MapReduce for Flajolet-Martin:

Let's implement FM algorithm for a stream of data using MapReduce on Hadoop.

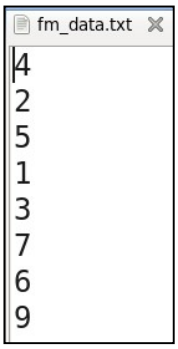
Step 1: Open Cloudera Quickstart VM.

Step 2: Clone the following repository on your local machine.

www.github.com/NSTiwari/Hadoop-MapReduce-Programs

You'll find the **`Flajolet-Martin-Algorithm`** folder along with some other folders. The **`Flajolet-Martin-Algorithm`** directory contains three files –

- **`fm_data.txt`** – A placeholder text file for input data stream.
- **`fm_mapper.py`** – The mapper file for FM algorithm.
- **`fm_reducer.py`** – The reducer file for FM algorithm.



fm_mapper.py:

```
#!/usr/bin/python

# import sys because we need to read and write data to STDIN and STDOUT

import sys
number={}
hash_values = {}
binary={}
trailing_zero={}
count=0

def trailing(s):
    return len(s) - len(s.rstrip('0'))
```

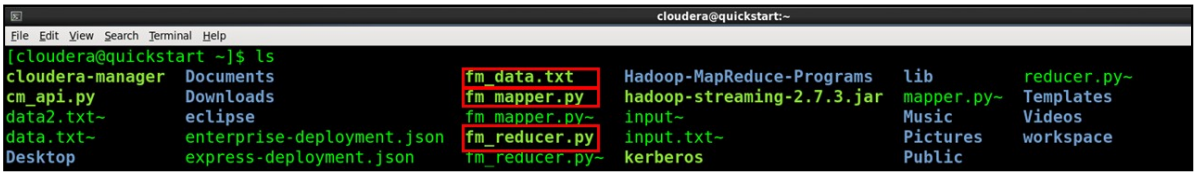
```
for line in sys.stdin:
    temp = 0
    number[count] = int(line)
    hash_values[count] = ((6 + number[count])) % 32
    binary[count]=bin(hash_values[count])[2:]
    trailing_zero[count]=int(trailing(str(binary[count])))
    print("%s\t%s\t%s" % (number[count], binary[count], trailing_zero[count]))
    count = count + 1
```

fm_reducer.py:

```
#!/usr/bin/python

from operator import itemgetter
import sys

maximum=0
M=0

for line in sys.stdin:
    num,binary,trailing_zero=line.split('\t')
    if int(maximum)<int(trailing_zero):
        maximum=trailing_zero

M=2**int(maximum)
print("Maximum trailing zeros = %sApproximate unique data elements = %s" %
(maximum, M))
```

Copy these three files inside **/home/cloudera** directory. Once done, check if they are copied at the desired location properly.

ls



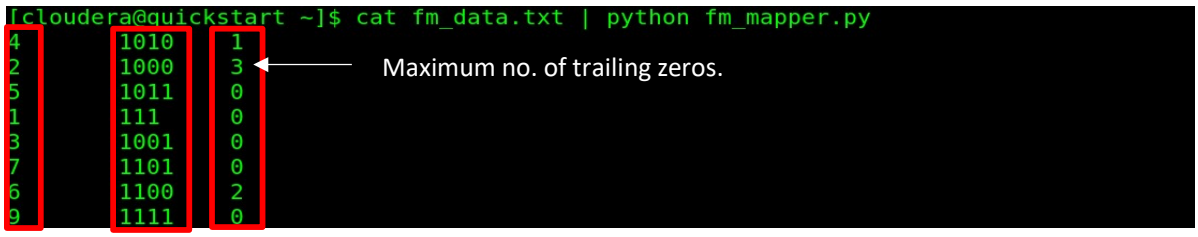
All the three required files are present.

Step 3: Test MapReduce program locally.

Before we run the MapReduce program on Hadoop, let's test it locally and see if the results obtained are correct as expected.

To do so, open the terminal and run the mapper program first.

cat fm_data.txt | python fm_mapper.py



The first column represents the input data stream, the second column corresponds to the binary value of the hash function h(x) = (x+6) mod 32 computed on each element of the data stream. The third column represents the no. of trailing zeros of the corresponding binary values of second column.

Now, run the complete MapReduce program.

cat fm_data.txt | python fm_mapper.py | python fm_reducer.py

```
[cloudera@quickstart ~]$ cat fm_data.txt | python fm_mapper.py | python fm_reducer.py
Maximum trailing zeros = 3
Approximate unique data elements = 8
```

As you can see, the maximum value in third column (no. of trailing zeros) is, R=3. Therefore, approximate distinct elements are, $2^R = 2^3 = 8$ with the elements being {1, 2, 3, 4, 5, 6, 7, 9} respectively.

The MapReduce program is running perfectly on local and the results obtained are exactly what we wanted.

> **Note:**
> The results would vary depending on the hash function you choose for the data stream. The hash function, h(x) = (x+6) mod 32 for the above data stream worked just perfectly and the results obtained were accurate. For different hash functions, different results are possible. That's the reason we consider 'approximate' unique data elements.

Let's now execute the same program on Hadoop.

Step 4: Create a directory on HDFS.

sudo -u hdfs hadoop fs -mkdir /flajolet_martin
hdfs dfs -ls /

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -mkdir /flajolet_martin
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 6 items
drwxr-xr-x   - hdfs   supergroup          0 2021-04-06 01:34 /flajolet_martin
drwxr-xr-x   - hbase  supergroup          0 2021-04-05 06:21 /hbase
drwxr-xr-x   - solr   solr                0 2015-06-09 03:38 /solr
drwxrwxrwx   - hdfs   supergroup          0 2021-04-05 06:45 /tmp
drwxr-xr-x   - hdfs   supergroup          0 2021-04-05 07:36 /user
drwxr-xr-x   - hdfs   supergroup          0 2015-06-09 03:36 /var
[cloudera@quickstart ~]$
```

Step 5: Copy input file on HDFS.

sudo -u hdfs hadoop fs -put /home/cloudera/fm_data.txt /flajolet_martin
hdfs dfs -ls /flajolet_martin

```
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -put /home/cloudera/fm_data.txt /flajolet_martin
[cloudera@quickstart ~]$ hdfs dfs -ls /flajolet_martin
Found 1 items
-rw-r--r--   1 hdfs supergroup         16 2021-04-06 01:37 /flajolet_martin/fm_data.txt
[cloudera@quickstart ~]$
```

The input file is copied successfully on HDFS inside **`flajolet_martin`** directory.

Step 6: Configure permissions to run MapReduce for FM algorithm on Hadoop.

Now, before we execute the MapReduce job on Hadoop, we need to give permission to read, write and execute the MapReduce program. We also need to provide permission for the default user (cloudera) to write the output file on HDFS.

To do so, run the following commands.

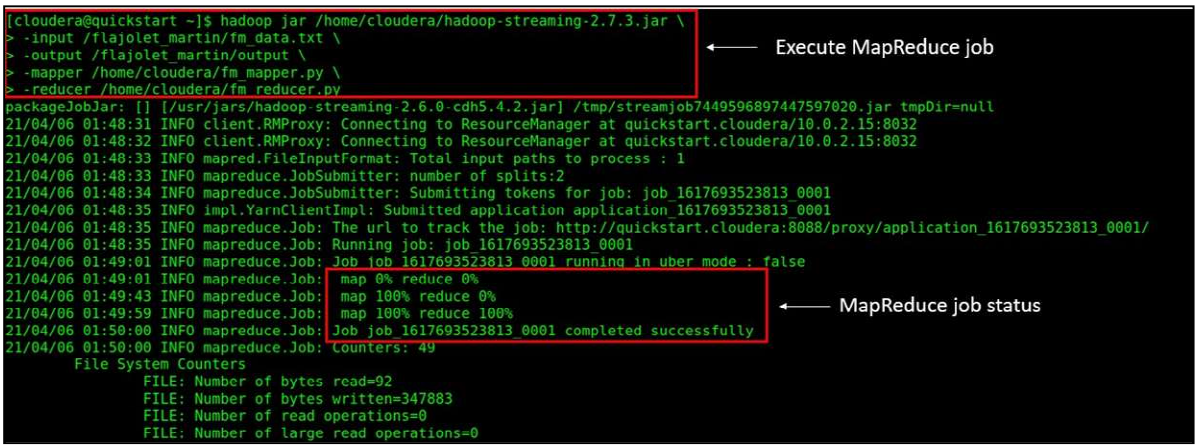
```
chmod 777 fm_mapper.py fm_reducer.py
sudo -u hdfs hadoop fs -chown cloudera /flajolet_martin
```

```
[cloudera@quickstart ~]$ chmod 777 fm_mapper.py fm_reducer.py
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -chown cloudera /flajolet_martin
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
```

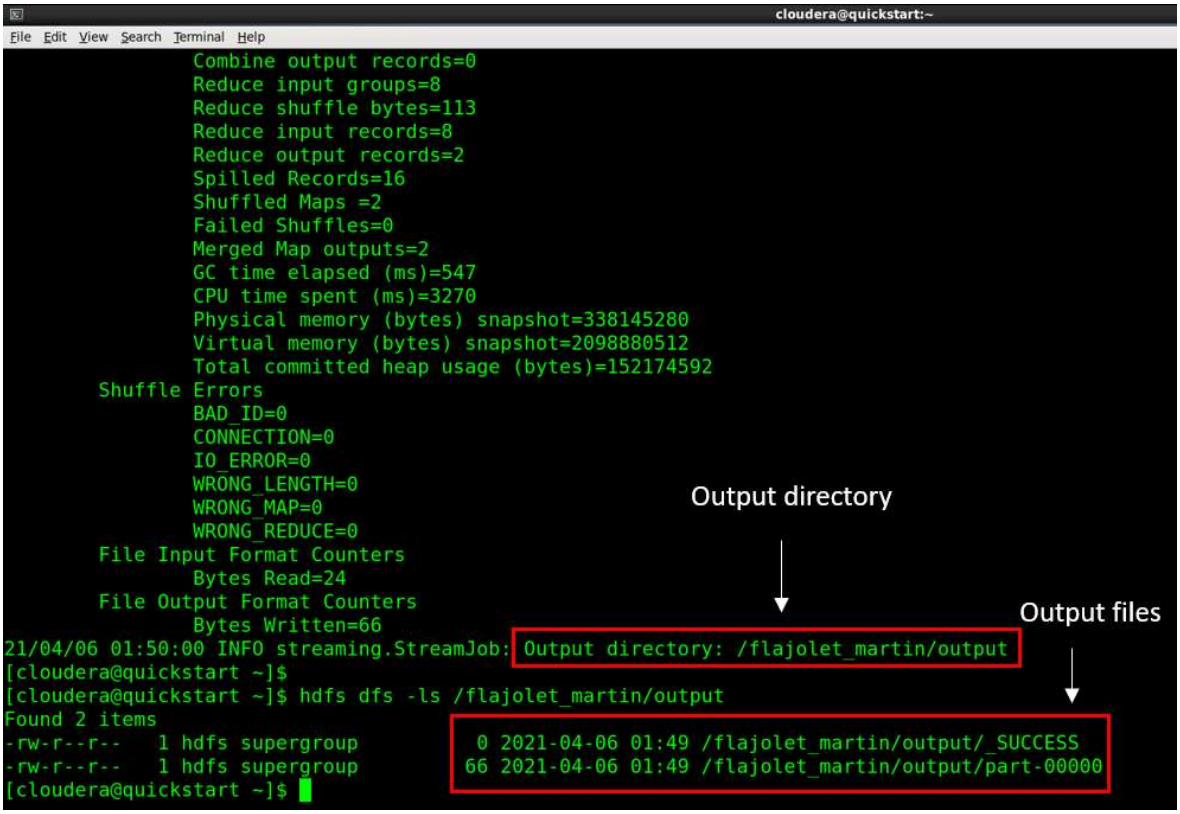
Step 7: Run MapReduce on Hadoop.

We can now execute the MapReduce job on Hadoop. Run the following command.

```
hadoop jar /home/cloudera/hadoop-streaming-2.7.3.jar \
> -input /flajolet_martin/fm_data.txt \
> -output /flajolet_martin/output \
> -mapper /home/cloudera/fm_mapper.py \
> -reducer /home/cloudera/fm_reducer.py
```




```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/hadoop-streaming-2.7.3.jar \
> -input /flajolet_martin/fm_data.txt \
> -output /flajolet_martin/output \
> -mapper /home/cloudera/fm_mapper.py \
> -reducer /home/cloudera/fm_reducer.py
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.4.2.jar] /tmp/streamjob7449596897447597020.jar tmpDir=null
21/04/06 01:48:31 INFO client.RMProxy: Connecting to ResourceManager at quickstart.cloudera/10.0.2.15:8032
21/04/06 01:48:32 INFO client.RMProxy: Connecting to ResourceManager at quickstart.cloudera/10.0.2.15:8032
21/04/06 01:48:33 INFO mapred.FileInputFormat: Total input paths to process : 1
21/04/06 01:48:33 INFO mapreduce.JobSubmitter: number of splits:2
21/04/06 01:48:34 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1617693523813_0001
21/04/06 01:48:35 INFO impl.YarnClientImpl: Submitted application application_1617693523813_0001
21/04/06 01:48:35 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1617693523813_0001/
21/04/06 01:48:35 INFO mapreduce.Job: Running job: job_1617693523813_0001
21/04/06 01:49:01 INFO mapreduce.Job: Job job_1617693523813_0001 running in uber mode : false
21/04/06 01:49:01 INFO mapreduce.Job:  map 0% reduce 0%
21/04/06 01:49:43 INFO mapreduce.Job:  map 100% reduce 0%
21/04/06 01:49:59 INFO mapreduce.Job:  map 100% reduce 100%
21/04/06 01:50:00 INFO mapreduce.Job: Job job_1617693523813_0001 completed successfully
21/04/06 01:50:00 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=92
                FILE: Number of bytes written=347883
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
```




```
                Combine output records=0
                Reduce input groups=8
                Reduce shuffle bytes=113
                Reduce input records=8
                Reduce output records=2
                Spilled Records=16
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=547
                CPU time spent (ms)=3270
                Physical memory (bytes) snapshot=338145280
                Virtual memory (bytes) snapshot=2098880512
                Total committed heap usage (bytes)=152174592
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=24
        File Output Format Counters
                Bytes Written=66
21/04/06 01:50:00 INFO streaming.StreamJob: Output directory: /flajolet_martin/output
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hdfs dfs -ls /flajolet_martin/output
Found 2 items
-rw-r--r--   1 hdfs supergroup          0 2021-04-06 01:49 /flajolet_martin/output/_SUCCESS
-rw-r--r--   1 hdfs supergroup         66 2021-04-06 01:49 /flajolet_martin/output/part-00000
[cloudera@quickstart ~]$
```

Step 8: Read MapReduce output.

Now, to see the output of the MapReduce job you just executed, run the following command.

hdfs dfs -cat /flajolet_martin/output/part-00000

```
[cloudera@quickstart ~]$ hdfs dfs -cat /flajolet_martin/output/part-00000
Maximum trailing zeros = 3
Approximate unique data elements = 8
```

And here we go, the results of the MapReduce job are exactly the same we obtained before, while testing locally.

Congratulations, for successfully executing FM algorithm using MapReduce on Hadoop. Just remember that the hash function you choose for your data stream will determine what your output would be.