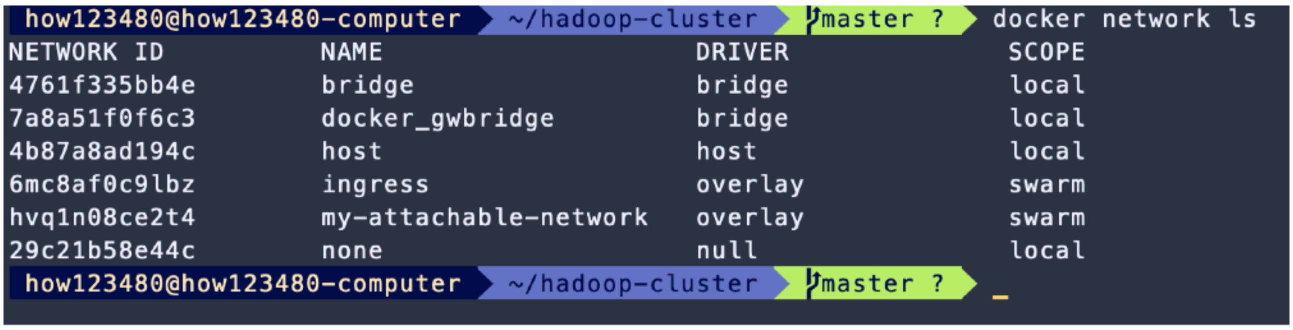
HW3 Hadoop Mapreduce

ID: r08921a07
NAME: 曾梓豪

1. Run the application in a Hadoop Docker container
the choose to use the docker container from the link:
https://github.com/sdwangntu/hadoop-cluster

First, I used command “swarm init” then created network with command “docker network create -d overlay --attachable my-attachable-network”



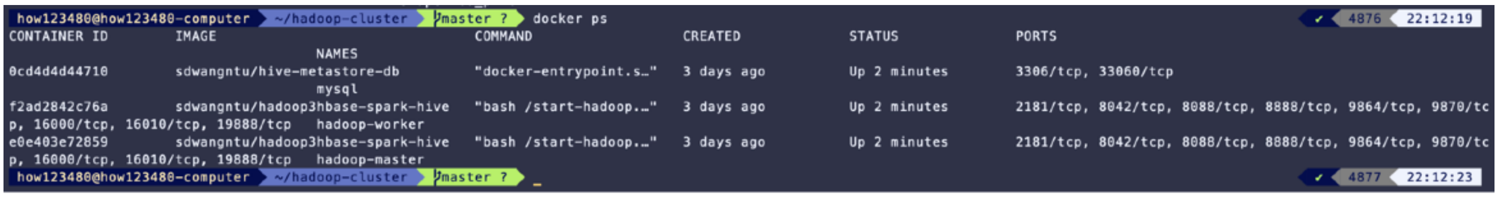
```
 how123480@how123480-computer  ~/hadoop-cluster  master ?  docker network ls
NETWORK ID          NAME                       DRIVER              SCOPE
4761f335bb4e        bridge                     bridge              local
7a8a51f0f6c3        docker_gwbridge            bridge              local
4b87a8ad194c        host                       host                local
6mc8af0c9lbz        ingress                    overlay             swarm
hvq1n08ce2t4        my-attachable-network      overlay             swarm
29c21b58e44c        none                       null                local
 how123480@how123480-computer  ~/hadoop-cluster  master ?  _
```

Second, I used commands below to launch the cluster:

1. docker run --hostname=mysql --name mysql --network my-attachable-network -d sdwangntu/hive-metastore-db

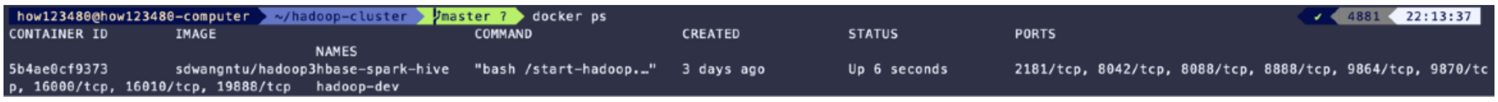
2. docker run --hostname=hadoop-master --name hadoop-master --network my-attachable-network -d sdwangntu/hadoop3hbase-spark-hive

3. docker run --hostname=hadoop-worker --name hadoop-worker --network my-attachable-network -d sdwangntu/hadoop3hbase-spark-hive



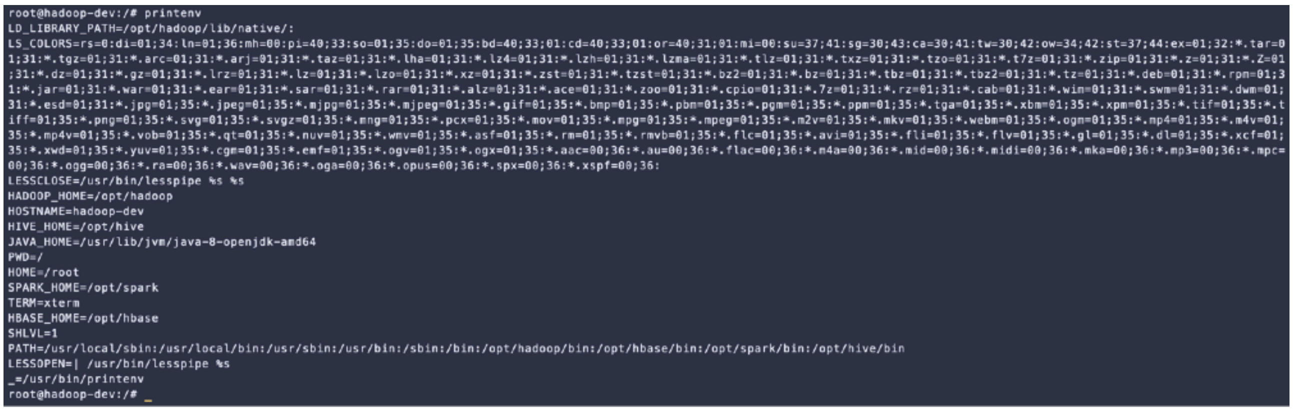
```
 how123480@how123480-computer  ~/hadoop-cluster  master ?  docker ps                                                                    ✓  4876  22:12:19
CONTAINER ID        IMAGE                                 COMMAND                  CREATED             STATUS              PORTS
                                  NAMES
0cd4d4d44710        sdwangntu/hive-metastore-db           "docker-entrypoint.s…"   3 days ago          Up 2 minutes        3306/tcp, 33060/tcp
                                  mysql
f2ad2842c76a        sdwangntu/hadoop3hbase-spark-hive     "bash /start-hadoop.…"   3 days ago          Up 2 minutes        2181/tcp, 8042/tcp, 8088/tcp, 8888/tcp, 9864/tcp, 9870/tc
p, 16000/tcp, 16010/tcp, 19888/tcp   hadoop-worker
e0e403e72859        sdwangntu/hadoop3hbase-spark-hive     "bash /start-hadoop.…"   3 days ago          Up 2 minutes        2181/tcp, 8042/tcp, 8088/tcp, 8888/tcp, 9864/tcp, 9870/tc
p, 16000/tcp, 16010/tcp, 19888/tcp   hadoop-master
 how123480@how123480-computer  ~/hadoop-cluster  master ?  _                                                                            ✓  4877  22:12:23
```

Third, I deployed the development container with command ”docker run --hostname=hadoop-dev --name hadoop-dev -v $(pwd):/home --network my-attachable-network -d sdwangntu/hadoop3hbase-spark-hive”



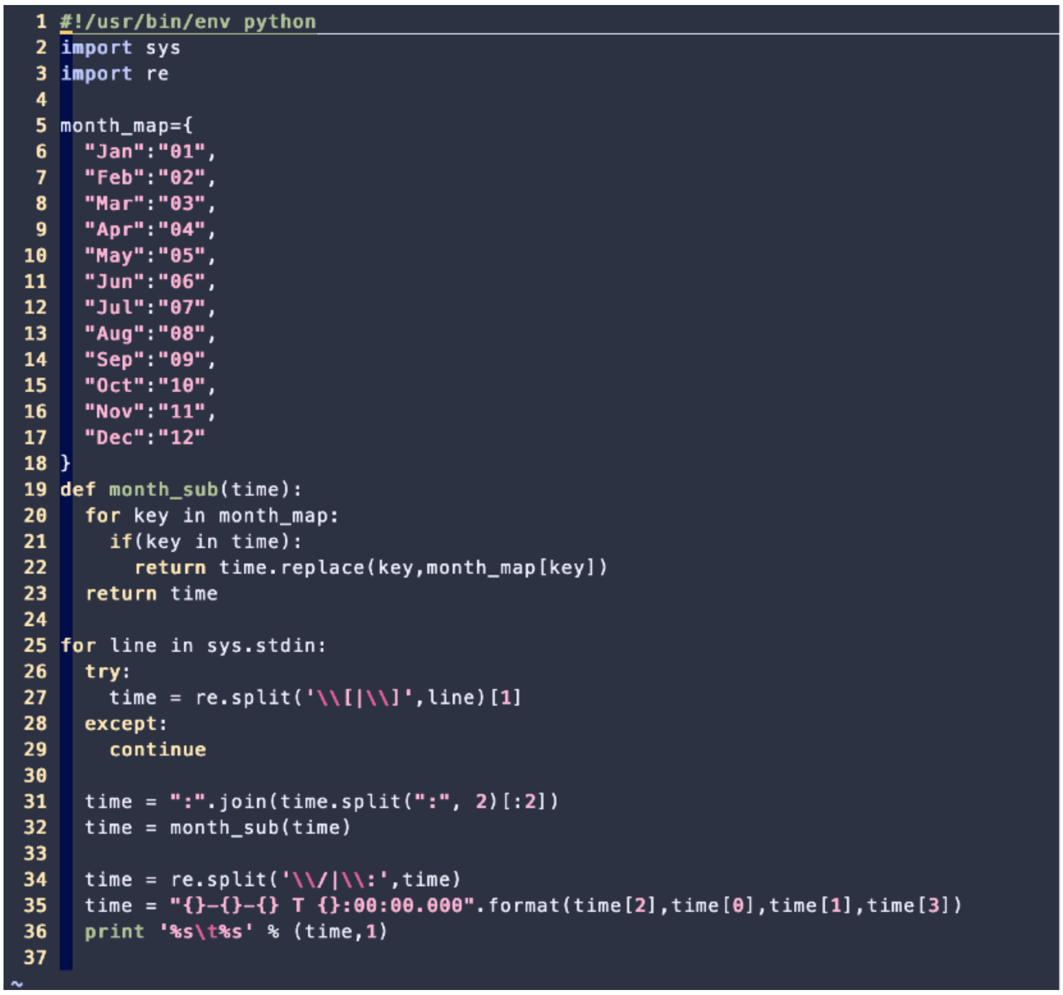
```
 how123480@how123480-computer  ~/hadoop-cluster  master ?  docker ps                                                                    ✓  4881  22:13:37
CONTAINER ID        IMAGE                                 COMMAND                  CREATED             STATUS              PORTS
                                  NAMES
5b4ae0cf9373        sdwangntu/hadoop3hbase-spark-hive     "bash /start-hadoop.…"   3 days ago          Up 6 seconds        2181/tcp, 8042/tcp, 8088/tcp, 8888/tcp, 9864/tcp, 9870/tc
p, 16000/tcp, 16010/tcp, 19888/tcp   hadoop-dev
```

Fourth, use command “docker exec -it {container ID} bash” to get the shell of the container. then used command “printenv” to print the environment variable. We can know hadoop in the directory “/opt/hadoop”



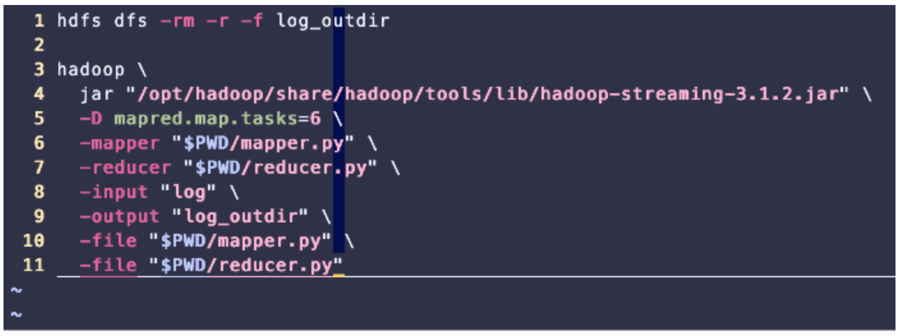
```
root@hadoop-dev:/# printenv
LD_LIBRARY_PATH=/opt/hadoop/lib/native/:
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=0
1;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01
;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;3
1:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;
31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.t
iff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;
35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;
35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=
00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
LESSCLOSE=/usr/bin/lesspipe %s %s
HADOOP_HOME=/opt/hadoop
HOSTNAME=hadoop-dev
HIVE_HOME=/opt/hive
JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
PWD=/
HOME=/root
SPARK_HOME=/opt/spark
TERM=xterm
HBASE_HOME=/opt/hbase
SHLVL=1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/opt/hadoop/bin:/opt/hbase/bin:/opt/spark/bin:/opt/hive/bin
LESSOPEN=| /usr/bin/lesspipe %s
_=/usr/bin/printenv
root@hadoop-dev:/# _
```

edit the “mapper” code of lecture example, “wordcount” to parse the log file.



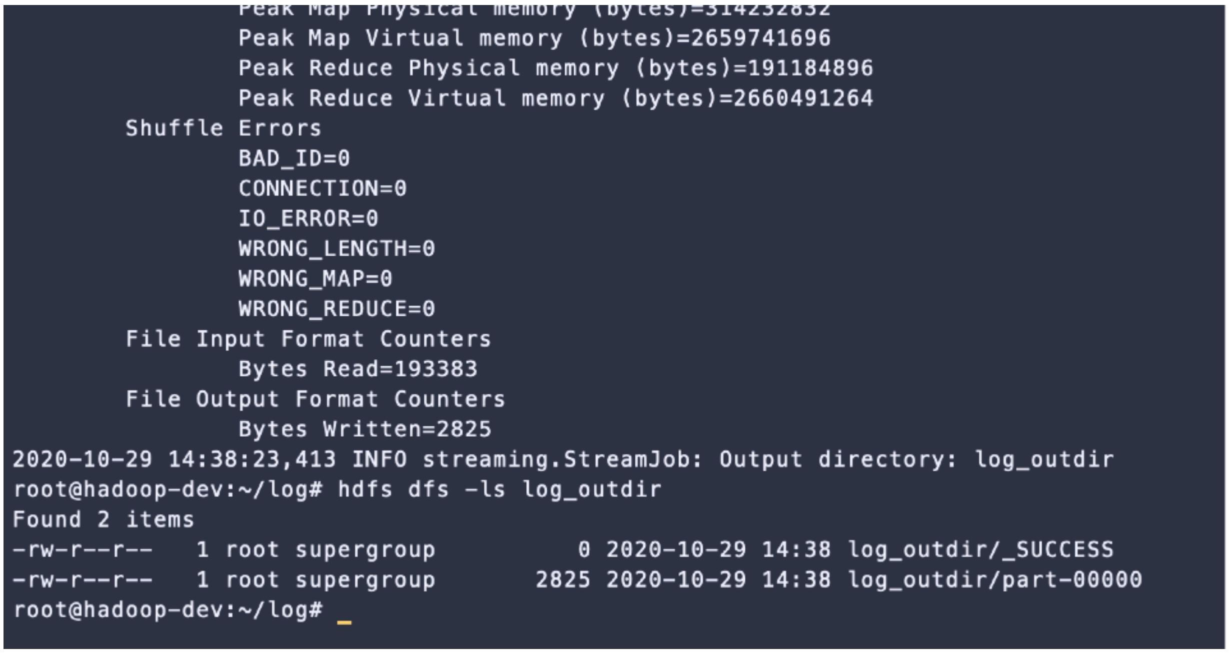
```python
#!/usr/bin/env python
import sys
import re

month_map={
  "Jan":"01",
  "Feb":"02",
  "Mar":"03",
  "Apr":"04",
  "May":"05",
  "Jun":"06",
  "Jul":"07",
  "Aug":"08",
  "Sep":"09",
  "Oct":"10",
  "Nov":"11",
  "Dec":"12"
}
def month_sub(time):
  for key in month_map:
    if(key in time):
      return time.replace(key,month_map[key])
  return time

for line in sys.stdin:
  try:
    time = re.split('\\[|\\]',line)[1]
  except:
    continue

  time = ":".join(time.split(":", 2)[:2])
  time = month_sub(time)

  time = re.split('\\/|\\:',time)
  time = "{}-{}-{} T {}:00:00.000".format(time[2],time[0],time[1],time[3])
  print '%s\t%s' % (time,1)

~
```

then use hadoop-streaming to start the program.

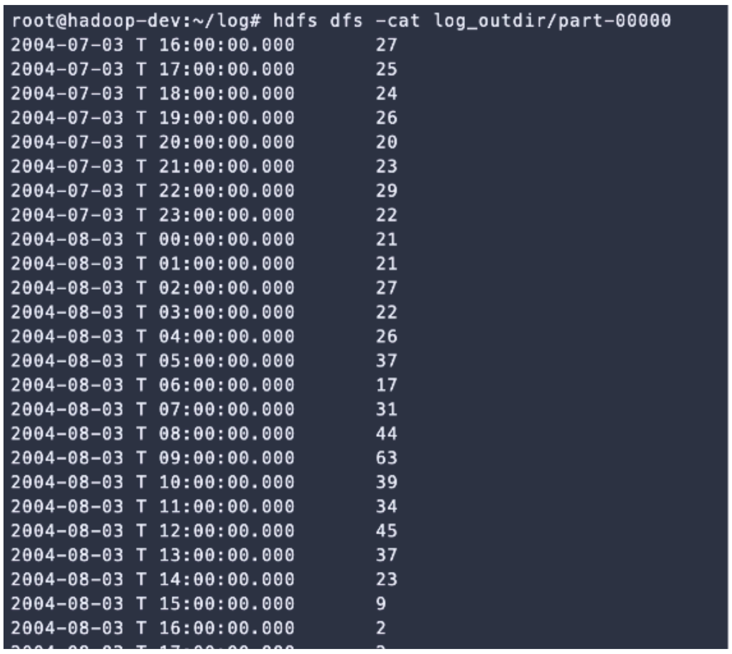


```bash
hdfs dfs -rm -r -f log_outdir

hadoop \
  jar "/opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.1.2.jar" \
  -D mapred.map.tasks=6 \
  -mapper "$PWD/mapper.py" \
  -reducer "$PWD/reducer.py" \
  -input "log" \
  -output "log_outdir" \
  -file "$PWD/mapper.py" \
  -file "$PWD/reducer.py"
~
~
```

Finally, I get the result of log analysis.



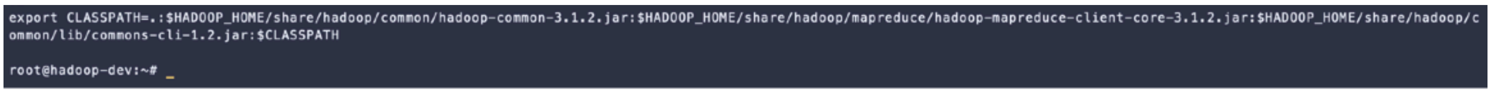
```
                Peak Map Physical memory (bytes)=314232832
                Peak Map Virtual memory (bytes)=2659741696
                Peak Reduce Physical memory (bytes)=191184896
                Peak Reduce Virtual memory (bytes)=2660491264
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=193383
        File Output Format Counters
                Bytes Written=2825
2020-10-29 14:38:23,413 INFO streaming.StreamJob: Output directory: log_outdir
root@hadoop-dev:~/log# hdfs dfs -ls log_outdir
Found 2 items
-rw-r--r--   1 root supergroup          0 2020-10-29 14:38 log_outdir/_SUCCESS
-rw-r--r--   1 root supergroup       2825 2020-10-29 14:38 log_outdir/part-00000
root@hadoop-dev:~/log# _
```



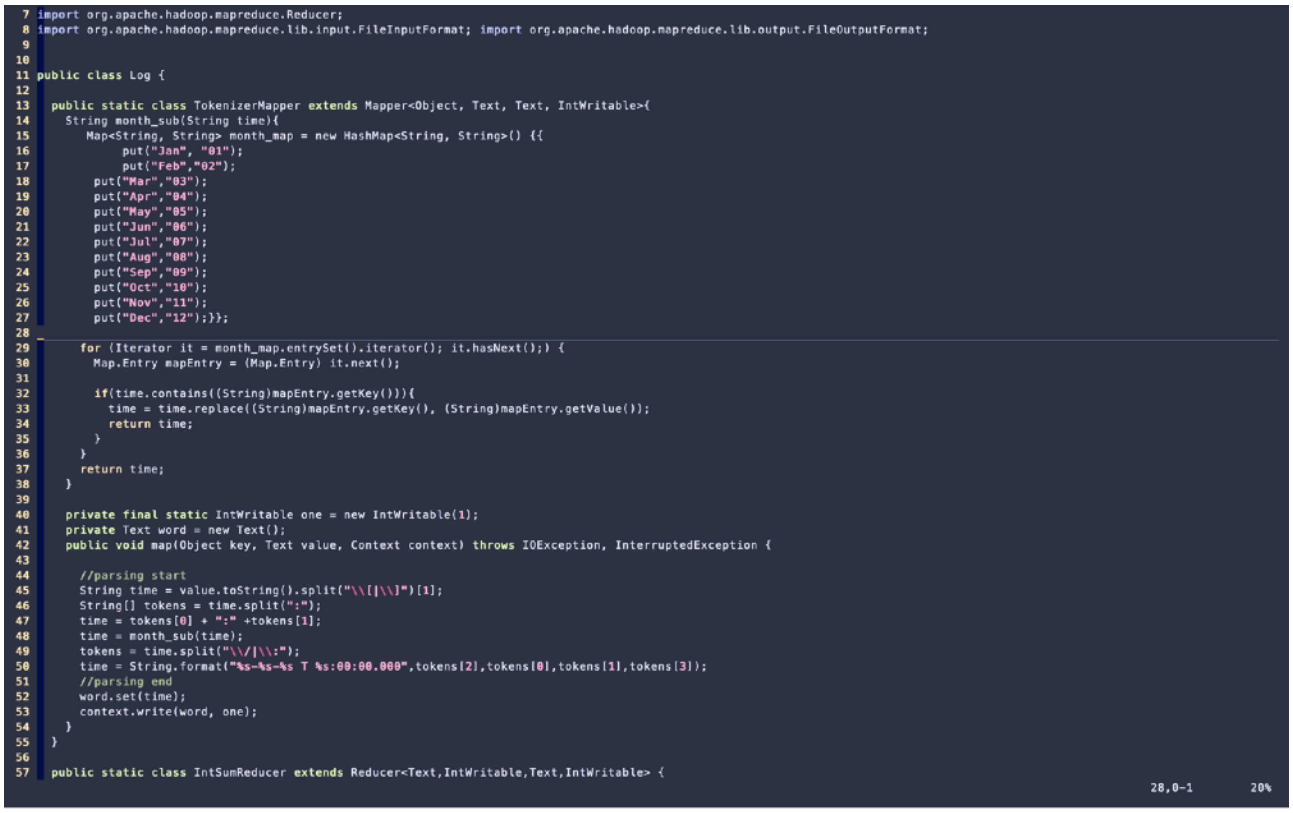
```
root@hadoop-dev:~/log# hdfs dfs -cat log_outdir/part-00000
2004-07-03 T 16:00:00.000       27
2004-07-03 T 17:00:00.000       25
2004-07-03 T 18:00:00.000       24
2004-07-03 T 19:00:00.000       26
2004-07-03 T 20:00:00.000       20
2004-07-03 T 21:00:00.000       23
2004-07-03 T 22:00:00.000       29
2004-07-03 T 23:00:00.000       22
2004-08-03 T 00:00:00.000       21
2004-08-03 T 01:00:00.000       21
2004-08-03 T 02:00:00.000       27
2004-08-03 T 03:00:00.000       22
2004-08-03 T 04:00:00.000       26
2004-08-03 T 05:00:00.000       37
2004-08-03 T 06:00:00.000       17
2004-08-03 T 07:00:00.000       31
2004-08-03 T 08:00:00.000       44
2004-08-03 T 09:00:00.000       63
2004-08-03 T 10:00:00.000       39
2004-08-03 T 11:00:00.000       34
2004-08-03 T 12:00:00.000       45
2004-08-03 T 13:00:00.000       37
2004-08-03 T 14:00:00.000       23
2004-08-03 T 15:00:00.000       9
2004-08-03 T 16:00:00.000       2
```

2. Write the java version of part(a)

I also edited the “mapper” code in lecture example “wordcount”, and following is the code of “mapper” section code.
In order to run the java, I set the environment variable in “/etc/profile” then “source“ it.

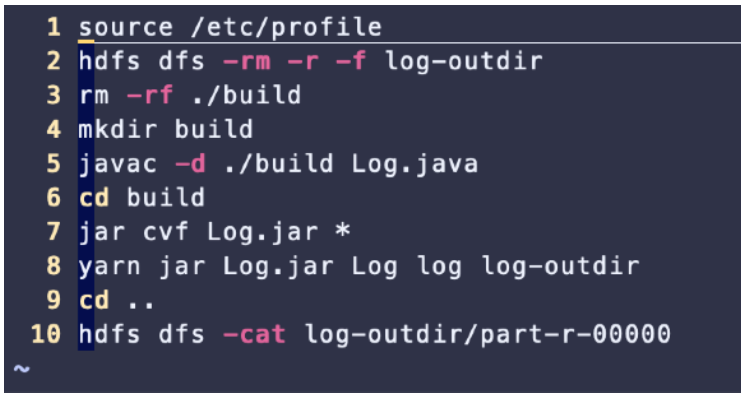


```
export CLASSPATH=.:$HADOOP_HOME/share/hadoop/common/hadoop-common-3.1.2.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.1.2.jar:$HADOOP_HOME/share/hadoop/c
ommon/lib/commons-cli-1.2.jar:$CLASSPATH

root@hadoop-dev:~# _
```



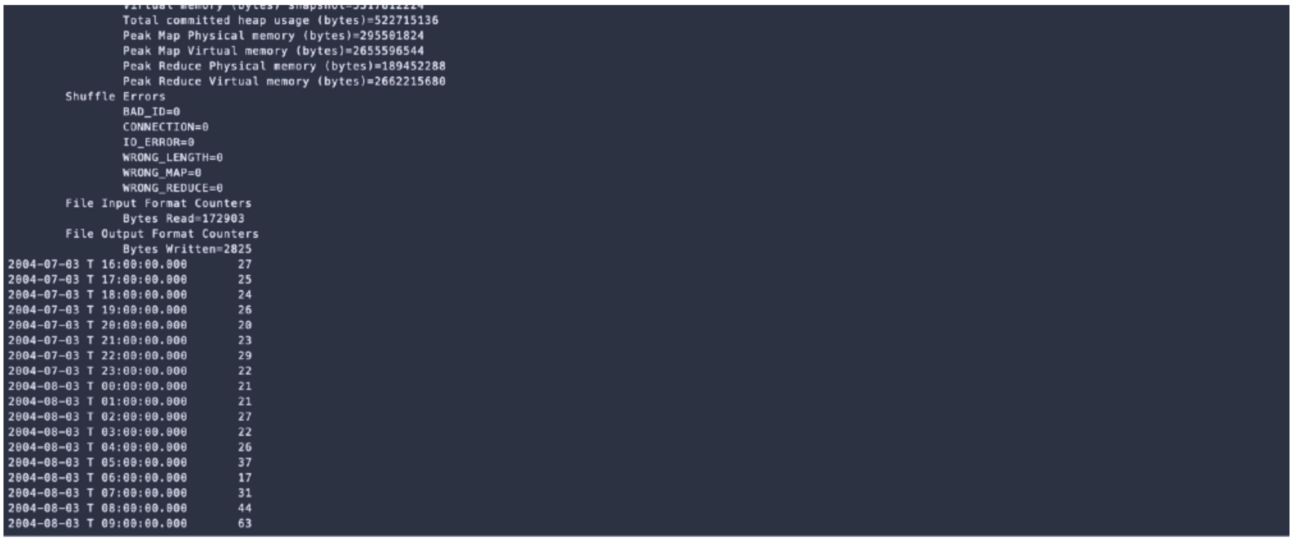
```java
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


public class Log {

  public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>{
    String month_sub(String time){
       Map<String, String> month_map = new HashMap<String, String>() {{
            put("Jan", "01");
            put("Feb","02");
        put("Mar","03");
        put("Apr","04");
        put("May","05");
        put("Jun","06");
        put("Jul","07");
        put("Aug","08");
        put("Sep","09");
        put("Oct","10");
        put("Nov","11");
        put("Dec","12");}};

      for (Iterator it = month_map.entrySet().iterator(); it.hasNext();) {
        Map.Entry mapEntry = (Map.Entry) it.next();

        if(time.contains((String)mapEntry.getKey())){
          time = time.replace((String)mapEntry.getKey(), (String)mapEntry.getValue());
          return time;
        }
      }
      return time;
    }

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

      //parsing start
      String time = value.toString().split("\\[|\\]")[1];
      String[] tokens = time.split(":");
      time = tokens[0] + ":" +tokens[1];
      time = month_sub(time);
      tokens = time.split("\\/|\\:");
      time = String.format("%s-%s-%s T %s:00:00.000",tokens[2],tokens[0],tokens[1],tokens[3]);
      //parsing end
      word.set(time);
      context.write(word, one);
    }
  }

  public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable> {
                                                                                      28,0-1        20%
```

Then I use the following script to start the hadoop program and display the result.



```bash
source /etc/profile
hdfs dfs -rm -r -f log-outdir
rm -rf ./build
mkdir build
javac -d ./build Log.java
cd build
jar cvf Log.jar *
yarn jar Log.jar Log log log-outdir
cd ..
hdfs dfs -cat log-outdir/part-r-00000
```

Following is the result of java version



```
                Total committed heap usage (bytes)=522715136
                Peak Map Physical memory (bytes)=295501824
                Peak Map Virtual memory (bytes)=2655596544
                Peak Reduce Physical memory (bytes)=189452288
                Peak Reduce Virtual memory (bytes)=2662215680
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=172903
        File Output Format Counters
                Bytes Written=2825
2004-07-03 T 16:00:00.000       27
2004-07-03 T 17:00:00.000       25
2004-07-03 T 18:00:00.000       24
2004-07-03 T 19:00:00.000       26
2004-07-03 T 20:00:00.000       20
2004-07-03 T 21:00:00.000       23
2004-07-03 T 22:00:00.000       29
2004-07-03 T 23:00:00.000       22
2004-08-03 T 00:00:00.000       21
2004-08-03 T 01:00:00.000       21
2004-08-03 T 02:00:00.000       27
2004-08-03 T 03:00:00.000       22
2004-08-03 T 04:00:00.000       26
2004-08-03 T 05:00:00.000       37
2004-08-03 T 06:00:00.000       17
2004-08-03 T 07:00:00.000       31
2004-08-03 T 08:00:00.000       44
2004-08-03 T 09:00:00.000       63
```

3. Source code and running script github link:
https://github.com/how123480/simple-hadoop