Hands-on Assignment 2

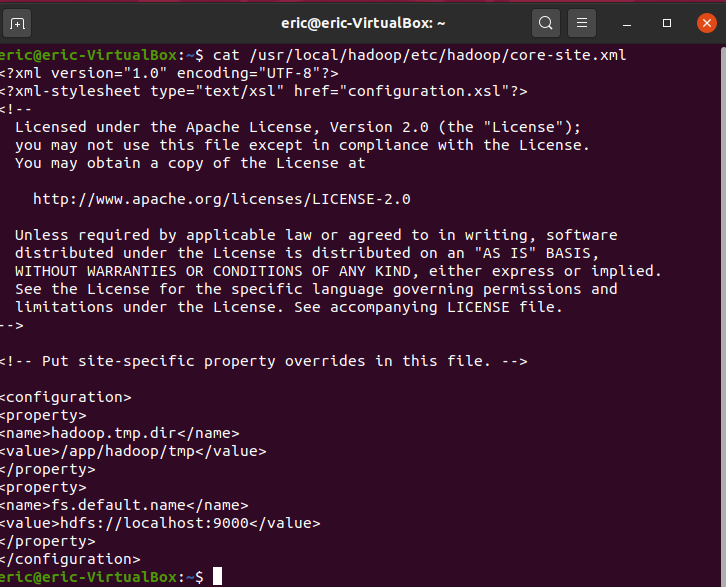
**Task 1**

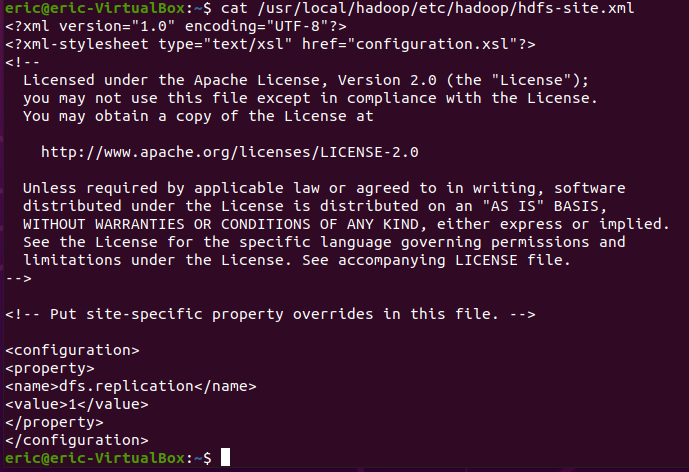
7.show the directory with its new owner  
  

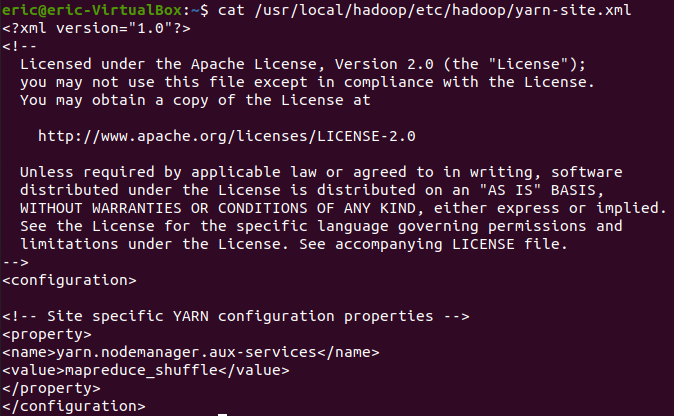

8.the directory for java  

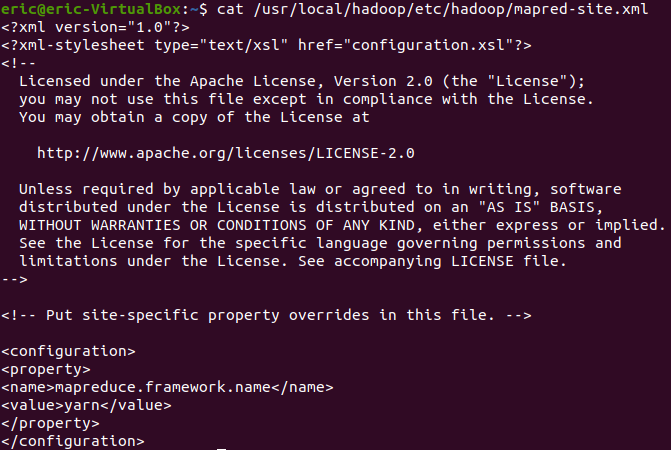

10. close .bashrc file and execute command: “. .bashrc”  


12.show the directory with its new owner  


13.show the output of the ‘core-site.xml’ file  


14. show the output of the ‘hdfs-site.xml’ file  


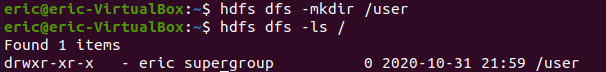
15. show the output of the ‘mapred-site.xml’ file  


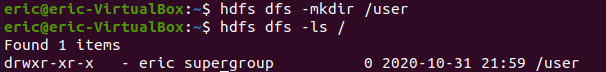
16. show the output of the ‘yarn-site.xml’ file  


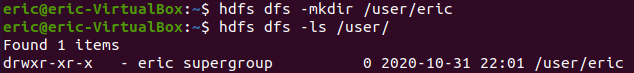
20.check services  

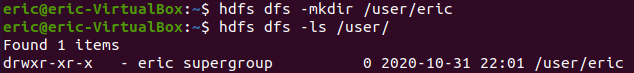

Commands:

2.examine files in HDFS cluster  


3.create a directory in HDFS cluster  


4.examine files again  


5.create a new directory  


6.examine the new directory  


8.upload the file in the cluster  


10.see the content of the file  


**Task2**

**A  
mapper\_Wei\_A.py**

#!/usr/bin/python3

import string

import sys

# get all lines from stdin

for line in sys.stdin:

val = line.strip()

words = val.split(' ')

for word in words:

for letter in word:

# remove punctuations from the word

punc = '''!()-[]{};:'"\\, <>./?@#$%^&\*\_~'''

if letter in punc:

word = word.replace(letter, '')

# uniform the word in lower case

word = word.lower()

print ('%s\t%s' % (word, 1))

**reducer\_Wei\_A.py**

#!/usr/bin/python3

import sys

from operator import itemgetter

items = []

new\_items = []

(last\_key, count) = (None, 0)

for line in sys.stdin:

(key, val) = line.strip().split('\t')

if last\_key and last\_key != key:

# print('%s\t%s' % (last\_key, count))

items.append((last\_key, count))

(last\_key, count) = (key, 1)

else:

(last\_key, count) = (key, count + 1)

if last\_key:

# print('%s\t%s' % (last\_key, count))

items.append((last\_key, count))

items = sorted(items, key = itemgetter(1), reverse = True)

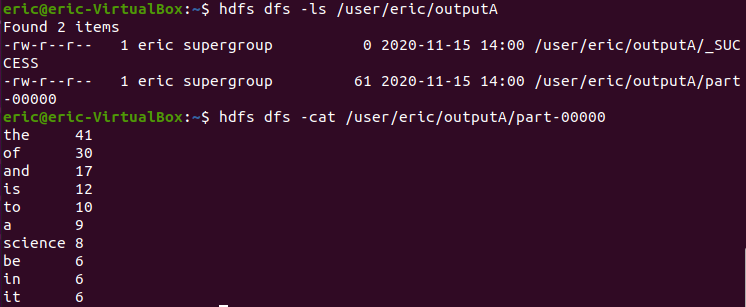
for item in items:

new\_items.append('%s\t%s' % (item[0], item[1]))

for newItem in new\_items[0:10]:

print(newItem)

**result:**



**B**

**mapper\_Wei\_B.py**

#!/usr/bin/python3

import string

import sys

for line in sys.stdin:

line = line.strip()

words = line.split(' ')

for word in words[2:5:2]:

print(str(word), end ='\t')

print()

**reducer\_Wei\_B.py**

#!/usr/bin/python3

import sys

from math import sqrt

def update(existingAggregate, newValue):

(count, mean, M2) = existingAggregate

count += 1

delta = newValue - mean

mean += delta / count

delta2 = newValue - mean

M2 += delta \* delta2

return (count, mean, M2)

(last\_key, last\_val) = (None, 0)

(count, mean, M2) = (0, 0, 0)

for line in sys.stdin:

(key, val) = line.strip().split('\t')

if last\_key and last\_key != key:

print('%s\t%s' % (last\_key, str(round(sqrt(M2/count), 2))))

(last\_key, last\_val) = (key, val)

(count, mean, M2) = update((0, 0, 0), float(val))

else:

(count, mean, M2) = update((count, mean, M2), float(val))

last\_key = key

if last\_key:

print('%s\t%s' % (last\_key, str(round(sqrt(M2/count), 2))))

**result:**

eric@eric-VirtualBox:~$ hdfs dfs -cat /user/eric/outputB/part-00000

Albuquerque 145.35

Anaheim 141.51

Anchorage 137.12

Arlington 138.74

Atlanta 131.66

Aurora 142.75

Austin 132.6

Bakersfield 161.24

Baltimore 156.49

Baton Rouge 119.82

Birmingham 155.63

Boise 133.11

Boston 149.35

Buffalo 151.36

Chandler 138.45

Charlotte 140.89

Chesapeake 126.01

Chicago 143.99

Chula Vista 154.27

Cincinnati 138.94

Cleveland 142.54

Colorado Springs 147.71

Columbus 145.86

Corpus Christi 136.15

Dallas 150.37

Denver 131.76

Detroit 161.75

Durham 142.4

El Paso 148.82

Fort Wayne 146.93

Fort Worth 136.55

Fremont 142.48

Fresno 138.77

Garland 139.71

Gilbert 155.16

Glendale 143.66

Greensboro 144.63

Henderson 151.35

Hialeah 141.06

Honolulu 130.38

Houston 128.65

Indianapolis 141.14

Irvine 132.53

Irving 146.97

Jacksonville 147.61

Jersey City 135.39

Kansas City 141.38

Laredo 138.55

Las Vegas 142.42

Lexington 146.06

Lincoln 135.74

Long Beach 142.41

Los Angeles 152.48

Louisville 133.3

Lubbock 133.52

Madison 156.93

Memphis 145.72

Mesa 126.01

Miami 141.81

Milwaukee 148.3

Minneapolis 124.7

Nashville 150.69

New Orleans 151.55

New York 130.33

Newark 141.97

Norfolk 148.19

North Las Vegas 153.36

Oakland 134.08

Oklahoma City 144.56

Omaha 144.09

Orlando 150.36

Philadelphia 129.64

Phoenix 142.5

Pittsburgh 164.64

Plano 136.16

Portland 147.56

Raleigh 141.99

Reno 139.49

Richmond 140.38

Riverside 143.97

Rochester 142.14

Sacramento 161.18

Saint Paul 147.12

San Antonio 132.92

San Bernardino 139.7

San Diego 142.81

San Francisco 137.7

San Jose 126.59

Santa Ana 151.96

Scottsdale 147.41

Seattle 154.1

Spokane 139.51

St. Louis 137.14

St. Petersburg 130.6

Stockton 141.37

Tampa 131.59

Toledo 143.93

Tucson 140.92

Tulsa 131.89

Virginia Beach 129.8

Washington 123.9

Wichita 136.08

Winston–Salem 126.26

**C**

**mapper\_Wei\_C.py**

#!/usr/bin/python3

import sys

for line in sys.stdin:

line = line.strip()

words = line.split(' ')

for word in words[0:7:6]:

print(word, end = '\t')

print()

**reducer\_Wei\_C.py**

#!/usr/bin/python3

import sys

(last\_client, last\_file) = (None, None)

total = 0

for line in sys.stdin:

(client, file) = line.strip().split('\t')

if last\_client == client and last\_file != file:

total += 1

(last\_client, last\_file) = (client, file)

elif last\_client == client and last\_file == file:

(last\_client, last\_file) = (client, file)

elif not last\_client:

(last\_client, last\_file) = (client, file)

total = 1

else:

print('%s\t%s' % (last\_client, str(total)))

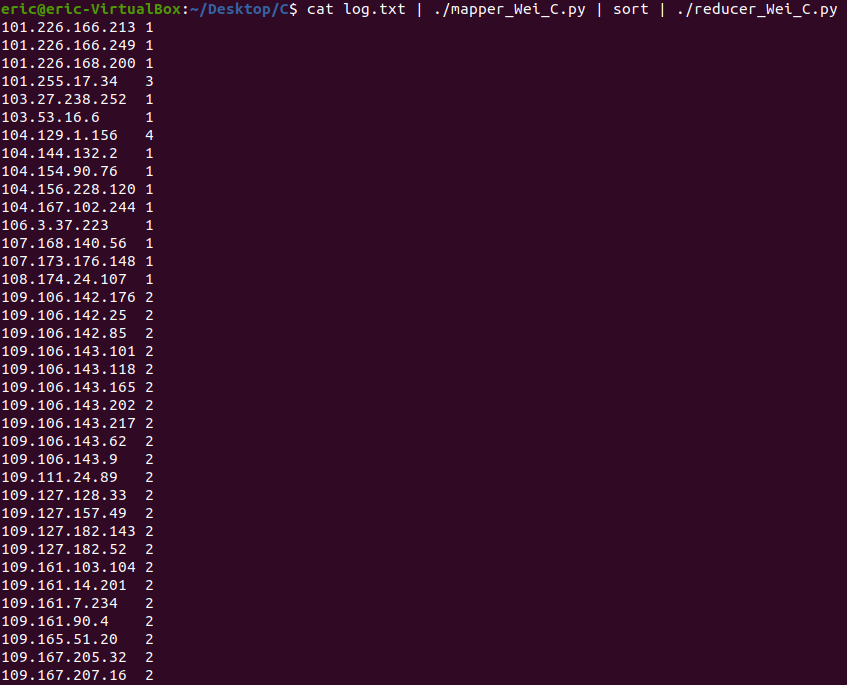
(last\_client, last\_file) = (client, file)

total = 1

if last\_client:

print('%s\t%s' % (last\_client, str(total)))

**result:**



(Professor, for the result of the part C, I screenshotted part of them since the answer is pretty massive and I think it’s a little bit clunky to lay them out here, could you please run my code if you wanna make a double check?)

**D**

**mapper\_Wei\_D.py**

#!/usr/bin/python3

import sys

import string

for line in sys.stdin:

line = line.strip()

words = line.split(',')

words.insert(0, words.pop())

for word in words:

print(word, end='\t')

print()

**reducer\_Wei\_D.py**

!/usr/bin/python3

import sys

(last\_key, last\_v1, last\_v2, last\_v3, last\_v4) = (None, 0, 0, 0, 0)

count = 0

for line in sys.stdin:

(key, v1, v2, v3, v4) = line.strip().split('\t')

if last\_key and last\_key != key:

last\_v1 = str(round(float(last\_v1/count), 2))

last\_v2 = str(round(float(last\_v2/count), 2))

last\_v3 = str(round(float(last\_v3/count), 2))

last\_v4 = str(round(float(last\_v4/count), 2))

print('%s\t%s\t%s\t%s\t%s' % (last\_key, last\_v1, last\_v2, last\_v3, last\_v4))

count = 1

(last\_key, last\_v1, last\_v2, last\_v3, last\_v4) = (key, v1, v2, v3, v4)

else:

count += 1

(last\_key, last\_v1, last\_v2, last\_v3, last\_v4) = (key, float(last\_v1) + float(v1), float(last\_v2) + float(v2), float(last\_v3) + float(v3), float(last\_v4) + float(v4))

if last\_key:

last\_v1 = str(round(float(last\_v1/count), 2))

last\_v2 = str(round(float(last\_v2/count), 2))

last\_v3 = str(round(float(last\_v3/count), 2))

last\_v4 = str(round(float(last\_v4/count), 2))

print('%s\t%s\t%s\t%s\t%s' % (last\_key, last\_v1, last\_v2, last\_v3, last\_v4))

**result:**

eric@eric-VirtualBox:~/Desktop/D$ hdfs dfs -cat /user/eric/outputD/part-00000

Iris-setosa 5.01 3.42 1.46 0.24

Iris-versicolor 5.94 2.77 4.26 1.33

Iris-virginica 6.59 2.97 5.55 2.03

**Task 3**

***Hadoop*** *was originated from 1997,* ***Doug Cutting*** *spent three months writing the first version of* ***Lucene****, a full text library for analyzing ordinary text with the purpose of index. In three years, he open sourced Lucene and surprisingly, many people found it useful and he received many great feedback. Then in 2001, Lucene moved to* ***Apache Software Foundation****. Later he was joined by* ***Mike Cafarella*** *for putting an effort into indexing the entire web and which derived a new web crawler called* ***Apache Nutch****. They deployed Nutch on a single machine which unexpectedly limited the total number of pages to 100 million, so that they needed a distributed storage layer for improving the scalability and handling component, till 2003, Google published the Google File System paper which perfectly could solve the very same problems, so they started to implement it in java and finished in 2004, naming it Nutch Distributed File System(****NDFS****) and which successfully helped them out by splitting file into chunks and chunks into nodes. After that, they embarked on the exploration between various data processing models, trying to figure out a way to achieve the parallelism, luckily they were enlightened again and worked out Parallelization, Distribution and Fault-tolerance. Then in 2006, Cutting created a new incubating project called* ***Hadoop****, consisting of* ***Hadoop Common****,* ***HDFS*** *and* ***MapReduce.*** *Meanwhile, Yahoo! also had the same problem as theirs, and after negotiation, they agreed to replace the original system by Hadoop and worked great. In the following years, Hadoop graduate to the top level and revolutionized data storage to possibly keep all the data, getting blooming and some services were also provided by Amazon. The emergence of relational databases effectively decrease the cost of memory and although these limitations are long gone, we need to remember the enormous benefit of information about history. In 2012, MapReduce v2 named* ***YARN*** *was pulled out from MapReduce codebase and marked a turn point.*