingOfFish.txt /Alpha
21/11/01 19:23:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
hadoop@hadoop-VirtualBox:~$ hadoop fs -ls /
21/11/01 19:24:15 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
drwxr-xr-x  - hadoop supergroup          0 2021-11-01 19:23 /Alpha
drwxr-xr-x  - hadoop supergroup          0 2021-11-01 18:31 /aquib_bigData
hadoop@hadoop-VirtualBox:~$ hadoop fs -ls /Alpha
21/11/01 19:24:25 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r--  1 hadoop supergroup      1284 2021-11-01 19:23 /Alpha/WeddingOfFish.txt
hadoop@hadoop-VirtualBox:~$ #
```

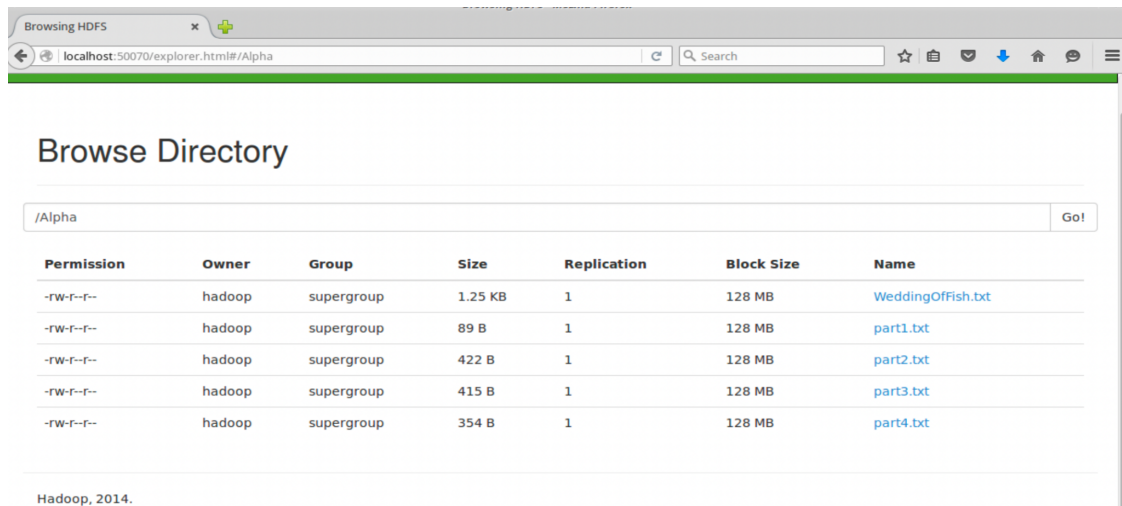
Figure 1.1: Transferring the file from Local system to HDFS.

```

copyFromLocal: /home/hadoop/Desktop/part2.txt: No such file or directory
hadoop@hadoop-VirtualBox:~$ hdfs dfs -copyFromLocal /home/hadoop/Desktop/part1.txt /Alpha
21/11/01 20:09:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
copyFromLocal: /home/hadoop/Desktop/part1.txt: No such file or directory
hadoop@hadoop-VirtualBox:~$ hdfs dfs -copyFromLocal /home/hadoop/Desktop/part1.txt /home/hadoop/Desktop/part2.txt /Alpha
21/11/01 20:10:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
copyFromLocal: /Alpha/part2.txt: File exists
hadoop@hadoop-VirtualBox:~$ hdfs dfs -copyFromLocal /home/hadoop/Desktop/part2.txt /home/hadoop/Desktop/part3.txt /Alpha
21/11/01 20:11:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
copyFromLocal: /Alpha/part3.txt: File exists
hadoop@hadoop-VirtualBox:~$ hdfs fs -ls /Alpha
21/11/01 20:12:25 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 5 items
-rw-r--r-- 1 hadoop supergroup 1284 2021-11-01 19:23 /Alpha/WeddingOffFish.txt
-rw-r--r-- 1 hadoop supergroup 89 2021-11-01 20:10 /Alpha/part1.txt
-rw-r--r-- 1 hadoop supergroup 422 2021-11-01 20:10 /Alpha/part2.txt
-rw-r--r-- 1 hadoop supergroup 415 2021-11-01 20:11 /Alpha/part3.txt
-rw-r--r-- 1 hadoop supergroup 354 2021-11-01 20:11 /Alpha/part4.txt
hadoop@hadoop-VirtualBox:~$

```

Figure 1.2: Uploading Multiple files to already existing Directory on HDFS



Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	1.25 KB	1	128 MB	<a href="#">WeddingOffFish.txt</a>
-rw-r--r--	hadoop	supergroup	89 B	1	128 MB	<a href="#">part1.txt</a>
-rw-r--r--	hadoop	supergroup	422 B	1	128 MB	<a href="#">part2.txt</a>
-rw-r--r--	hadoop	supergroup	415 B	1	128 MB	<a href="#">part3.txt</a>
-rw-r--r--	hadoop	supergroup	354 B	1	128 MB	<a href="#">part4.txt</a>

Hadoop, 2014.

Figure 1.3: HDFS Directory with partition files

**Project 2: Design of a Map Reduce Program**  
**MGS 655: Distributed Computing & Big Data**  
**Submitted by: Aqib Junaid - 50412947, Akhilesh Anand Undralla - 50365549**

---

a)

The following flow diagram illustrates step by step functioning of mapper and reducer. We have included the dictionary library in the mapper and associate the flag with every mapped word from the text file.

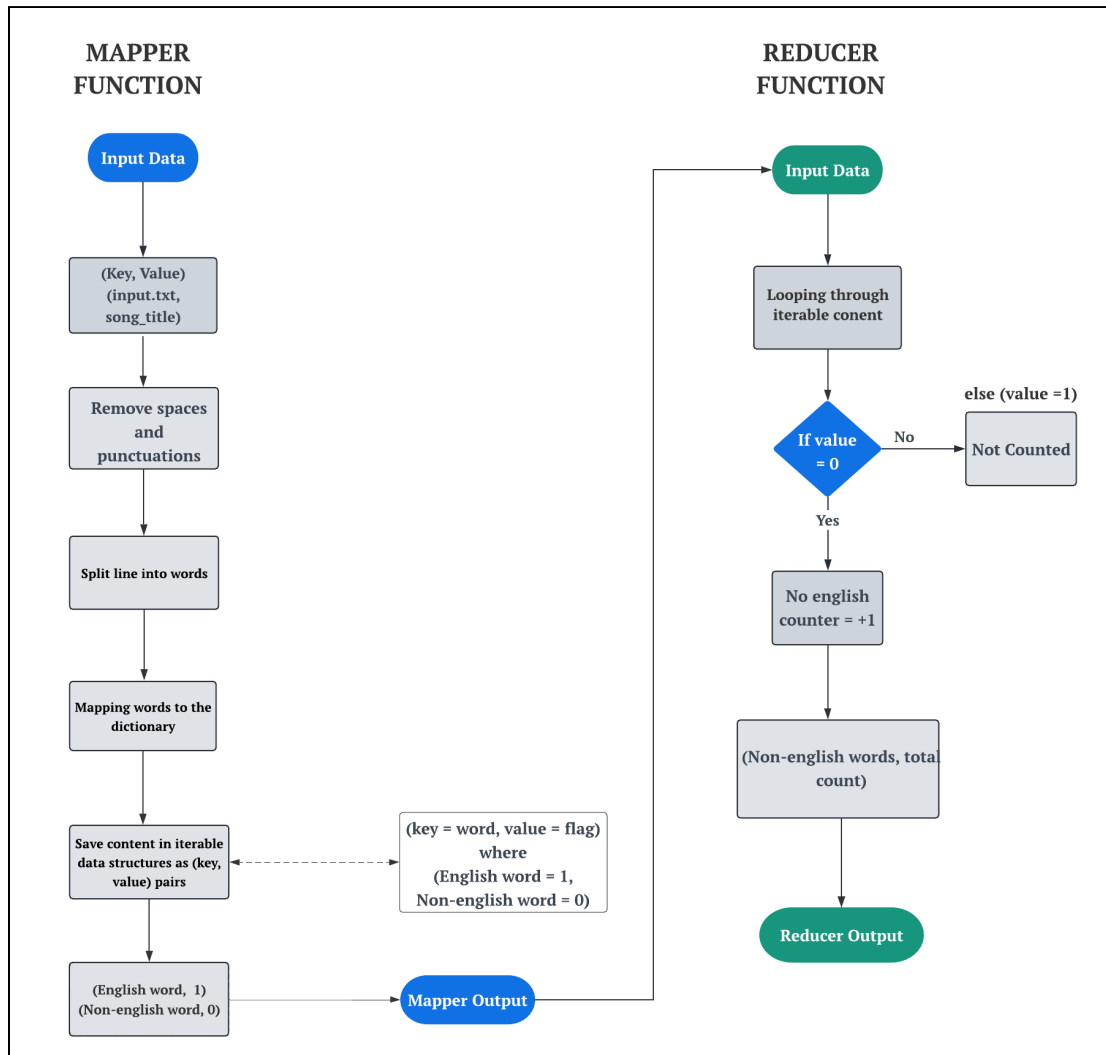


Figure: Flowchart for Map-Reduce algorithm

Even though we do not have hands-on experience in python or java, we used pseudo programming understanding to write the functions.

Mapper (Input.txt, content of the file)

The key in mapper is the input.txt song file, and value is the content of the file. We used `line.strip()` to remove the leading and trailing white spaces. Code to remove the punctuations is added later. Line were split into words using `words = line.split()`

The words are made to pass through the dictionary and then a flag is raised for every word that matches with the dict library. The value is returned with the word and its flag. Using binary as a flag, words are flagged as 1 that matches the one in the library and flagged as 0 if there is no match. Thus, the output from the mapper will look like this (W1, 1) where W1 is an english word or (W2,0) for W2 is a non-English word. All these tuples will be the input for a reducer function. The reducer will take the output from the mapper as its input which will be (W1, 1) or (W2,0) and programmed with a condition.

```
If value =1 then, do nothing
Else value =0, increase non-english counter = ++ (increment)
Only the counter which will show non-English count will be the output from
reducer.
```

b)

### Example 1:

```
Class: Mapper
Method: Map(Tsunami.text, context)
Remove spacing, punctuation (punc='''!()-[]{};:'"\,<>./?@#$$%^&*~'''')
Now Split the line into words
Mapping the words to the dictionary and putting in the flags
If present in dictionary put 1 or else put 0 in the value attribute for
forming a tuple
```

### Output:

```
(Oh,1) (mother,1) (of,1) (the,1) (ocean,1), (Mother,1) (of,1) (Ganga,0)
(why,1)(did,1) (you,1) (take,1) (so,1) (many,1) (lives,1) (forcibly,1) (What,1)
(pain,1) (Oh,1) (Dayal,0) (merciful,1) (one,1) (of,1) (my,1) (heart,1) (how,1) (my,1)
(heart,1) (weeps,1) (The,1) (sky,1) (weeps,1) (the,1) (wind,1) (cries,1) (the,1)
(unlucky,1) (mother,1) (weeps,1) (for,1) (her,1) (lost,1) (kids,1) (It,1) (is,1)
(hard,1) (to,1) (understand,1) (your,1) (play,1) (Lila,0) (khela,0)
```

```
Class: Reducer
Method:
Reduce(word , counts [0, 0, . . .])
sum ← 0
for all counts [0,0,...] do
sum ← sum + 0 Emit(non-english, count sum)
```

**Output:** (Non-english,3)

### Example 2:

```
Class: Mapper
Method: Map(victimizationwomen.text, context)
Remove spacing, punctuation (punc='''!()-[]{};:'"\,<>./?@#$$%^&*~'''')
Now Split the line into words
Mapping the words to the dictionary and putting in the flags
```

## Output:

```
(Listen,1)(to,1)(me,1)(everyone,1)(listen,1)(carefully,1)(Let,1)(me,1)(speak,1)
)(about,1)(the,1)(murder,1)(of,1)(women,1)(Scientists,1)(have,1)(invented,1)(s
uch,1)(a,1)(machine,1)(that,1)(everything,1)(inside,1)(a,1)(pregnant,1)(woman,
1)(can,1)(be,1)(seen,1)(If,1)(it,1)(is,1)(a,1)(baby,1)(girl,1)(they,1)(suck,1)
(her,1)(out,1)(and,1)(kill,1)(her,1)(Parents,1)(are,1)(unhappy,1)(if,1)(a,1)(g
irl,1)(child,1)(is,1)(born,1)(They,1)(dont,1)(send,1)(her,1)(to,1)(school,1)(b
ut,1)(to,1)(the,1)(kitchen,1)(Women,1)(work,1)(beside,1)(their,1)(husbands,1)(
on,1)(the,1)(fields,1)(They,1)(come,1)(back,1)(home,1)(and,1)(do,1)(all,1)(the
,1)(housework,1)(There,1)(is,1)(no,1)(peace,1)(in,1)(a,1)(womans,1)(life,1)(So
,1)(many,1)(women,1)(are,1)(killed,1)(for,1)(dowry,1)(demands,1)(Oh,1)(destiny
,1)(women,1)(are,1)(put,1)(on,1)(the,1)(funeral,1)(pyre,1)(with,1)(their,1)(hu
sbands,1)(Women,1)(fight,1)(for,1)(freedom,1)(Rani,0)(Rasmani,0)(Matangini,0)(
Hajra,0)(died,1)(from,1)(bullets,1)(Children,1)(get,1)(so,1)(much,1)(love,1)(f
rom,1)(their,1)(mothers,1)(So,1)(how,1)(can,1)(the,1)(same,1)(girls,1)(are,1)(
murdered,1)(off,1)(If,1)(women,1)(are,1)(finished,1)(off,1)(like,1)(this,1)(th
e,1)(human,1)(race,1)(will,1)(die,1)(out,1)
```

Class: Reducer

Method:

```
Reduce(word , counts [0, 0, . . .])
sum ← 0
for all counts [0,0,...] do
sum ← sum + 0 Emit(non-english, count sum)
Output: (Non-english,4)
```

e)

We have not executed this mapper-reducer program in a hadoop environment, but if we had to improve the program, combiners could be used thereby no involvement of reducer is needed. Also, we would like to find a way by which all these text files can be executed at once because the size of the files is small. This can also be achieved by creating a combined file for all the songs but getting an individual count on the files can not be achieved, rather we would have a total counter for all the non-english words in the archive.

f)

Using parallel programming speeds up the process as compared to serial execution. But as discussed in the class, speedup of the parallel programming depends on various parameters like parallel overhead, hardware infrastructure, style of writing a program may increase the execution time but if we are able to optimize these parameters we will be able to use the potential of parallel programming. Even though we try to convert the serial sequential programs to parallel, it becomes expensive and should be done after proper evaluation of process outcomes comparing these two procedures.