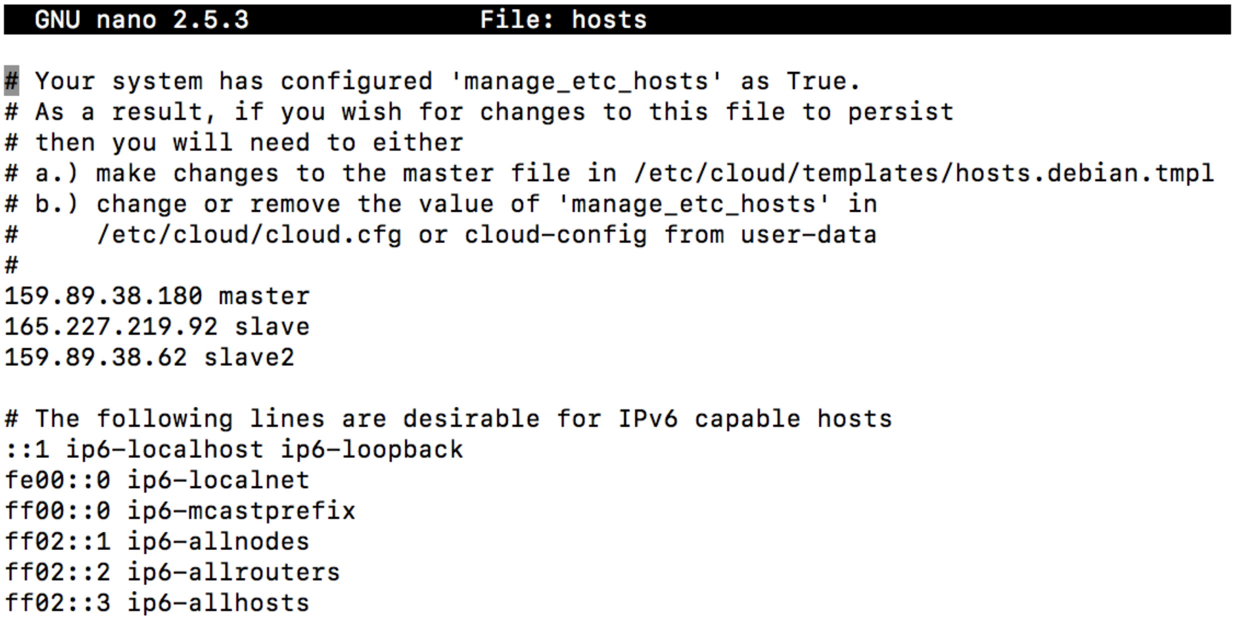
INFSCI 2750: Cloud Computing

Mini Project 1

Weiqi Yu (WEY46) | Ningjuan Zhu (NIZ21) | Han-Tse Cheng(HAC117)

Part 1.1 Setting up Hadoop:

For the first part of the project, a Hadoop cluster with 3 VMs have been built and configured. One Name Node and two Data Nodes were assigned as follow:



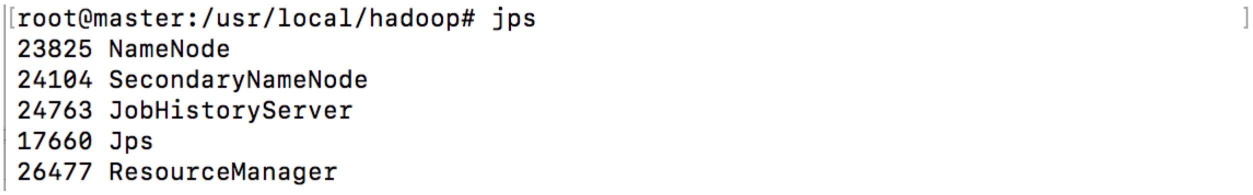
```
  GNU nano 2.5.3                 File: hosts

# Your system has configured 'manage_etc_hosts' as True.
# As a result, if you wish for changes to this file to persist
# then you will need to either
# a.) make changes to the master file in /etc/cloud/templates/hosts.debian.tmpl
# b.) change or remove the value of 'manage_etc_hosts' in
#     /etc/cloud/cloud.cfg or cloud-config from user-data
#
159.89.38.180 master
165.227.219.92 slave
159.89.38.62 slave2

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

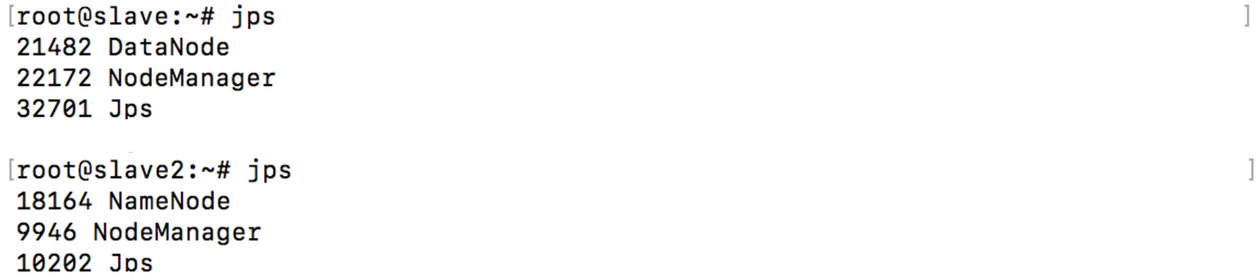
Screenshot 1. Hosts of the Hadoop cluster

To verify the cluster has been successfully setup, we have processed the *wordcount* example. While running, the daemons on the different VMs are shown in below respectively:



```
root@master:/usr/local/hadoop# jps
23825 NameNode
24104 SecondaryNameNode
24763 JobHistoryServer
17660 Jps
26477 ResourceManager
```

Screenshot 2. jsp of Master Node VM

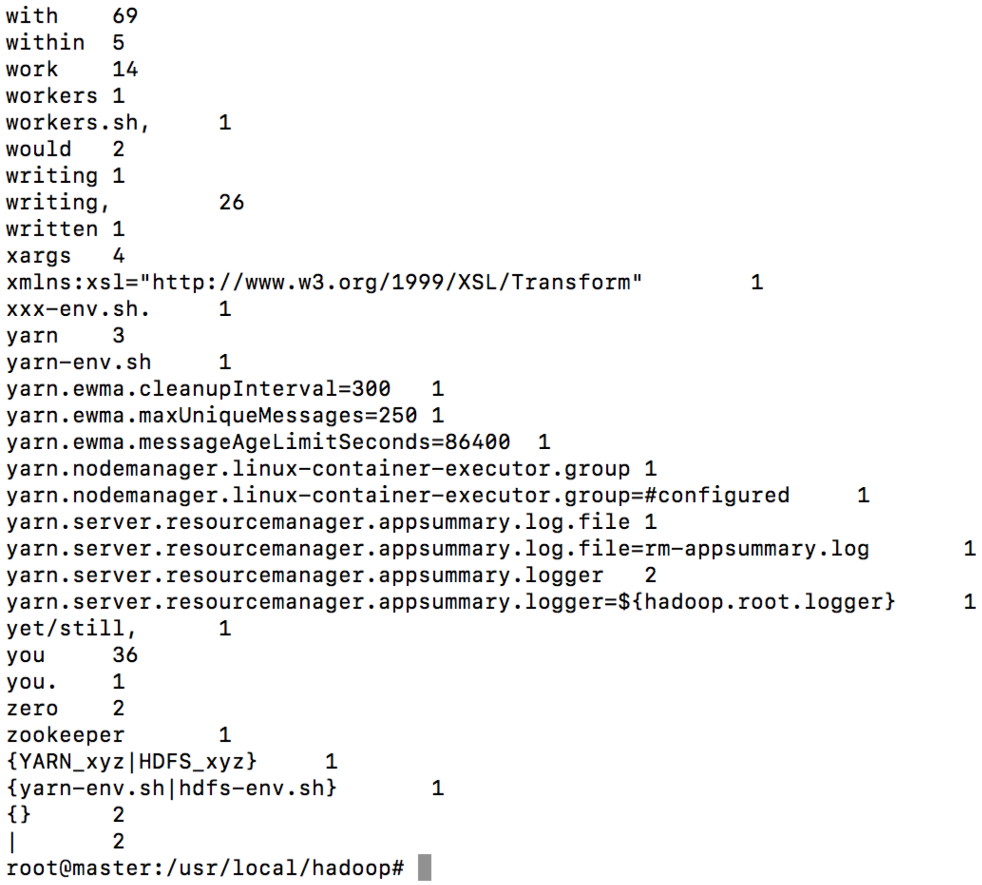


```
root@slave:~# jps
21482 DataNode
22172 NodeManager
32701 Jps

root@slave2:~# jps
18164 NameNode
9946 NodeManager
10202 Jps
```

Screenshot 3. jsp of Data Node VMs

The *wordcount* example has been finished successfully, part of the output as:

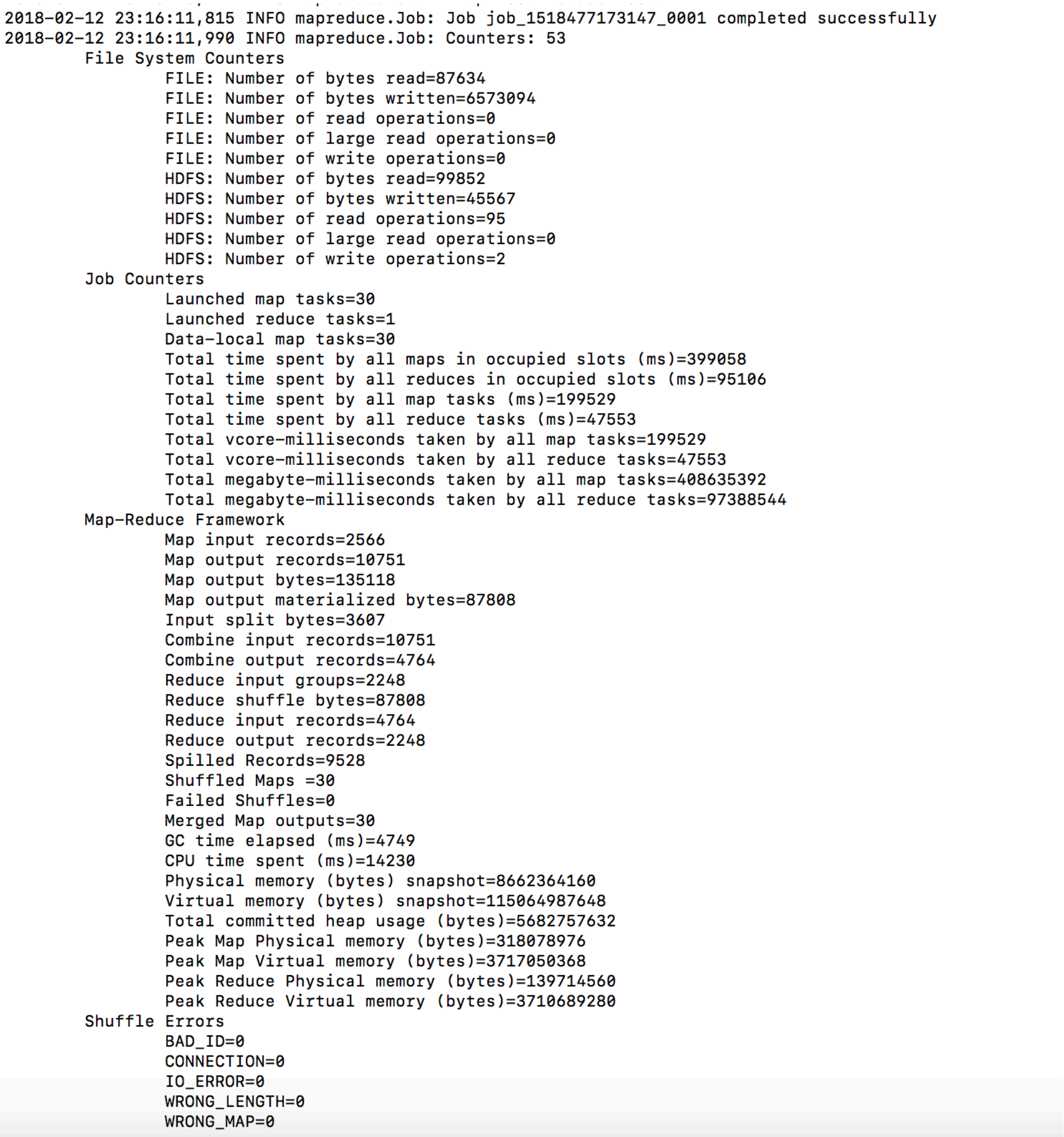


```
with    69
within  5
work    14
workers 1
workers.sh,     1
would   2
writing 1
writing,        26
written 1
xargs   4
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"        1
xxx-env.sh.     1
yarn    3
yarn-env.sh     1
yarn.ewma.cleanupInterval=300   1
yarn.ewma.maxUniqueMessages=250 1
yarn.ewma.messageAgeLimitSeconds=86400  1
yarn.nodemanager.linux-container-executor.group 1
yarn.nodemanager.linux-container-executor.group=#configured     1
yarn.server.resourcemanager.appsummary.log.file 1
yarn.server.resourcemanager.appsummary.log.file=rm-appsummary.log       1
yarn.server.resourcemanager.appsummary.logger   2
yarn.server.resourcemanager.appsummary.logger=${hadoop.root.logger}     1
yet/still,      1
you     36
you.    1
zero    2
zookeeper       1
{YARN_xyz|HDFS_xyz}     1
{yarn-env.sh|hdfs-env.sh}       1
{}      2
|       2
root@master:/usr/local/hadoop# 
```

Screenshot 4. cat of the output of the *worldcount* example

Part 1.2 Docker Image:

Based on the previous part, one Hadoop Docker image has been built which can quickly deploy Hadoop as we did in the previous step. The Dockerfile and other support files are included separately in the "part_1_2_docker_hadoop3" folder. The built Docker image has been tested by running the same *wordcount* job using a bootstrap script:

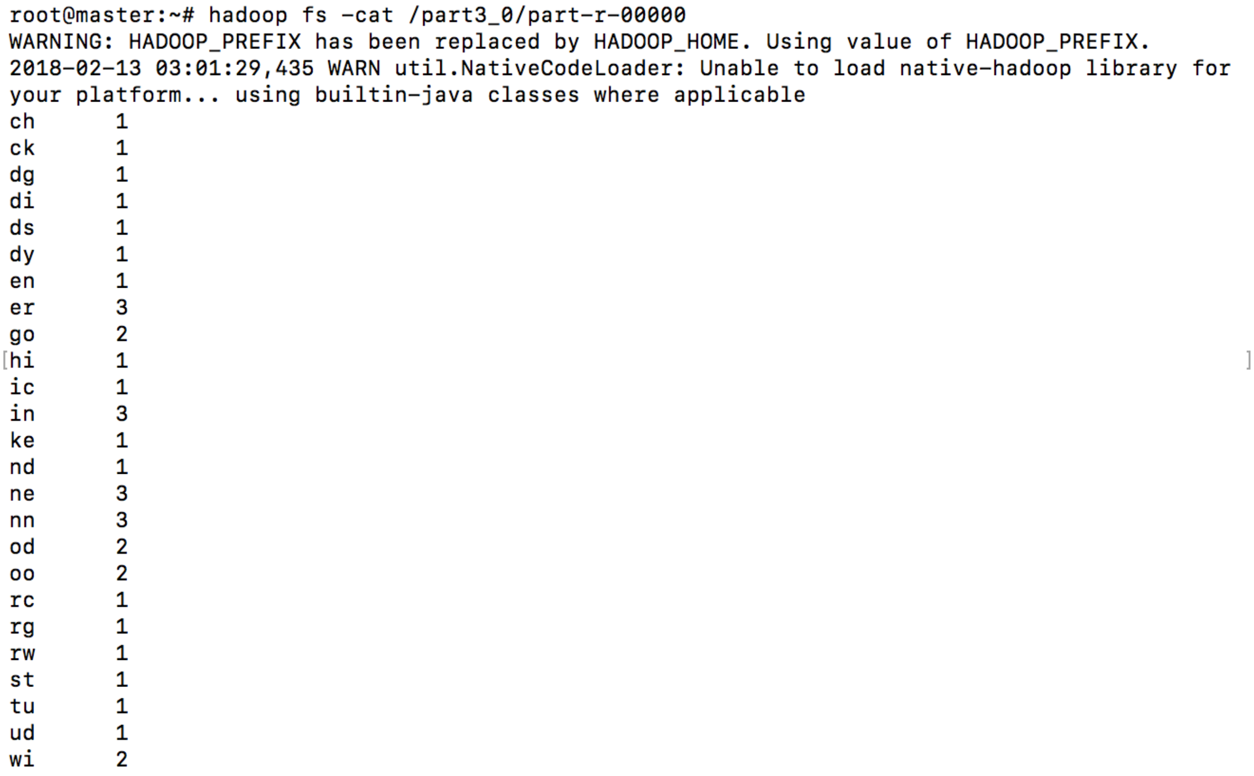


```
2018-02-12 23:16:11,815 INFO mapreduce.Job: Job job_1518477173147_0001 completed successfully
2018-02-12 23:16:11,990 INFO mapreduce.Job: Counters: 53
        File System Counters
                FILE: Number of bytes read=87634
                FILE: Number of bytes written=6573094
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=99852
                HDFS: Number of bytes written=45567
                HDFS: Number of read operations=95
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=30
                Launched reduce tasks=1
                Data-local map tasks=30
                Total time spent by all maps in occupied slots (ms)=399058
                Total time spent by all reduces in occupied slots (ms)=95106
                Total time spent by all map tasks (ms)=199529
                Total time spent by all reduce tasks (ms)=47553
                Total vcore-milliseconds taken by all map tasks=199529
                Total vcore-milliseconds taken by all reduce tasks=47553
                Total megabyte-milliseconds taken by all map tasks=408635392
                Total megabyte-milliseconds taken by all reduce tasks=97388544
        Map-Reduce Framework
                Map input records=2566
                Map output records=10751
                Map output bytes=135118
                Map output materialized bytes=87808
                Input split bytes=3607
                Combine input records=10751
                Combine output records=4764
                Reduce input groups=2248
                Reduce shuffle bytes=87808
                Reduce input records=4764
                Reduce output records=2248
                Spilled Records=9528
                Shuffled Maps =30
                Failed Shuffles=0
                Merged Map outputs=30
                GC time elapsed (ms)=4749
                CPU time spent (ms)=14230
                Physical memory (bytes) snapshot=8662364160
                Virtual memory (bytes) snapshot=115064987648
                Total committed heap usage (bytes)=5682757632
                Peak Map Physical memory (bytes)=318078976
                Peak Map Virtual memory (bytes)=3717050368
                Peak Reduce Physical memory (bytes)=139714560
                Peak Reduce Virtual memory (bytes)=3710689280
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
```

Screenshot 5. testing Docker image result

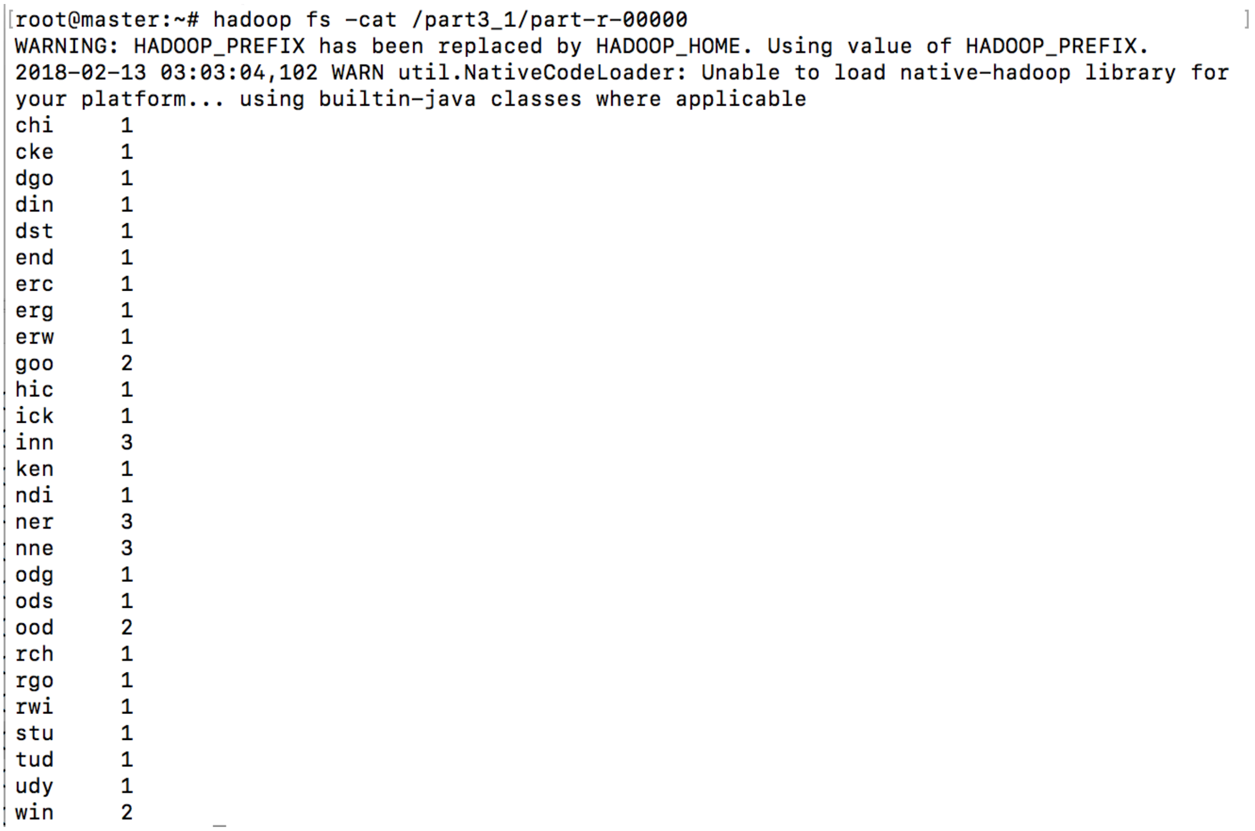
Part 3 Hadoop program - n-gram:

A n-gram Hadoop program has been implemented to produce the n-gram frequencies of the input file with given *n* as a parameter. To test our program, we have performed n-gram with *n* equals to 2 and 3 respectively on the input file with simple text “winner winner chicken dinner good good study”. The *n* must be passed as a parameter in args[2], otherwise an error will occur. The results are shown below:



```
root@master:~# hadoop fs -cat /part3_0/part-r-00000
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
2018-02-13 03:01:29,435 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
ch      1
ck      1
dg      1
di      1
ds      1
dy      1
en      1
er      3
go      2
hi      1
ic      1
in      3
ke      1
nd      1
ne      3
nn      3
od      2
oo      2
rc      1
rg      1
rw      1
st      1
tu      1
ud      1
wi      2
```

Screenshot 6. N-gram result with n = 2



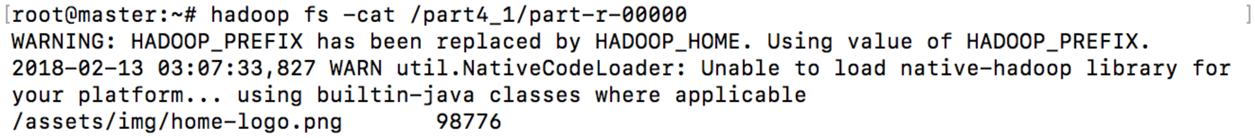
```
root@master:~# hadoop fs -cat /part3_1/part-r-00000
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
2018-02-13 03:03:04,102 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
chi     1
cke     1
dgo     1
din     1
dst     1
end     1
erc     1
erg     1
erw     1
goo     2
hic     1
ick     1
inn     3
ken     1
ndi     1
ner     3
nne     3
odg     1
ods     1
ood     2
rch     1
rgo     1
rwi     1
stu     1
tud     1
udy     1
win     2
```

Screenshot 7. N-gram result with n = 3

Part 4 Hadoop program - Log analysis:

1. How many hits were made to the website item “/assets/img/home- logo.png”?

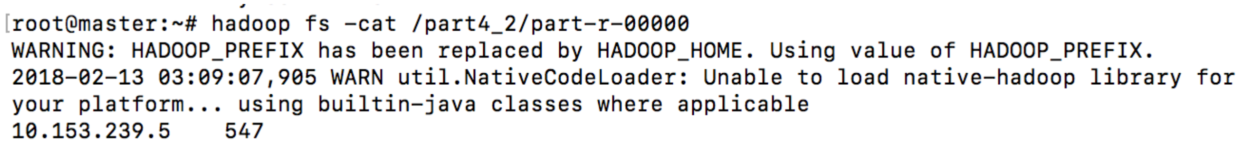
Answer: 98776 hits



```
root@master:~# hadoop fs -cat /part4_1/part-r-00000
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
2018-02-13 03:07:33,827 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
/assets/img/home-logo.png       98776
```

2. How many hits were made from the IP: 10.153.239.5

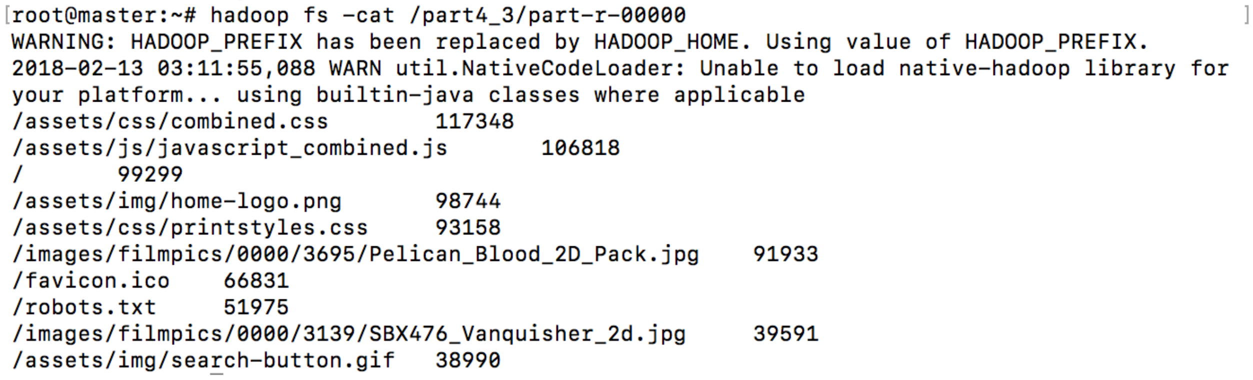
Answer: 547



```
root@master:~# hadoop fs -cat /part4_2/part-r-00000
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
2018-02-13 03:09:07,905 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
10.153.239.5    547
```

3. Which path in the website has been hit most? How many hits were made to the path?

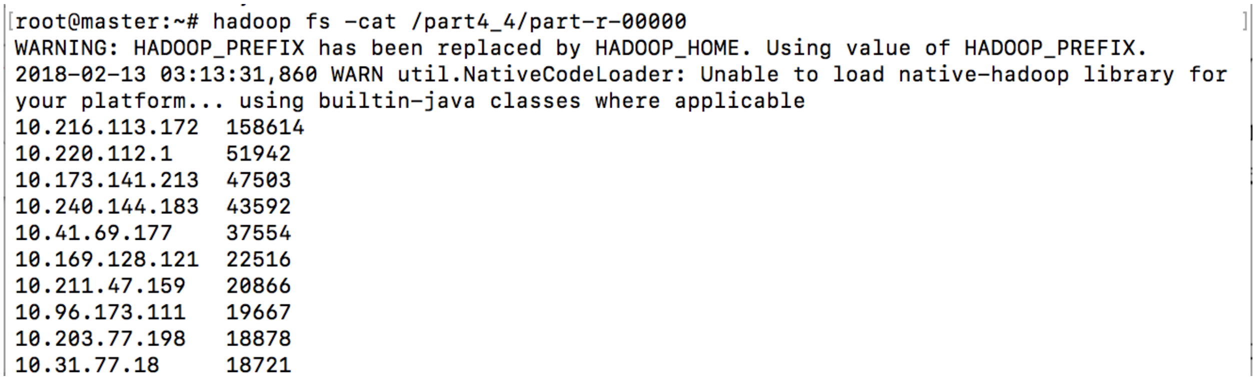
Answer: /assets/css/combined.css, with 117348 hits



```
root@master:~# hadoop fs -cat /part4_3/part-r-00000
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
2018-02-13 03:11:55,088 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
/assets/css/combined.css        117348
/assets/js/javascript_combined.js       106818
/       99299
/assets/img/home-logo.png       98744
/assets/css/printstyles.css     93158
/images/filmpics/0000/3695/Pelican_Blood_2D_Pack.jpg    91933
/favicon.ico    66831
/robots.txt     51975
/images/filmpics/0000/3139/SBX476_Vanquisher_2d.jpg     39591
/assets/img/search-button.gif   38990
```

4. Which IP accesses the website most? How many accesses were made by it?

Answer:10.216.113.172, with 158614 hits



```
root@master:~# hadoop fs -cat /part4_4/part-r-00000
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
2018-02-13 03:13:31,860 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
10.216.113.172  158614
10.220.112.1    51942
10.173.141.213  47503
10.240.144.183  43592
10.41.69.177    37554
10.169.128.121  22516
10.211.47.159   20866
10.96.173.111   19667
10.203.77.198   18878
10.31.77.18     18721
```

Note: for the questions 3 and 4, we developed Hadoop programs to get top 10 hits by IP/URL