

# Assignment 1

By

Vishwajeet Ekal [vekal@iu.edu](mailto:vekal@iu.edu)

## Hadoop Installation:

Below are the steps that I performed for the Hadoop installation:

- Step 1: Create a VM of Ubuntu for installing Hadoop.
- Step 2: Updated JDK 11 as a prerequisite for running Hadoop.
- Step 3: Downloaded Hadoop 3.3.6.
- Step 4: I updated configuration files such as `hdfs-site.xml`, `mapred-site.xml`, they contain important settings for Hadoop's distributed file system and MapReduce framework.
- Step 5: I setup environment variables such as `JAVA_HOME`, `HADOOP_HOME` in `.bashrc`.
- Step 6: I started Hadoop with `start-all.sh` script. This script includes NameNode, DataNode, NodeManager, ResourceManager startup process.

```
91 alias ll='ls -alF'
92 alias la='ls -A'
93 alias l='ls -CF'
94
95 # Add an "alert" alias for long running commands.  Use like so:
96 #   sleep 10; alert
97 alias alert='notify-send --urgency=low -i "${[ $? = 0 ]} && echo terminal || echo error" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\''")"'
98
99 # Alias definitions.
100 # You may want to put all your additions into a separate file like
101 # ~/.bash_aliases, instead of adding them here directly.
102 # See /usr/share/doc/bash-doc/examples in the bash-doc package.
103
104 if [ -f ~/.bash_aliases ]; then
105     . ~/.bash_aliases
106 fi
107
108 # enable programmable completion features (you don't need to enable
109 # this, if it's already enabled in /etc/bash.bashrc and /etc/profile
110 # sources /etc/bash.bashrc).
111 if ! shopt -oq posix; then
112     if [ -f /usr/share/bash-completion/bash_completion ]; then
113         . /usr/share/bash-completion/bash_completion
114     elif [ -f /etc/bash_completion ]; then
115         . /etc/bash_completion
116     fi
117 fi
118
119 export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
120 export HADOOP_HOME=/home/hadoop/hadoop
121 export HADOOP_INSTALL=$HADOOP_HOME
122 export HADOOP_MAPRED_HOME=$HADOOP_HOME
123 export HADOOP_COMMON_HOME=$HADOOP_HOME
124 export HADOOP_HDFS_HOME=$HADOOP_HOME
125 export HADOOP_YARN_HOME=$HADOOP_HOME
126 export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
127 export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
128 export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

## Environment variables

```
hadoop@ECC:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [ECC]
Starting resourcemanager
Starting nodemanagers
hadoop@ECC:~$
```

## Start-all.sh Output

```

hadoop@ECC:~$ jps
2093 DataNode
3938 Jps
3202 SecondaryNameNode
3573 NodeManager
3433 ResourceManager
2863 NameNode
hadoop@ECC:~$

```

All process Up and Running

Overview 'localhost:9000' (✓active)

<b>Started:</b>	Sat Mar 09 19:42:48 -0500 2024
<b>Version:</b>	3.3.6, r1be78238728da9266a4f88195058f08fd012bf9c
<b>Compiled:</b>	Sun Jun 18 04:22:00 -0400 2023 by ubuntu from (HEAD detached at release-3.3.6-RC1)
<b>Cluster ID:</b>	CID-b22e9052-97d4-41e4-8866-9345f69a1592
<b>Block Pool ID:</b>	BP-1986937257-127.0.1.1-1709677560910

**Summary**

Security is off.  
 Safemode is off.  
 41 files and directories, 25 blocks (25 replicated blocks, 0 erasure coded block groups) = 66 total filesystem object(s).  
 Heap Memory used 166.58 MB of 273 MB Heap Memory. Max Heap Memory is 869.5 MB.  
 Non Heap Memory used 52.14 MB of 53.56 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

<b>Configured Capacity:</b>	28.87 GB
-----------------------------	----------

Hadoop UI

**All Applications**

Apps Running	Apps Completed	Containers Running	Used Resources	Total Res
0	0	0	<memory:0 B, vCores:0>	<memory:8 GB, vCores:4>

Decommissioning Nodes	Decommissioned Nodes	Lost Nodes
0	0	0

Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory
0	0	0	0	0	0 B

Scheduling Resource Type	Minimum Allocation	Maximum Allocation
mb (unit=Mi), vcores	<memory:1024, vCores:1>	<memory:8192, vCores:4>

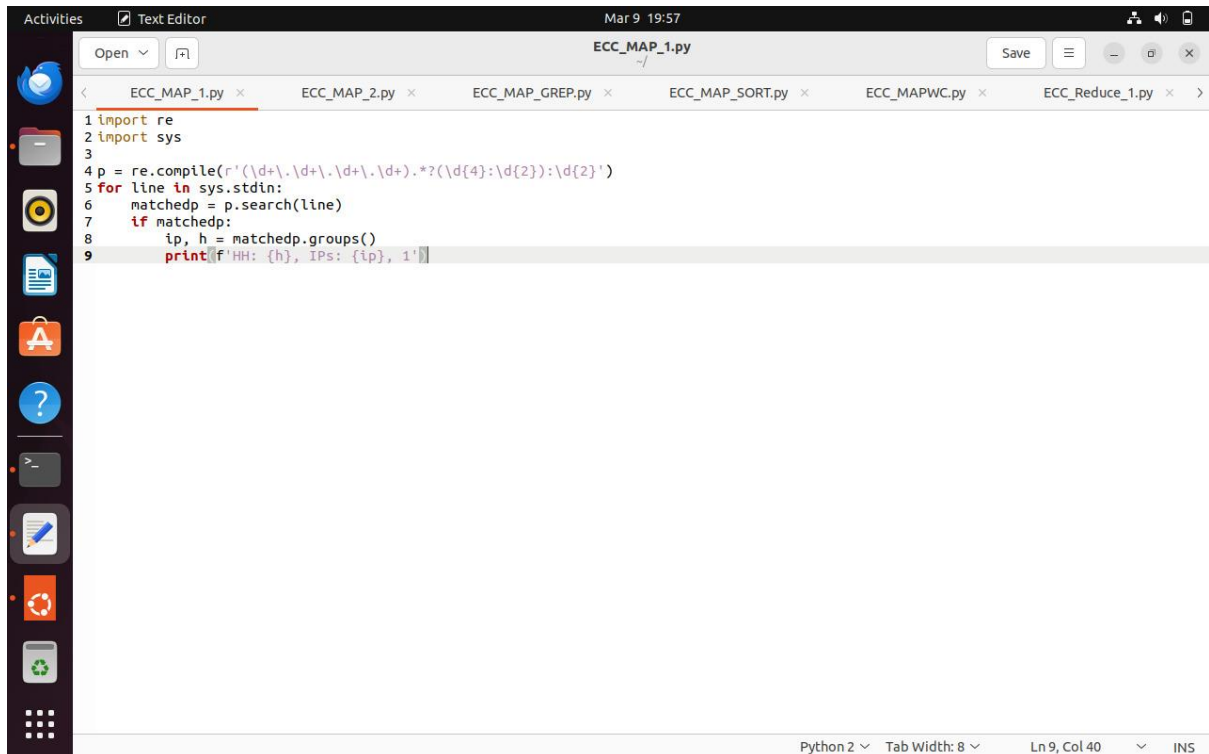
Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory
No data available in table										

Hadoop Cluster

## ❖ PART 1. Output the top-3 IP addresses with the granularity of an hour

Mapper (ECC\_MAP\_1.py):

- Below mapper code searches and extracts IP addresses and hours from input log file.
- For each match it prints the hour and IP along 1 to indicate the count which is passed to reducer.



```
1 import re
2 import sys
3
4 p = re.compile(r'(\d+\.\d+\.\d+\.\d+).*?(\d{4}:\d{2}):\d{2}')
5 for line in sys.stdin:
6     matchedp = p.search(line)
7     if matchedp:
8         ip, h = matchedp.groups()
9         print(f'HH: {h}, IPs: {ip}, 1')
```

ECC\_MAP\_1.py

Reducer ((ECC\_Reduce\_1.py):

- It aggregates the count of each IP address for specific hour, then sorts the IP addresses according to their count in descending order.
- Finally prints the top 3 IPs with their count for each hour.

```
1 import sys
2 from collections import defaultdict
3
4 hstats = defaultdict(lambda: defaultdict(int))
5
6 for line in sys.stdin:
7     line = line.strip()
8     h, ip, c = line.split(',')
9
10    hstats[h][ip] += int(c)
11
12 for h, s in hstats.items():
13     hdiv = h.split(':')
14     hour = hdiv[2]
15
16     resIPs = sorted(s.items(), key=lambda x: x[1], reverse=True)[:3]
17
18     for ip, c in resIPs:
19         print("-----")
20         print(f"TOTAL COUNT: {c}, {ip}, Hour: {hour}:00")
21 print("-----")
```

## ECC\_Reduce\_1.py

Command to run Map Reduce:

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -files  
ECC_MAP_1.py,ECC_Reduce_1.py -mapper 'python3 ECC_MAP_1.py' -reducer 'python3  
ECC_Reduce_1.py' -input /home/hadoop/sample.log -output /home/hadoop/ECC_OUTPUT/
```

Logs:

```
hadoop@ECC: ~
hadoop@ECC:~$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -files ECC_MAP_1.py,ECC_Reduce_1.py -mapper 'python3 ECC_MAP_1.py' -reducer 'python3 ECC_Reduce_1.py' -input /home/hadoop/sample.log -output /home/hadoop/ECC_OUTPUT/
packageJobJar: [/tmp/hadoop-unjar497173251588385524/] [] /tmp/streamjob1228383590632640183.jar tmpDir=null
2024-03-09 20:18:48,545 INFO client.DefaultHadoopFollowerProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-03-09 20:18:48,890 INFO client.DefaultHadoopFollowerProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-03-09 20:18:49,186 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/staging/job_1710031384272_0003
2024-03-09 20:18:49,584 INFO mapreduce.FileInputFormat: Total input files to process : 1
2024-03-09 20:18:50,061 INFO mapreduce.JobSubmitter: number of splits:2
2024-03-09 20:18:50,210 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1710031384272_0003
2024-03-09 20:18:50,211 INFO mapreduce.JobSubmitter: Executing with Tokens: []
2024-03-09 20:18:50,400 INFO conf.Configuration: resource-types.xml not found
2024-03-09 20:18:50,400 INFO resource.ResourceUtil: Unable to find 'resource-types.xml'
2024-03-09 20:18:50,402 INFO impl.YarnClientImpl: Submitted application application_1710031384272_0003
2024-03-09 20:18:50,531 INFO mapreduce.Job: The url to track the job: http://ECC.myquest.virtualbox.org:8088/proxy/application_1710031384272_0003/
2024-03-09 20:18:50,532 INFO mapreduce.Job: Running job: job_1710031384272_0003
2024-03-09 20:18:50,533 INFO mapreduce.Job: Job job_1710031384272_0003 running in uber mode : false
2024-03-09 20:18:50,533 INFO mapreduce.Job: map 0% reduce 0%
2024-03-09 20:18:50,533 INFO mapreduce.Job: map 100% reduce 0%
2024-03-09 20:18:50,533 INFO mapreduce.Job: map 100% reduce 100%
2024-03-09 20:18:50,533 INFO mapreduce.Job: map 100% reduce 100%
2024-03-09 20:18:50,533 INFO mapreduce.Job: Job job_1710031384272_0003 completed successfully
2024-03-09 20:18:50,533 INFO mapreduce.Job: Counters: 54
File System Counters
  FILE: Number of bytes read=11882
  FILE: Number of bytes written=863383
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=106487
  HDFS: Number of bytes written=379
  HDFS: Number of read operations=11
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
  HDFS: Number of bytes read erasure-coded=0
Job Counters
  Launched map tasks=2
  Launched reduce tasks=1
Data-local map tasks=2
Total time spent by all maps in occupied slots (ms)=7251
Total time spent by all reduces in occupied slots (ms)=3599
Total time spent by all map tasks (ms)=7251
Total time spent by all reduce tasks (ms)=3599
Total vcore-millisecods taken by all map tasks=7251
Total vcore-millisecods taken by all reduce tasks=3599
Total megabyte-millisecods taken by all map tasks=7425024
Total megabyte-millisecods taken by all reduce tasks=3685376
Map-Reduce Framework
  Map input records=319
  Map output records=319
  Map output bytes=11228
  Map output materialized bytes=11888
  Input split bytes=192
  Combine input records=0
  Combine output records=0
  Reduce input groups=46
  Reduce shuffle bytes=11888
  Reduce input records=319
  Reduce output records=7
  Spilled Records=58
  Shuffled Maps =2
  Failed Shuffles=0
  Merged Map outputs=2
GC time elapsed (ms)=310
CPU time spent (ms)=2450
Physical memory (bytes) snapshot=882515968
Virtual memory (bytes) snapshot=7716880384
```

## Logs

Command to check output: `hdfs dfs -cat /home/hadoop/ECC_OUTPUT/part-00000`

Final Output:

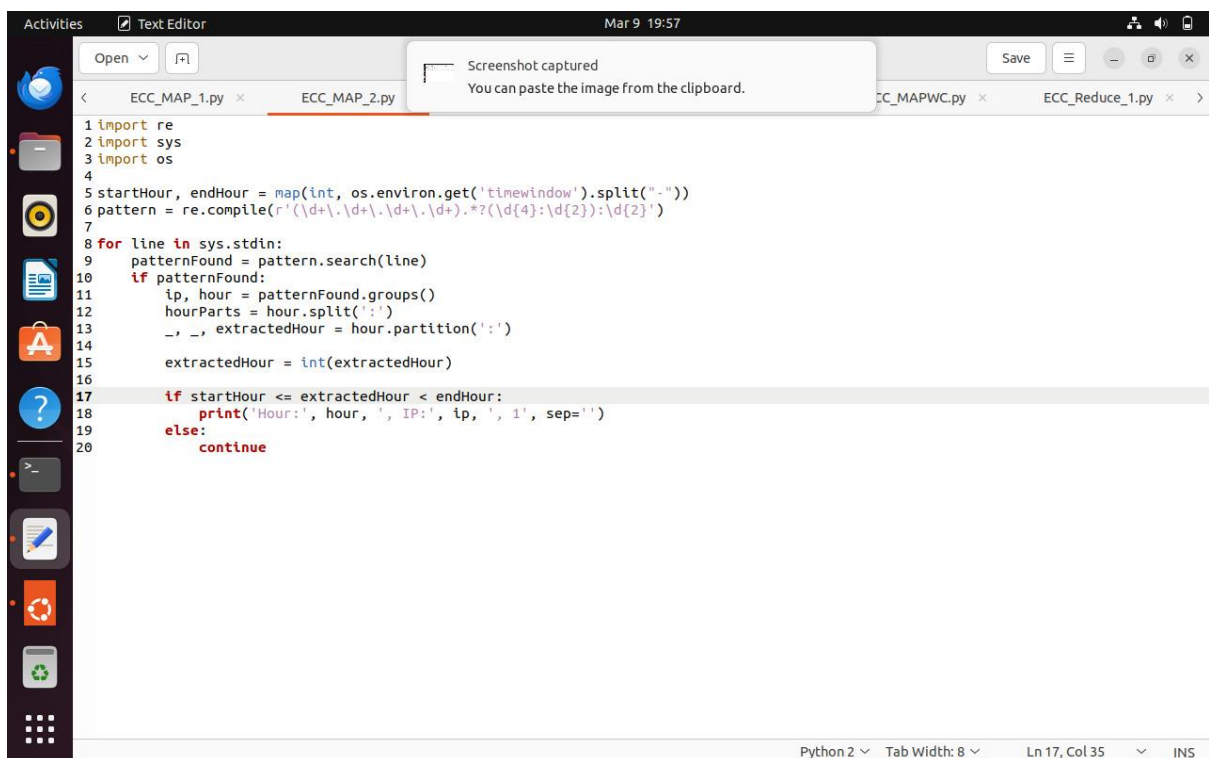
```
hadoop@ECC:~$ hdfs dfs -cat /home/hadoop/ECC_OUTPUT/part-00000
-----
TOTAL COUNT: 38, IPs: 66.111.54.249, Hour: 03:00
-----
TOTAL COUNT: 36, IPs: 5.211.97.39, Hour: 03:00
-----
TOTAL COUNT: 31, IPs: 66.249.66.194, Hour: 03:00
-----
hadoop@ECC:~$
```

## Final Output

- ❖ PART 2.1: Make your program like a database search. Your program should be able to accept parameters from users, such as 0-1, which means from time 00:00 to 01:00, and output the top 3 IP addresses in the given time period.

Mapper (ECC\_MAP\_2.py):

- I have modified the mapper to accept command line argument called time window.
- It is extracting the start hour and end hour from the time window mentioned in command line.
- And based on this, it filters and print entries that fall within the specified time window.



```
1 import re
2 import sys
3 import os
4
5 startHour, endHour = map(int, os.environ.get('timewindow').split("-"))
6 pattern = re.compile(r'(\d+\.\d+\.\d+\.\d+).*?(\d{4}):\d{2}:\d{2}')
7
8 for line in sys.stdin:
9     patternFound = pattern.search(line)
10    if patternFound:
11        ip, hour = patternFound.groups()
12        hourParts = hour.split(':')
13        _, _, extractedHour = hour.partition(':')
14
15        extractedHour = int(extractedHour)
16
17    if startHour <= extractedHour < endHour:
18        print('Hour:', hour, 'IP:', ip, '1', sep='')
19    else:
20        continue
```

## ECC\_MAP\_2.py

### Reducer (ECC\_Reduce\_1.py):

- Reducer for this part is the same as part 1.
- I have taken two test cases; first time window is 00 to 02 which has no IP addresses and second time window is 01 to 04 which has IP addresses.

Test case 1 Time Window 00 to 02:

### Command to run Map Reduce:

```
$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -files  
ECC_MAP_2.py,ECC_Reduce_1.py -mapper 'python3 ECC_MAP_2.py' -reducer 'python3  
ECC_Reduce_1.py' -input /home/hadoop/sample.log -output /home/hadoop/REDUCE02/ -  
cmdenv timewindow="00-02"
```

Logs:

The image shows a terminal window with a dark theme. The title bar at the top indicates the date and time as 'Mar 9 21:32'. The terminal prompt is 'hadoop@ECC: ~'. The output of a command is displayed, showing logs for a Hadoop job. The logs include information about the client, resource manager, and the job's progress. The job is identified by ID 'job\_1710837867746\_0001'. The logs show that the job is running and has completed successfully. The output also includes a section for 'File System Counters' and 'Job Counters'.

```
packageJobJar: [/tmp/hadoop-unjar-421207755108730267/] [] /tmp/streamjob6233392806669497725.jar tmpDir=null
2024-03-09 21:31:43.989 INFO client.DefaultHadoopFallbackProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-03-09 21:31:44.255 INFO client.DefaultHadoopFallbackProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-03-09 21:31:46.629 INFO mapreduce.JobResourceMonitor: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1710837867746_0001
2024-03-09 21:31:46.944 INFO mapreduce.FileInputFormat: Total input files to process : 1
2024-03-09 21:31:48.150 INFO mapreduce.JobSubmitter: number of splits:2
2024-03-09 21:31:48.383 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1710837867746_0001
2024-03-09 21:31:48.383 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-03-09 21:31:48.651 INFO conf.Configuration: resource-types.xml not found
2024-03-09 21:31:48.612 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-03-09 21:31:49.112 INFO impl.YarnClientImpl: Submitted application application_1710837867746_0001
2024-03-09 21:31:49.182 INFO mapreduce.Job: The url to track the job: http://ECC-mysite.virtuallab.org:8088/proxy/application_1710837867746_0001/
2024-03-09 21:31:49.184 INFO mapreduce.Job: Running job: job_1710837867746_0001
2024-03-09 21:31:50.110 INFO mapreduce.Job: map 1710837867746_0001 running in user mode : false
2024-03-09 21:31:50.240 INFO mapreduce.Job: map 0% reduce 0%
2024-03-09 21:32:05.496 INFO mapreduce.Job: map 100% reduce 0%
2024-03-09 21:32:11.542 INFO mapreduce.Job: map 100% reduce 100%
2024-03-09 21:32:11.560 INFO mapreduce.Job: Job job_1710837867746_0001 completed successfully
2024-03-09 21:32:12.693 INFO mapreduce.Job: Counters: 54

File System Counters
  FILE: Number of bytes read=6
  FILE: Number of bytes written=839876
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=106687
  HDFS: Number of bytes written=0
  HDFS: Number of read operations=11
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
  HDFS: Number of bytes read erasure-coded=0

Job Counters
  Launched map tasks=2
  Launched reduce tasks=1
  Data-local map tasks=2
  Total time spent by all maps in occupied slots (ms)=9761
  Total time spent by all reduces in occupied slots (ms)=3208
  Total time spent by all map tasks (ms)=9761
  Total time spent by all reduce tasks (ms)=3208
  Total vcore-millisecods taken by all map tasks=9761
  Total vcore-millisecods taken by all reduce tasks=3208
  Total megabyte-millisecods taken by all map tasks=995264
  Total megabyte-millisecods taken by all reduce tasks=3284992

Map-Reduce Framework
  Map input records=319
  Map output records=0
  Map output bytes=0
  Map output materialized bytes=12
  Input split bytes=192
  Combine input records=0
  Combine output records=0
  Reduce input groups=0
  Reduce shuffle bytes=12
  Reduce input records=0
  Reduce output records=0
  Spilled Records=0
  Shuffled Maps=2
```

## Logs

Command to check output: `hdfs dfs -cat /home/Hadoop/REDUC02/part-00000`

### Final Output:

```
hadoop@ec2-54-185-100-100:~$ hadoop@ECC:~$  
hadoop@ECC:~$  
hadoop@ECC:~$ hdfs dfs -cat /home/hadoop/REDUCE02/part-00000  
hadoop@ECC:~$
```

## Final Output



### Command to run Map Reduce:

```
$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -files ECC_MAP_2.py,ECC_Reduce_1.py -mapper 'python3 ECC_MAP_2.py' -reducer 'python3 ECC_Reduce_1.py' -input /home/hadoop/sample.log -output /home/hadoop/REDUCE14 -cmdenv timewindow="01-04"
```

Logs:

```
Activities Terminal Mar 9 22:27

hadoop@ECC: ~

hadoop@ECC: ~
hadoop@ECC: ~

hadoop@ECC: ~$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -files ECC_MAP_2.py,ECC_Reduce_1.py -mapper 'python3 ECC_MAP_2.py' -reducer 'python3 ECC_Reduce_1.py' -input /home/hadoop/sample.log -output /home/hadoop/REDUCE14/ -cmdenv timeout=60 -packageJobJar: [/tmp/hadoop-unjar5148840826989154284/] [] /tmp/streamjob3847983044124539376.jar tmpDir=null
2024-03-09 22:26:05,303 INFO client.DefaultHadoopRMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-03-09 22:26:05,549 INFO client.DefaultHadoopRMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-03-09 22:26:05,817 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1710040267176_0003
2024-03-09 22:26:06,247 INFO mapred.FileInputFormat: Total input files to process : 1
2024-03-09 22:26:06,326 INFO mapreduce.JobSubmitter: number of splits:2
2024-03-09 22:26:06,497 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1710040267176_0003
2024-03-09 22:26:06,497 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-03-09 22:26:06,697 INFO conf.Configuration: resource-types.xml not found
2024-03-09 22:26:06,697 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-03-09 22:26:06,777 INFO impl.YarnClientImpl: Submitted application application_1710040267176_0003
2024-03-09 22:26:06,844 INFO mapreduce.Job: The url to track the job: http://ECC.myguest.virtualbox.org:8088/proxy/application_1710040267176_0003/
2024-03-09 22:26:06,846 INFO mapreduce.Job: Running job: job_1710040267176_0003
2024-03-09 22:26:15,018 INFO mapreduce.Job: Job job_1710040267176_0003 running in uber mode : false
2024-03-09 22:26:15,019 INFO mapreduce.Job: map 0% reduce 0%
2024-03-09 22:26:21,145 INFO mapreduce.Job: map 50% reduce 0%
2024-03-09 22:26:22,151 INFO mapreduce.Job: map 100% reduce 0%
2024-03-09 22:26:28,208 INFO mapreduce.Job: map 100% reduce 100%
2024-03-09 22:26:29,224 INFO mapreduce.Job: Job job_1710040267176_0003 completed successfully
2024-03-09 22:26:29,352 INFO mapreduce.Job: Counters: 54

File System Counters
  FILE: Number of bytes read=11563
  FILE: Number of bytes written=862990
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=106687
  HDFS: Number of bytes written=373
  HDFS: Number of read operations=11
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
  HDFS: Number of bytes read erasure-coded=0

Job Counters
  Launched map tasks=2
  Launched reduce tasks=1
  Data-local map tasks=2
  Total time spent by all maps in occupied slots (ms)=8155
  Total time spent by all reduces in occupied slots (ms)=3589
  Total time spent by all map tasks (ms)=8155
  Total time spent by all reduce tasks (ms)=3589
  Total vcore-milliseonds taken by all map tasks=8155
  Total vcore-milliseonds taken by all reduce tasks=3589
  Total megabyte-milliseonds taken by all map tasks=8350720
  Total megabyte-milliseonds taken by all reduce tasks=3589
```

## Logs

Command to check output: `hdfs dfs -cat /home/Hadoop/REDUC14/part-00000`

### Final Output:

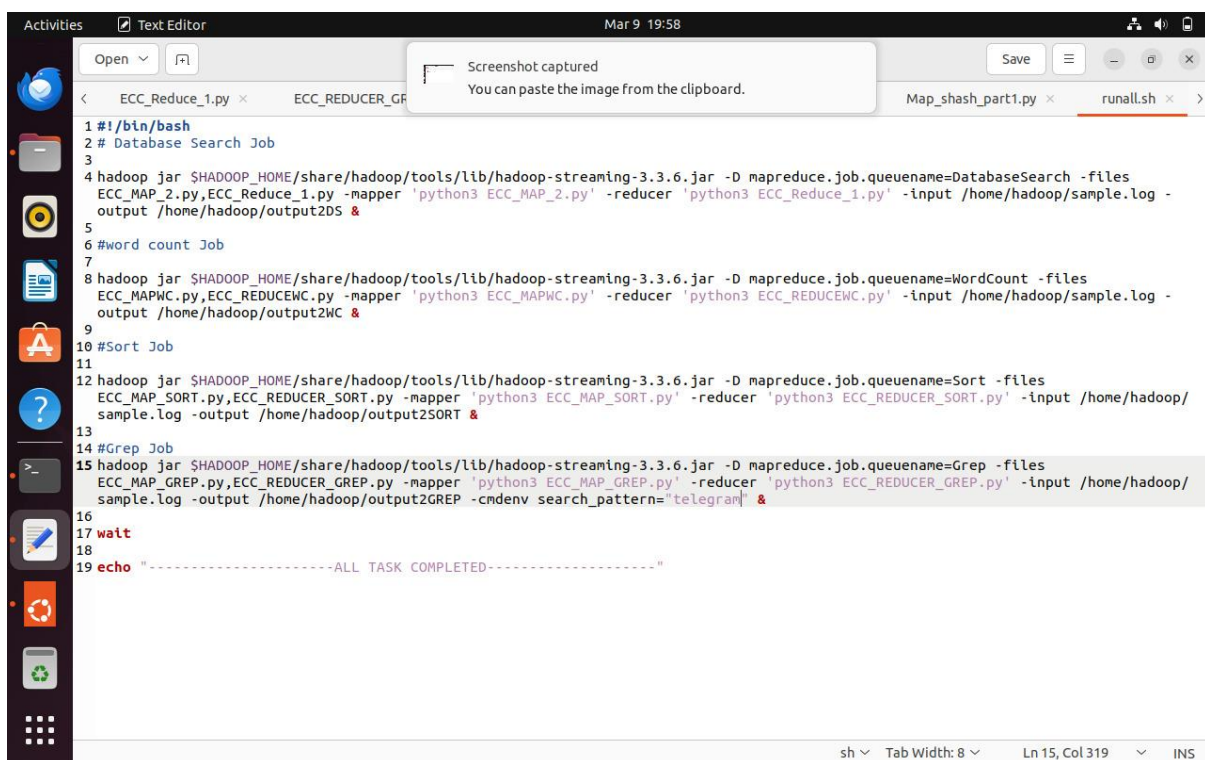
```
hadoop@ECC:~$ hdfs dfs -cat /home/hadoop/REDUCE14/part-00000
-----
TOTAL COUNT: 38, IP:66.111.54.249, Hour: 03:00
-----
TOTAL COUNT: 36, IP:5.211.97.39, Hour: 03:00
-----
TOTAL COUNT: 31, IP:66.249.66.194, Hour: 03:00
-----
hadoop@ECC:~$
```

## Final Output

## ❖ PART 2.2: Run it along with three other examples, WordCount, Sort, Grep, at the same time, and test fair and capacity schedulers.

Running all task:

- To run all the task in parallel that are wordcount, sort, database search and grep I developed a script called runAll.sh.
- I ran these jobs individually and added them in script with & at the end to make it run in background and concurrently.
- I have also added individual mapper and reducer code and output for each of the above task.
- I have added each queue name in each command with the flag -D mapreduce.job.queueName. Below is the screenshot of runall.sh.
- 



```
1 #!/bin/bash
2 # Database Search Job
3
4 hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -D mapreduce.job.queueName=DatabaseSearch -files
ECC_MAP_2.py,ECC_Reduce_1.py -mapper 'python3 ECC_MAP_2.py' -reducer 'python3 ECC_Reduce_1.py' -input /home/hadoop/sample.log -
output /home/hadoop/output2DS &
5
6 #word count Job
7
8 hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -D mapreduce.job.queueName=WordCount -files
ECC_MAPWC.py,ECC_REDUCEWC.py -mapper 'python3 ECC_MAPWC.py' -reducer 'python3 ECC_REDUCEWC.py' -input /home/hadoop/sample.log -
output /home/hadoop/output2WC &
9
10 #Sort Job
11
12 hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -D mapreduce.job.queueName=Sort -files
ECC_MAP_SORT.py,ECC_REDUCER_SORT.py -mapper 'python3 ECC_MAP_SORT.py' -reducer 'python3 ECC_REDUCER_SORT.py' -input /home/hadoop/
sample.log -output /home/hadoop/output2SORT &
13
14 #Grep Job
15 hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -D mapreduce.job.queueName=Grep -files
ECC_MAP_GREP.py,ECC_REDUCER_GREP.py -mapper 'python3 ECC_MAP_GREP.py' -reducer 'python3 ECC_REDUCER_GREP.py' -input /home/hadoop/
sample.log -output /home/hadoop/output2GREP -cmdenv search_pattern='telegram' &
16
17 wait
18
19 echo "-----ALL TASK COMPLETED-----"
```

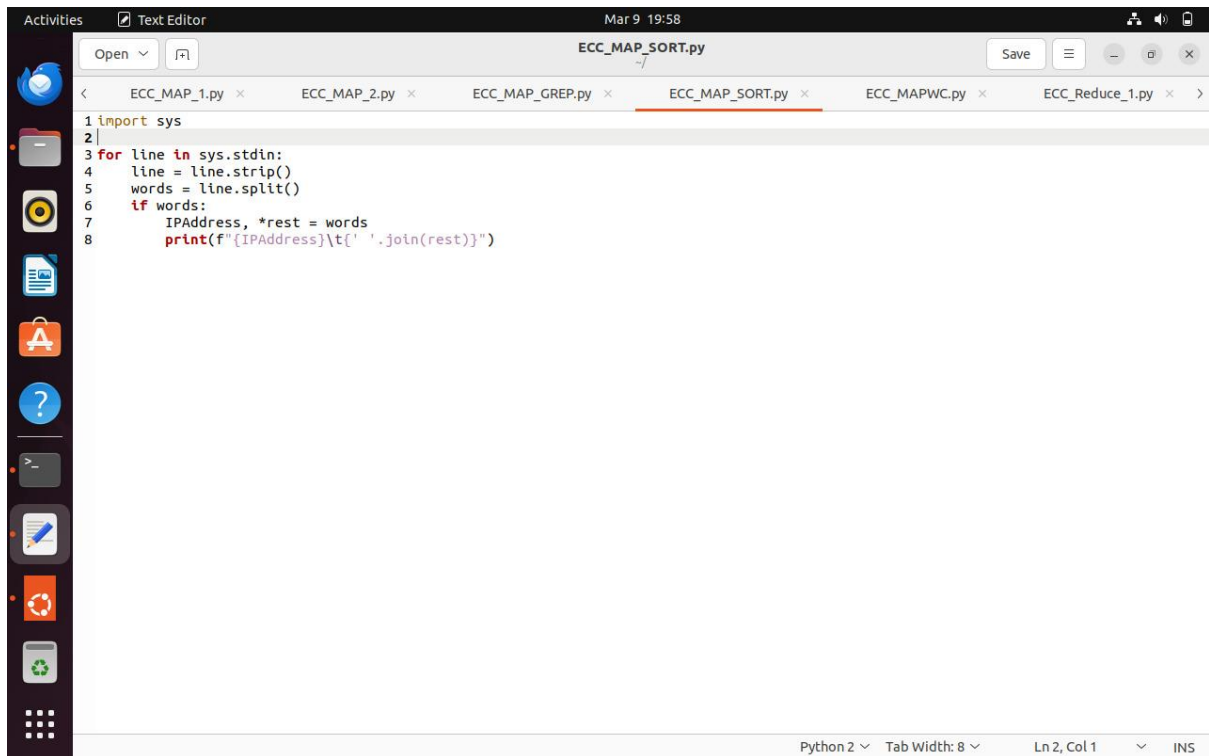
runall.sh

## ❖ Sort Task:

Mapper Sort (ECC\_MAP\_SORT.py):

- It extracts the first word as IP and remaining words as string.
- It then prints IP along with rest words in tab-separated format and sends it to reducer.



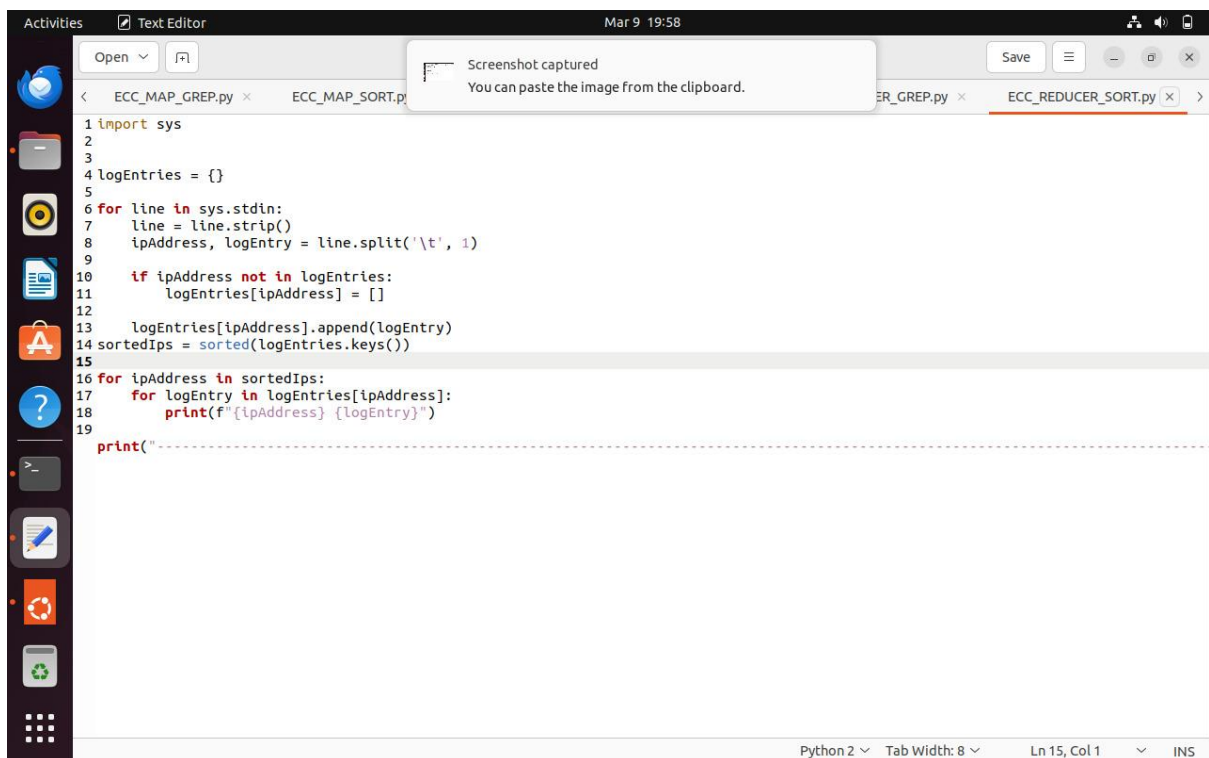


```
1 import sys
2
3 for line in sys.stdin:
4     line = line.strip()
5     words = line.split()
6     if words:
7         IPAddress, *rest = words
8         print(f'{IPAddress}\t{'.join(rest)}')
```

ECC\_MAP\_SORT.py

Reducer Sort (ECC\_REDUCER\_SORT.py):

- It stores log entries in dictionary and then sorts the IP address and prints it alongside the corresponding log entries.



```
1 import sys
2
3
4 logEntries = {}
5
6 for line in sys.stdin:
7     line = line.strip()
8     ipAddress, logEntry = line.split('\t', 1)
9
10    if ipAddress not in logEntries:
11        logEntries[ipAddress] = []
12
13    logEntries[ipAddress].append(logEntry)
14 sortedIps = sorted(logEntries.keys())
15
16 for ipAddress in sortedIps:
17     for logEntry in logEntries[ipAddress]:
18         print(f'{ipAddress} {logEntry}')
19
20 print("-----")
```

ECC\_REDUCER\_SORT.py

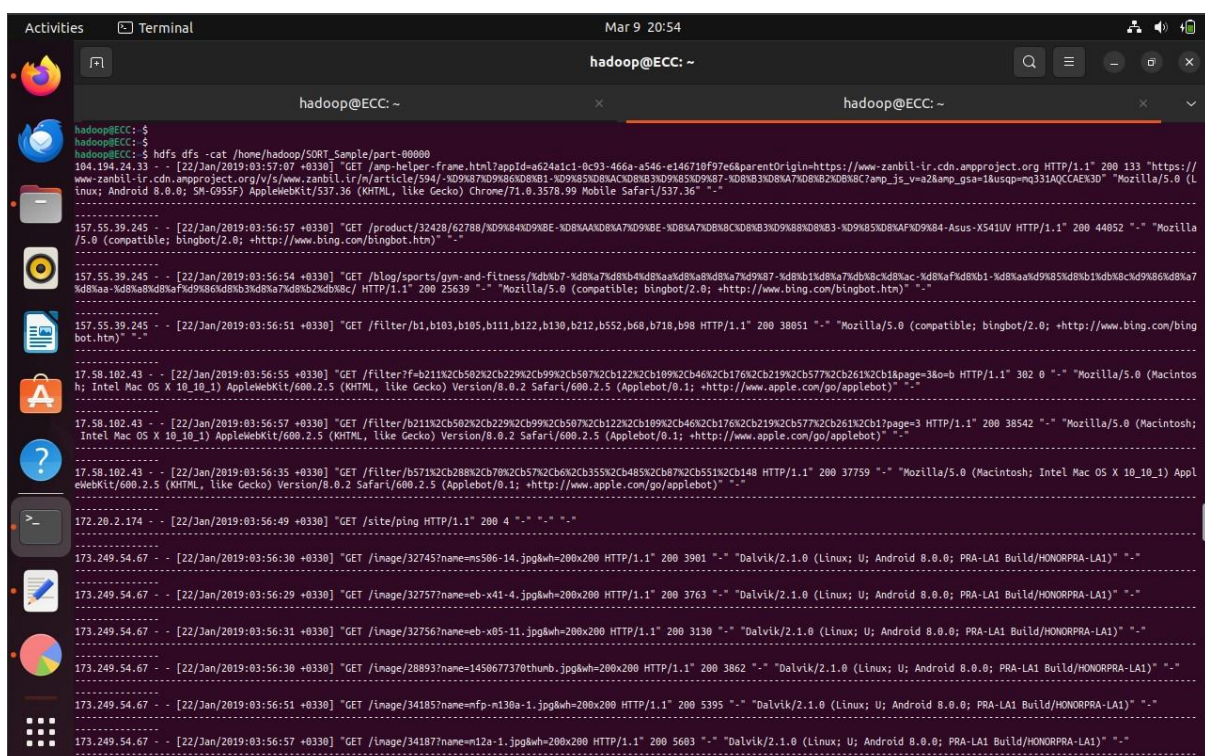
Command to run Map Reduce:

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -files  
ECC_MAP_SORT.py,ECC_REDUCER_SORT.py -mapper 'python3 ECC_MAP_SORT.py' -reducer  
'python3 ECC_REDUCER_SORT.py' -input /home/hadoop/sample.log -output  
/home/hadoop/SORT_Sample
```

Command to check output: `hdfs dfs -cat /home/Hadoop/SORT_Sample/part-00000`

Final Output:

- It shows logs sorted according to IP address.



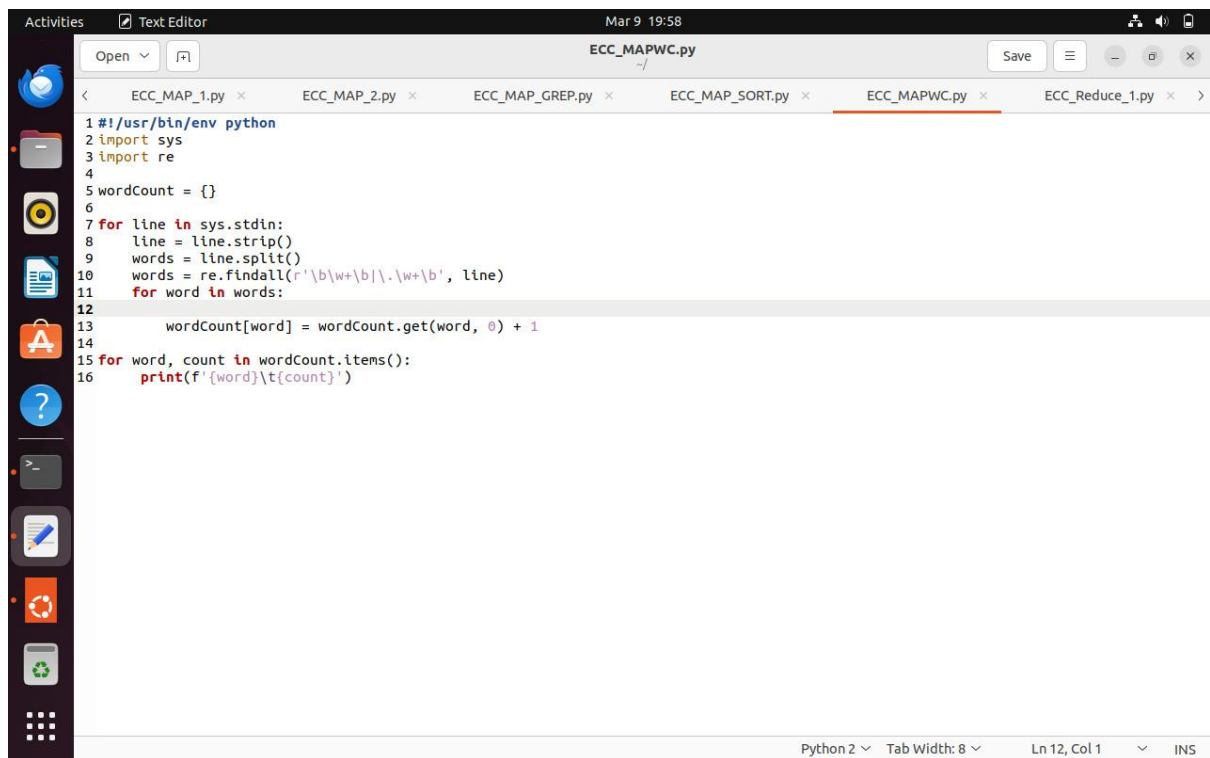
```
hadoop@ECC: ~  
hadoop@ECC: $ hdfs dfs -cat /home/hadoop/SORT_Sample/part-00000  
104.194.24.33 - - [22/Jun/2019:03:57:07 +0330] "GET /amp-helper-frame.html?appid=a624a1c1-0c93-466a-a546-e146718f97e6&parentOrigin=https://www-zanbil-ir.cdn.ampproject.org HTTP/1.1" 200 133 "https://  
www-zanbil-ir.cdn.ampproject.org/v/s/www.zanbil.ir/n/article/594/-XD9K87AD9K86KD8K81-XD9K85XD8KACXD8B3XD9K85XD9K87-XD8B3XD8KA7XD8B2XD8B8C7amp_js_v=a2&mp_gsa=1&usqp=mq33IAQCCAEK3D" "Mozilla/5.0 (L  
inux; Android 8.0.0; SM-G955F) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.99 Mobile Safari/537.36" "-"  
-----  
157.55.39.245 - - [22/Jun/2019:03:56:57 +0330] "GET /product/32428/62788/XD9K84XD9K8E-XD8KAAKD8KA7XD9K8E-XD8KA7XD8K8CND8B3XD9K88XD8B3-XD9K85XD8KAFXD9K84-Asus-K541UV HTTP/1.1" 200 44852 "-" "Mozilla  
/5.0 (compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm)" "-"  
-----  
157.55.39.245 - - [22/Jun/2019:03:56:54 +0330] "GET /blog/sports/gym-and-fitness/kdbkb7-kd8Ka7kd8b4kd8KaaKD8Ka8kd8Ka7kd9K87-kd8b1kd8Ka7kd8K8Ckd8Kac-kd8Kafkd8b1-kd8KaaKD9K85kd8B1kd8K8kd9K86kd8Ka7  
kd8Kaa-kd8KaaKD8Ka7kd9K86kd8B3kd8Ka7kd8B2kd8B8C HTTP/1.1" 200 25639 "-" "Mozilla/5.0 (compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm)" "-"  
-----  
157.55.39.245 - - [22/Jun/2019:03:56:51 +0330] "GET /filter/b1,b103,b105,b111,b122,b130,b212,b552,b68,b718,b98 HTTP/1.1" 200 38051 "-" "Mozilla/5.0 (compatible; bingbot/2.0; +http://www.bing.com/bing  
bot.htm)" "-"  
-----  
17.58.102.43 - - [22/Jun/2019:03:56:55 +0330] "GET /filter?fb=b211k2Cb502k2Cb229k2Cb99k2Cb507k2Cb122k2Cb109k2Cb46k2Cb176k2Cb219k2Cb577k2Cb261k2Cb1&page=3 HTTP/1.1" 302 0 "-" "Mozilla/5.0 (Macintos  
h; Intel Mac OS X 10_10_1) AppleWebKit/600.2.5 (KHTML, like Gecko) Version/8.0.2 Safari/600.2.5 (Applebot/0.1; +http://www.apple.com/go/applebot)" "-"  
-----  
17.58.102.43 - - [22/Jun/2019:03:56:57 +0330] "GET /filter/b211k2Cb502k2Cb229k2Cb99k2Cb507k2Cb122k2Cb109k2Cb46k2Cb176k2Cb219k2Cb577k2Cb261k2Cb1?page=3 HTTP/1.1" 200 38542 "-" "Mozilla/5.0 (Macintosh;  
Intel Mac OS X 10_10_1) AppleWebKit/600.2.5 (KHTML, like Gecko) Version/8.0.2 Safari/600.2.5 (Applebot/0.1; +http://www.apple.com/go/applebot)" "-"  
-----  
17.58.102.43 - - [22/Jun/2019:03:56:35 +0330] "GET /filter/b571k2Cb288k2Cb70k2Cb57k2Cb6k2Cb355k2Cb485k2Cb87k2Cb551k2Cb148 HTTP/1.1" 200 37759 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) Appl  
eWebKit/600.2.5 (KHTML, like Gecko) Version/8.0.2 Safari/600.2.5 (Applebot/0.1; +http://www.apple.com/go/applebot)" "-"  
-----  
172.20.2.174 - - [22/Jun/2019:03:56:49 +0330] "GET /site/ping HTTP/1.1" 200 4 "-" "-" "-"  
-----  
173.249.54.67 - - [22/Jun/2019:03:56:30 +0330] "GET /image/32745?name=ms506-14.jpg&wh=200x200 HTTP/1.1" 200 3901 "-" "Dalvik/2.1.0 (Linux; U; Android 8.0.0; PRA-LA1 Build/HONORPRA-LA1)" "-"  
-----  
173.249.54.67 - - [22/Jun/2019:03:56:29 +0330] "GET /image/32757?name=eb-x41-4.jpg&wh=200x200 HTTP/1.1" 200 3763 "-" "Dalvik/2.1.0 (Linux; U; Android 8.0.0; PRA-LA1 Build/HONORPRA-LA1)" "-"  
-----  
173.249.54.67 - - [22/Jun/2019:03:56:31 +0330] "GET /image/32756?name=eb-x05-11.jpg&wh=200x200 HTTP/1.1" 200 3130 "-" "Dalvik/2.1.0 (Linux; U; Android 8.0.0; PRA-LA1 Build/HONORPRA-LA1)" "-"  
-----  
173.249.54.67 - - [22/Jun/2019:03:56:30 +0330] "GET /image/28893?name=1450677370thumb.jpg&wh=200x200 HTTP/1.1" 200 3862 "-" "Dalvik/2.1.0 (Linux; U; Android 8.0.0; PRA-LA1 Build/HONORPRA-LA1)" "-"  
-----  
173.249.54.67 - - [22/Jun/2019:03:56:51 +0330] "GET /image/34105?name=nfp-m130a-1.jpg&wh=200x200 HTTP/1.1" 200 5395 "-" "Dalvik/2.1.0 (Linux; U; Android 8.0.0; PRA-LA1 Build/HONORPRA-LA1)" "-"  
-----  
173.249.54.67 - - [22/Jun/2019:03:56:57 +0330] "GET /image/34187?name=m12a-1.jpg&wh=200x200 HTTP/1.1" 200 5603 "-" "Dalvik/2.1.0 (Linux; U; Android 8.0.0; PRA-LA1 Build/HONORPRA-LA1)" "-"
```

Final Output

## ❖ Word Count Task:

Mapper Word Count (ECC\_MAPWC.py):

- It uses dictionary to store word counts and sends the word counts to reducer.

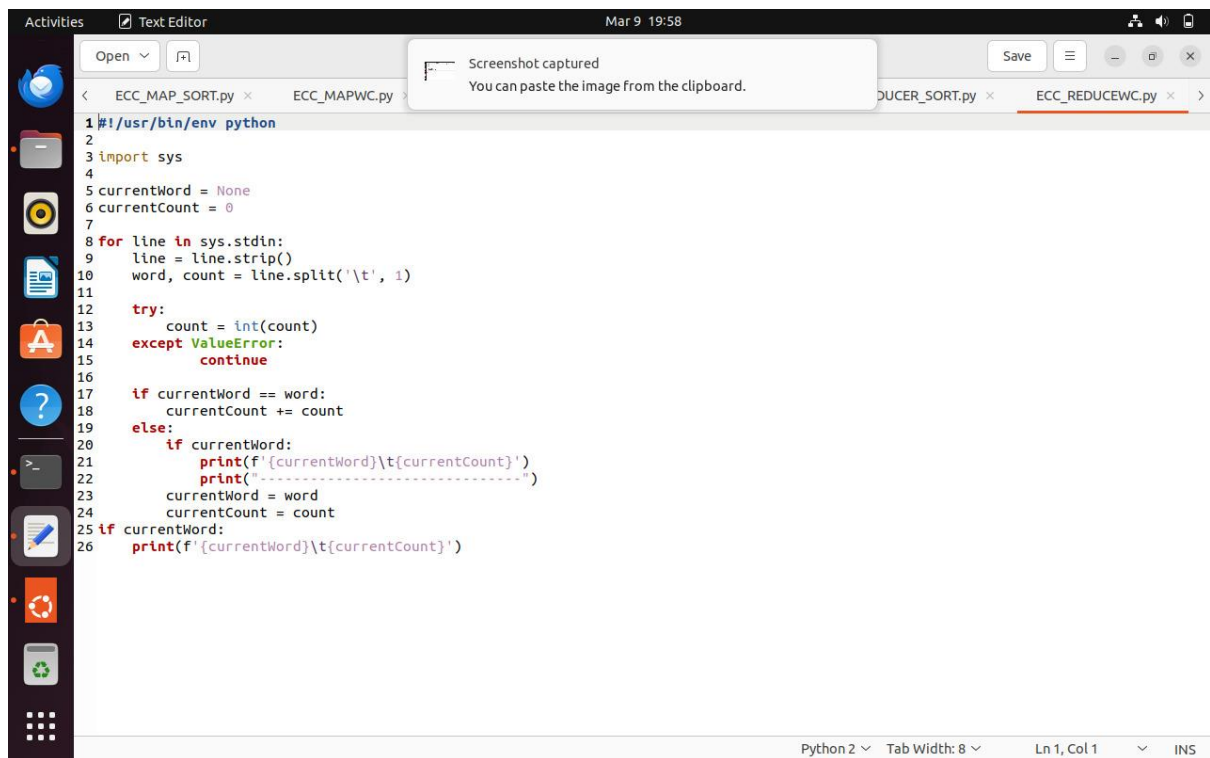


```
1#!/usr/bin/env python
2import sys
3import re
4
5wordCount = {}
6
7for line in sys.stdin:
8    line = line.strip()
9    words = line.split()
10    words = re.findall(r'\b\w+\b', line)
11    for word in words:
12
13        wordCount[word] = wordCount.get(word, 0) + 1
14
15for word, count in wordCount.items():
16    print(f'{word}\t{count}')
```

## ECC\_MAPWC.py

Reducer Word Count (ECC\_REDUCEWC.py):

- It processes word count pairs and consolidates the counts for each word.
- It sums the count for each word and when word changes it prints the word and count.



```
1#!/usr/bin/env python
2
3import sys
4
5currentWord = None
6currentCount = 0
7
8for line in sys.stdin:
9    line = line.strip()
10    word, count = line.split('\t', 1)
11
12    try:
13        count = int(count)
14    except ValueError:
15        continue
16
17    if currentWord == word:
18        currentCount += count
19    else:
20        if currentWord:
21            print(f'{currentWord}\t{currentCount}')
22            print("-----")
23        currentWord = word
24        currentCount = count
25
26if currentWord:
27    print(f'{currentWord}\t{currentCount}')
```

## ECC\_REDUCEWC.py

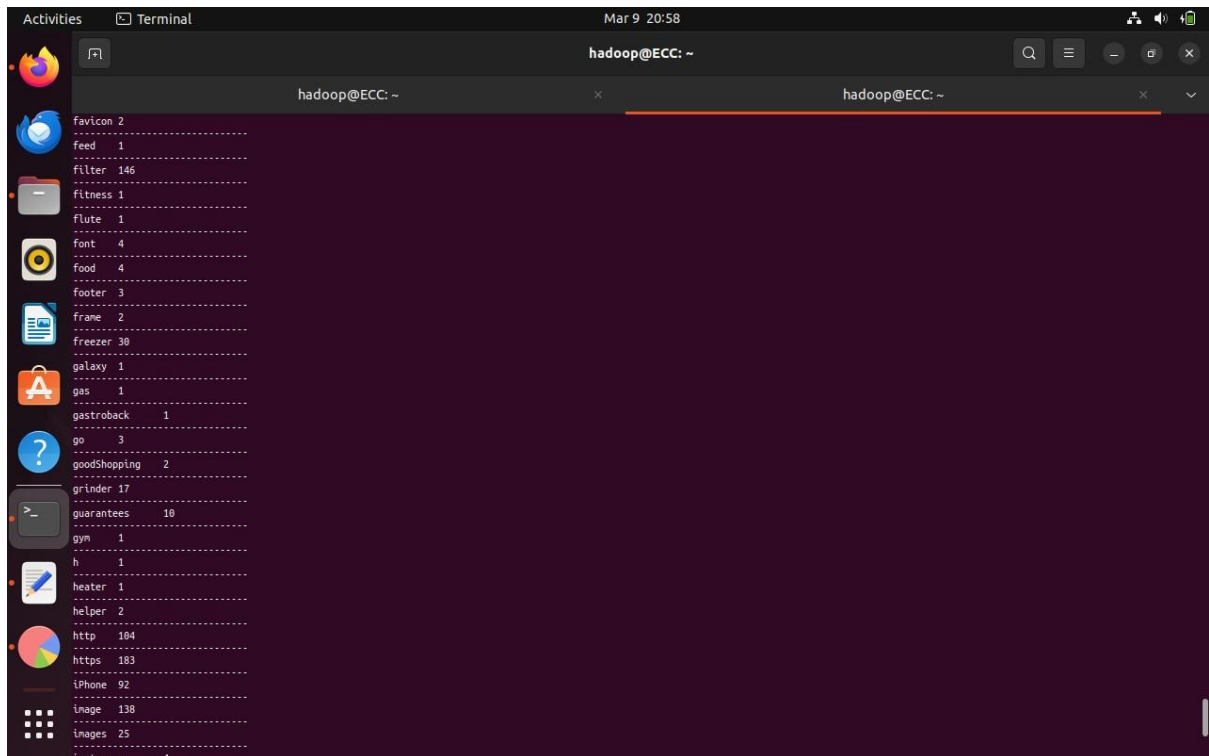
Command to run Map Reduce:

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -files  
ECC_MAPWC.py,ECC_REDUCEWC.py -mapper 'python3 ECC_MAPWC.py' -reducer 'python3  
ECC_REDUCEWC.py' -input /home/hadoop/sample.log -output /home/hadoop/WC_Sample
```

Command to check output: *hdfs dfs -cat /home/Hadoop/WC\_Sample/part-00000*

Final Output:

- It shows word count of each word from the log file.



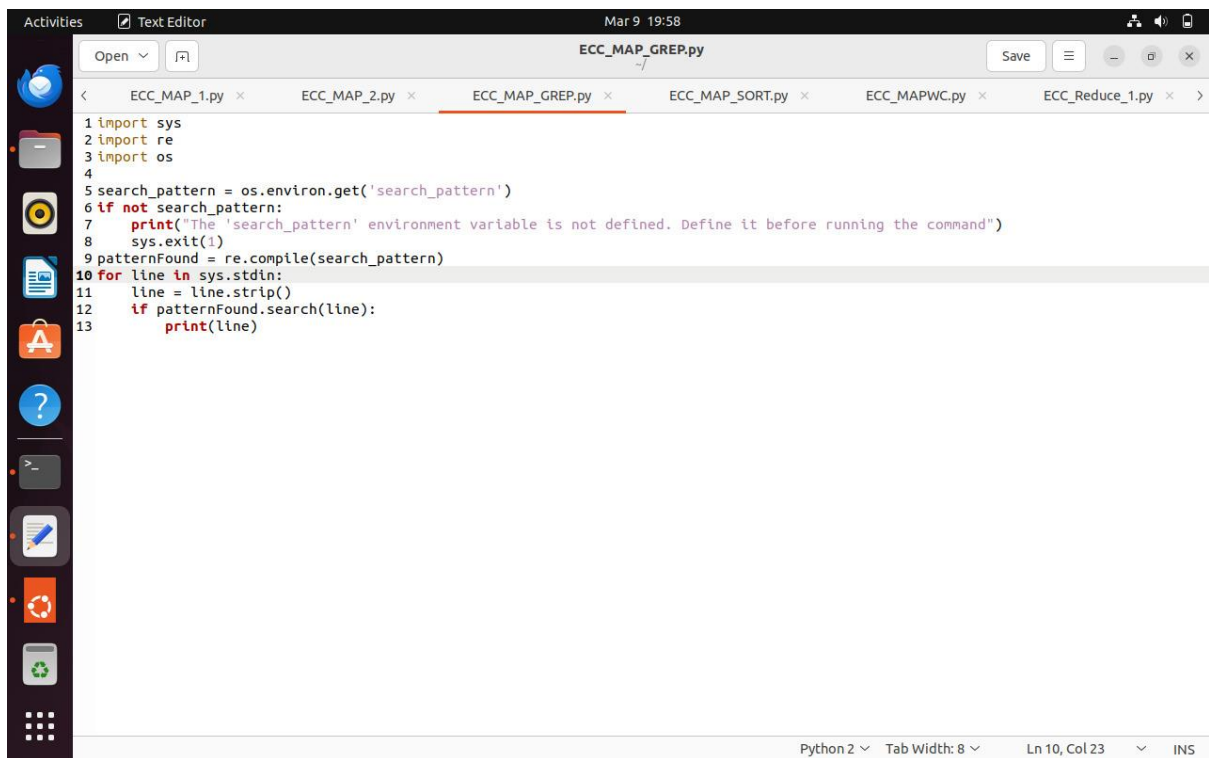
Final Output

## ❖ Grep Task:

Mapper Grep (ECC\_MAP\_GREP.py):

- It takes the input for specific search pattern from user via command line and checks if it matches.
- If the line matches the search pattern it sends it to reducer.



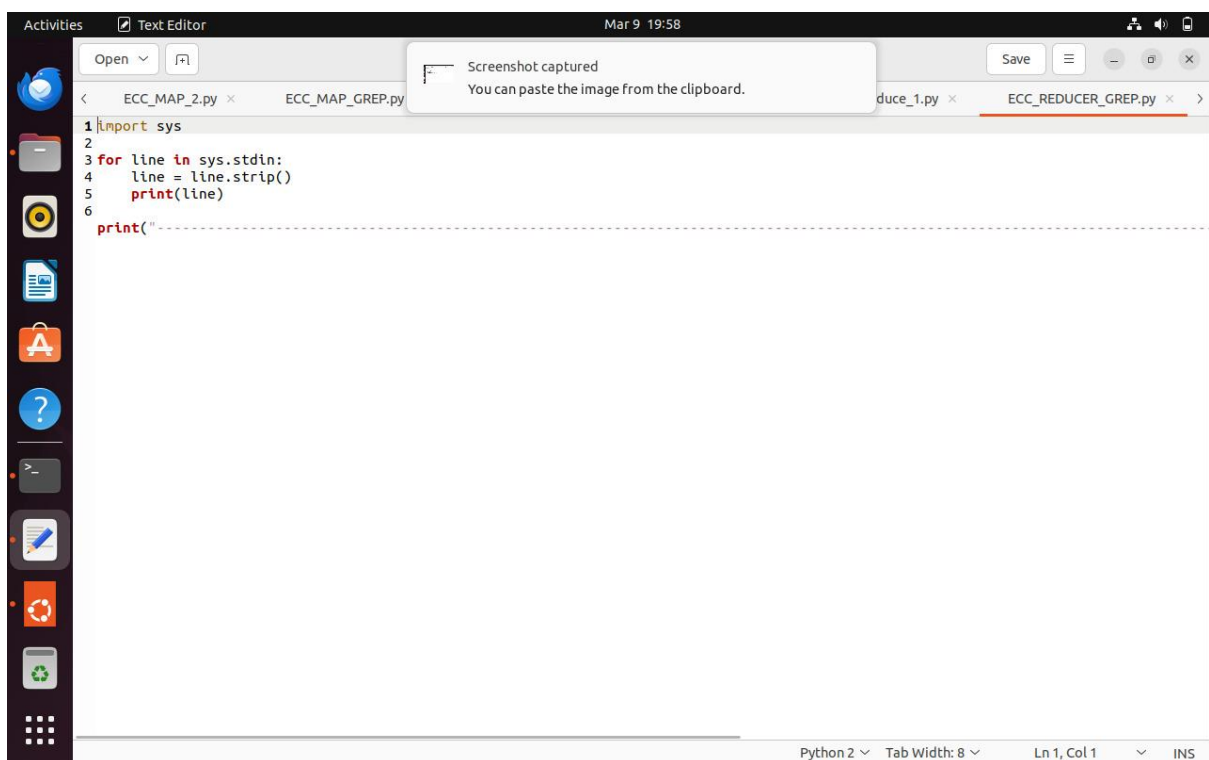


```
1 import sys
2 import re
3 import os
4
5 search_pattern = os.environ.get('search_pattern')
6 if not search_pattern:
7     print("The 'search_pattern' environment variable is not defined. Define it before running the command")
8     sys.exit(1)
9 patternFound = re.compile(search_pattern)
10 for line in sys.stdin:
11     line = line.strip()
12     if patternFound.search(line):
13         print(line)
```

## ECC\_MAP\_GREP.py

Reducer Grep (ECC\_REDUCER\_GREP.py):

- It prints out the line which it receives from the mapper after filtering according to search pattern.



```
1 import sys
2
3 for line in sys.stdin:
4     line = line.strip()
5     print(line)
6
7 print("-----")
```

## ECC\_REDUCER\_GREP.py

Command to check output: `hdfs dfs -cat /home/hadoop/GREPTELE/part-00000`

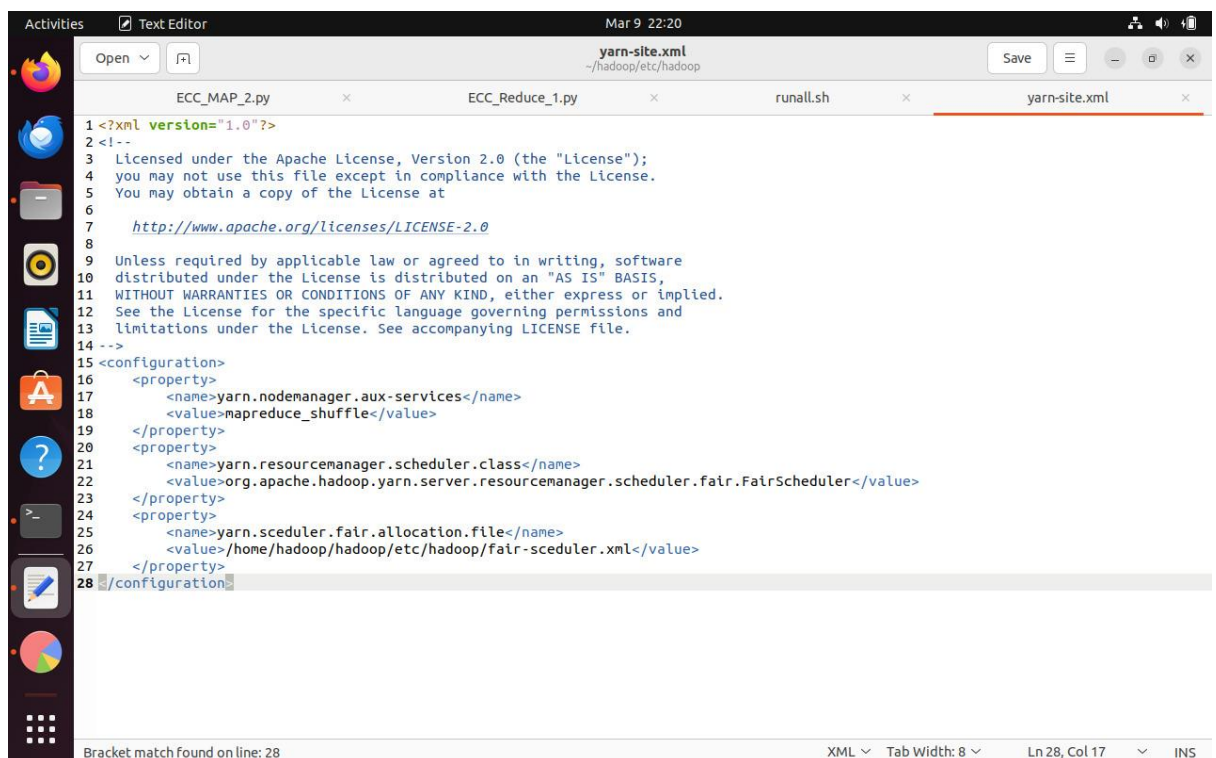
## Final Output:

```
hadoop@ECC:~$ hdfs dfs -cat /home/hadoop/GREPTELE/part-00000
2.177.12.140 - - [22/Jun/2019:03:56:24 +0330] "GET /static/images/amp/telegram.png HTTP/1.1" 200 4859 "https://www.zanbil.ir/m/product/33606/%D8%AA%D9%84%D9%88%D8%8C%D8%B2%D8%8C%D9%88%D9%86-%D8%A7%D9%84-%D8%A7%D8%8C-%D8%AF%D8%8C-%D8%B3%D8%A7%D9%85%D8%B3%D9%88%D9%86%D8%AF-%D9%85%D8%AF%D9%84-55NU8950-Ultra-HD-4K" "Mozilla/5.0 (Android 7.1.1; Mobile; rv:64.0) Gecko/64.0 Firefox/64.0" "-"
-----
31.56.96.51 - - [22/Jun/2019:03:56:32 +0330] "GET /static/images/amp/telegram.png HTTP/1.1" 200 4859 "https://www.zanbil.ir/m/filter/b113" "Mozilla/5.0 (Linux; Android 6.0; ALE-L21 Build/HuaweiALE-L21) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.158 Mobile Safari/537.36" "-"
-----
5.209.200.218 - - [22/Jun/2019:03:57:07 +0330] "GET /static/images/amp/telegram.png HTTP/1.1" 200 4859 "https://www.zanbil.ir/m/filter/b99%2Cp4510%2Cstexists%2Ct116" "Mozilla/5.0 (Linux; Android 5.1.1; SM-G361H Build/LMY48B) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.91 Mobile Safari/537.36" "-"
-----
66.111.54.249 - - [22/Jun/2019:03:57:02 +0330] "GET /static/images/amp/telegram.png HTTP/1.1" 200 4859 "https://www.zanbil.ir/m/browse/refrigerator-and-freezer/%D8%8C%D8%AE%D8%A7%D9%84-%D9%81%D8%B1%D8%8C%D8%B2%D8%B1" "Mozilla/5.0 (Linux; Android 5.0; SM-G900H Build/LRX21T) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.93 Mobile Safari/537.36" "-"
-----
```

## Final Output

### ❖ Testing the Fair Scheduler:

→ To test the fair scheduler, I added below properties in \$HADOOP\_HOME/yarn-site.xml. It is fair scheduler module which is required to change the scheduler to fair.



The screenshot shows a text editor window titled 'yarn-site.xml' with the following content:

```
1 <?xml version="1.0"?>
2 <!--
3 Licensed under the Apache License, Version 2.0 (the "License");
4 you may not use this file except in compliance with the License.
5 You may obtain a copy of the License at
6
7 http://www.apache.org/licenses/LICENSE-2.0
8
9 Unless required by applicable law or agreed to in writing, software
10 distributed under the License is distributed on an "AS IS" BASIS,
11 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 See the License for the specific language governing permissions and
13 limitations under the License. See accompanying LICENSE file.
14 -->
15 <configuration>
16   <property>
17     <name>yarn.nodemanager.aux-services</name>
18     <value>mapreduce_shuffle</value>
19   </property>
20   <property>
21     <name>yarn.resourcemanager.scheduler.class</name>
22     <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler</value>
23   </property>
24   <property>
25     <name>yarn.sceduler.fair.allocation.file</name>
26     <value>/home/hadoop/hadoop/etc/hadoop/fair-sceduler.xml</value>
27   </property>
28 </configuration>
```

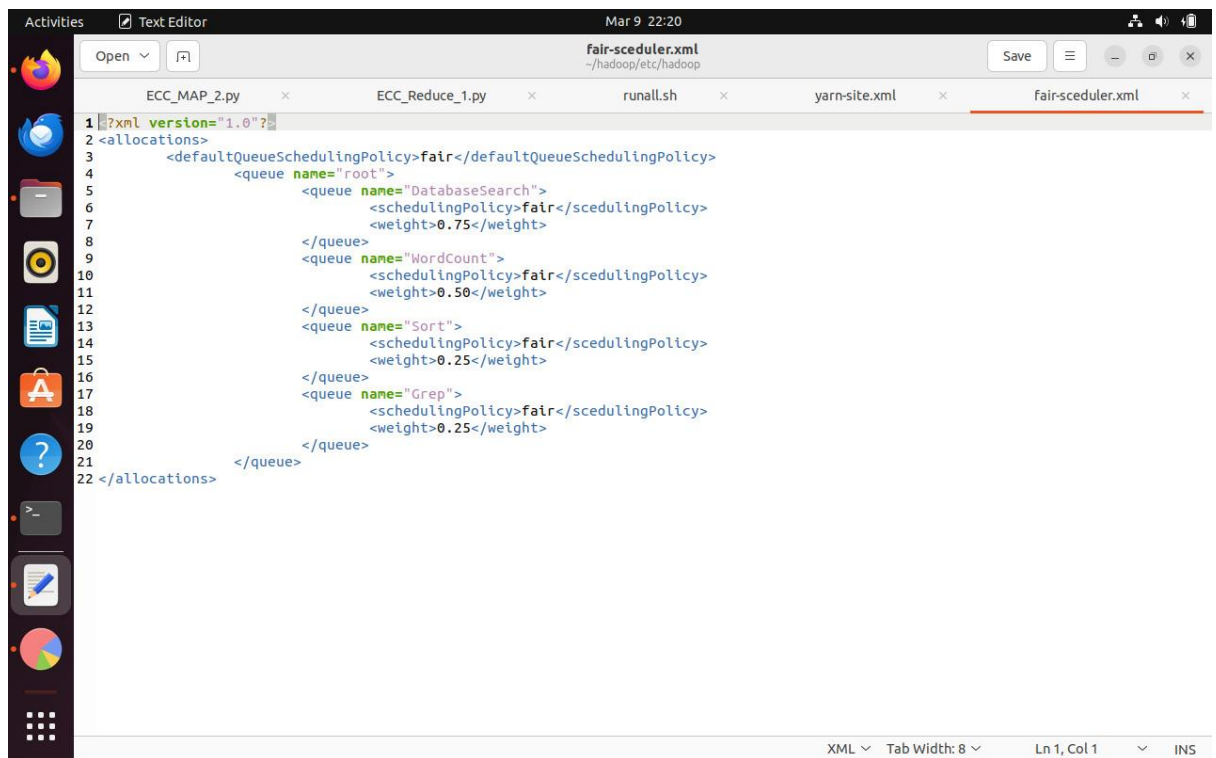
## yarn-site.xml for fair scheduler

→ I also created fair-sheduler.xml file and allocated below queues in it.

1. Database Search
2. Word Count
3. Sort



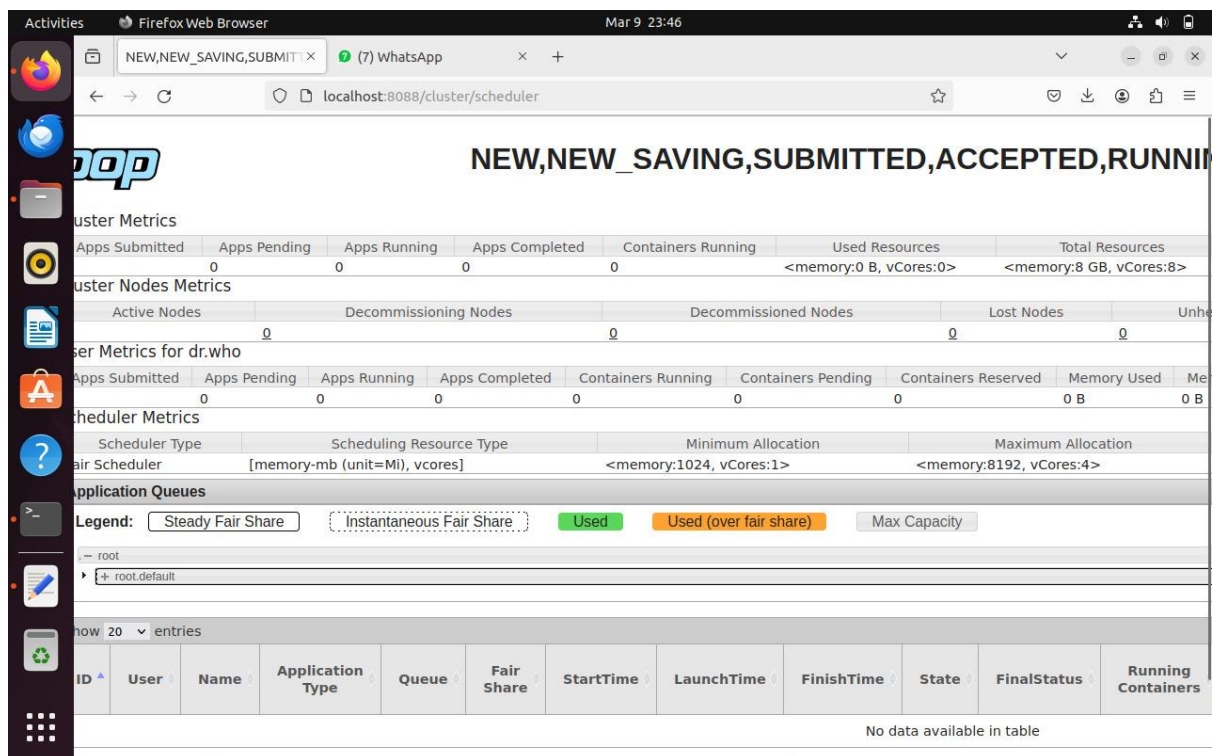
#### 4. Grep



```
1 <?xml version="1.0"?>
2 <allocations>
3   <defaultQueueSchedulingPolicy>fair</defaultQueueSchedulingPolicy>
4   <queue name="root">
5     <queue name="DatabaseSearch">
6       <schedulingPolicy>fair</schedulingPolicy>
7       <weight>0.75</weight>
8     </queue>
9     <queue name="WordCount">
10      <schedulingPolicy>fair</schedulingPolicy>
11      <weight>0.50</weight>
12    </queue>
13    <queue name="Sort">
14      <schedulingPolicy>fair</schedulingPolicy>
15      <weight>0.25</weight>
16    </queue>
17    <queue name="Grep">
18      <schedulingPolicy>fair</schedulingPolicy>
19      <weight>0.25</weight>
20    </queue>
21  </queue>
22 </allocations>
```

fair-scheduler.xml

→ Below are the queues which are created in Hadoop.



NEW,NEW\_SAVING,SUBMITTED,ACCEPTED,RUNNING

Cluster Metrics	
Apps Submitted	Apps Pending
0	0

Cluster Nodes Metrics	
Active Nodes	Decommissioning Nodes
0	0

Scheduler Metrics	
Scheduler Type	Scheduling Resource Type
Fair Scheduler	[memory-mb (unit=Mi), vcores]

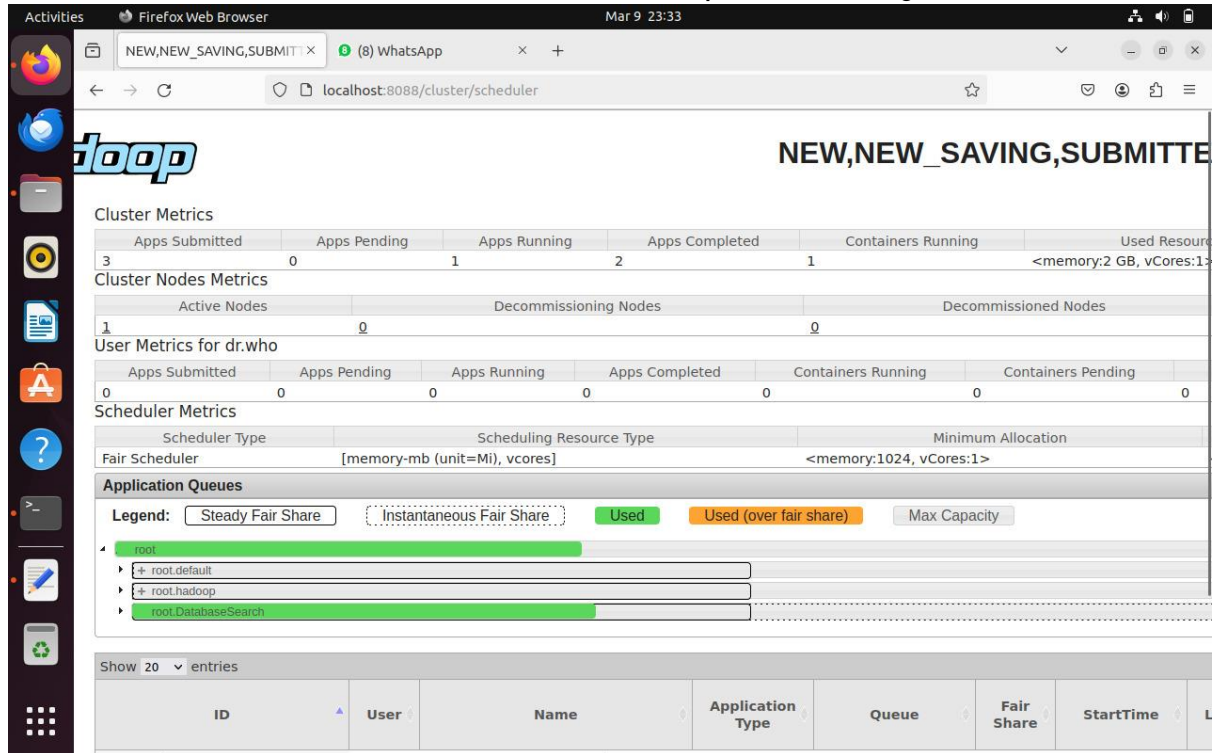
Application Queues	
Queue	Fair Share
root	0

ID	User	Name	Application Type	Queue	Fair Share	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers
No data available in table											

Initial Queues in Hadoop

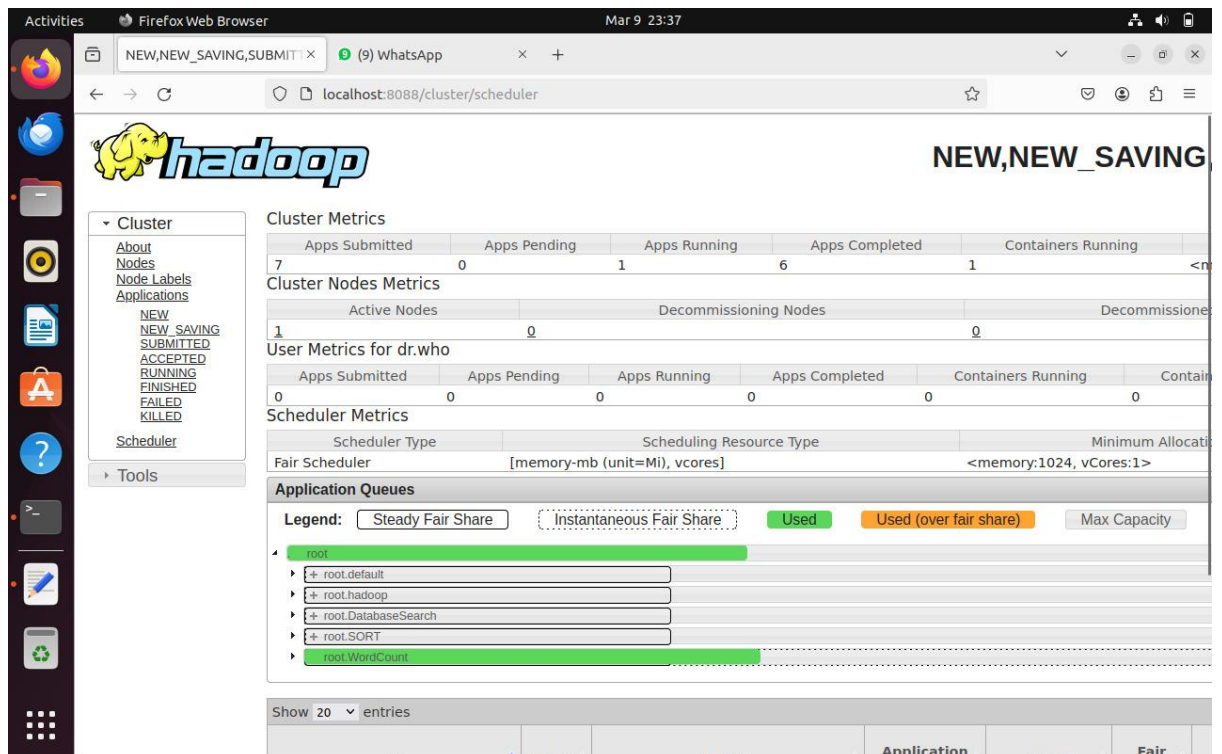
→ In fair-scheduler.xml, I have given more weight database search as it was consuming more resources and initially it failed hence, I increased the weight.

→ In the below screenshot, we can see DatabaseSearch queue is running first.



DatabaseSearch Queue running first

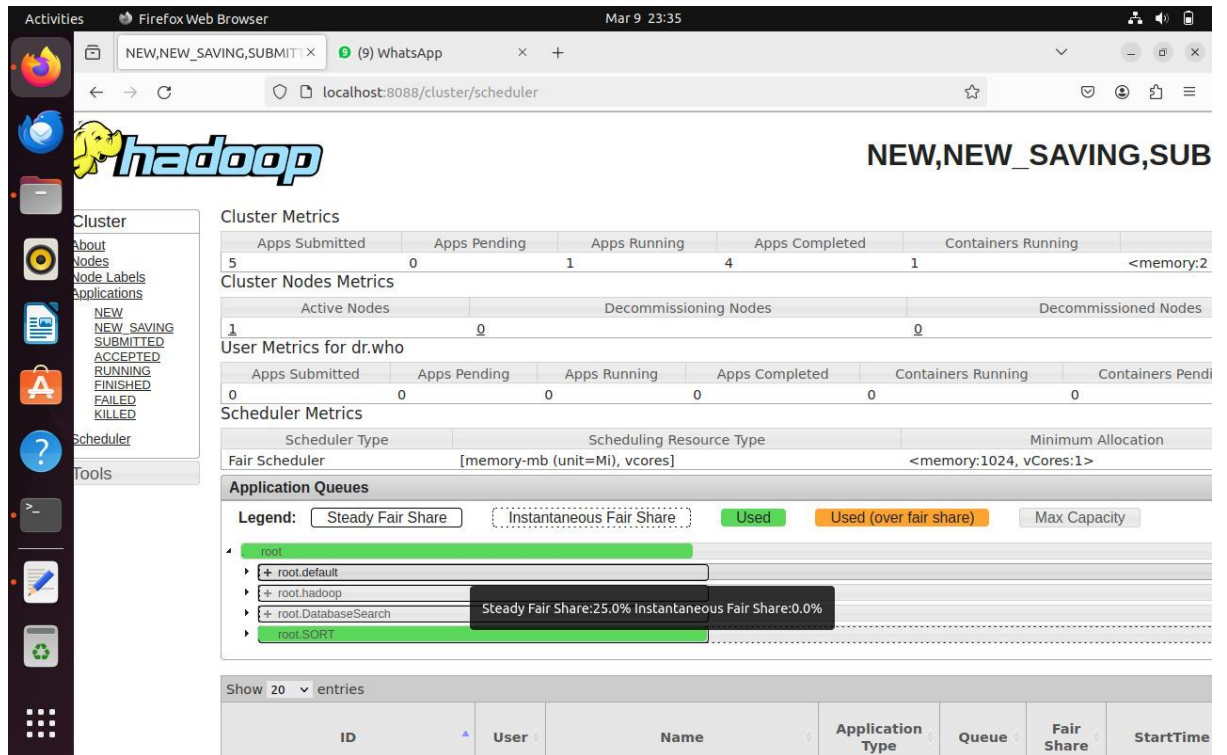
→ I also gave name WordCount slightly less weight than DatabaseSearch but more than other two and hence it is running on second number as seen in below screenshot.



WordCount queue running second



→ I gave equal weights to both Sort and Grep as they were relatively less bulky and executed fast. As seen in below screenshot they are running concurrently.



Grep and Sort queue running concurrently

## ❖ Testing the Capacity Scheduler:

- To implement capacity scheduler, I modified capacity-scheduler.xml.
- I have configured databaseSearch, wordcount, sort, grep queues in my capacity-scheduler.xml as shown in below screenshot.

The screenshot shows a text editor window with the file capacity-scheduler.xml open. The XML configuration is as follows:

```

231 </property>
232
233 <property>
234   <name>yarn.scheduler.capacity.workflow-priority-mappings-override.enable</name>
235   <value>>false</value>
236   <description>
237     If a priority mapping is present, will it override the value specified
238     by the user? This can be used by administrators to give applications a
239     priority that is different than the one specified by the user.
240     The default is false.
241   </description>
242 </property>
243
244 <property>
245   <name>yarn.scheduler.capacity.capacity.root.DatabaseSearch.capacity</name>
246   <value>35</value>
247   <description>DatabaseSearch queue target capacity.</description>
248 </property>
249
250 <property>
251   <name>yarn.scheduler.capacity.root.WordCount.capacity</name>
252   <value>25</value>
253   <description>WordCount queue target capacity.</description>
254 </property>
255
256 <property>
257   <name>yarn.scheduler.capacity.root.Sort.capacity</name>
258   <value>25</value>
259   <description>Sort queue target capacity.</description>
260 </property>
261
262 <property>
263   <name>yarn.scheduler.capacity.root.Grep.capacity</name>
264   <value>15</value>
265   <description>Grep queue target capacity.</description>
266 </property>
267 </configuration>
  
```

The status bar at the bottom indicates: XML Tab Width: 8 Ln 15, Col 1 INS.

## Capacity-scheduler.xml with sort, grep, wordcount and DatabaseSearch

### Queue

The screenshot shows the Hadoop Capacity Scheduler web interface in a Firefox browser window. The URL is localhost:8088/cluster/scheduler. The interface displays various metrics and queues.

**Cluster Metrics**

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources
4	0	0	4	0	<memory:0 B, vCores:0>	<memory:8192 B, vCores:1>

**Cluster Nodes Metrics**

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes
1	0	0	0

**User Metrics for dr.who**

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved
0	0	0	0	0	0	0

**Scheduler Metrics**

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Fair Scheduler	[memory-mb (unit=Mb), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:1>

**Application Queues**

Legend: ☐ Steady Fair Share ☐ Instantaneous Fair Share ☒ Used ☐ Used (over fair share) ☐ Max Capacity

- root
  - root.default
  - root.DatabaseSearch
  - root.Sort
  - root.WordCount
  - root.GREP

At the bottom, there is a table showing application entries with columns: Application, Queue, Fair, StartTime, LaunchTime, FinishTime, State, and FinalState.

### Capacity Scheduler with sort, grep, wordcount and DatabaseSearch Queue

- Soon I noticed from below that queues reached at max capacity and got stuck hence I implemented priority handling in fair scheduling.
- I saw the queues executed smoothly in fair scheduling.

### ❖ Conclusion:

I got to learn many topics related to scheduling and map-reduce framework along with Hadoop.