Assignment 1

By

Vishwajeet Ekal 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 conuration 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.

```
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Add an "alert" alias for long running commands.  Use like so:
#   sleep 10; alert
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$//'\'')"'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

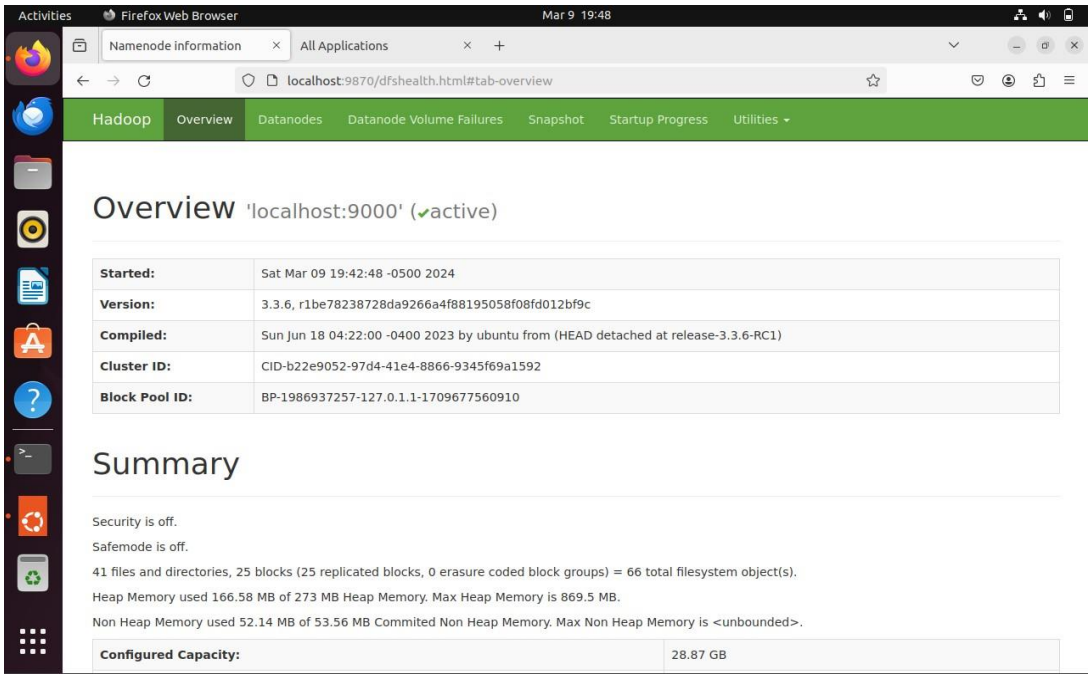
Environment variables

```
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:~$
```

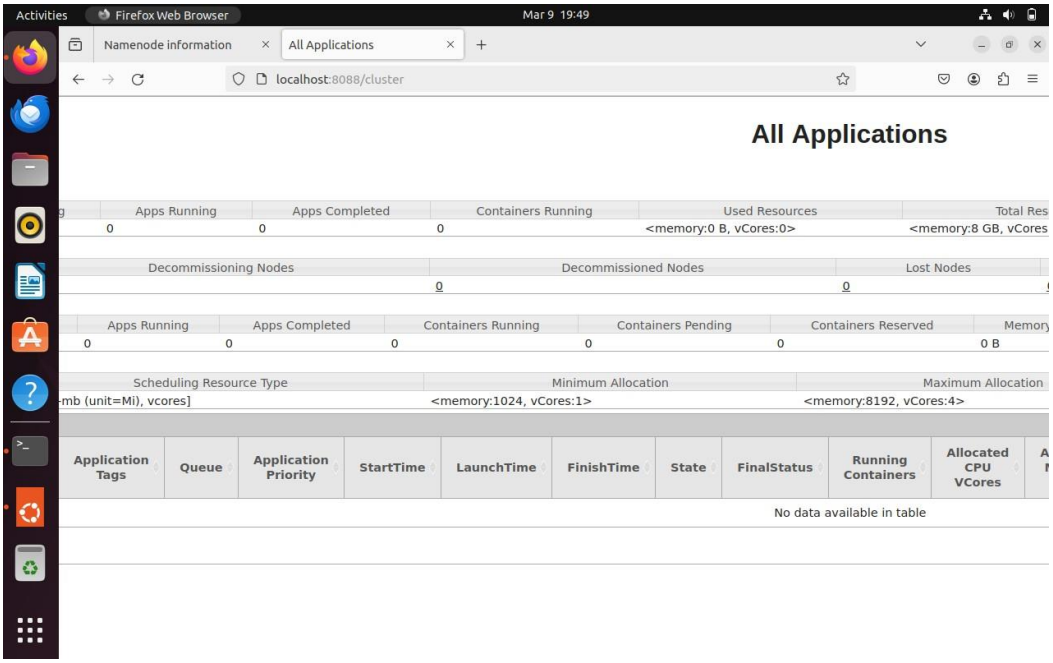
Start-all.sh Output

```
hadoop@ECC:~$ jps
2993 DataNode
3938 Jps
3202 SecondaryNameNode
3573 NodeManager
3433 ResourceManager
2863 NameNode
hadoop@ECC:~$
```

All process Up and Running



Hadoop UI

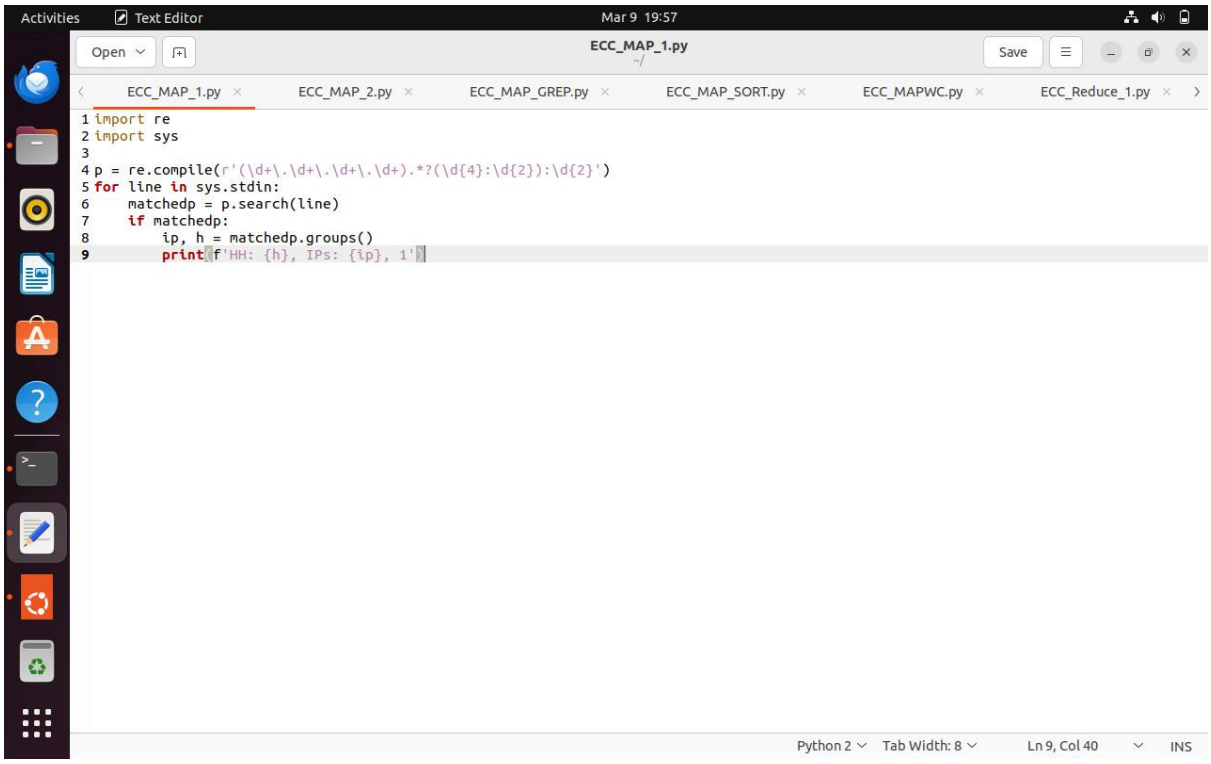


Hadoop Cluster

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

Mapper (ECC_MAP_1.py):

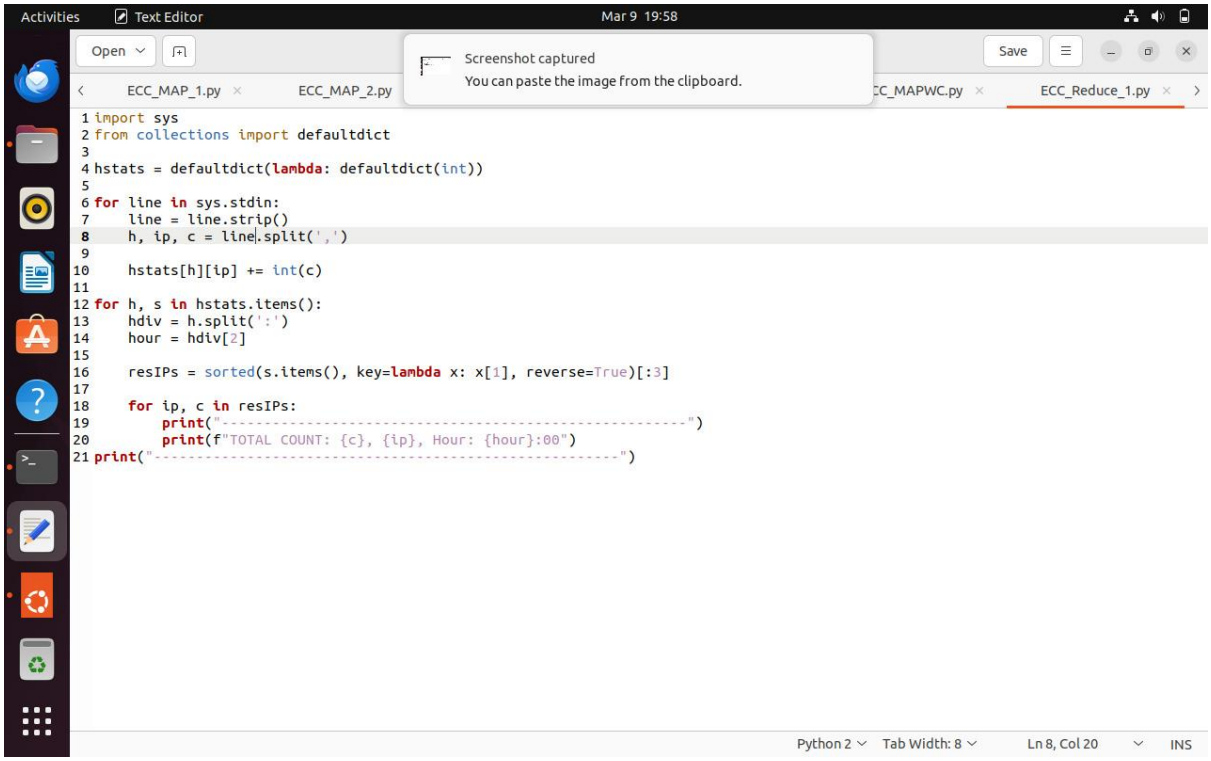
- Below mapper code searches and extracts IP addresses and hours from input log file.
- For each match it prints the hour and IP along 1 to indicate the count which is passed to reducer.



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.

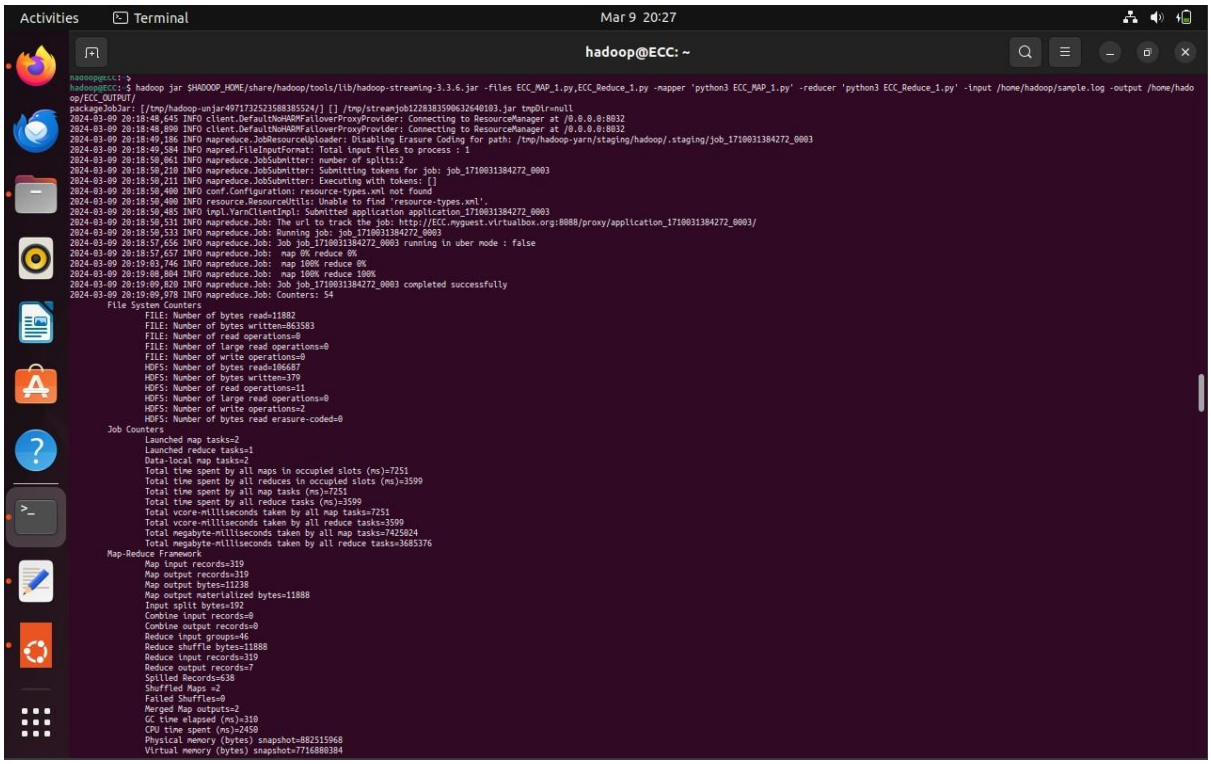


ECC_Reduce_1.py

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:



Logs

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

Final Output*:*

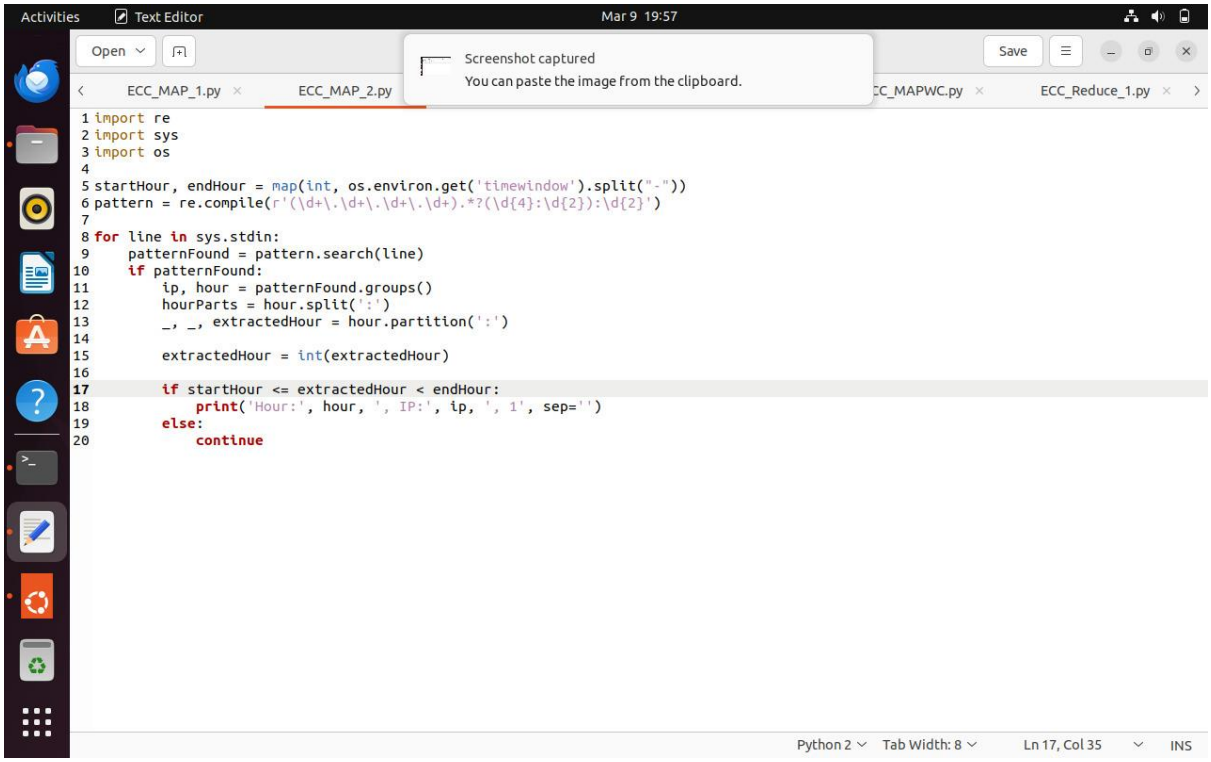
```
hadoop@ECC:~$ hdfs dfs -cat /home/hadoop/ECC_OUTPUT/part-00000
-----------------------------------------------------
TOTAL COUNT: 38,  IPs: 66.111.54.249, Hour: 03:00
-----------------------------------------------------
TOTAL COUNT: 36,  IPs: 5.211.97.39, Hour: 03:00
-----------------------------------------------------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

Final Output

❖ <u>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.</u>

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.



ECC_MAP_2.py

Reducer (ECC_Reduce_1.py):

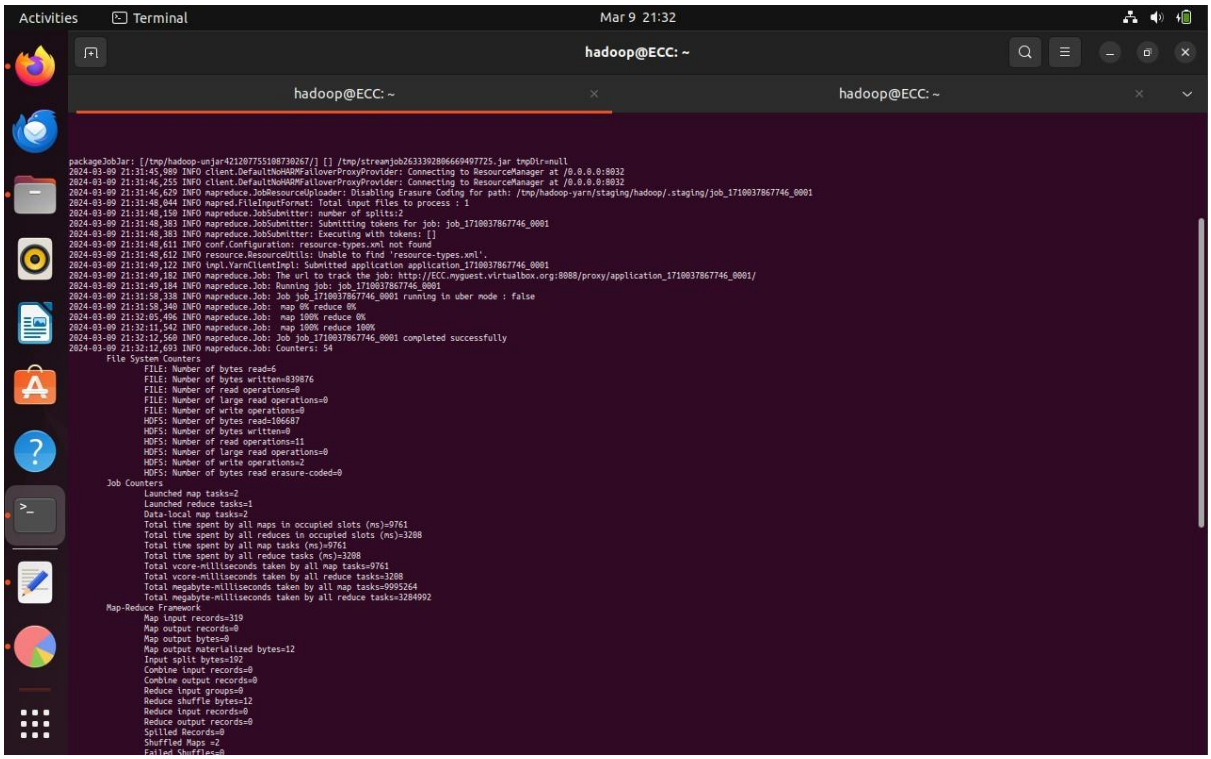
- Reducer for this part is the same as part 1.
- I have taken two test cases; first time window is 00 to 02 which has no IP addresses and second time window is 01 to 04 which has IP addresses.

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:



Logs

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

Final Output:

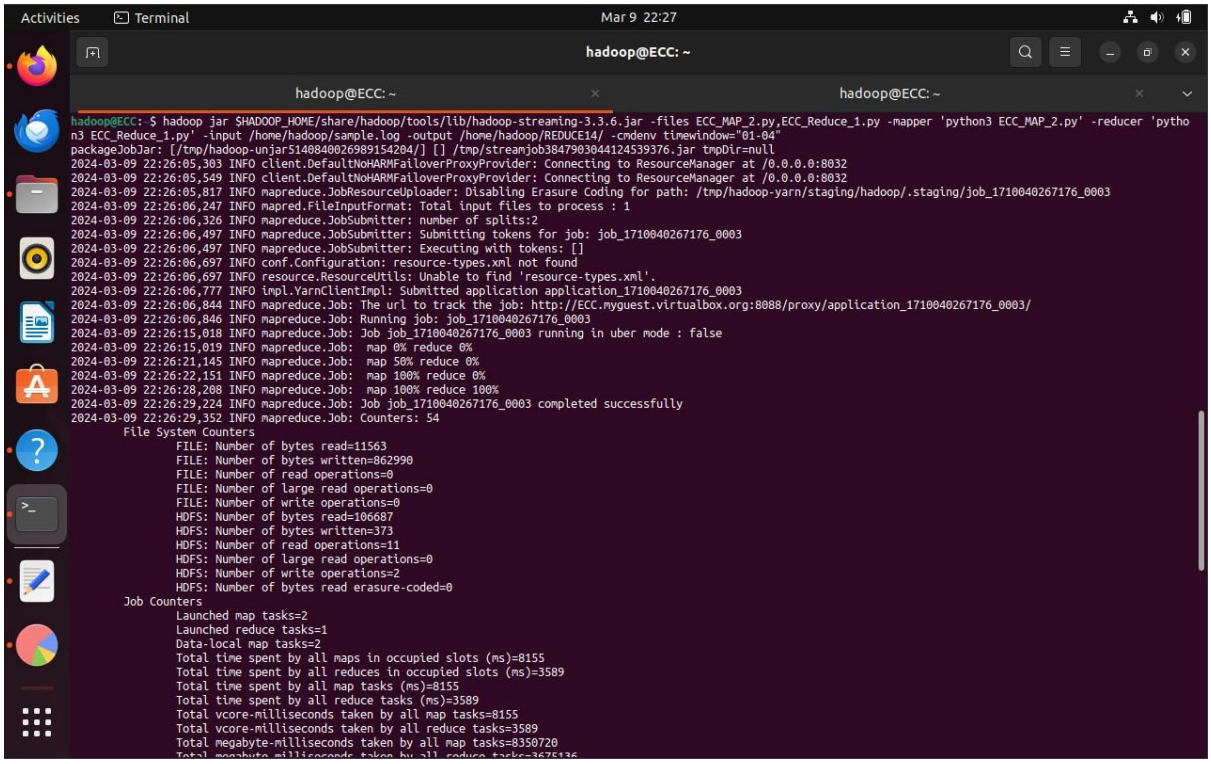
Final Output

Test Case 2 Time Window 01 to 04:

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:



Logs

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

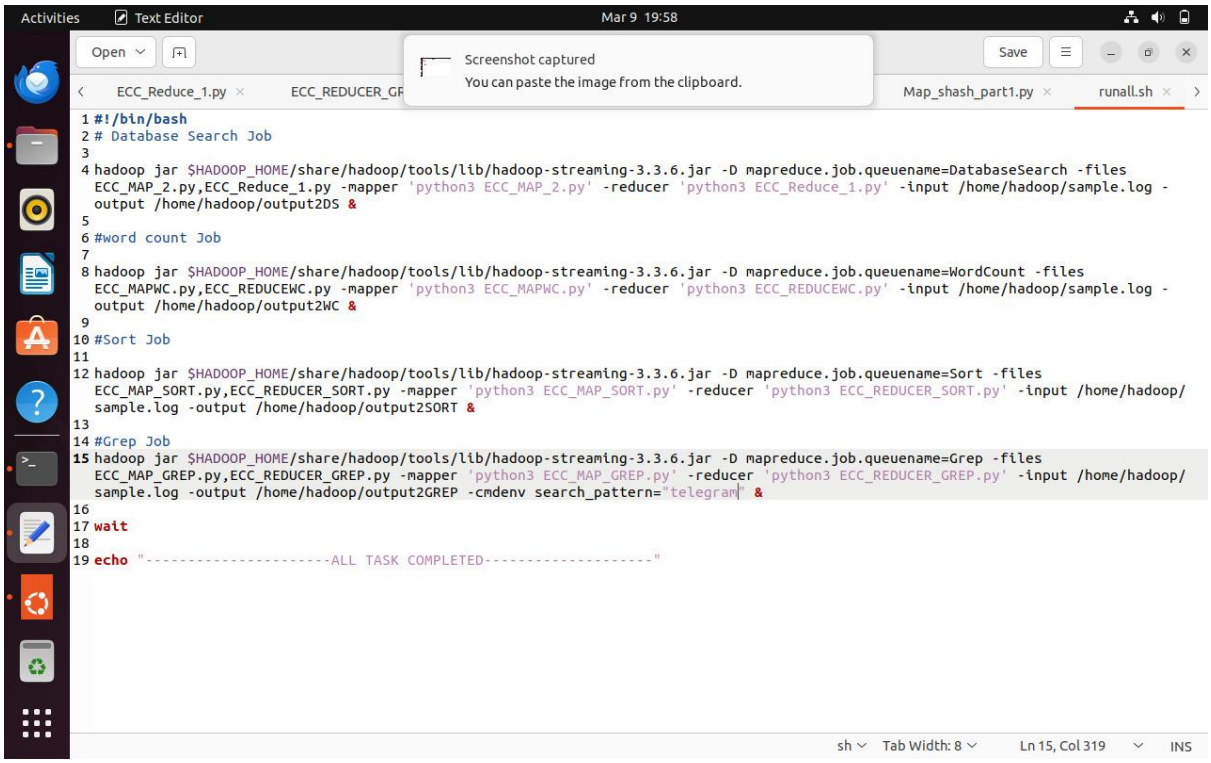
Final Output:

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.jobs.queuename. Below is the screenshot of runall.sh.
-

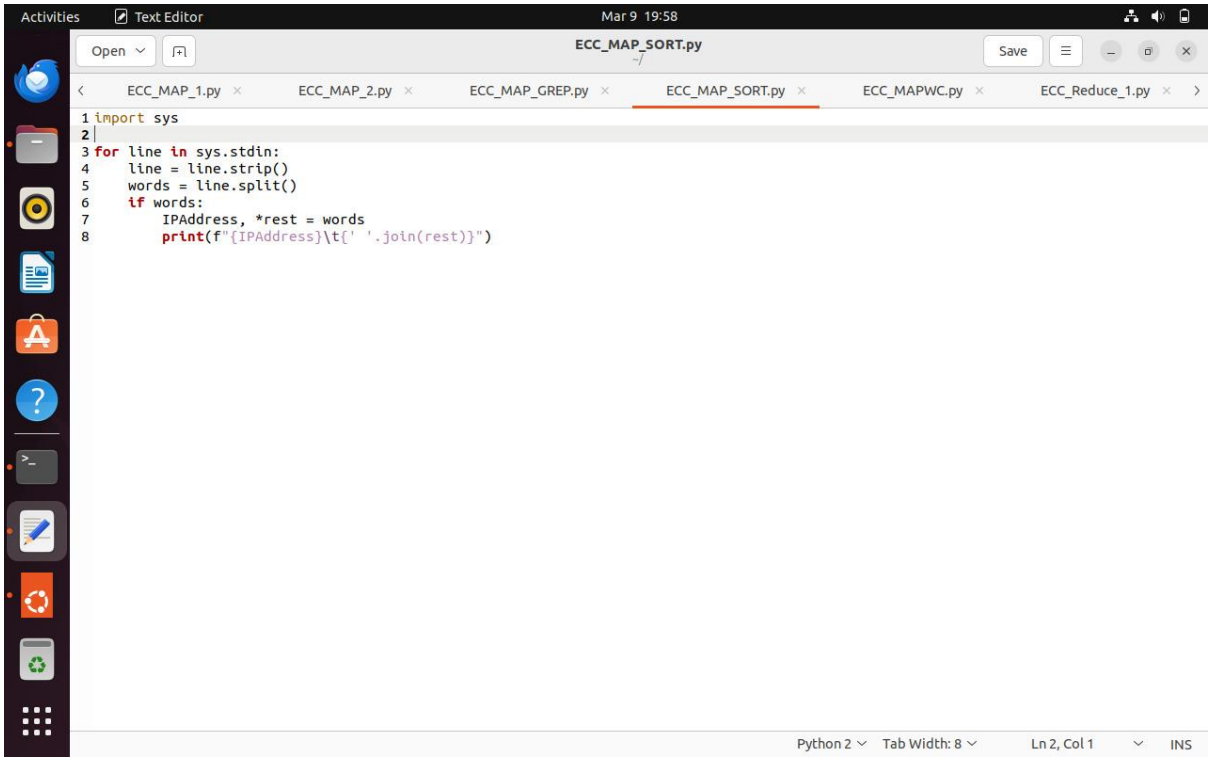


runall.sh

❖ Sort Task:

Mapper Sort (ECC_MAP_SORT.py):

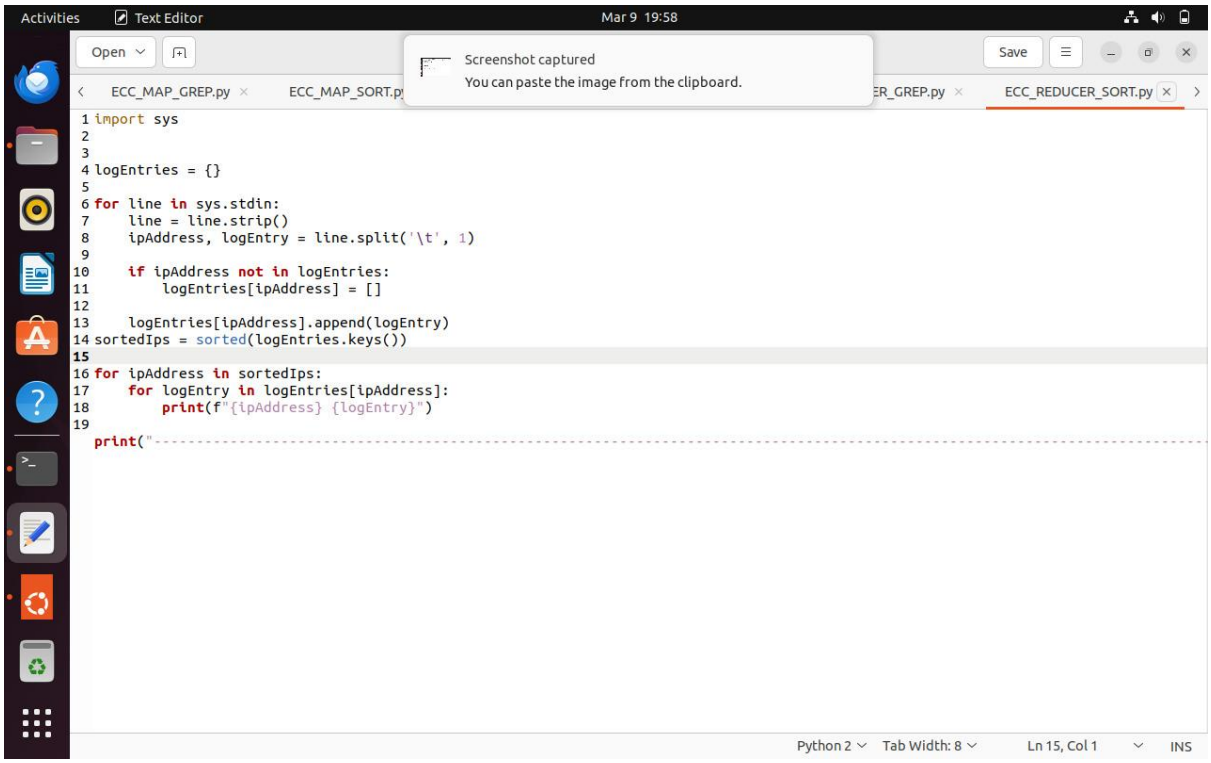
- It extracts the first word as IP and remaining words as string.
- It then prints IP along with rest words in tab-separated format and sends it to reducer.



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.



ECC_REDUCER_SORT.py

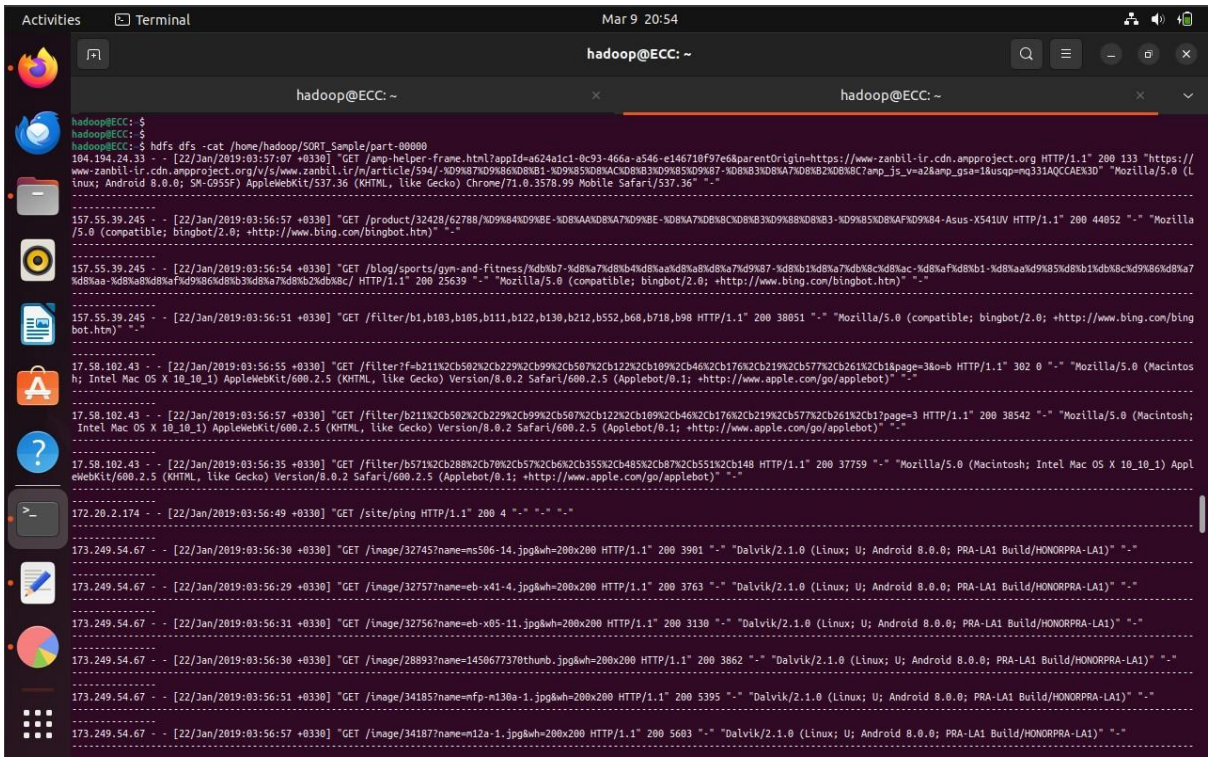
Command to run Map Reduce:

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.

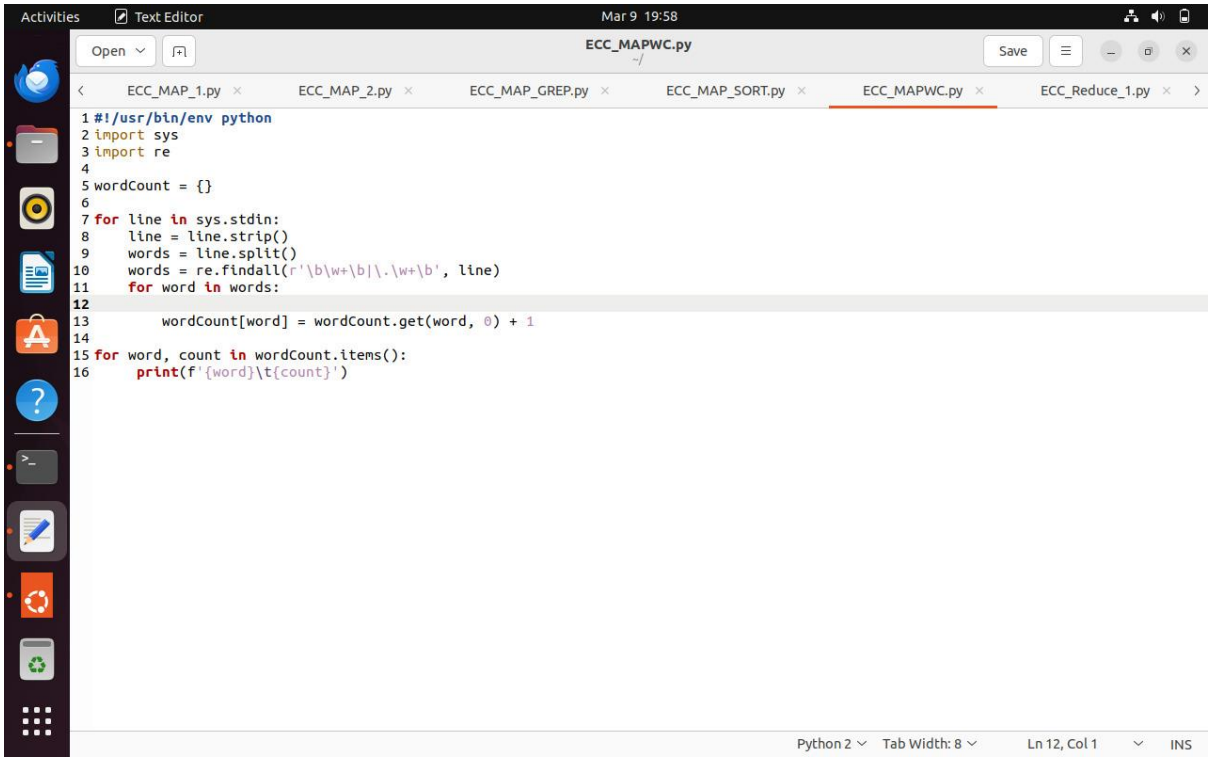


Final Output

❖ Word Count Task:

Mapper Word Count (ECC_MAPWC.py):

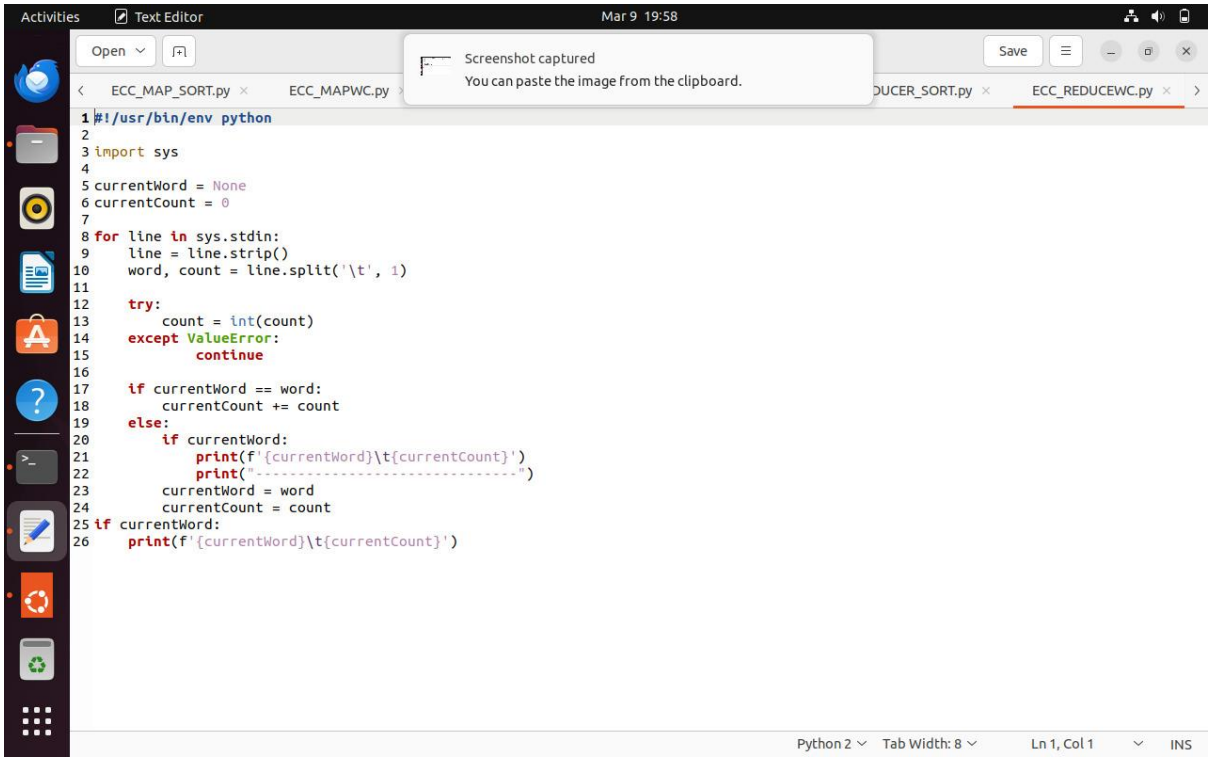
- It uses dictionary to store word counts and sends the word counts to reducer.



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.



ECC_REDUCEWC.py

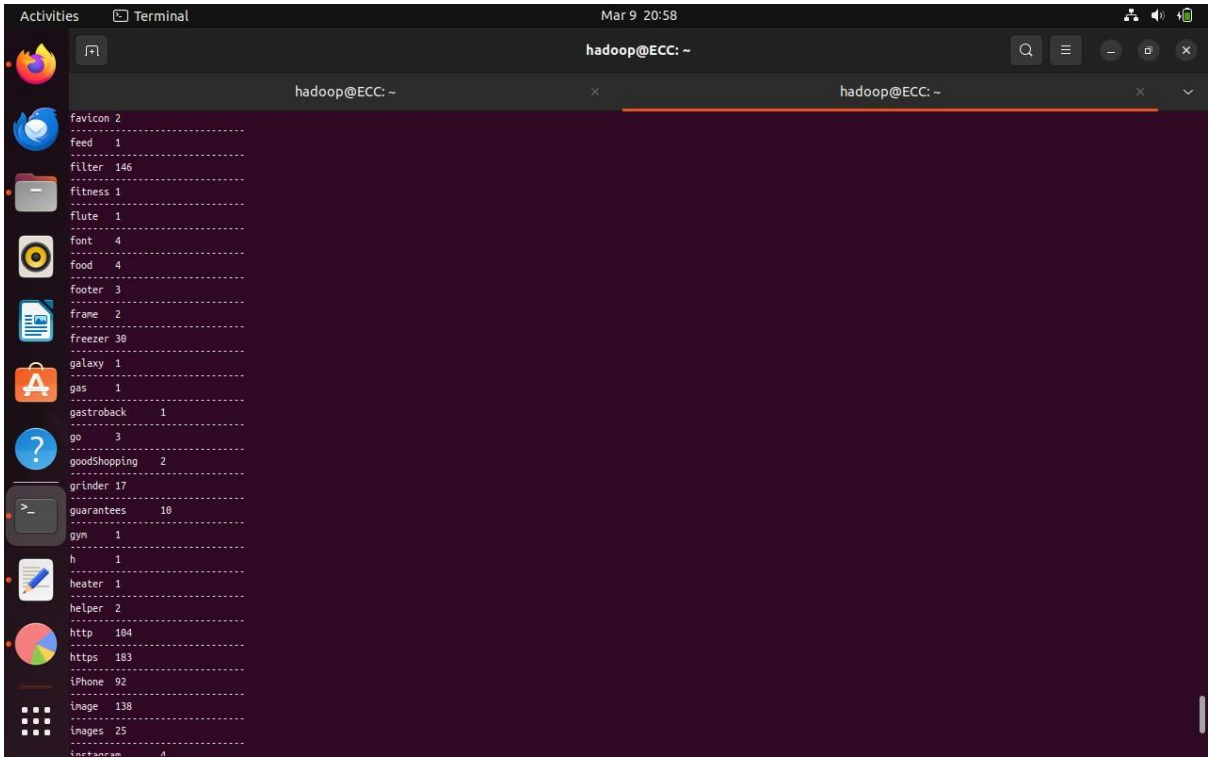
Command to run Map Reduce:

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.

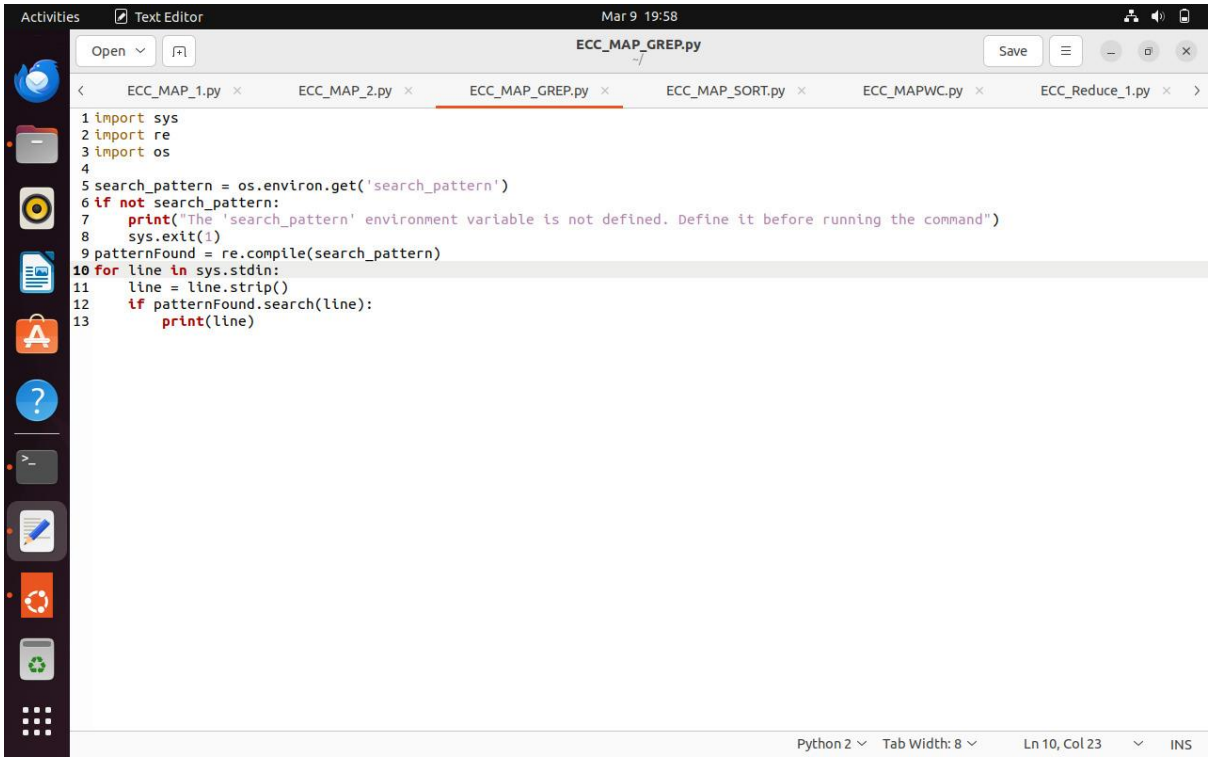


Final Output

❖ Grep Task:

Mapper Grep (ECC_MAP_GREP.py):

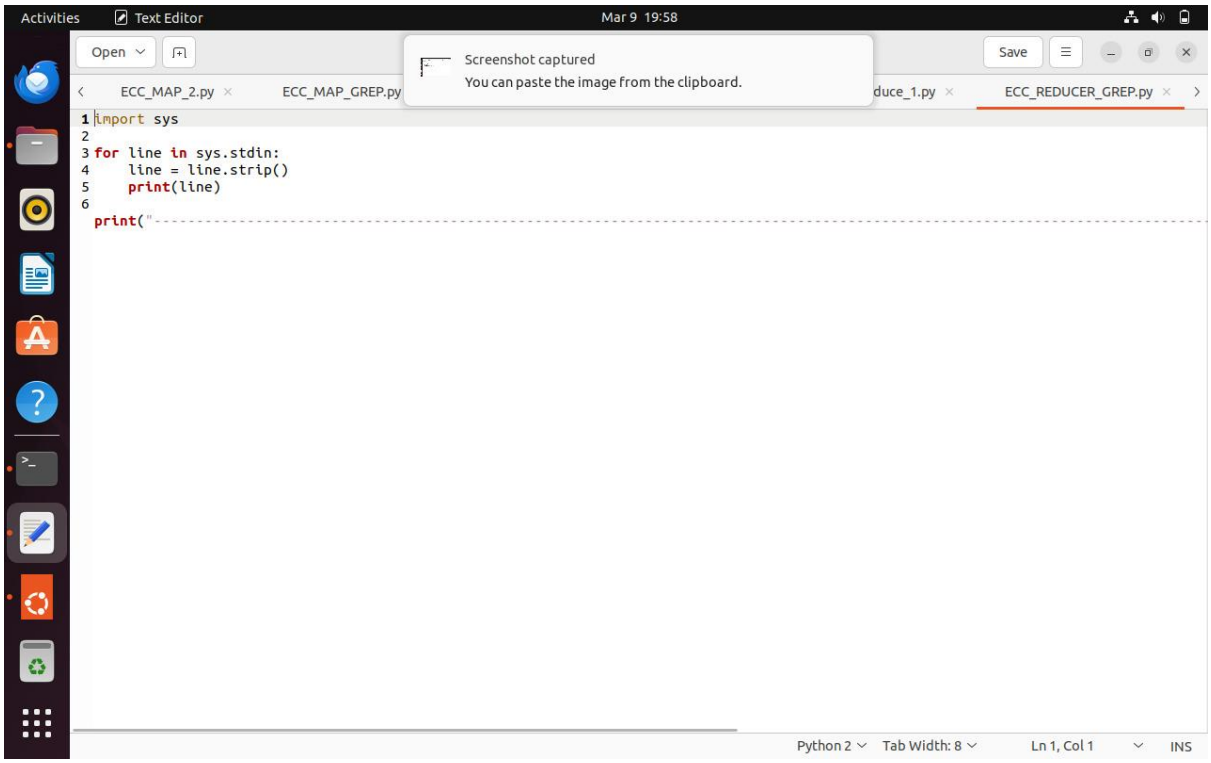
- ➢ It takes the input for specific search pattern from user via command line and checks if it matches.
- ➢ If the line matches the search pattern it sends it to reducer.



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.



ECC_REDUCER_GREP.py

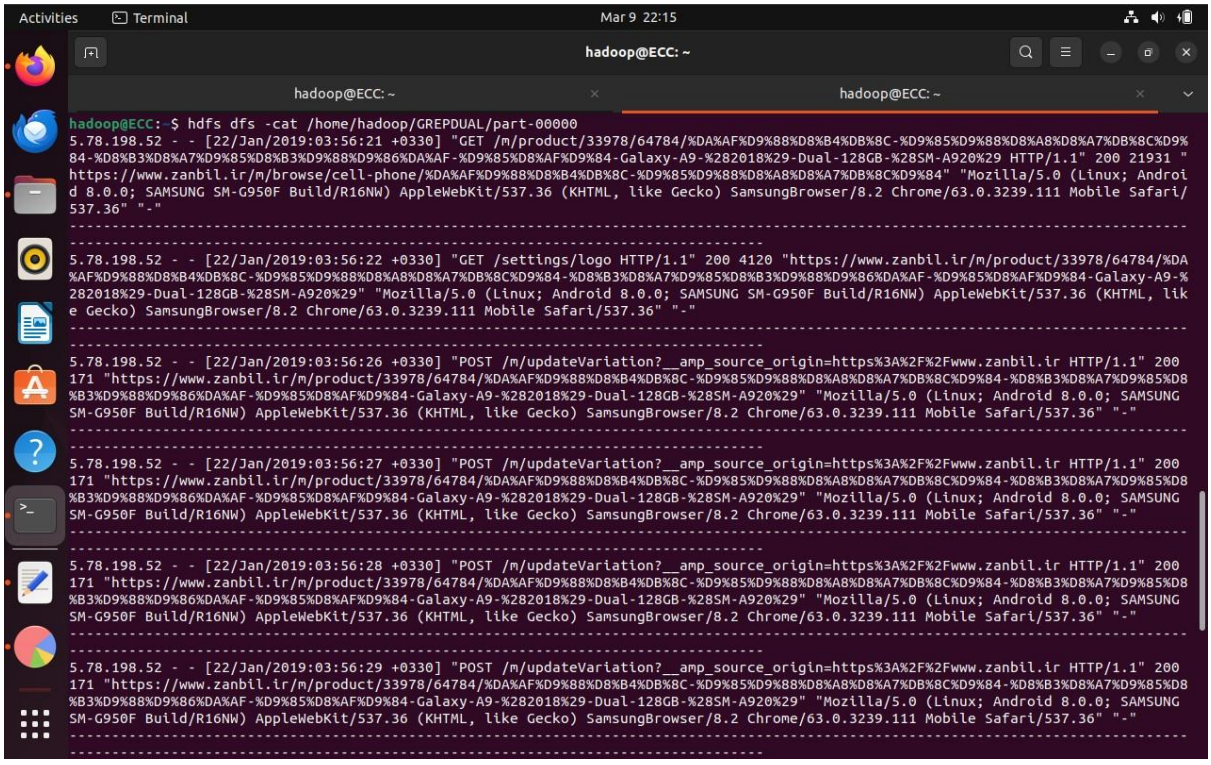
TestCase 1 “Dual” as search pattern:

Command to run Map Reduce:

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -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/GREPTELE/ -cmdenv search_pattern="Dual"

Command to check output*: hdfs dfs -cat /home/hadoop/GREPDUAL/part-00000*

Final Output for Test Case 1:



Final Output

Test Case 2 “telegram” as search pattern:

Command to run Map Reduce:

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -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/GREPTELE/ -cmdenv search_pattern="telegram"

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

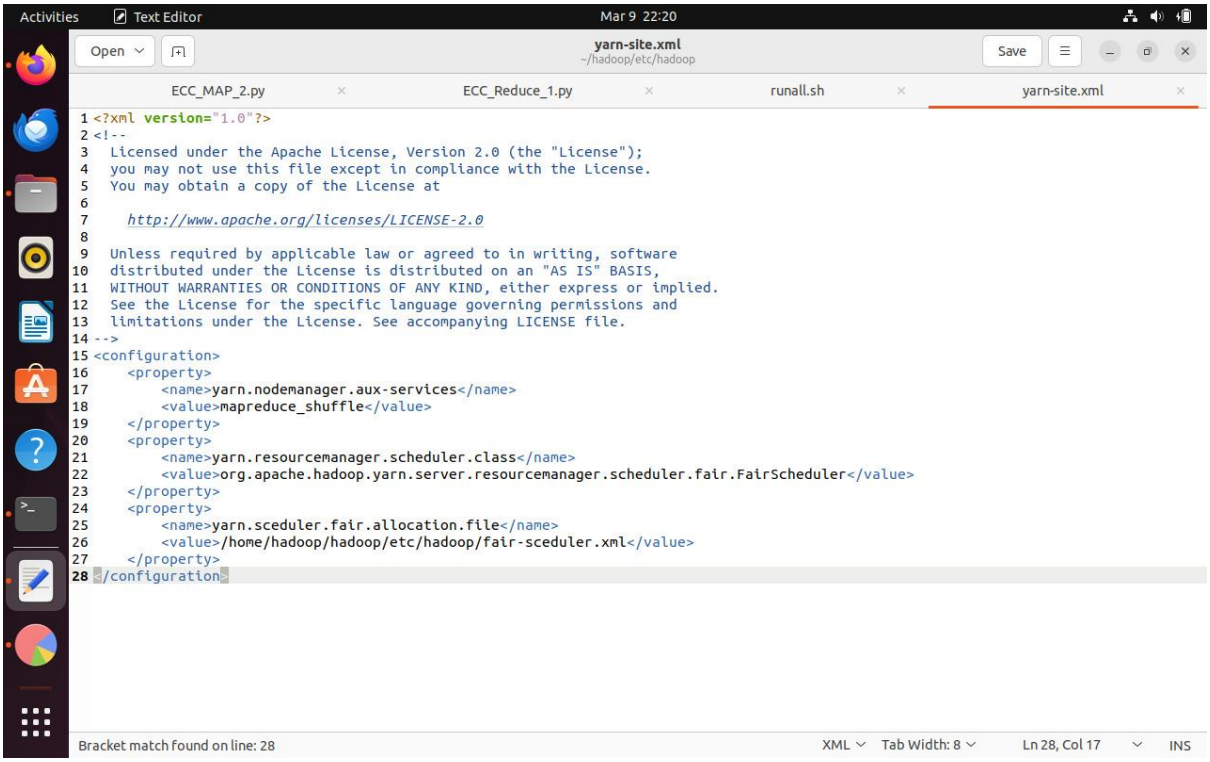
Final Output:

```
hadoop@ECC:~$ hdfs dfs -cat /home/hadoop/GREPTELE/part-00000
2.177.12.140 - - [22/Jan/2019:03:56:24 +0330] "GET /static/images/amp/telegram.png HTTP/1.1" 200 4859 "https://www.zanbil.ir/m/produ
ct/33606/%D8%AA%D9%84%D9%88%DB%8C%D8%B2%DB%8C%D9%88%D9%86-%D8%A7%D9%84-%D8%A7%DB%8C-%D8%AF%DB%8C-%D8%B3%D8%A7%D9%85%D8%B3%D9%88%D9%8
6%DA%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/Jan/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 Mo
bile Safari/537.36" "-"
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------
5.209.200.218 - - [22/Jan/2019:03:57:07 +0330] "GET /static/images/amp/telegram.png HTTP/1.1" 200 4859 "https://www.zanbil.ir/m/filt
er/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/Jan/2019:03:57:02 +0330] "GET /static/images/amp/telegram.png HTTP/1.1" 200 4859 "https://www.zanbil.ir/m/brow
se/refrigerator-and-freezer/%DB%8C%D8%AE%DA%86%D8%A7%D9%84-%D9%81%D8%B1%DB%8C%D8%B2%D8%B1" "Mozilla/5.0 (Linux; Android 5.0; SM-G900
H 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.

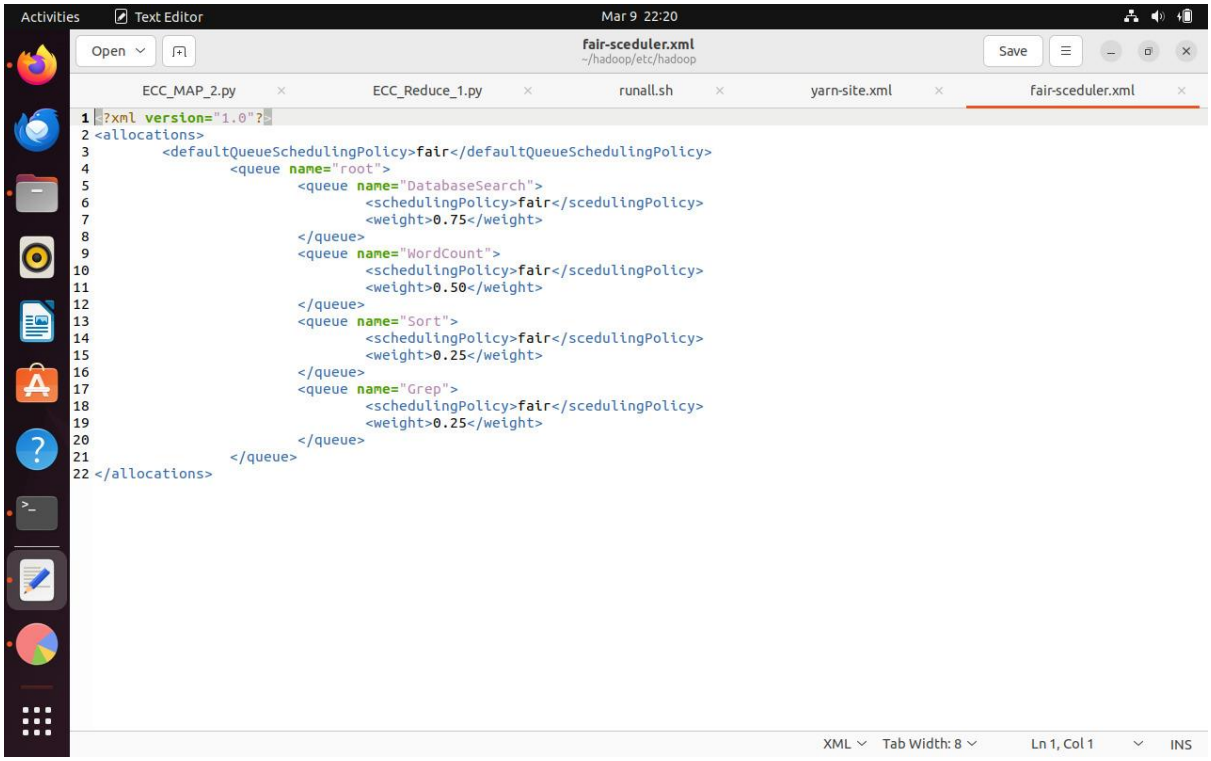


yarn-site.xml for fair scheduler

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

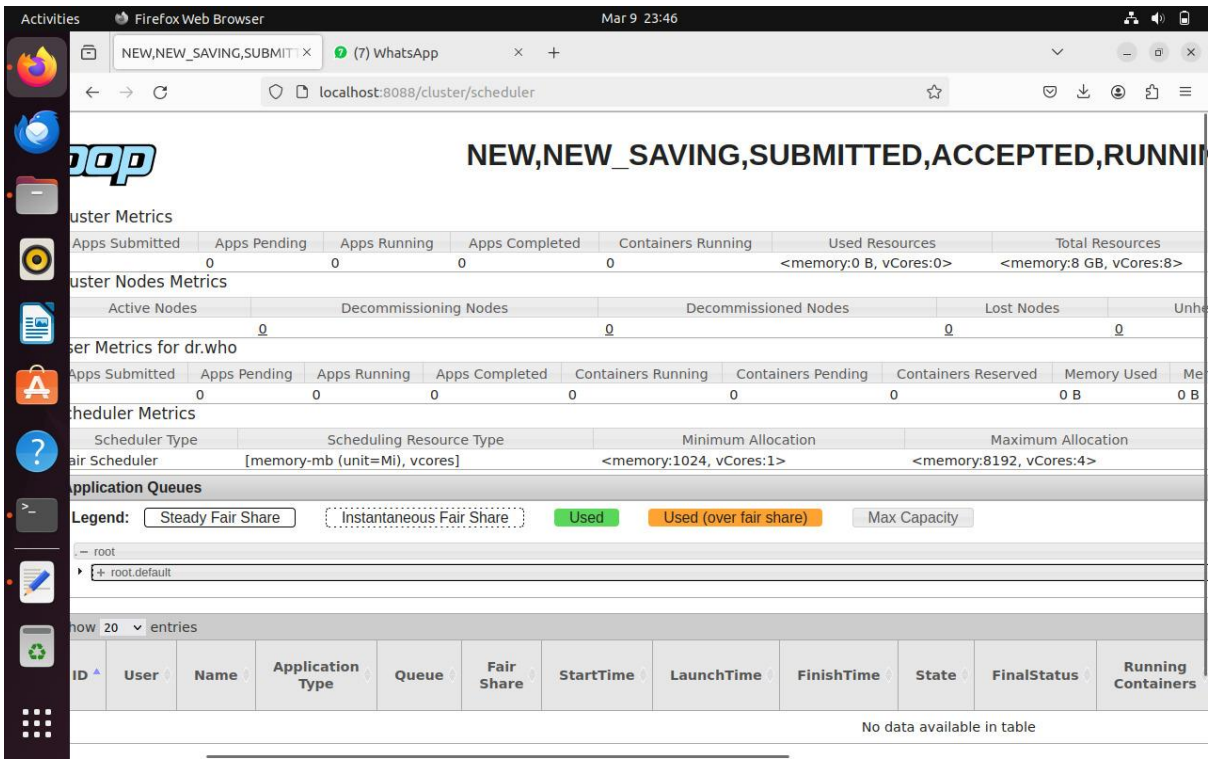
1. Database Search
2. Word Count
3. Sort

4. Grep



fair-scheduler.xml

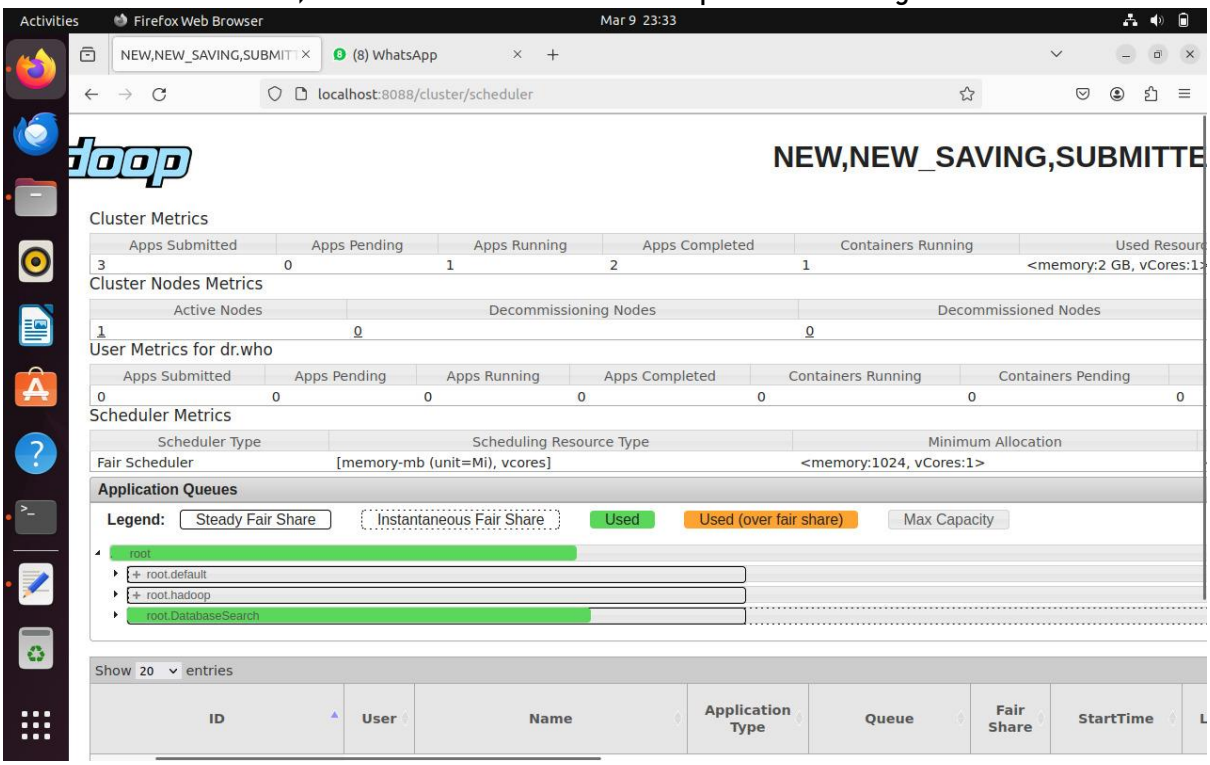
→ Below are the queues which are created in Hadoop.



Initial Queues in Hadoop

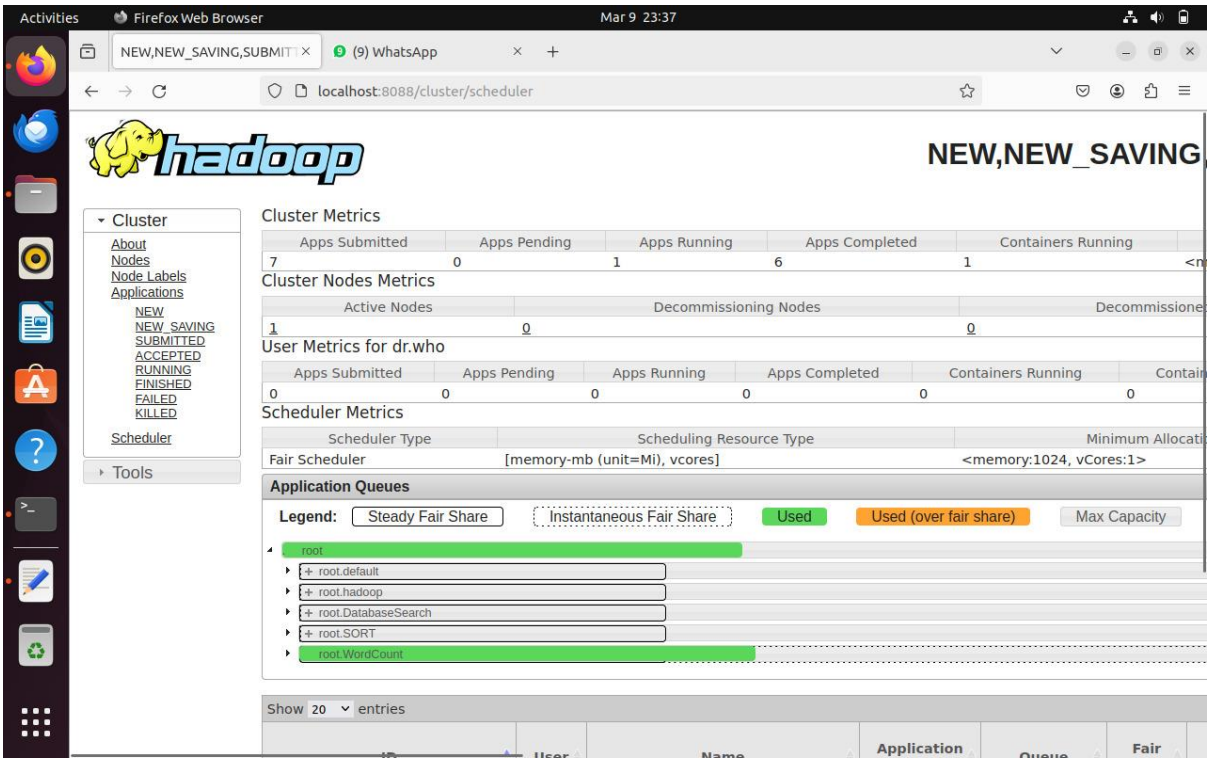
→ In fair-scheduler.xml, I have given more weight database search as it was consuming more resources and initially it failed hence, I increased the weight.

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



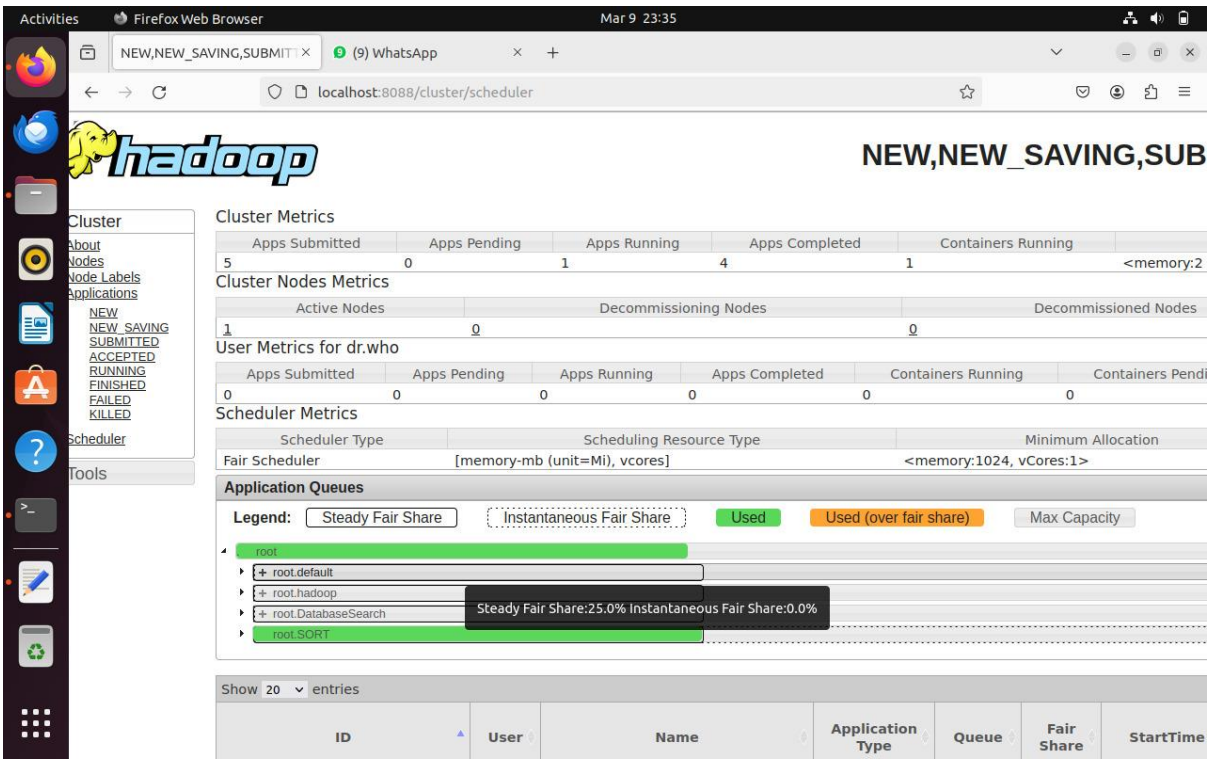
DatabaseSearch Queue running first

→ 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.

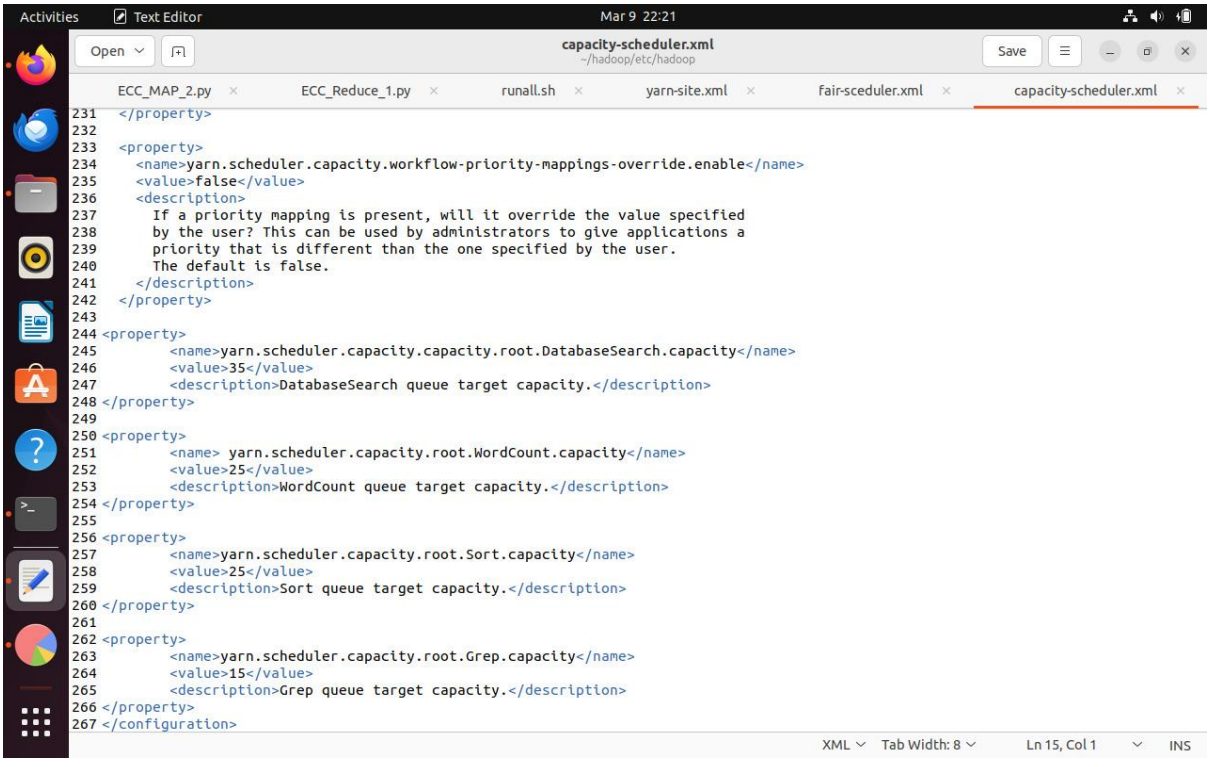


Grep and Sort queue running concurrently

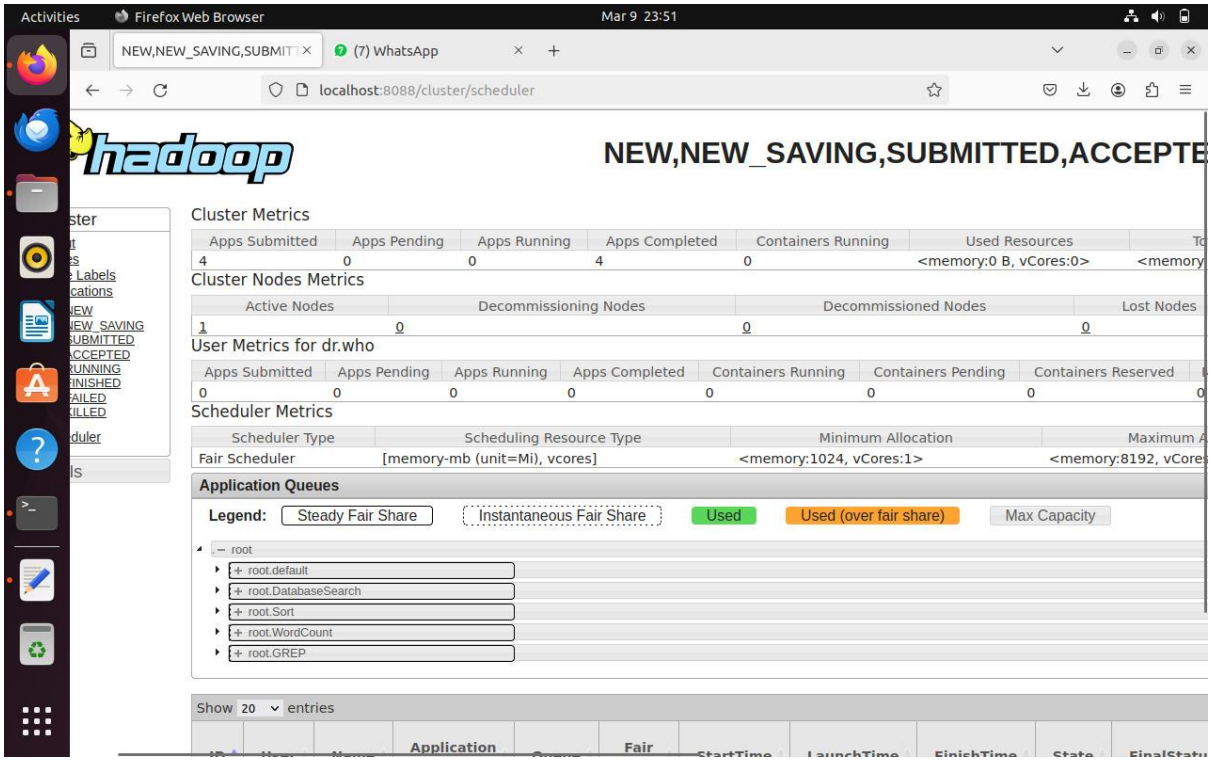
❖ Testing the Capacity Scheduler:

→ To implement capacity scheduler, I modified capacity-scheduler.xml.

→ I have conured databaseSearch, wordcount, sort, grep queues in my capacity-scheduler.xml as shown in below screenshot.



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



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 smoothy in fair scheduling.

❖ Conclusion:

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