

MapReduce for matrix multiplication on Hadoop:

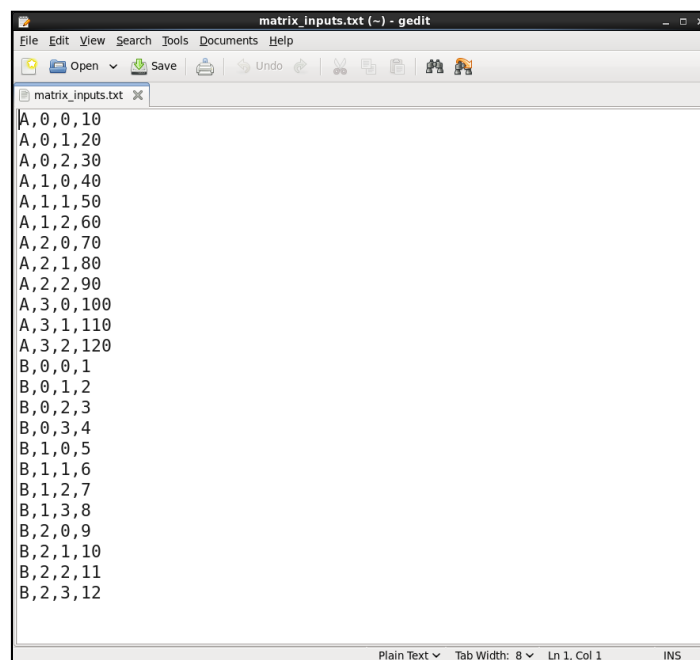
Let's solve a matrix-matrix multiplication problem using MapReduce on Hadoop.

Step 1: Open Cloudera Quickstart VM.



Step 2: Create input data.

Create a .txt data file inside `/home/cloudera` directory that will be passed as an input to MapReduce program. For simplicity purpose, we name it as `matrix_inputs.txt`.



Input Format:

- The first column represents the matrix name to which the element belongs.
- The second column corresponds to the row to which the matrix element belongs.
- The third column corresponds to the column to which the matrix element belongs.
- The fourth column represents the actual element value.
- Each column is separated by a comma.
- The tuple (`matrix_name`, `row_value`, `column_value`, `element_value`) for every element is written on a new line.

Step 3: Mapper and Reducer files.

Create `mapper_matrix.py` and `reducer_matrix.py` files inside `/home/cloudera` directory.

mapper_matrix.py

```
#!/usr/bin/python

import sys
row_a, col_b = 4,4 # dimensions of resultant matrix

for line in sys.stdin:
    matrix_index, row, col, value = line.rstrip().split(",")

    if matrix_index == "A":
        for i in range(0,col_b):
            key = row + "," + str(i)
            print("%s\t%s\t%s"%(key,col,value))

    else:
        for j in range(0,row_a):
            key = str(j) + "," + col
            print("%s\t%s\t%s"%(key,row,value))
```

reducer_matrix.py

```
#!/usr/bin/python

import sys
from operator import itemgetter

prev_index = None
value_list = []

for line in sys.stdin:
    curr_index, index, value = line.rstrip().split("\t")
    index, value = map(int,[index,value])

    if curr_index == prev_index:
        value_list.append((index,value))

    else:
        if prev_index:
            value_list = sorted(value_list,key=itemgetter(0))
            i = 0
            result = 0

            while i < len(value_list) - 1:
                if value_list[i][0] == value_list[i + 1][0]:
                    result += value_list[i][1]*value_list[i + 1][1]
                    i += 2
                else:
                    i += 1

            print("%s,%s"%(prev_index,str(result)))
```

```

prev_index = curr_index
value_list = [(index,value)]

if curr_index == prev_index:
    value_list = sorted(value_list,key=itemgetter(0))
    i = 0
    result = 0

    while i < len(value_list) - 1:
        if value_list[i][0] == value_list[i + 1][0]:
            result += value_list[i][1]*value_list[i + 1][1]
            i += 2

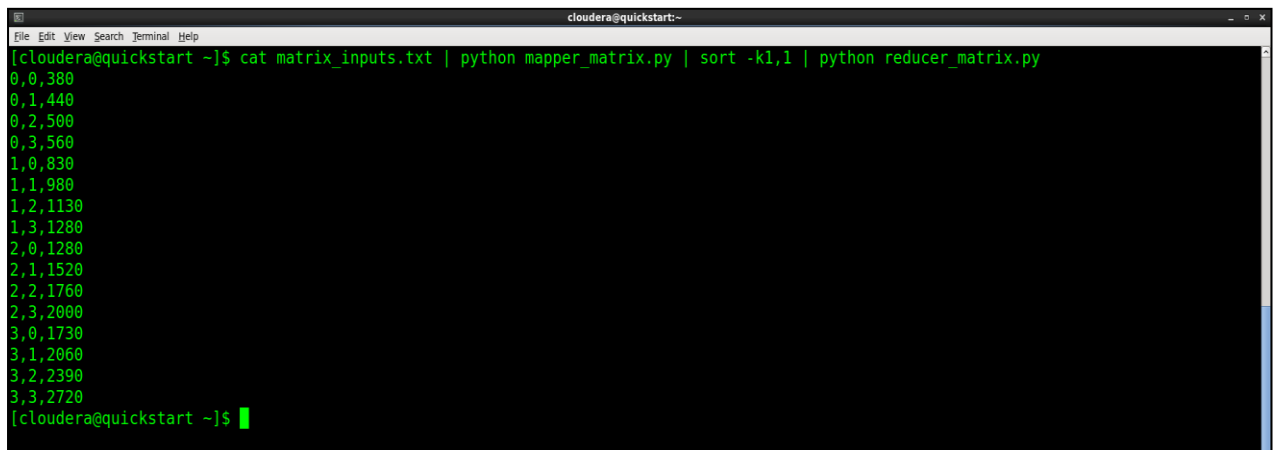
        else:
            i += 1

    print("%s,%s"%(prev_index,str(result)))

```

Step 4: Test the matrix multiplication MapReduce program locally to check if the program is correct before we run it on Hadoop.

`cat matrix_inputs.txt | python mapper_matrix.py | sort -k1,1 | python reducer_matrix.py`



```

[cloudera@quickstart ~]$ cat matrix_inputs.txt | python mapper_matrix.py | sort -k1,1 | python reducer_matrix.py
0,0,380
0,1,440
0,2,500
0,3,560
1,0,830
1,1,980
1,2,1130
1,3,1280
2,0,1280
2,1,1520
2,2,1760
2,3,2000
3,0,1730
3,1,2060
3,2,2390
3,3,2720
[cloudera@quickstart ~]$

```

The output obtained is exactly the way we wanted. Again, the first column represents the **row_value** of element, second column represents the **column_value** of element and the third column represents the actual **element_value** in the output matrix.

The program is now good to be tested on Hadoop.

Step 5: Create a directory on HDFS.

Make sure that the HDFS and YARN services are started in the background from the Cloudera dashboard before you create a directory on HDFS.

Create a directory named **matrix_multiplication** on HDFS where our input matrix file and the resultant output matrix would be stored. Use the following command for it.

`sudo -u hdfs hadoop fs -mkdir /matrix_multiplication`

List the HDFS directory items using the following command to check if the directory was successfully created or not.

`hdfs dfs -ls /`

```
cloudera@quickstart:~$ sudo -u hdfs hadoop fs -mkdir /matrix_multiplication
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 7 items
drwxr-xr-x  - hbase      supergroup      0 2021-02-16 08:01 /hbase
drwxr-xr-x  - hdfs       supergroup      0 2021-02-27 10:43 /matrix_multiplication
drwxr-xr-x  - solr       solr            0 2015-06-09 03:38 /solr
drwxrwxrwx  - hdfs       supergroup      0 2021-02-24 05:53 /tmp
drwxr-xr-x  - hdfs       supergroup      0 2021-02-24 06:41 /user
drwxr-xr-x  - hdfs       supergroup      0 2015-06-09 03:36 /var
drwxr-xr-x  - cloudera   supergroup      0 2021-02-25 05:59 /word_count_map_reduce
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
```

Created "matrix_multiplication" directory

Step 6: Copy input data file to HDFS.

Copy the `matrix_inputs.txt` file to `matrix_multiplication` directory on HDFS using the following command.

```
sudo -u hdfs hadoop fs -put /home/cloudera/matrix_inputs.txt /matrix_multiplication
```

Check if file was copied successfully to the desired location.

```
hdfs dfs -ls /matrix_multiplication
```

```
cloudera@quickstart:~$ sudo -u hdfs hadoop fs -put /home/cloudera/matrix_inputs.txt /matrix_multiplication
[cloudera@quickstart ~]$ hdfs dfs -ls /matrix_multiplication
Found 1 items
-rw-r--r--  1 hdfs supergroup      210 2021-02-27 11:00 /matrix_multiplication/matrix_inputs.txt
[cloudera@quickstart ~]$
```

File copied successfully

Step 7: Configure permissions to execute MapReduce matrix multiplication on Hadoop.

To run the `mapper_matrix.py` and `reducer_matrix.py` files in Hadoop, we need to give the read, write and execute permissions to it. We also need to give the default user (cloudera), the permission to write the output file inside the `matrix_multiplication` HDFS directory.

Run the following commands to do so.

```
chmod 777 mapper_matrix.py reducer_matrix.py
sudo -u hdfs hadoop fs -chown cloudera /matrix_multiplication
```

```
cloudera@quickstart:~$ chmod 777 mapper_matrix.py reducer_matrix.py
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -chown cloudera /matrix_multiplication
[cloudera@quickstart ~]$
```

Permission to read, write and execute files on HDFS

Step 8: Run matrix multiplication MapReduce on Hadoop.

All the settings and files have been configured. Now, we're all set to run MapReduce for the multiplication of two matrices on Hadoop.

Execute the command below to start the MapReduce job.

```
hadoop jar /home/cloudera/hadoop-streaming-2.7.3.jar \
> -input /matrix_multiplication/matrix_inputs.txt \
> -output /matrix_multiplication/output \
> -mapper /home/cloudera/mapper_matrix.py \
> -reducer /home/cloudera/reducer_matrix.py
```

```
cloudera@quickstart:~$ hadoop jar /home/cloudera/hadoop-streaming-2.7.3.jar \
> -input /matrix_multiplication/matrix_inputs.txt \
> -output /matrix_multiplication/output \
> -mapper /home/cloudera/mapper_matrix.py \
> -reducer /home/cloudera/reducer_matrix.py
packageJobJar: [/usr/jars/hadoop-streaming-2.6.0-cdh5.4.2.jar] /tmp/streamjob8715669728874717245.jar tmpDir=
null
21/02/27 11:15:04 INFO client.RMProxy: Connecting to ResourceManager at quickstart.cloudera/10.0.2.15:8032
21/02/27 11:15:05 INFO client.RMProxy: Connecting to ResourceManager at quickstart.cloudera/10.0.2.15:8032
21/02/27 11:15:06 INFO mapred.FileInputFormat: Total input paths to process : 1
21/02/27 11:15:06 INFO mapreduce.JobSubmitter: number of splits:2
21/02/27 11:15:06 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1614441235029_0001
21/02/27 11:15:07 INFO impl.YarnClientImpl: Submitted application application_1614441235029_0001
21/02/27 11:15:07 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/applicati
on_1614441235029_0001/
21/02/27 11:15:07 INFO mapreduce.Job: Running job: job_1614441235029_0001
21/02/27 11:15:22 INFO mapreduce.Job: map 0% reduce 0%
21/02/27 11:15:39 INFO mapreduce.Job: map 100% reduce 0%
21/02/27 11:15:50 INFO mapreduce.Job: map 100% reduce 100%
21/02/27 11:15:51 INFO mapreduce.Job: Job job_1614441235029_0001 completed successfully
21/02/27 11:15:51 INFO mapreduce.Job: Counters: 49
File System Counters
  FILE: Number of bytes read=531
  FILE: Number of bytes written=348820
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=561
  HDFS: Number of bytes written=154
  HDFS: Number of read operations=9
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
Job Counters
  Launched map tasks=2
  Launched reduce tasks=1
```

```
Map output bytes=840
Map output materialized bytes=539
Input split bytes=246
Combine input records=0
Combine output records=0
Reduce input groups=16
Reduce shuffle bytes=539
Reduce input records=96
Reduce output records=16
Spilled Records=192
Shuffled Maps =2
Failed Shuffles=0
Merged Map outputs=2
GC time elapsed (ms)=328
CPU time spent (ms)=2680
Physical memory (bytes) snapshot=379772928
Virtual memory (bytes) snapshot=2100383744
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=315
File Output Format Counters
  Bytes Written=154
21/02/27 11:15:51 INFO streaming.StreamJob: Output directory: /matrix_multiplication/output
cloudera@quickstart:~$
cloudera@quickstart:~$ hdfs dfs -ls /matrix_multiplication/output
Found 2 items
-rw-r--r-- 1 hdfs supergroup 0 2021-02-27 11:15 /matrix_multiplication/output/ SUCCESS
-rw-r--r-- 1 hdfs supergroup 154 2021-02-27 11:15 /matrix_multiplication/output/part-00000
```

If you see the output on terminal as shown in the above two images, then the MapReduce job was executed successfully.

Step 9: Read the matrix multiplication MapReduce output.

The output for the MapReduce job for matrix multiplication has been written into HDFS inside `/matrix_multiplication/output` directory.

To check the output of the matrix multiplication between matrices A and B, run the following command.

```
hdfs dfs -cat /matrix_multiplication/output/part-00000
```

```
cloudera@quickstart:~$ hdfs dfs -cat /matrix_multiplication/output/part-00000
0,0,380
0,1,440
0,2,500
0,3,560
1,0,830
1,1,980
1,2,1130
1,3,1280
2,0,1280
2,1,1520
2,2,1760
2,3,2000
3,0,1730
3,1,2060
3,2,2390
3,3,2720
[cloudera@quickstart ~]$
```

Great, the output for matrix multiplication MapReduce job on Hadoop matches with the one previously executed locally.

Key-value pairs were produced for each element of matrix A and matrix B which were then mapped and reduced to the above final values which is nothing but the product of the two matrices. The resulting matrix is a 4x4 matrix.