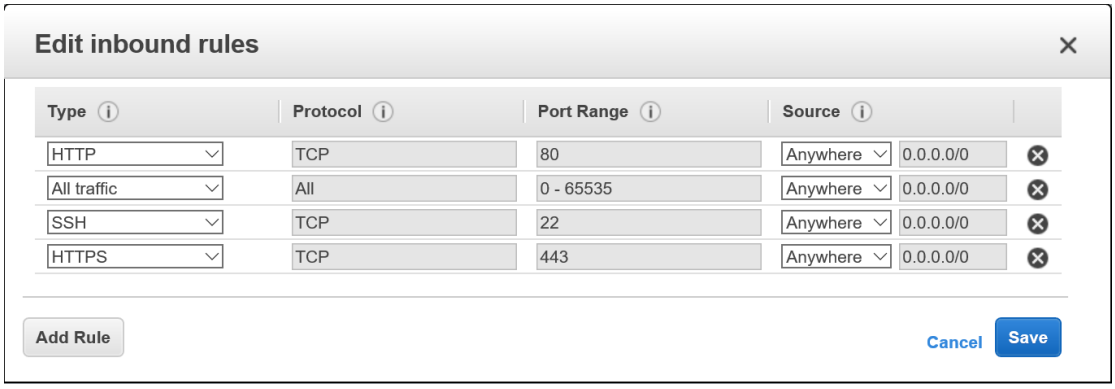
IEMS 5709 Homework 1 Sample Solution

Q1: Multi-node Hadoop cluster setup

a) Install a single-node Hadoop

i. Update EC2 Security Group

In order to allow Hadoop servers communicate with outside as well as each other, we should update EC2 Security Group to let network traffic on certain ports to pass through. In this case, to reduce the difficulty of configuration, I set the Security Group to allow any network traffic to pass.

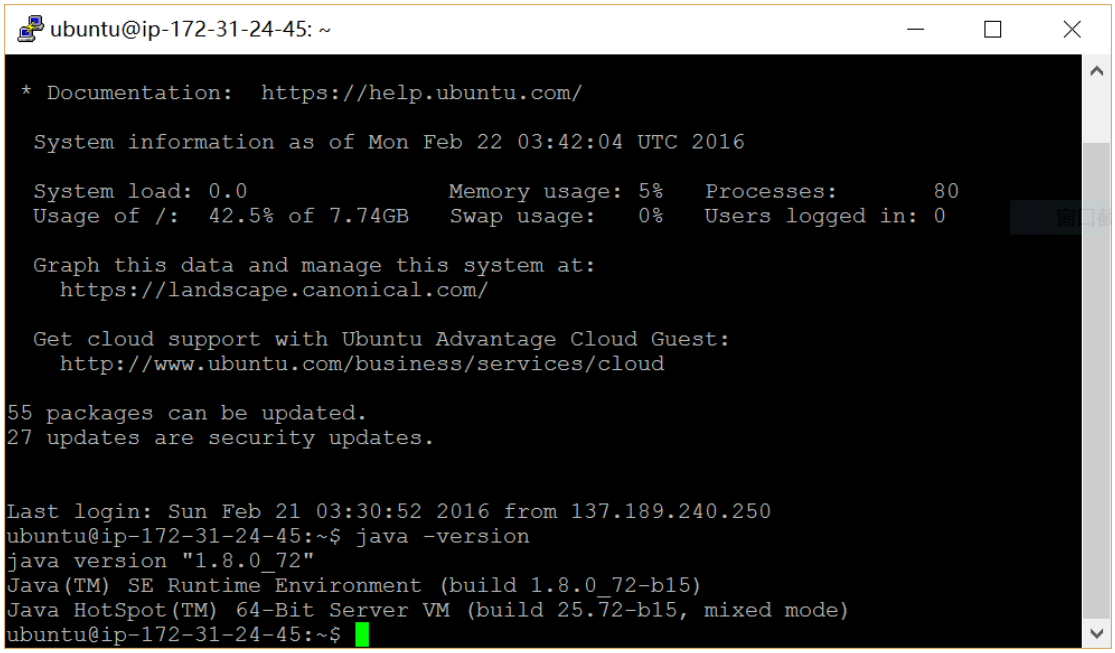


ii. Install Java on server

In Linux shell, I use below commands to install and get install result of Java.

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
java –version
```

Below is the screenshot of Java version information.

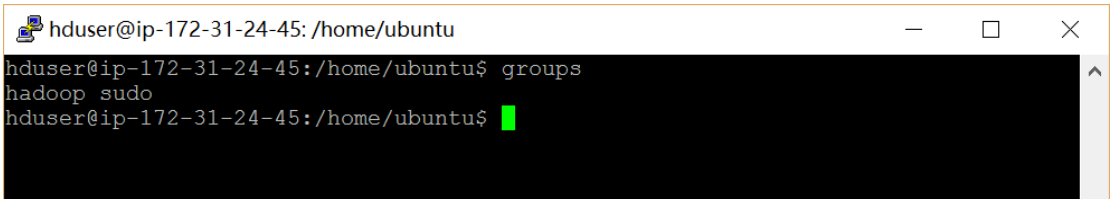


iii. Create Hadoop user and set proper privilege

In Linux shell, I use below commands to do these work.

```
sudo addgroup hadoop
sudo adduser --ingroup hadoop hduser
sudo addgroup hadoop
sudo adduser --ingroup hadoop hduser
sudo usermod -a -G sudo hduser
```

Use groups command to check user group information.

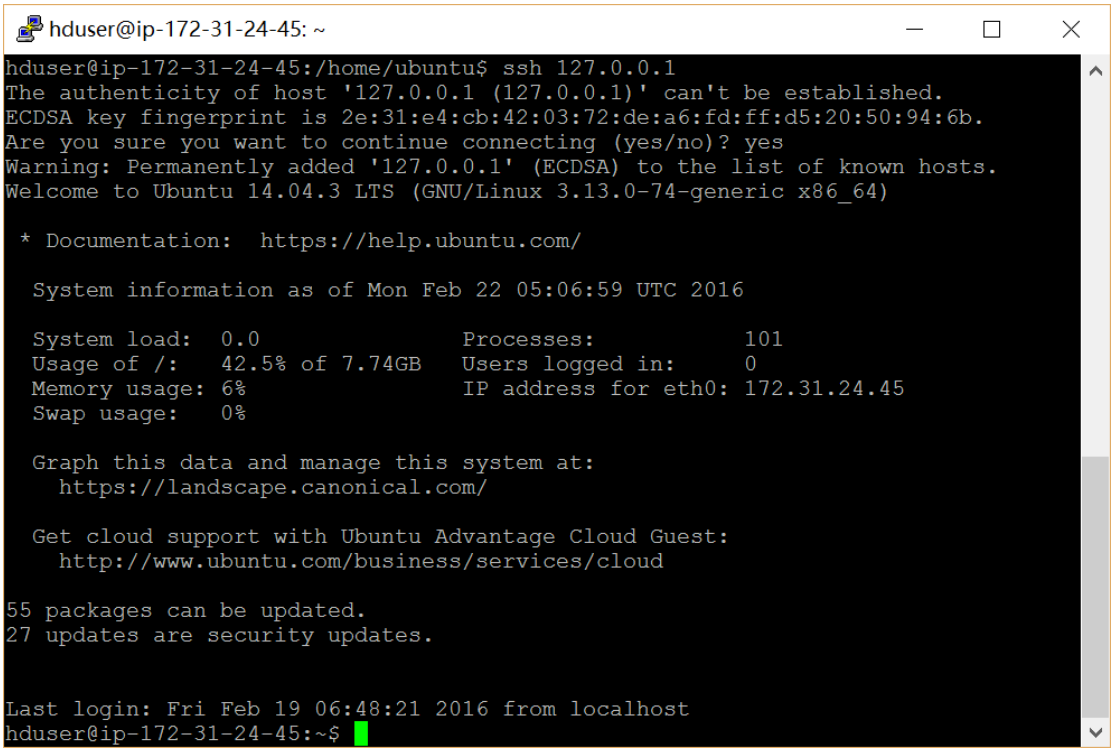


iv. Install and Configure SSH

In Linux shell, I use below commands to install and configure SSH Client.

```
sudo apt-get install openssh-server
sudo su hduser
ssh-keygen -t rsa -P ""
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

After doing these, I use ssh 127.0.0.1 command to test whether all settings are fine.



v. Download and set up Hadoop resources

In Linux shell, I use below commands to do these work.

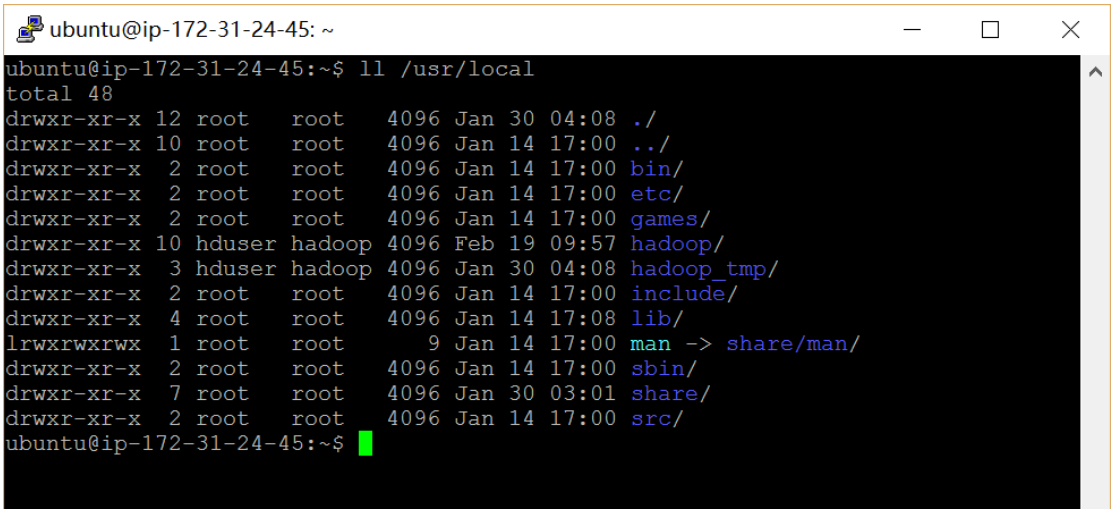
```
wget http://ftp.cuhk.edu.hk/pub/packages/apache.org/hadoop/common/hadoop-2.7.1/hadoop-2.7.1.tar.gz
sudo tar -xzvf hadoop-2.7.1.tar.gz
sudo mv hadoop-2.7.1/* /usr/local/hadoop
sudo chown hduser:hadoop -R /usr/local/Hadoop
```

sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode

sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode

sudo chown hduser:hadoop -R /usr/local/hadoop_tmp/

After doing these, I use ll /usr/local/ command to check the result.



vi. Update Hadoop configuration files

In this part I update bellowing files according to the guidance of *How to install Apache Hadoop 2.6.0 in Ubuntu (Single node setup).*

$HOME/.bashcr hadoop-env.sh core-site.xml hdfs-site.xml

yarn-site.xml Mapred-site.xml

vii. Format Namenode and start Hadoop Daemons

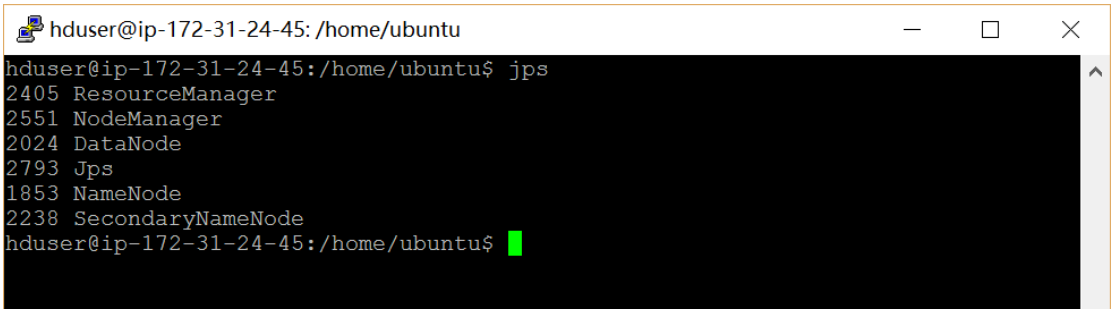
I use below commands to do these works.

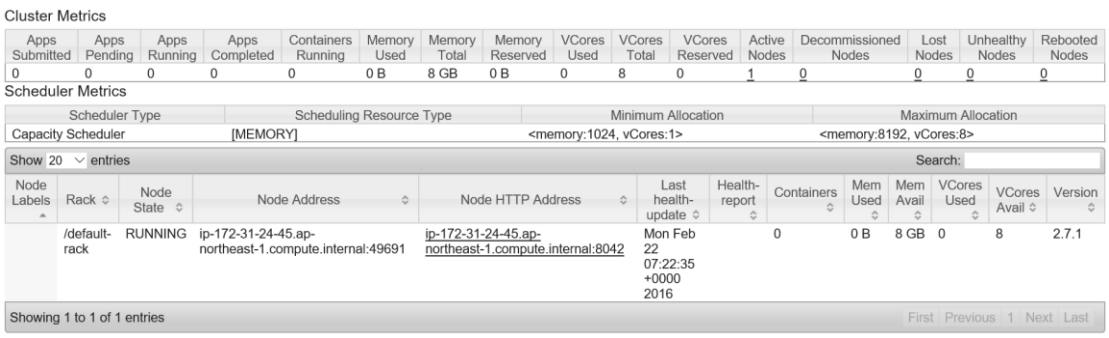
hdfs namenode –format

start-dfs.sh

start-yarn.sh

After doing all these, I use jps command and website to check the Hadoop statue.





b) Install and setup a multi-node Hadoop cluster with 4 VMs

i. Create Image and Use it launch three more instances

Using Single-node Hadoop server created before, I built an AMI image, and created three more instance based on it.

Name	Instance ID	Instance Type	Public IP	Key Name	Security Groups	Image ID
HadoopMaster	i-2e68418b	t2.micro	52.193.207.30	howard-key-pair	howard-SG-tky	ami-a21529cc
HadoopSlave1	i-50ae87f5	t2.micro	52.192.40.106	howard-key-pair	howard-SG-tky	ami-b75962d9
HadoopSlave2	i-6eac85cb	t2.micro	52.192.139.9	howard-key-pair	howard-SG-tky	ami-b75962d9
HadoopSlave3	i-aea1880b	t2.micro	52.193.223.99	howard-key-pair	howard-SG-tky	ami-b75962d9

ii. Edit hosts file

On both HadoopMaster and three HadoopSlave servers, edit their hosts files like below, thus they can communicate using hostnames, and here I used EC2 private IP instead of public IP.

```
127.0.0.1 localhost
172.31.27.227 HadoopMaster
172.31.23.146 HadoopSlave1
172.31.21.248 HadoopSlave2
172.31.25.125 HadoopSlave3
```

iii. Create ssh keys and register them into other servers

To let HadoopMaster control HadoopSlaves, we should create ssh key for it, and register the key to all HadoopSlaves. Moreover, to ease the remote control between servers, I also create ssh keys for all HadoopSlave servers and register them into each other and HadoopMaster. Commands below shows how I do it in HadoopMaster, the process is similar on all other HadoopSlave server.

```
ssh-keygen -t rsa -P ""
ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@HadoopSlave1
ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@HadoopSlave2
ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@HadoopSlave3
```

iv. Update Hadoop configuration files

In this part I update bellowing files on HadoopMaster according to the guidance of How to install Apache Hadoop 2.6.0 in Ubuntu (Multi node/Cluster setup), and copy them to the same path of three HadoopSlaves.

```
core-site.xml    hdfs-site.xml    yarn-site.xml    Mapred-site.xml
```

v. Update master and nodes files

To let Hadoop system successfully find Master and Slave nodes, we should update master and nodes configuration files on each node, below is the content of these two files.

Master file content:

```
HadoopMaster
```

Slave file content:

```
HadoopSlave1
HadoopSlave2
HadoopSlave3
```

vi. Set privilege in HadoopMaster

In Linux shell, I use below commands to set proper privilege for HadoopMaster.

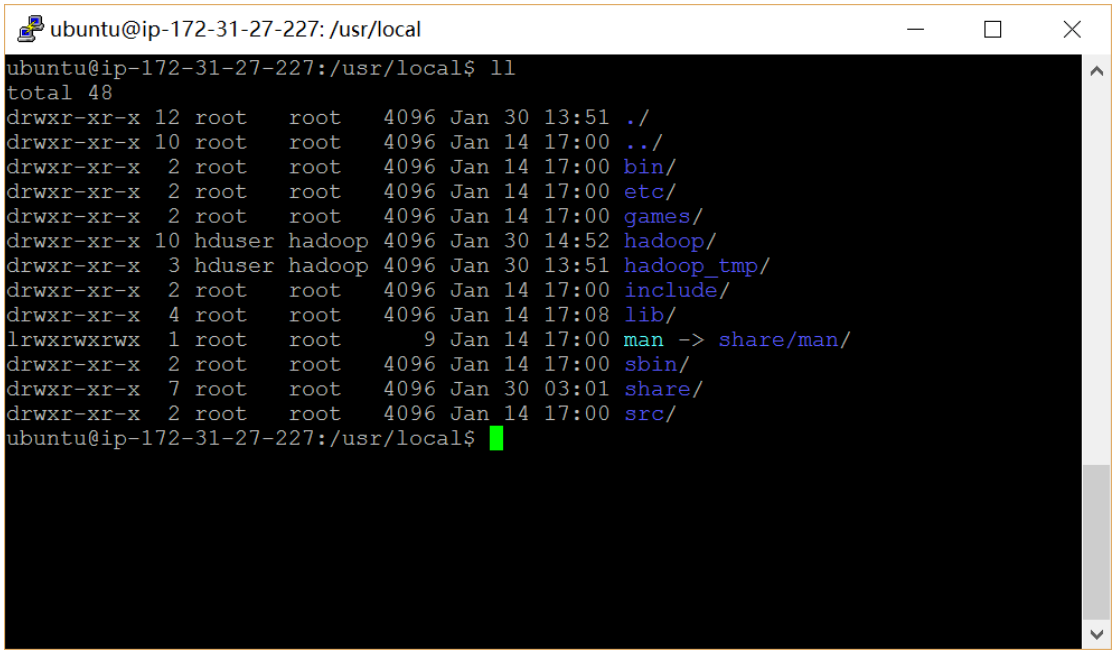
sudo rm -rf /usr/local/hadoop_tmp/

sudo mkdir -p /usr/local/hadoop_tmp/

sudo mkdir -p /usr/local/hadoop_tmp/hdfs/Namenode

sudo chown hduser:hadoop -R /usr/local/hadoop_tmp/

After doing these, I use ll command to check the result.



vii. Set privilege in HadoopSlaves

In Linux shell, I use below commands to set proper privilege for HadoopSlaves.

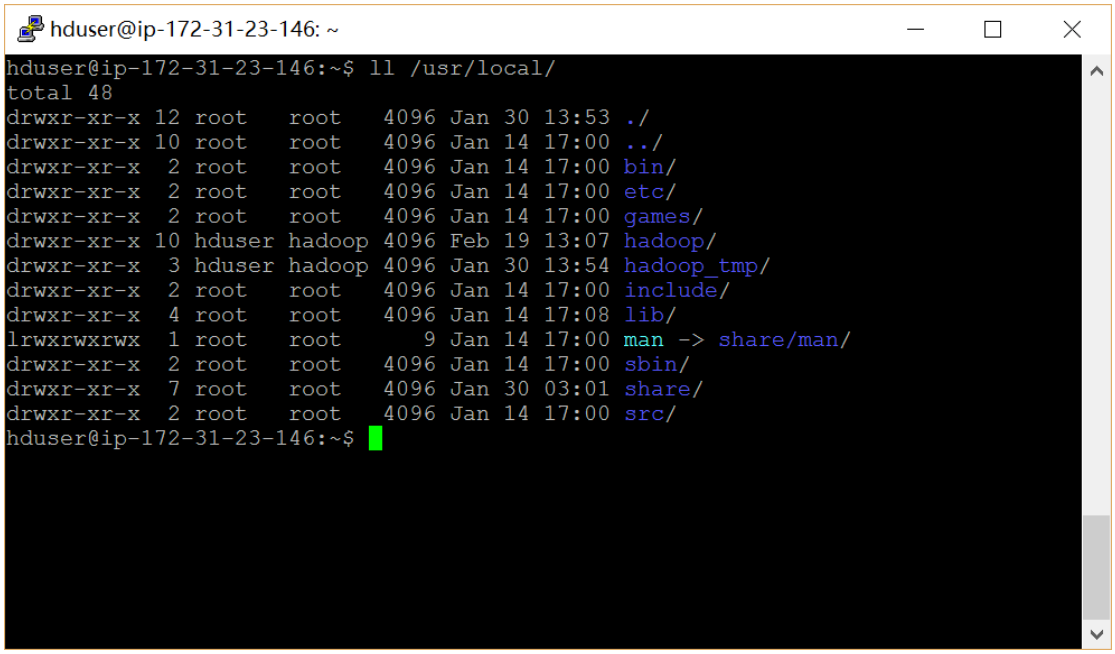
sudo rm -rf /usr/local/hadoop_tmp/hdfs/

sudo mkdir -p /usr/local/hadoop_tmp/

sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode

sudo chown hduser:hadoop -R /usr/local/hadoop_tmp/

After doing these, I use ll command to check the result.



viii. Format Namenode and start Hadoop Daemons

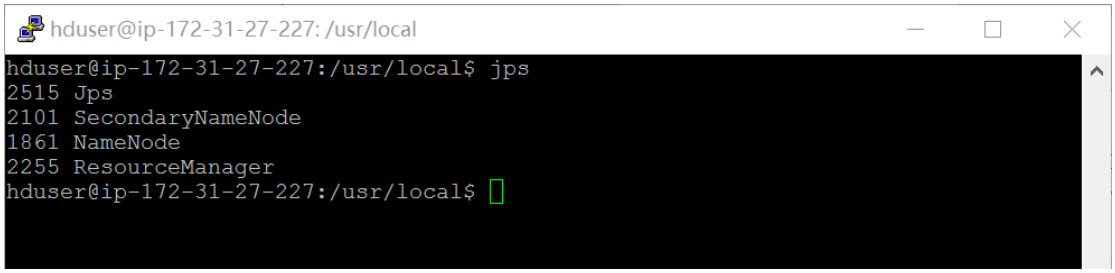
I use below commands to do these works.

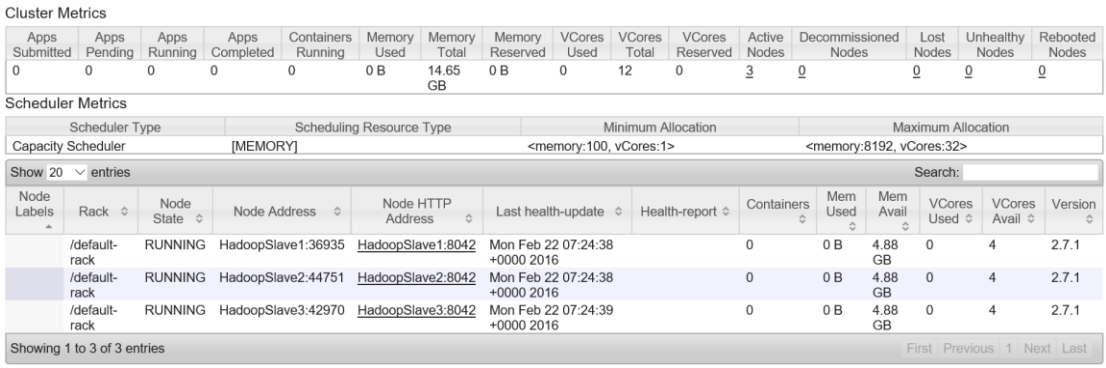
`hdfs namenode –format`

`start-dfs.sh`

`start-yarn.sh`

After doing all these, I use `jps` command and website to check the Hadoop statue.





Q2: Basic text processing with Hadoop

a) Setup

i. Download and unzip files

In Linux shell, I use following commands to do these works.

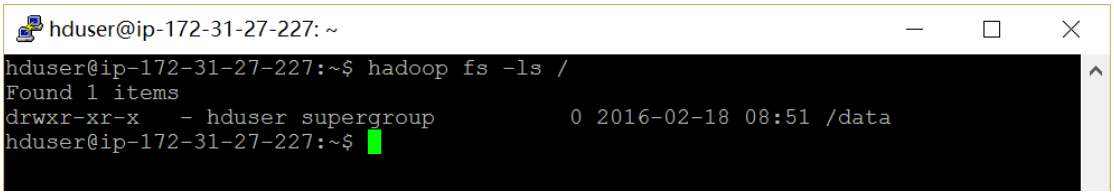
```
wget -c "https://github.com/YangRonghai/100ebooks/blob/master/data.tar.gz?raw=true" -O data.tar.gz
tar -xzvf data.tar.gz
```

ii. Upload /data to Hadoop File System

In Linux shell, I use below commands to upload files.

```
hadoop fs –put /home/hduser/data/ /
```

After doing this, I use `hadoop fs -ls /` command to check the result.



b) Basic word counting

In this task, I write two Python Script files mapper.py and reduer.py to do the work. Below is the code of these two files.

mapper.py

```
#!/usr/bin/env python
import string
import sys
delchars = string.punctuation + string.digits
for line in sys.stdin:
    line = line.translate(None, delchars)
    line = line.strip()
    line = line.lower()
    words = line.split()
    for word in words:
        print "%s\t\t%s" %(word,1)
```

reducer.py

```
#!/usr/bin/env python
import operator
import sys
wordcount = {}
totalword = 0
distinct = 0
for line in sys.stdin:
    line = line.strip()
    word, count = line.split("\t", 1)
    try:
        count = int(count)
        accumu = wordcount.get(word,0)
        wordcount[word] = accumu + count
        totalword += count
        if accumu==0:
            distinct += 1
        else:
            pass
    except ValueError:
        pass
```

```
ftotal = float(totalword)
sorted_wordcount = sorted(wordcount.items(), key=operator.itemgetter(1), reverse=Tru
e)
print "total words: %s" %totalword
print "distinct words: %s" %distinct
for word, count in sorted_wordcount:
print "%s\t\t%s\t\t%.4f%%" %(word,count,count/ftotal*100)
```

Then I use below command to start MapReduce job.

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.7.1.jar -map
per /home/hduser/mapper.py -reducer /home/hduser/reducer.py -input /data/* -outp
ut /withstop
```

Below is part of execution logs of this MapReduce job.

```
hduser@ip-172-31-27-227: ~
16/02/22 07:59:08 INFO mapreduce.Job:  map 100% reduce 89%
16/02/22 07:59:11 INFO mapreduce.Job:  map 100% reduce 92%
16/02/22 07:59:14 INFO mapreduce.Job:  map 100% reduce 95%
16/02/22 07:59:17 INFO mapreduce.Job:  map 100% reduce 99%
16/02/22 07:59:19 INFO mapreduce.Job:  map 100% reduce 100%
16/02/22 07:59:19 INFO mapreduce.Job: Job job_1456125517597_0001 completed success
fully
16/02/22 07:59:19 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=211614513
                FILE: Number of bytes written=413290341
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=96337573
                HDFS: Number of bytes written=5645686
                HDFS: Number of read operations=891
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=296
                Launched reduce tasks=1
                Data-local map tasks=296
                Total time spent by all maps in occupied slots (ms)=45279080
                Total time spent by all reduces in occupied slots (ms)=4531505
                Total time spent by all map tasks (ms)=4116280
                Total time spent by all reduce tasks (ms)=411955
                Total vcore-seconds taken by all map tasks=4116280
                Total vcore-seconds taken by all reduce tasks=411955
                Total megabyte-seconds taken by all map tasks=4215070720
                Total megabyte-seconds taken by all reduce tasks=421841920
        Map-Reduce Framework
```

And through the output file of this job, we can easily get the total number of words in these EBooks is **15939232**.

```
hduser@ip-172-31-27-227: ~
total words: 15939232
distinct words: 243584
the             931432          5.8436%
and             559045          3.5074%
of              526127          3.3008%
"./withstop/part-00000" [converted] 243586L, 5924054C          1,1           Top
```

c) File counting

By checking below part of log during last job, we can easily find out there are **296** map tasks and **1** reduce task executed during execution. According to Hadoop scheduling, we known there are **297** works executed during the whole job.

```
        Job Counters
                Launched map tasks=296
                Launched reduce tasks=1
                Data-local map tasks=296
                Total time spent by all maps in occupied slots (ms)=45279080
                Total time spent by all reduces in occupied slots (ms)=4531505
                Total time spent by all map tasks (ms)=4116280
                Total time spent by all reduce tasks (ms)=411955
                Total vcore-seconds taken by all map tasks=4116280
                Total vcore-seconds taken by all reduce tasks=411955
                Total megabyte-seconds taken by all map tasks=4215070720
                Total megabyte-seconds taken by all reduce tasks=421841920
```

d) Distinct word counting

The reducer.py used in question a) already realized the function of distinct word counting, by looking at the output file of job execution, we can easily get the total number of distinct words in these EBooks is **243584**.

```
hduser@ip-172-31-27-227: ~
total words: 15939232
distinct words: 243584
the             931432          5.8436%
and             559045          3.5074%
of              526127          3.3008%
"./withstop/part-00000" [converted] 243586L, 5924054C          1,1           Top
```

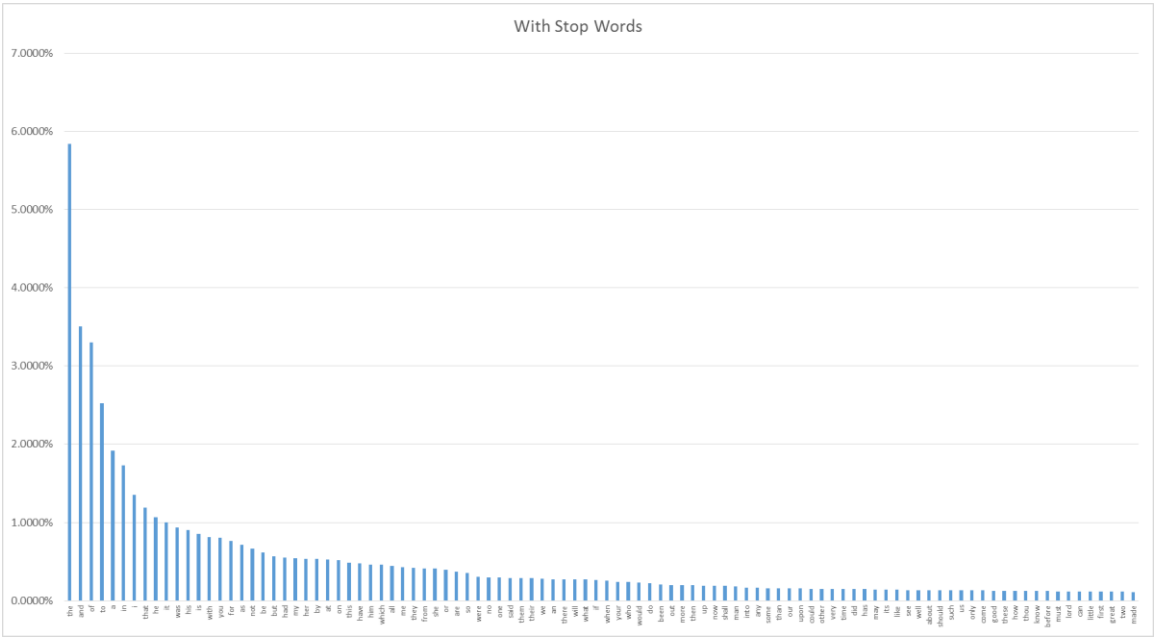
e) Frequent words

The previous used reducer.py also realized the function of counting word frequency, by looking at the output file of job execution, we can easily get the top 100 most frequently used words and their corresponding relative frequencies.

the	931432	5.8436%	him	73615	0.4618%
and	559045	3.5074%	which	73026	0.4582%
of	526127	3.3008%	all	70987	0.4454%
to	402016	2.5222%	me	67794	0.4253%
a	305117	1.9143%	they	67318	0.4223%
in	275915	1.7310%	from	65880	0.4133%
i	215111	1.3496%	she	65871	0.4133%
that	189661	1.1899%	or	62923	0.3948%
he	169506	1.0635%	are	58531	0.3672%
it	158865	0.9967%	so	56903	0.3570%
was	148961	0.9346%	were	48011	0.3012%
his	143113	0.8979%	no	47198	0.2961%
is	135655	0.8511%	one	46839	0.2939%
with	129826	0.8145%	said	46261	0.2902%
you	127885	0.8023%	them	45775	0.2872%
for	121834	0.7644%	their	45684	0.2866%
as	113778	0.7138%	we	44719	0.2806%
not	105755	0.6635%	an	43685	0.2741%
be	98442	0.6176%	there	43541	0.2732%
but	89911	0.5641%	will	43208	0.2711%
had	87515	0.5491%	what	43151	0.2707%
my	85763	0.5381%	if	42117	0.2642%
her	85566	0.5368%	when	41051	0.2575%
by	85183	0.5344%	your	37869	0.2376%
at	83467	0.5237%	who	37546	0.2356%
on	82999	0.5207%	would	36704	0.2303%
this	77692	0.4874%	do	35484	0.2226%
have	75966	0.4766%	been	33063	0.2074%

out	31992	0.2007%	see	21118	0.1325%
more	31949	0.2004%	well	21101	0.1324%
then	31862	0.1999%	about	21086	0.1323%
up	30122	0.1890%	should	20950	0.1314%
now	30028	0.1884%	such	20945	0.1314%
shall	29966	0.1880%	us	20661	0.1296%
man	28891	0.1813%	only	20582	0.1291%
into	26652	0.1672%	come	20446	0.1283%
any	26104	0.1638%	good	20222	0.1269%
some	25272	0.1586%	these	20106	0.1261%
than	24862	0.1560%	how	19651	0.1233%
our	24447	0.1534%	thou	19631	0.1232%
upon	24301	0.1525%	know	19385	0.1216%
could	24080	0.1511%	before	19154	0.1202%
other	24041	0.1508%	must	18951	0.1189%
very	23811	0.1494%	lord	18845	0.1182%
time	23130	0.1451%	can	18523	0.1162%
did	23127	0.1451%	little	18437	0.1157%
has	23006	0.1443%	first	18138	0.1138%
may	22577	0.1416%	great	17909	0.1124%
its	22484	0.1411%	two	17806	0.1117%
like	21992	0.1380%	made	17667	0.1108%

And below is the histogram showing their relative frequency percentage.



f) Frequent words excluding stop words

To filter out stop words and eliminate their affection on frequency calculation, I build a new sw-mapper.py to rerun the word counting job. Below is the code of new reducer.

sw-mapper.py

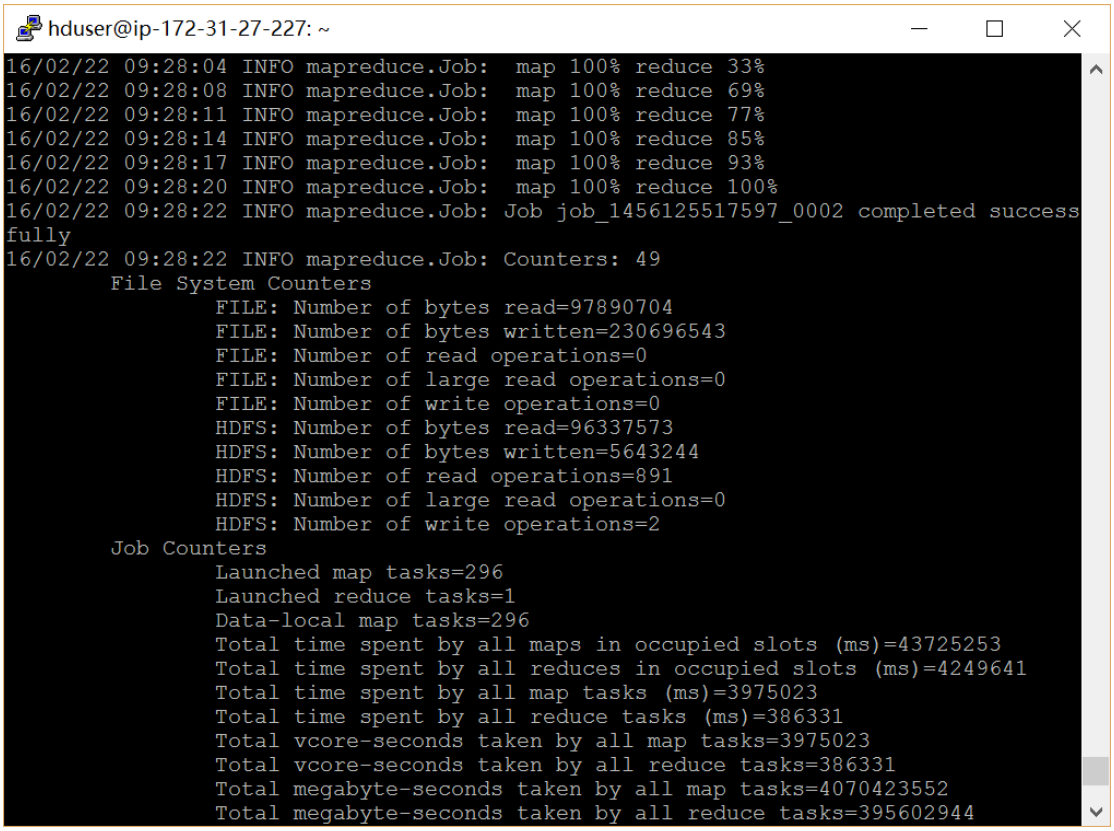
```
#!/usr/bin/env python
import string
import sys
f = open("common-english-words.txt","r")
fline = f.readline()
stopwords = fline.split(",")
```

```
delchars = string.punctuation + string.digits
for line in sys.stdin:
    line = line.translate(None, delchars)
    line = line.strip()
    line = line.lower()
    words = line.split()
    for word in words:
        if word not in stopwords:
            print "%s\t\t%s" %(word,1)
        else:
            pass
```

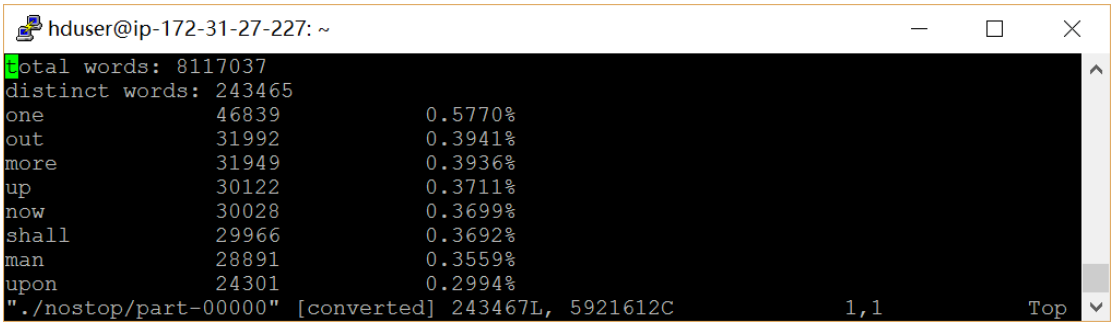
Then I use below command to start MapReduce job.

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.7.1.jar -mapper /home/hduser/sw-mapper.py -reducer /home/hduser/reducer.py -input /data/* -output /nostop
```

Below is part of execution logs of this MapReduce job.



By looking at the output file of job execution, we can easily get the top 100 most frequently used words and their corresponding relative frequencies.



one	46839	0.5770%
out	31992	0.3941%
more	31949	0.3936%
up	30122	0.3711%
now	30028	0.3699%
shall	29966	0.3692%
man	28891	0.3559%
upon	24301	0.2994%
very	23811	0.2933%
time	23130	0.2850%
see	21118	0.2602%
well	21101	0.2600%
such	20945	0.2580%
come	20446	0.2519%
good	20222	0.2491%
thou	19631	0.2418%
know	19385	0.2388%
before	19154	0.2360%
lord	18845	0.2322%
little	18437	0.2271%
first	18138	0.2235%
great	17909	0.2206%
two	17806	0.2194%
made	17667	0.2177%
much	16870	0.2078%
go	16604	0.2046%
over	16475	0.2030%
down	16433	0.2025%
here	15768	0.1943%
thy	15657	0.1929%
men	14935	0.1840%
make	14926	0.1839%
never	14886	0.1834%
again	14865	0.1831%
old	14631	0.1803%
came	14547	0.1792%
being	14279	0.1759%
day	14089	0.1736%
unto	13976	0.1722%
himself	13971	0.1721%
those	13552	0.1670%
mr	13552	0.1670%
way	13521	0.1666%
life	13478	0.1660%
people	13442	0.1656%
god	13410	0.1652%
even	13303	0.1639%
without	12889	0.1588%
hand	12846	0.1583%
work	12669	0.1561%
long	12402	0.1528%
away	12402	0.1528%
many	12395	0.1527%
thee	12195	0.1502%
take	11915	0.1468%
king	11822	0.1456%
think	11725	0.1444%
same	11724	0.1444%
sir	11662	0.1437%
went	11615	0.1431%
house	11592	0.1428%
through	11470	0.1413%
give	11378	0.1402%
still	11198	0.1380%
nothing	11110	0.1369%
project	11091	0.1366%
eyes	11023	0.1358%
under	10894	0.1342%
back	10546	0.1299%
love	10507	0.1294%
though	10470	0.1290%
another	10413	0.1283%
o	10309	0.1270%
against	10266	0.1265%
thought	10205	0.1257%
things	10083	0.1242%
place	9955	0.1226%
last	9936	0.1224%
head	9878	0.1217%
tell	9502	0.1171%
father	9495	0.1170%
found	9416	0.1160%
face	8985	0.1107%
name	8944	0.1102%
right	8902	0.1097%
young	8877	0.1094%
new	8809	0.1085%
once	8750	0.1078%
put	8660	0.1067%
both	8555	0.1054%
three	8553	0.1054%
night	8430	0.1039%
saw	8422	0.1038%
part	8403	0.1035%
thing	8391	0.1034%
look	8366	0.1031%
ill	8353	0.1029%
took	8344	0.1028%
ye	8289	0.1021%
years	8235	0.1015%

And below is the histogram showing their relative frequency percentage.

g) Run with different number of mappers and reducers

i. Start History server

To monitor and analyze detailed information about MapReduce job, we should first start up History server in each HadoopSlave. We should update mapred-site.xml file like below (take HadoopSlave2 as example).

```
<property>
        <name>mapreduce.jobhistory.address</name>
        <value>HadoopSlave2:10020</value>
</property>
<property>
        <name>mapreduce.jobhistory.webapp.address</name>
        <value>HadoopSlave2:19888</value>
</property>
```

Then we can use below command to start the server.

```
/usr/local/hadoop/sbin/mr-jobhistory-daemon.sh start historyserver
```

At last I use `jps` command and website to check whether History server is running.

ii. Run MapReduce Job with different number of mapper and reducer.

By using parameters -jobconf mapred.map.tasks=x and -jobconf mapred.reduce.tasks=x, we can indicate the number of mappers and reducers used in a MapReduce job execution. Below is an example of indicating a MapReduce job using 40 mappers and 10 reducers.

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.7.1.jar -mapper /home/hduser/mapper.py -reducer /home/hduser/reducer.py -input /data/* -output /m40r10 -jobconf mapred.map.tasks=40 -jobconf mapred.reduce.tasks=10
```

And by checking execution logs from History server, I generate a statistic table like below.

Condition	Maximum mapper Time	Minimum Mapper Time	Average Mapper Time	Maximum Reducer Time	Minimum Reducer Time	Average Reducer Time	Total Job
Mapper:20 Reducer:2	31sec	3sec	13sec	37sec	28sec	33sec	8mins 45sec
Mapper:20 Reducer:5	44sec	3sec	11sec	22sec	13sec	17sec	9mins 0sec
Mapper:40 Reducer:5	32sec	3sec	11sec	18sec	16sec	17sec	9mins 16sec
Mapper:40 Reducer:2	26sec	5sec	12sec	19sec	14sec	17sec	8mins 8sec
Mapper:40 Reducer:10	28sec	3sec	11sec	15sec	10sec	12sec	9mins 17sec