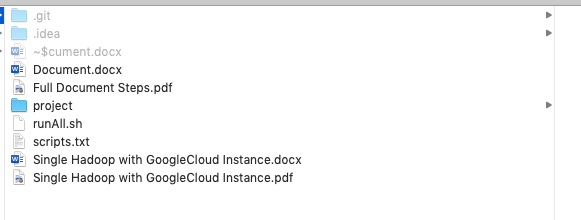
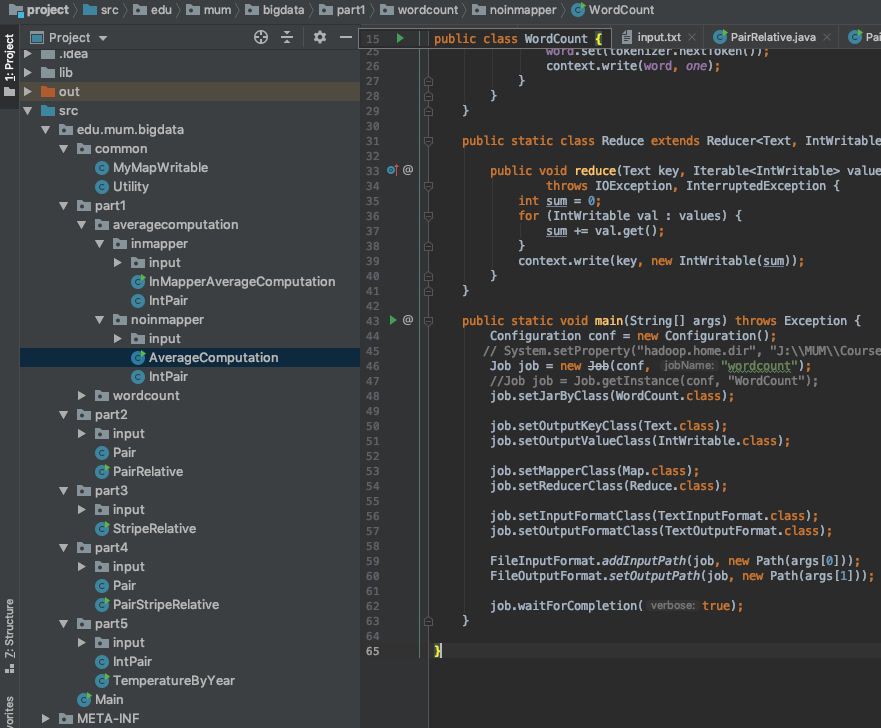
# **PROJECT – PART 1-5**

**Source code:** [**https://github.com/hantruongth/MapReduceBD**](https://github.com/hantruongth/MapReduceBD)



*Project on IDE Intellij*



# **Part 1: Set up a single node cluster and optionally an eclipse development environment to create and test your programs**

1. Setup environment on Virtual box, Docker and Google Cloud

Ref Single Hadoop with GoogleCloud Instance.docx

1. Get Cloudera

Ref Single Hadoop with GoogleCloud Instance.docx

1. WordCount – run on google cloud

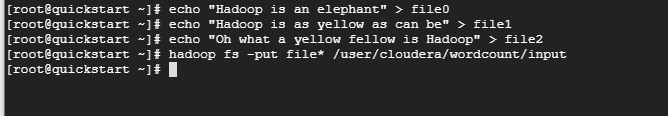
*Prepare data*

echo "Hadoop is an elephant" > file0

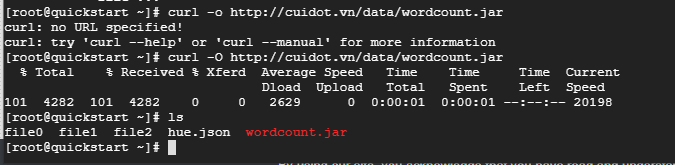
echo "Hadoop is as yellow as can be" > file1

echo "Oh what a yellow fellow is Hadoop" > file2

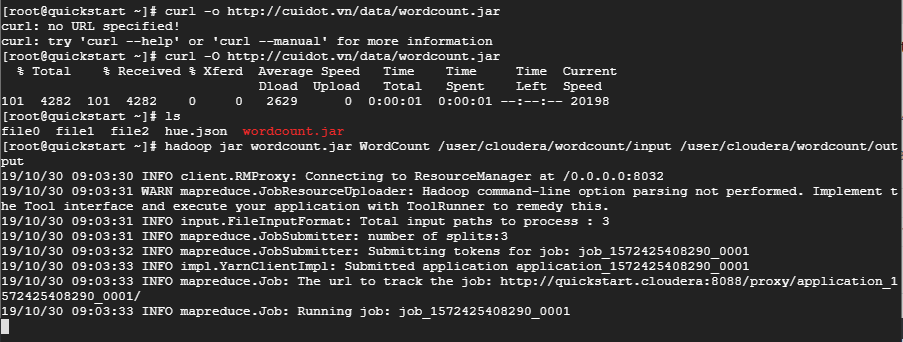
hadoop fs -put file\* /user/cloudera/wordcount/input



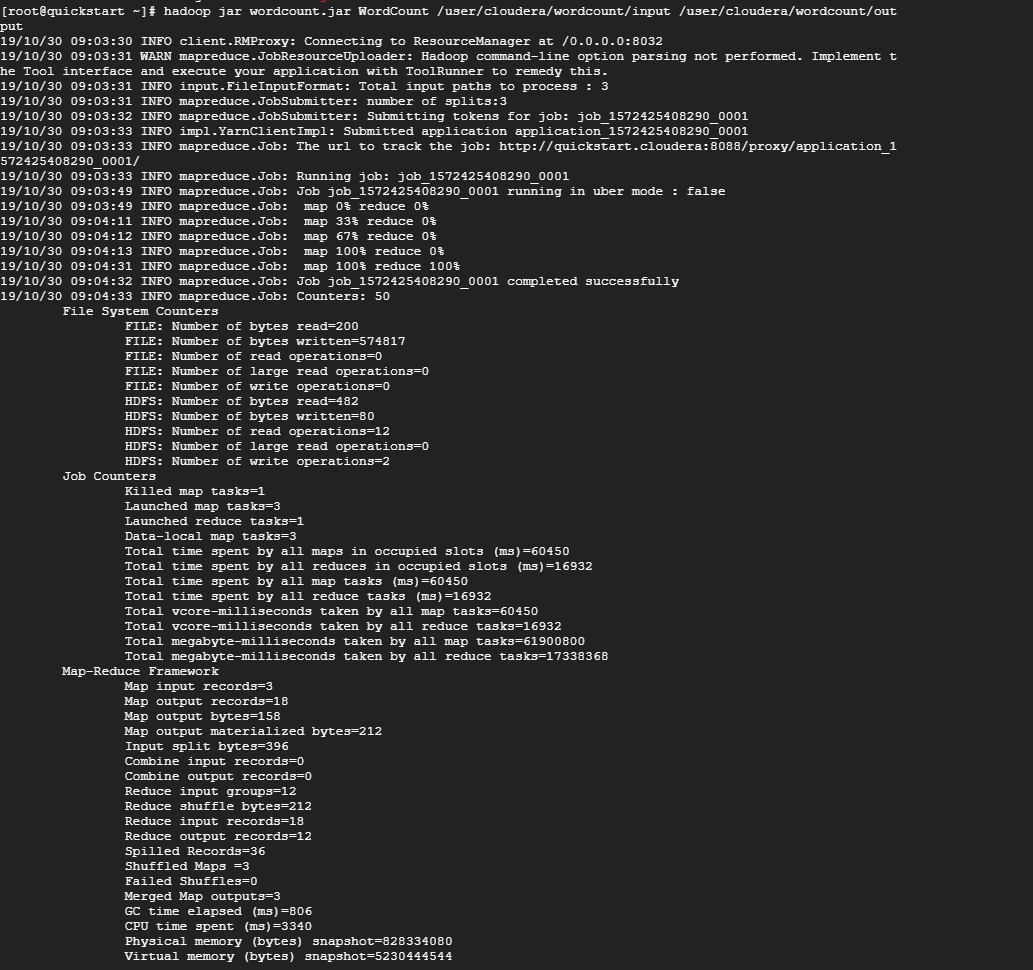
*Load jar file into cloudera*



*Run wordcount*



*Wordcount result:*



1. **InMapperWordCount – run on google cloud**

*Prepare data (skip if use current data that wordcount used)*

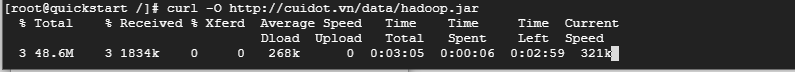
echo "Hadoop is an elephant" > file0

echo "Hadoop is as yellow as can be" > file1

echo "Oh what a yellow fellow is Hadoop" > file2

hadoop fs -put file\* /user/cloudera/wordcount/input

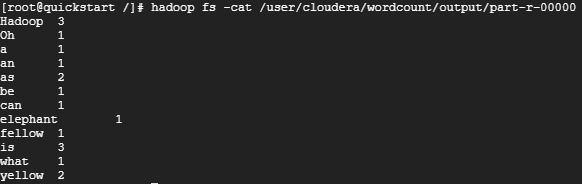
*Load jar file into cloudera*



*Run inMapperWordCount*

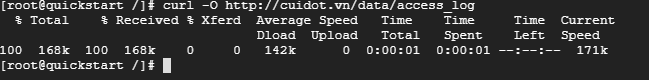


*InMapperWordCount Result:*

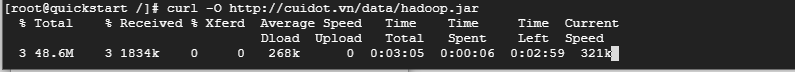


1. **Average Computation Algorithm – run on google cloud**

*Load data into cloudera*



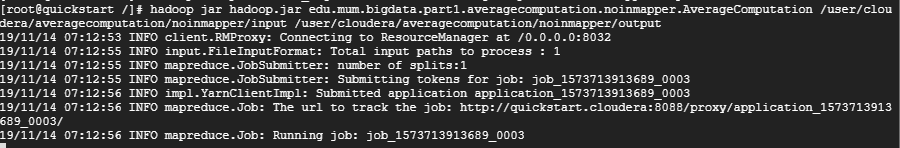
*Load jar file into cloudera*



*Organize folder and copy file into hdfs*



*Run AverageComputation algorithm*



*AverageComputation results:*

lj1212.inktomisearch.com 3169.0

lj1216.inktomisearch.com 209.0

lj1220.inktomisearch.com 209.0

lj1223.inktomisearch.com 1941.5

lj1231.inktomisearch.com 209.0

lordgun.org 2869.0

mail.geovariances.fr 6012.217391304348

market-mail.panduit.com 3427.344827586207

mcl02.cnc.bc.ca 10879.5

mmscrm07-2.sac.overture.com 68.0

mth-fgw.ballarat.edu.au 5448.714285714285

nb-bolz.cremona.polimi.it 2300.0

ns.mou.cz 2300.0

ns.wtbts.org 2311.3333333333335

ns3.vonroll.ch 5971.666666666667

ogw.netinfo.bg 2758.0

osdlab.eic.nctu.edu.tw 269.0

p213.54.168.132.tisdip.tiscali.de 5785.75

p5083cd5d.dip0.t-ipconnect.de 7368.0

pc-030-040.eco.rug.nl 7368.0

pc3-registry-stockholm.telia.net 9452.692307692309

pd95f99f2.dip.t-dialin.net 2869.0

pd9e50809.dip.t-dialin.net 2869.0

pd9e761cf.dip.t-dialin.net 2300.0

pd9eb1396.dip.t-dialin.net 2300.0

pntn02m05-129.bctel.ca 3095.0

pool-68-160-195-60.ny325.east.verizon.net 3724.2

ppp2.p33.is.com.ua 3582.0

proxy0.haifa.ac.il 3271.157894736842

prxint-sxb2.e-i.net 4022.0

prxint-sxb3.e-i.net 9254.07142857143

px7wh.vc.shawcable.net 7649.0

rouble.cc.strath.ac.uk 2869.0

spica.ukc.ac.uk 1973.5

spot.nnacorp.com 4632.4

trrc02m01-40.bctel.ca 3071.75

ts04-ip92.hevanet.com 4431.5

ts05-ip44.hevanet.com 7854.5625

user-0c8hdkf.cable.mindspring.com 5372.2

vlp181.vlp.fi 2869.0

watchguard.cgmatane.qc.ca 5741.0

wc03.mtnk.rnc.net.cable.rogers.com 10936.0

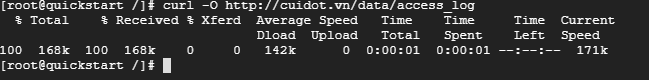
wc09.mtnk.rnc.net.cable.rogers.com 10860.666666666666

wwwcache.lanl.gov 2869.0

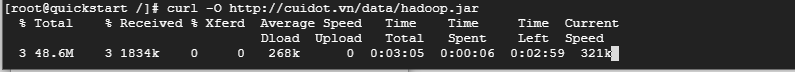
yongsan-cache.korea.army.mil 3056.0

1. **InMapper Average Computation Algorithm – run on google cloud**

*Load data into cloudera*



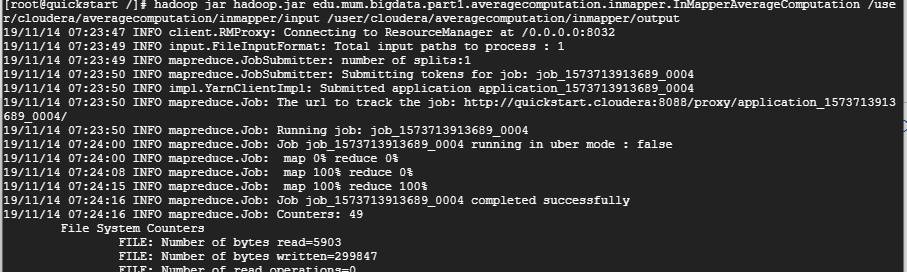
*Load jar file into cloudera*



*Organize folder and copy file into hdfs*



*Run AverageComputation algorithm*



*AverageComputation results:*

**[root@quickstart /]# hadoop fs -cat /user/cloudera/averagecomputation/inmapper/output/part-r-00000**

0x503e4fce.virnxx2.adsl-dhcp.tele.dk 1315.6666666666667

1-320.cnc.bc.ca 10879.5

1-729.cnc.bc.ca 3262.5714285714284

10.0.0.153 4444.981481481482

12.22.207.235 7368.0

128.227.88.79 5841.785714285715

142.27.64.35 1923.5714285714287

145.253.208.9 3728.285714285714

1513.cps.virtua.com.br 309.0

194.151.73.43 10879.5

195.11.231.210 6032.0

195.230.181.122 2300.0

195.246.13.119 5128.583333333333

2-110.cnc.bc.ca 7912.363636363636

2-238.cnc.bc.ca 3169.0

200-55-104-193.dsl.prima.net.ar 2179.4615384615386

200.160.249.68.bmf.com.br 6634.5

200.222.33.33 2300.0

203.147.138.233 2164.3076923076924

206-15-133-153.dialup.ziplink.net 0.0

206-15-133-154.dialup.ziplink.net 0.0

206-15-133-181.dialup.ziplink.net 0.0

207.195.59.160 4053.05

208-186-146-13.nrp3.brv.mn.frontiernet.net 1689.0

208-38-57-205.ip.cal.radiant.net 3830.3636363636365

208.247.148.12 3067.0

212.21.228.26 2869.0

212.92.37.62 5212.928571428572

213.181.81.4 7649.0

216-160-111-121.tukw.qwest.net 2317.5

216.139.185.45 6051.0

219.95.17.51 3169.0

3\_343\_lt\_someone 6277.2

4.37.97.186 2446.0

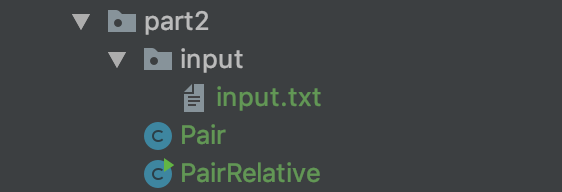
61.165.64.6 3056.0

61.9.4.61 2645.3333333333335

64-249-27-114.client.dsl.net 7368.0

# **Part 2: Implement Pairs algorithm to compute relative frequencies.**

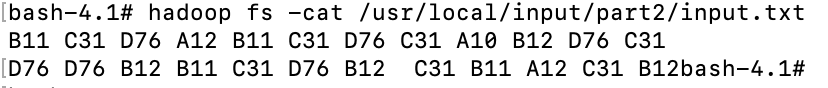
There are the .java files for the program.



*Run in Hadoop Docker*

# **Show input data content**

hadoop fs -cat /usr/local/input/part2/input.txt

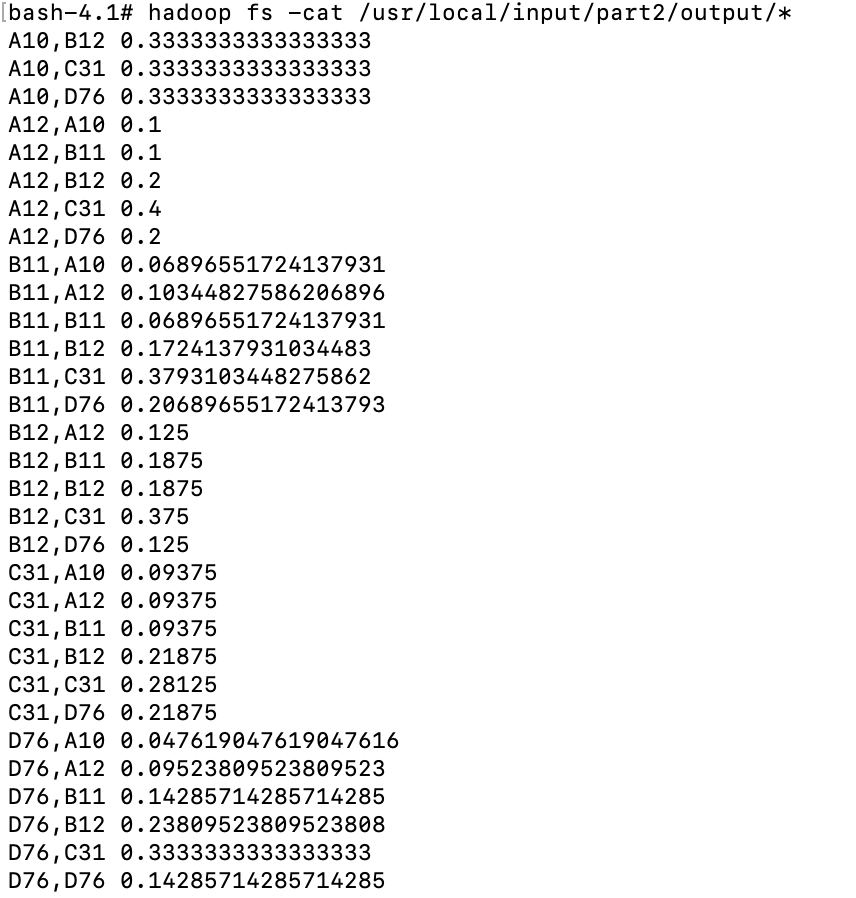


# **Execute the MapReduce program**

# hadoop jar project.jar edu.mum.bigdata.part2.PairRelative /usr/local/input/part2/input.txt /usr/local/input/part2/output

# **Program output**

**hadoop fs -cat /usr/local/input/part2/output/\***



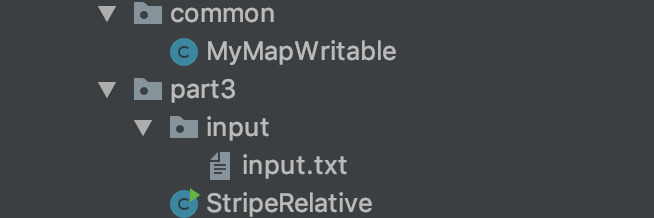
# **Run in Cloudera environment**

**# ./runAll.sh part2/ edu.mum.bigdata.part2.PairRelative**

# **Part 3: Implement Stripes algorithm to compute relative frequencies.**

# ***Run in Hadoop Docker***

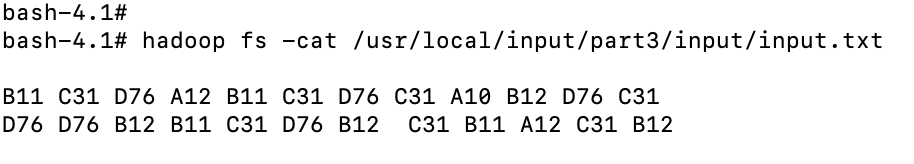
There are the .java files of program.



Build the whole project into .jar file and copy it into Hadoop docket and execute these commands:

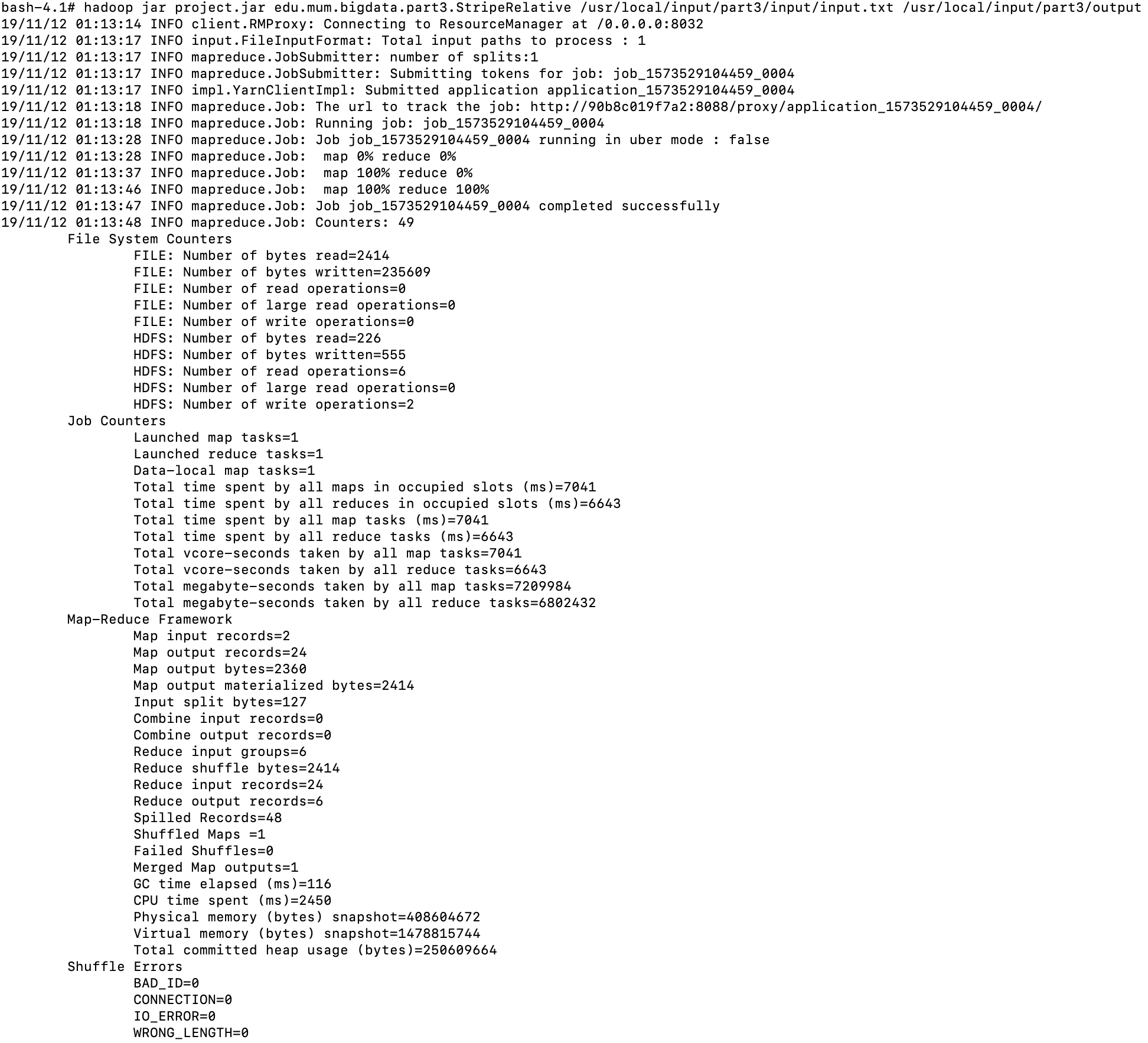
# **Show the input content**

hadoop fs -cat /usr/local/input/part3/input/input.txt



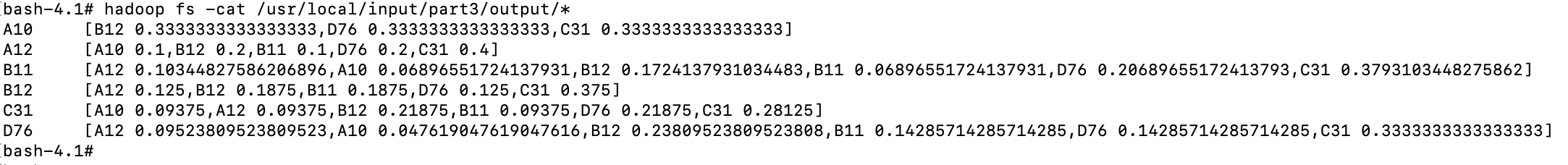
# **Execute the MapReduce program**

hadoop jar project.jar edu.mum.bigdata.part3.StripeRelative /usr/local/input/part3/input/input.txt /usr/local/input/part3/output



hadoop fs -cat /usr/local/input/part3/output/\*

# **Program output**



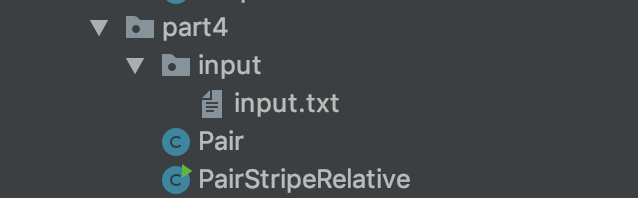
# **Run in Cloudera environment**

**# ./runAll.sh part3/ edu.mum.bigdata.part3.StripeRelative**

# **Part 4: Implement Pairs in Mapper and Stripes in Reducer to compute relative frequencies (Hybrid)**

# ***Run in Hadoop Docker***

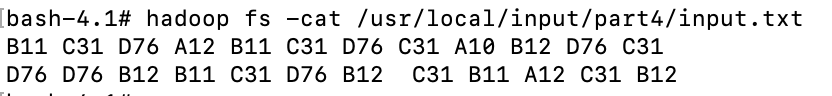
These are the .java files for the program.



Build the whole project into .jar file and copy it to Hadoop docker and execute these commands:

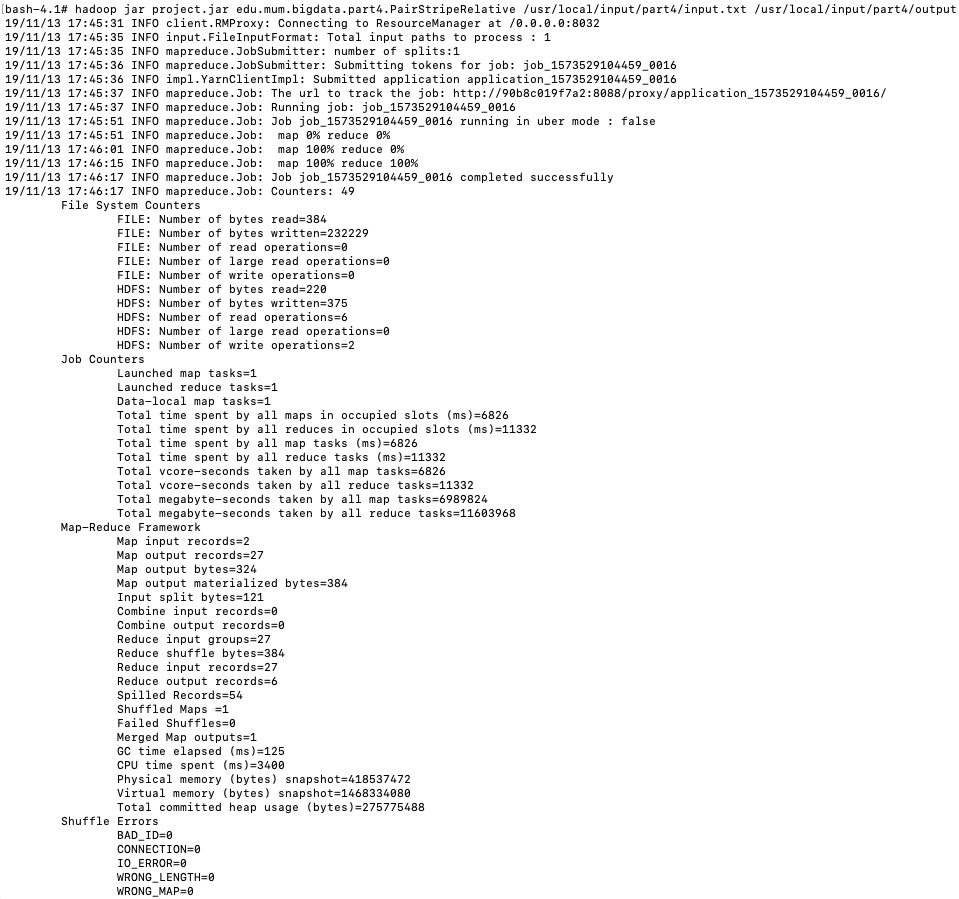
# **Show the content of input files**

hadoop fs -cat /usr/local/input/part4/input.txt



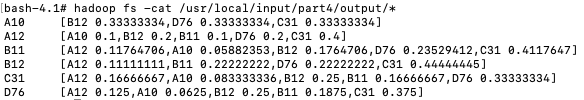
# **Execute the MapRecure program**

hadoop jar project.jar edu.mum.bigdata.part4.PairStripeRelative /usr/local/input/part4/input.txt /usr/local/input/part4/output



# **Show the program output**

hadoop fs -cat /usr/local/input/part4/output/\*



# **Run in Cloudera environment**

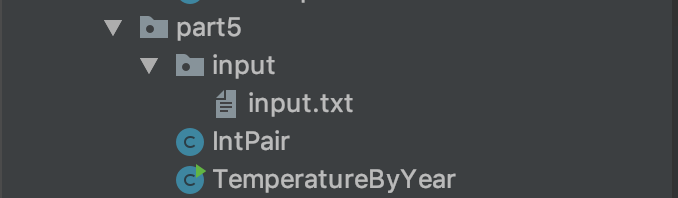
**# ./runAll.sh part4/ edu.mum.bigdata.part4.PairStripeRelative**

# **Part 5: Solve a MapReduce problem of your choice!**

# **Run in Hadoop Docker**

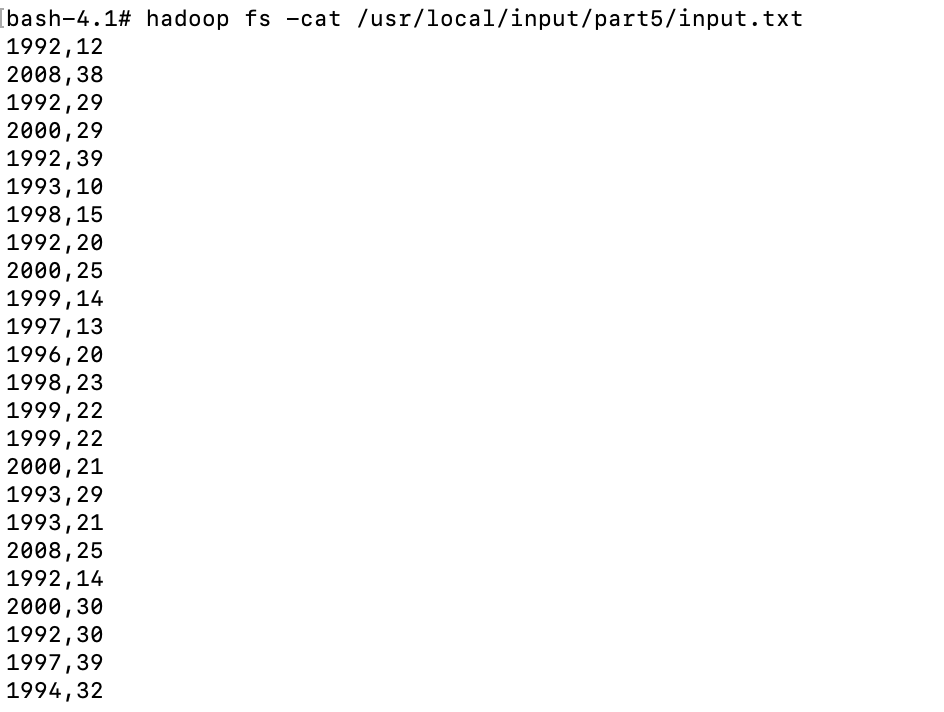
The problem is to compute the average temperature by year. Build the whole project into .jar file and copy it into Hadoop docker and execute with the follow commands.

The following is the .java files for this part and the testing input data.



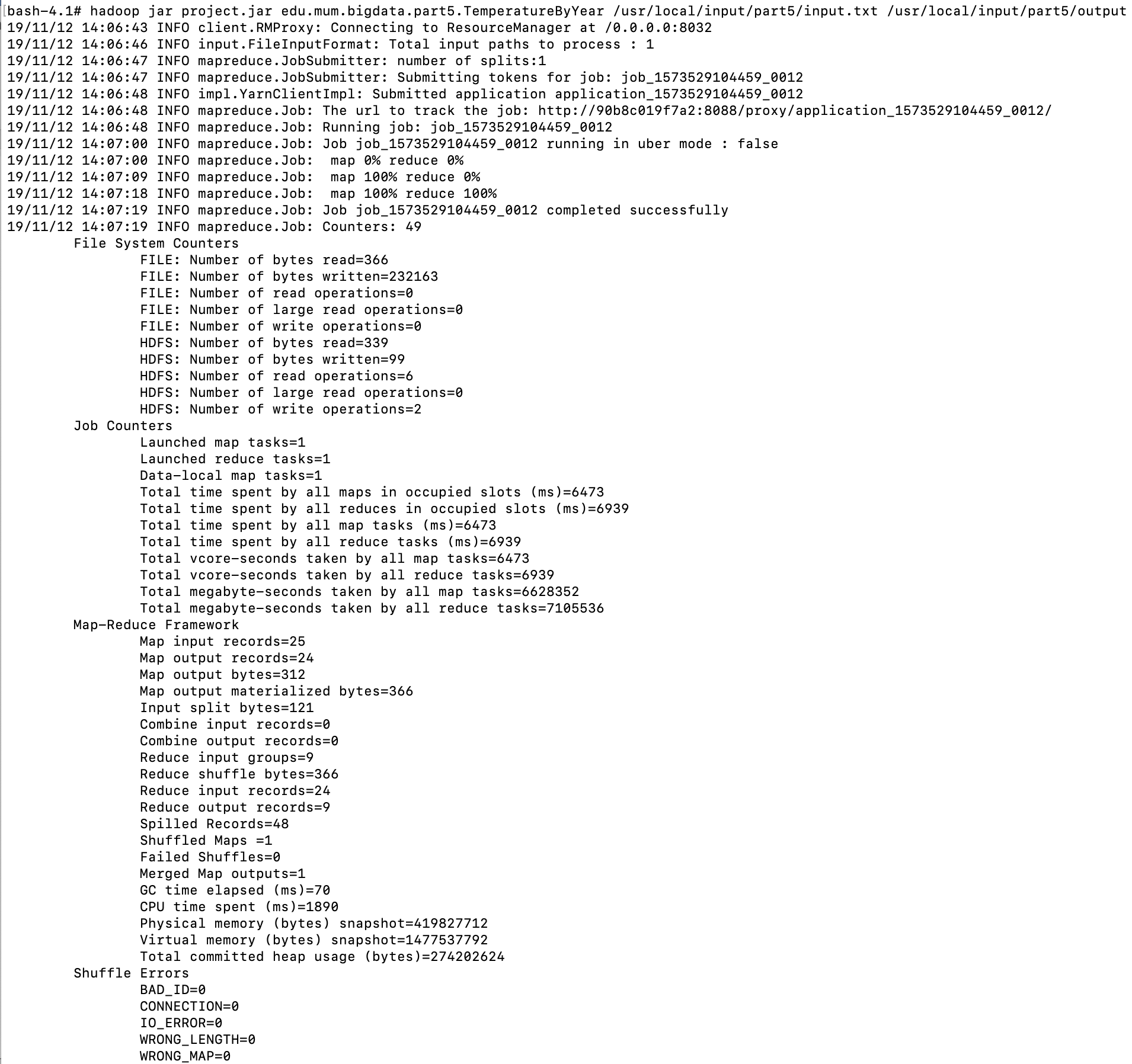
# **Show the content of input data.**

hadoop fs -cat /usr/local/input/part5/input.txt



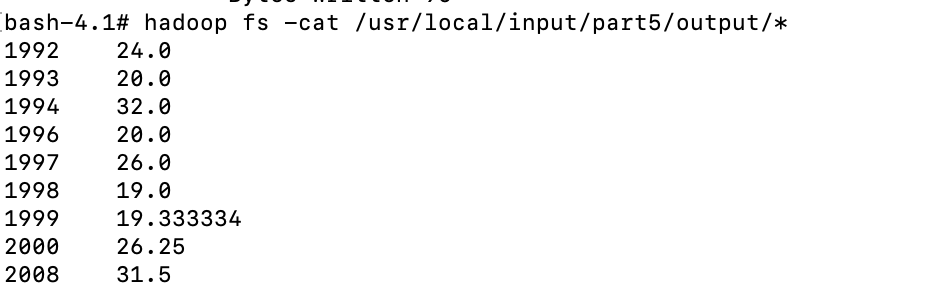
# **Execute the MapReduce program**

hadoop jar project.jar edu.mum.bigdata.part5.TemperatureByYear /usr/local/input/part5/input.txt /usr/local/input/part5/output



# **Show the output data.**

hadoop fs -cat /usr/local/input/part5/output/\*



# **Run in Cloudera environment**

**# ./runAll.sh part5/ edu.mum.bigdata.part5.TemperatureByYear**