

Hadoop Presentation Day – 2

Overview on Map-Reduce and an Example of Graphically Representing the number of Api Calls made w.r.t date by performing Map-Reduce on RSVP (Resource Reservation Protocol) Log using Matplotlib Library in Python.

Required Files for this session:

1. **mylog.txt** (Contains the log data of RSVP)
2. **ApiCallCount.java** (MapReduce function written in Java)
3. **ApiRequestManifest.txt** (included the name of the Main-Class to compile and create a jar file for Hadoop map-reduce job)
4. **ApiRequestGraphGenerator.py** (python script that reads the data from the map reduce output file and generates a graph using matplotlib library)

Note: All these files are added in the git repository:

https://github.com/DineshVasireddy/Docker_Hadoop

Pre-requisites:

1. Python, Matplotlib library:
 - a. If you have Python installed in your local system you can install the matplotlib library using “**pip install matplotlib**” in your terminal.
 - b. If the “pip” command does not work and you have python installed use this command first before step-a “**python get-pip.py**” and go back to step – a.
 - c. **Python** can be installed in your system (if not present) using this link: <https://www.python.org/downloads/> and clicking on “**Download Python 3.12.2**” button.
2. VS Code or any IDE that can run the python script.

Installation of Hadoop (Different Methods):

As currently we are using Hadoop with docker, I would like to mention it is not the only way to install Hadoop and work on it. other way to install Hadoop is through a VM (Virtual Machine) using Virtual Box:

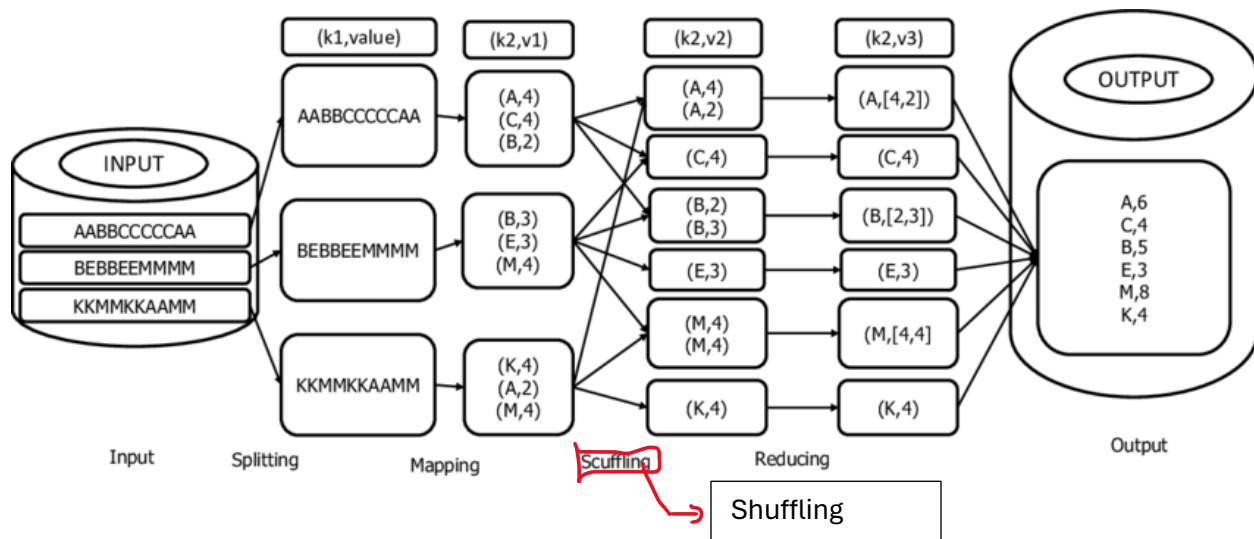
https://www.youtube.com/watch?v=Na0q6J_hi8M

You can also install Hadoop on your machine if you are on Debian-based systems like Ubuntu using “apt-get” or “yum” for Red-Hat based systems like CentOS.

More Information on Hadoop for setting up a Single Node Cluster can be found here:<https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html>

Start of the Session – 2

Map-Reduce Workflow Overview for Better Understanding:



Reference for detailed notes:

https://www.researchgate.net/publication/323845087_A_Blast_implementation_in_Hadoop_MapReduce_using_low_cost_commodity_hardware

More Information on Map-reduce and Examples can be found in this official Apache Hadoop documentation - https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

As we noticed lot of our peers' facing problems in understanding the Linux commands we are being used in our previous sessions and that we will be using in current sessions as well we would like to post this referral link of Linux commands cheat-sheet:

<https://www.hostinger.com/tutorials/linux-commands>

Note: The only difference between the general Linux commands and the commands used to access hdfs file system in that container is “**hdfs dfs -**” before the command you want to execute.

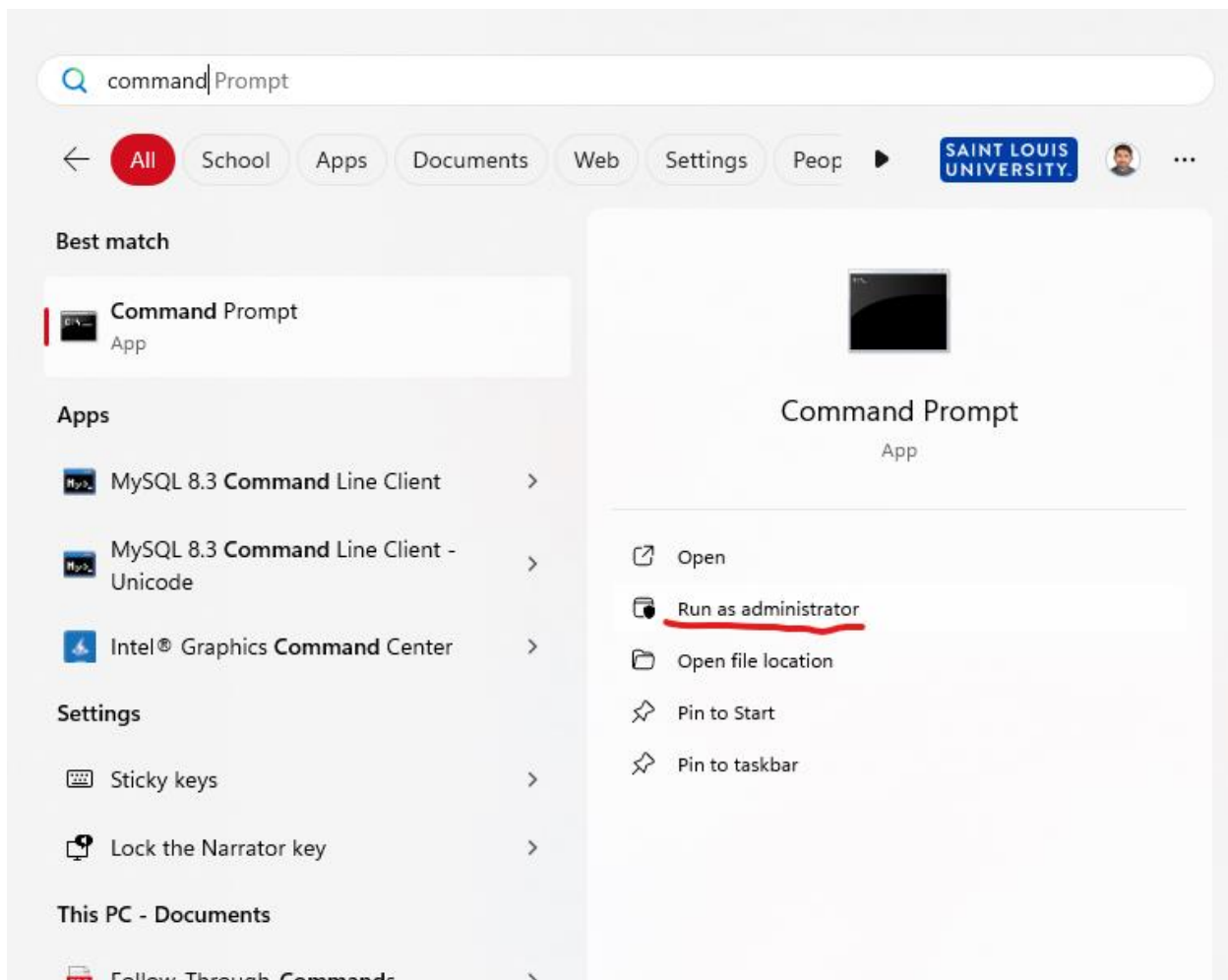
Implementation of Map-Reduce and Python-Script on the Map-Reduce output file for Graphical Representation

To make everything easy we have placed all the new files required in the main directory of the repository.

Step-1 (Moving Files from the Repository to Docker container namenode):

1. We need to open the terminal as an administrator:

Windows:



Macusers: you can open the terminal in a normal way and later perform the commands that require admin privileges using “**sudo**” command we will be talking about in detail about this later by the end of this session.

2. **Now**, we need to move these following files from the Repository to the namenode Container in our docker:

- a. mylog.txt
- b. ApiCallCount.java
- c. ApiRequestManifest.txt
- d. ApiRequestGraphGenerator.py

Using the command:

“docker cp <path_to_source> namenode:<path_to_destination>”

3. After moving every file into the namenode container you can execute **“docker exec -it namenode bash”**

(or)

“docker exec -i -t namenode bash”

to access the namenode container using bash as a root user

4. Now navigate to the path you have copied these files using

“cd <path_to_files_copied>”

5. Now perform “**ls**” operation and you will see similar output:

```
root@2a6e8af0e6bb:/ApiCallCount# ls
ApiCallCount.java  ApiRequestGraphGenerator.py  ApiRequestManifest.txt  mylog.txt
```

6. Now we will be moving the input file, Which in this case it is **“mylog.txt”** to the hdfs file system inside the namenode container to the path we created yesterday using this command:

“hdfs dfs -put mylog.txt <path_to_inputfolder_created_in_hdfs>”

Example: `hdfs dfs -put mylog.txt /users/saidinesh/Input`

Where **“/users/saidinesh/Input”** is the path I created my input folder in hdfs

7. Once we copy the file into the hdfs we need to compile the ApiCallCount.java file using this command:

```
"javac -classpath $HADOOP_HOME/share/hadoop/common/hadoop-common-3.2.1.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.2.1.jar ApiCallCount.java "
```

8. After you perform the above command and perform "ls" command now you will see following output:

```
root@2a6e8af0e6bb:/ApiCallCount# ls
ApiCallCount$IntSumReducer.class  ApiCallCount$TokenizerMapper.class  ApiCallCount.class  ApiCallCount.java  ApiRequestGraphGenerator.py  ApiRequestManifest.txt  mylog.txt
```

The image is not clear so I will be performing "ls -l" command to get the vertical output with detailed information on contents of the folder.

```
root@2a6e8af0e6bb:/ApiCallCount# ls -l
total 56
-rw-r--r-- 1 root root 1748 Mar 21 02:26 ApiCallCount$IntSumReducer.class
-rw-r--r-- 1 root root 1836 Mar 21 02:26 ApiCallCount$TokenizerMapper.class
-rw-r--r-- 1 root root 1507 Mar 21 02:26 ApiCallCount.class
-rwxr-xr-x 1 root root 2310 Mar 21 00:23 ApiCallCount.java
-rwxr-xr-x 1 root root 473 Mar 21 00:49 ApiRequestGraphGenerator.py
-rwxr-xr-x 1 root root 24 Mar 20 23:49 ApiRequestManifest.txt
-rwxr-xr-x 1 root root 30244 Mar 21 00:31 mylog.txt
root@2a6e8af0e6bb:/ApiCallCount#
```

9. As it can be seen new .class files have been created now we create the jar executable file for our hadoop mapreduce job using following command:

```
"jar cvmf ApiRequestManifest.txt ApiCallCount.jar *.class"
```

Where "ApiCallCount.jar" is the name of the jar file I want it to be created as.

Now perform "ls" operation again to confirm that the file has been Created and showing as follows:

```
root@2a6e8af0e6bb:/ApiCallCount# jar cvmf ApiRequestManifest.txt ApiCallCount.jar *.class
added manifest
adding: ApiCallCount$IntSumReducer.class(in = 1748) (out= 743)(deflated 57%)
adding: ApiCallCount$TokenizerMapper.class(in = 1836) (out= 805)(deflated 56%)
adding: ApiCallCount.class(in = 1507) (out= 819)(deflated 45%)
root@2a6e8af0e6bb:/ApiCallCount# ls
ApiCallCount$IntSumReducer.class  ApiCallCount$TokenizerMapper.class  ApiCallCount.class  ApiCallCount.jar
```

10. Now we have our Input file in hdfs and also the jar file to run hadoop job so we can run following command to execute the mapreduce job and store the output:

“hadoop jar ApiCallCount.jar ApiCallCount <path_to_input_file> <path_to_store_output>”

My Example:

**hadoop jar ApiCallCount.jar ApiCallCount /user/saidinesh/Input/mylog.txt
/user/saidinesh/Output/ApiCallCountOutput/**

11. After performing the above command you will see the job being executed and once it is done your screen will look like this:

```
Total megabyte-milliseconds taken by all reduce tasks=11427840
Map-Reduce Framework
  Map input records=374
  Map output records=75
  Map output bytes=750
  Map output materialized bytes=43
  Input split bytes=116
  Combine input records=75
  Combine output records=4
  Reduce input groups=4
  Reduce shuffle bytes=43
  Reduce input records=4
  Reduce output records=4
  Spilled Records=8
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=105
  CPU time spent (ms)=560
  Physical memory (bytes) snapshot=535560192
  Virtual memory (bytes) snapshot=13573963776
  Total committed heap usage (bytes)=472383488
  Peak Map Physical memory (bytes)=310104064
  Peak Map Virtual memory (bytes)=5113184256
  Peak Reduce Physical memory (bytes)=225456128
  Peak Reduce Virtual memory (bytes)=8460779520
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=30244
File Output Format Counters
  Bytes Written=35
root@2a6e8af0e6bb:/ApiCallCount#
```

If you see similar screen it means that the job has run successfully and you can find your output file under the path as “part-r-00000”

12. Now copy this file back to the container namenode from hdfs and later copy the file from the container to local system using following commands:

a. **hdfs dfs -get <path_to_part-r-00000-file-inHdfs>**

b. rename the file using this command:

“mv part-r-00000 mapreduce_output.txt”

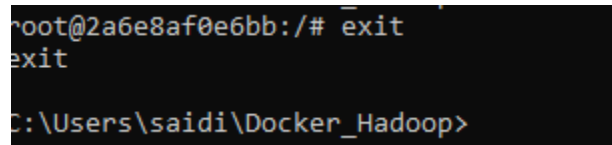
c. Now exit the container using one of the following commands (whatever works for you):

i. Ctrl + z **(key input)**

ii. Cmd + z **(key input)**

iii. exit **(command)**

13. Now you will find yourself outside container in your local system path like this



```
root@2a6e8af0e6bb:/# exit
exit
C:\Users\saidi\Docke_Hadoop>
```

14. Now copy the output file into this repository path using this command:

“docker cp namenode:<path_to_mapreduce_output.txt_file> <path_to_home_dir_repo>”

To perform above operation we need admin privileges

If on Mac perform the above command as follows if it does not work:

“sudo docker cp namenode:<path_to_mapreduce_output.txt_file> <path_to_home_dir_repo>”

15. Now after you copy the output file into your local system and specifically inside the home directory of the repository, There is a .py file named “ApiRequestGraphGenerator.py” run this script using any IDLE and you will see the API requests count made w.r.t the date as a graph.

Note: Not adding the output of the python script (graph) image in this document to display it in live class

Thanks to Everyone for listening to our presentation. Hope we have created an Interest in learning Hadoop atleast among one of our Peer. As Hadoop is a vast application and deals mostly with Linux concepts. Learning Hadoop needs to start with the understanding of Linux commands. We Have added the Learning Notes created by my Peer “Josh” as he was learning this concept for the first time and we feel this document might help more new learners and Added it as “**Hadoop Learning Notes**” in the repository. It might not contain all the information but it certainly contains the basics.

Note: The Hadoop Learning Notes.pdf in the repo is the notes prepared by Josh as a Speaking notes so it will contain a speech text. But it helps in understanding the things as a new learner.

Thank you.

Special Thanks:

Professor. Md. Mainul Islam Mamun

TA. Mr. Abhilash Akula

For giving us this opportunity.