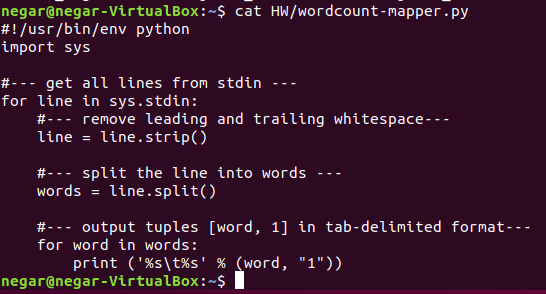
**درس : Big Data استاد : خانم دکتر صفری**

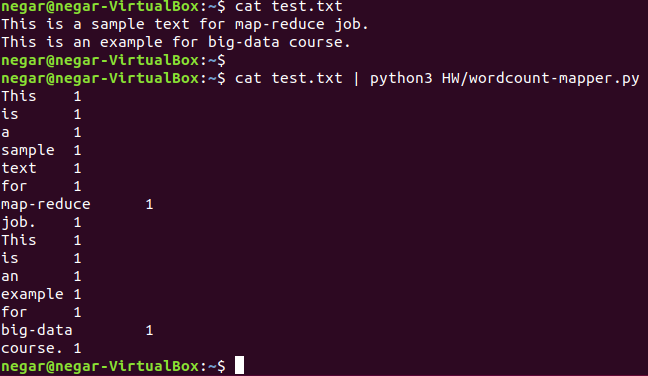
**دانشجو : رقیه یزدانی تمرین : MapReduce**

# MapReduce

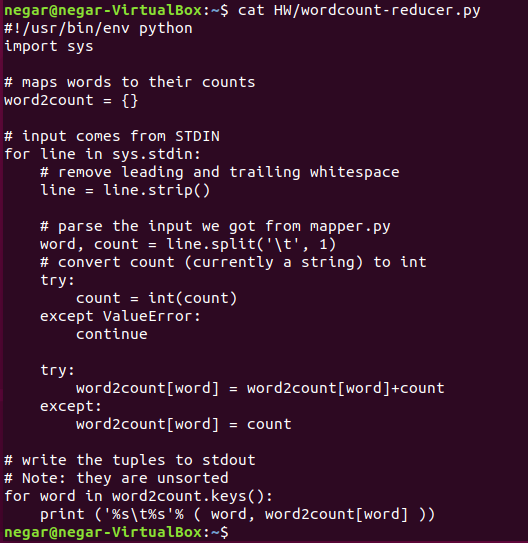
# انجام wordCount با استفاده از MapReduce

برای انجام این مثال، روی متنی از کتاب شکسپیر که برای تمرین hadoop دانلود شد، کار کرده و سعی می‌شود که دفعات تکرار کلمات در این فایل شمارش شود و از زبان پایتون برای نوشتن برنامه mapper و reducer استفاده شده است. فایل wordcount-mapper.py بصورت زیر، به ازای هر خط ورودی، کلمات آن را جدا کرده و لیستی از (key,value) که key هر کلمه و value مقدار 1 است را برمی‌گرداند. در ادامه مثالی از اجرای آن هم آورده شده است:

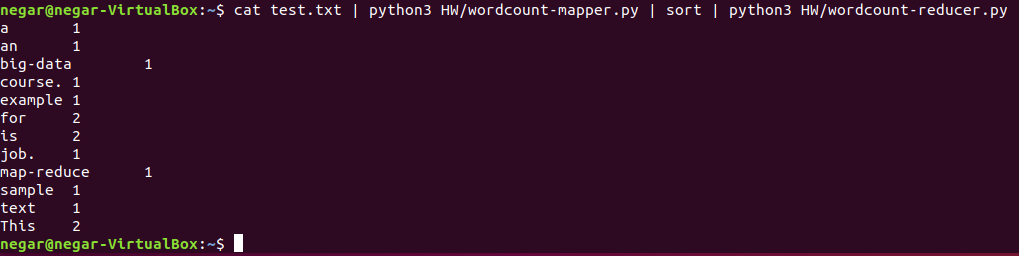




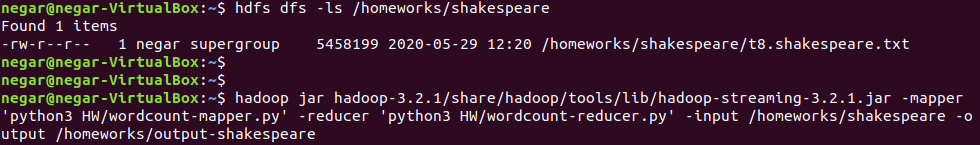
فایل wordcount-reducer.py هم لیستی از (key,value) ها را می‌گیرد و یک map می‌سازد که هر بار مقدار value هر pair را به count معادل آن کلمه اضافه می‌کند.



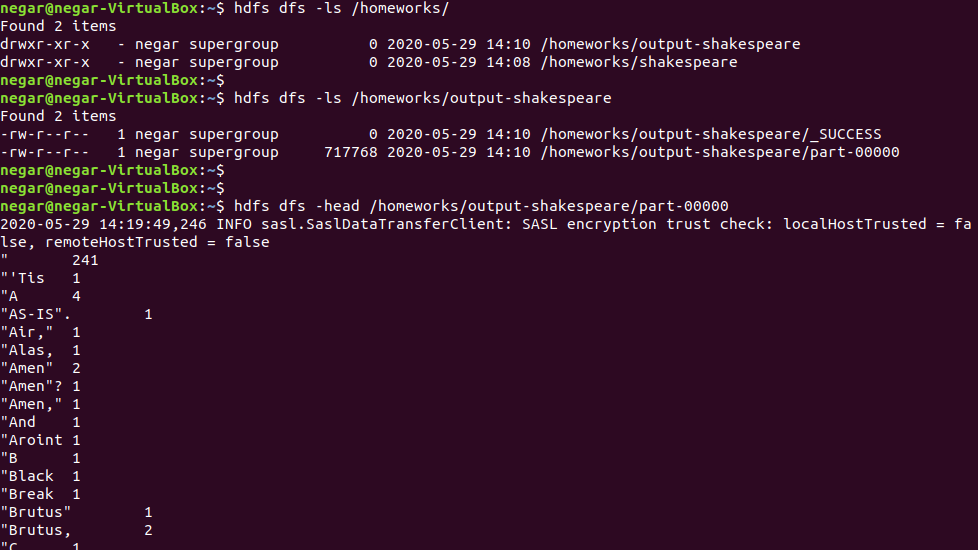
در ادامه برای اطمینان از صحت عملکرد برنامه، برای فایل test.txt ابتدا برنامه mapper اجرا شده و سپس key ها مرتب شده و به reducer داده می‌شود. این عملیات بصورت local انجام می‌شود و خارج از hdfs بوده و تنها برای تست برنامه است :



حال که از درستی عملکرد برنامه mapper و reducer اطمینان حاصل شد، باید برای فایل شکسپیر و روی hdfs این کار انجام شود. فایل متن با نام t8.shakespeare.txt در مسیر /homeworks/shakespeare روی hdfs قرار داده شده است و برنامه های mapper و reducer روی ماشین local و در مسیر HW/ قرار دارند. با دستور hadoop jar بصورتی که در شکل زیر نشان داده شده، با -mapper برنامه mapper و با -reducer برنامه reducer و با -input مسیر فایل ورودی روی hdfs و با -output مسیر خروجی را تعیین می‌کنیم.



پس از اجرای دستور بالا؛ همانطور که در شکل زیر مشخص شده، فایلی به نام part-00000 در مسیر تعیین شده برای خروجی، ساخته می‌شود که محتوای چند سطر اول آن در زیر؛ نشان داده شده است.

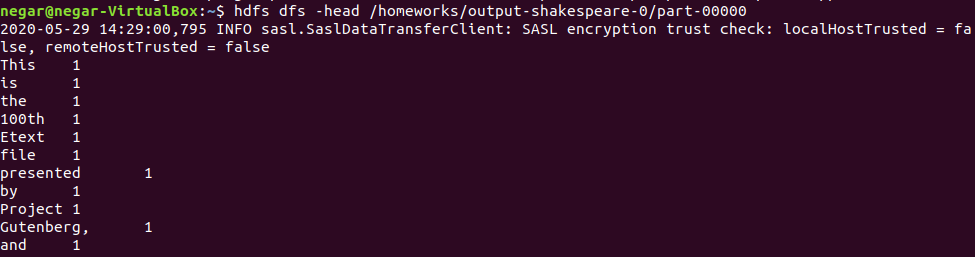


-numReduceTasks

هنگام استفاده از دستور hadoop jar می‌توان از -numReduceTasks نیز استفاده کرد، مقدار پیش‌فرض آن 1 است و به معنای تعداد reducer ها می‌باشد. اگر به آن مقدار 0 بدهیم، در واقع مرحله reducer را حذف کرده‌ایم و اجرا نخواهد شد. تمرین بالا را با این حالت نیز اجرا می کنیم :



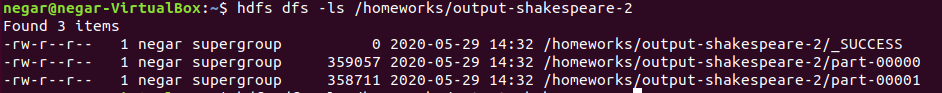
خروجی زیر نشان می دهد؛ که تنها مرحله mapper اجرا شده است:



یکبار دیگر این تمرین با مقدار 2 برای پارامتر -numReduceTasks اجرا می‌کنیم :



این بار دو فایل خروجی می سازد، فایلهای part-00000 و part-00001 که هر کدام خروجی یک reducer است:



بررسی فایلها نشان می‌دهد، کلیدها در هر کدام از این فایلها، متفاوت هستند و در واقع ترکیب این دو فایل معادل خروجی زمانی است که برنامه با یک reducer اجرا شد. به عبارت دیگر برنامه چه با یک reducer و چه با دو reducer اجرا شود، تعداد دفعات تکرار هر کلمه در کل متن شمرده می‌شود و نتیجه برای هر کلمه در هر دو حالت یکسان است.

# انجام join با استفاده از MapReduce

این بخش شامل دو تمرین جدا است که به نحوی تکمیل کننده هم هستند.

# تمرین اول :

دو فایل داریم به نامهای join1\_FileA.txt و join1\_FileB.txt که در ادامه محتوای هر کدام آورده شده است. هر سطر از فایل اول، شامل یک کلمه به عنوان key و یک عدد به عنوان value است که با , از هم جدا شده‌اند و هر سطر از فایل دوم، شامل یک تاریخ و یک کلمه به عنوان key و یک عدد بعنوان value که با , از هم جدا شده‌اند.

Join1\_FileA.txt

able,991

about,11

burger,15

actor,22

join1\_FileB.txt

Jan-01 able,5

Feb-02 about,3

Mar-03 about,8

Apr-04 able,13

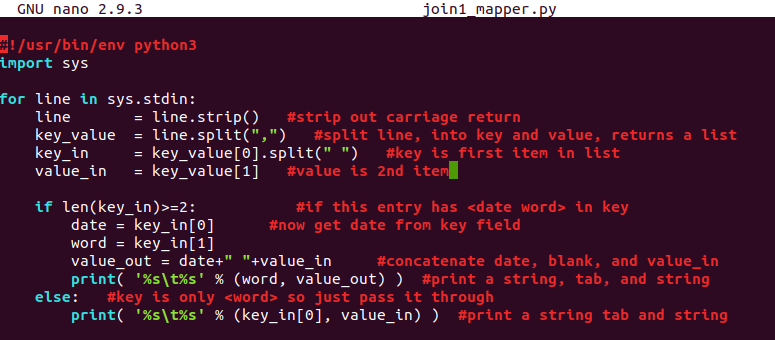
Feb-22 actor,3

Feb-23 burger,5

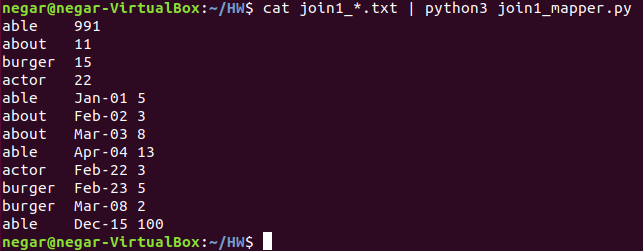
Mar-08 burger,2

Dec-15 able,100

هدف این است که این دو فایل روی "کلمه" با هم join شوند و خروجی ترکیبی بسازند. برای انجام آن با استفاده از MapReduce نیاز به یک برنامه mapper و یک برنامه reducer داریم. کد پایتون برنامه mapper در زیر آورده شده است :



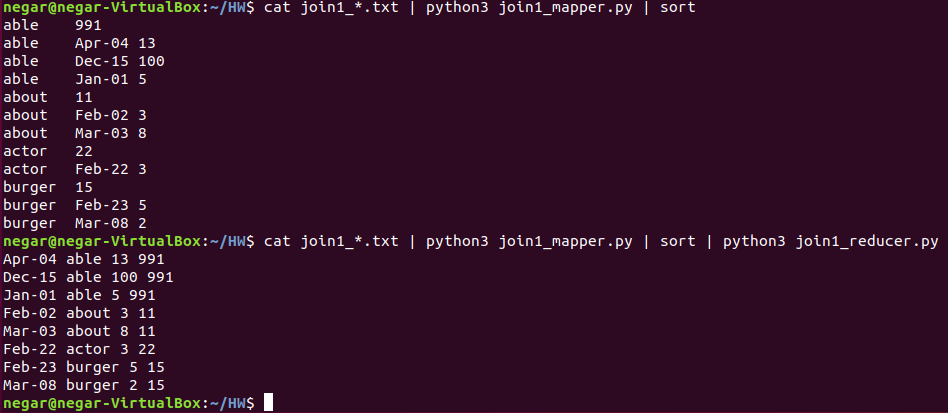
نتیجه اجرای برنامه join1\_mapper.py روی فایلهای ورودی بصورت زیر است. همانطور که مشخص است با فایل اول که کلید، یک کلمه است، کاری ندارد اما برای فایل دوم، تاریخ؛ از بخش key جدا شده و به بخش value اضافه شده است :



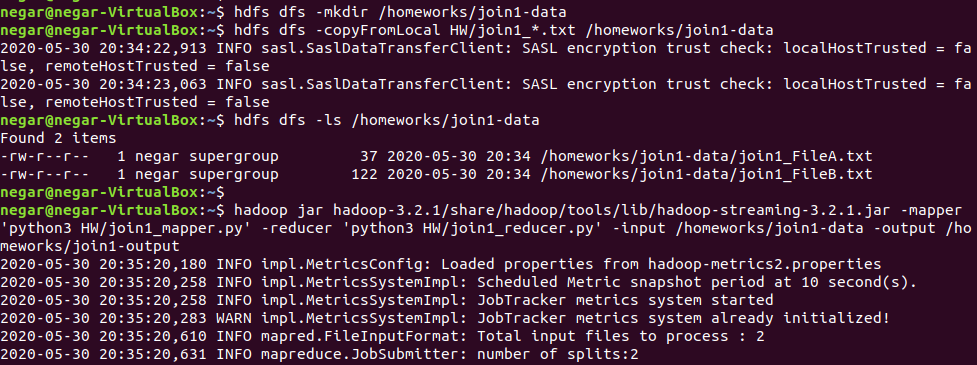
سپس نیاز به یک برنامه reducer داریم که در شکل زیر کد پایتون آن آورده شده است :

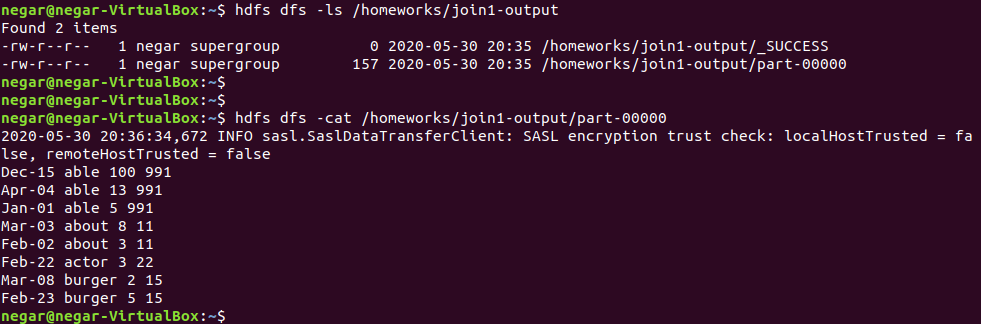


در شکل زیر ابتدا مرحله sort بعد از اجرای mapper انجام شده و خروجی آن نشان داده است و سپس مرحله reducer با اجرای برنامه بالا انجام شده و نتیجه کار که حاصل join اطلاعات می باشد، آورده شده است.



همین کار را مشابه قبل باید روی hdfs انجام دهیم، که لازم است ابتدا فایلهای داده، به hdfs منتقل شوند و سپس برنامه MapReduce اجرا شود. در شکلهای زیر، مراحل این کار نشان داده است و سپس خروجی کار نمایش داده شده است.





# تمرین دوم :

برای این بخش، برنامه پایتون به نام make\_join2data.py در اختیار قرار داده شده تا بکمک آن بتوان فایلهای داده را ایجاد کرد. کد این برنامه در ادامه آمده است :

make\_join2data.py

#!/usr/bin/env python

import sys

# --------------------------------------------------------------------------

# (make\_join2data.py) Generate a random combination of titles and viewer

    counts, or channels

# this is a simple version of a congruential generator,

# not a great random generator but enough

# --------------------------------------------------------------------------

chans = ['ABC','DEF','CNO','NOX','YES','CAB','BAT','MAN','ZOO','XYZ','BOB']

sh1 =['Hot','Almost','Hourly','PostModern','Baked','Dumb','Cold','Surreal'

    ,'Loud']

sh2 =['News','Show','Cooking','Sports','Games','Talking','Talking']

vwr =range(17,1053)

chvnm=sys.argv[1] #get number argument, if its n, do numbers not channels,

lch=len(chans)

lsh1=len(sh1)

lsh2=len(sh2)

lvwr=len(vwr)

ci=1

s1=2

s2=3

vwi=4

ri=int(sys.argv[3])

for i in range(0,int(sys.argv[2])): #arg 2 is the number of lines to output

if chvnm=='n': #no numuber

print('{0}\_{1},{2}'.format(sh1[s1],sh2[s2],chans[ci]))

else:

print('{0}\_{1},{2}'.format(sh1[s1],sh2[s2],vwr[vwi]))

ci=(5\*ci+ri) % lch

s1=(4\*s1+ri) % lsh1

s2=(3\*s1+ri+i) % lsh2

vwi=(2\*vwi+ri+i) % lvwr

if (vwi==4): vwi=5

این برنامه داده هایی تصادفی ایجاد می کند که با شش بار اجرای آن بصورت زیر، شش فایل تولید خواهد شد :

python3 make\_join2data.py y 1000 13 > join2\_gennumA.txt

python3 make\_join2data.py y 2000 17 > join2\_gennumB.txt

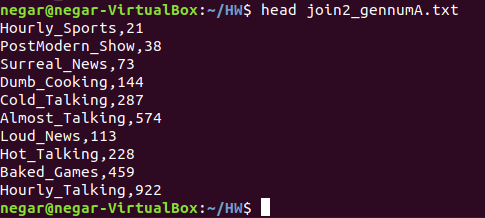
python3 make\_join2data.py y 3000 19 > join2\_gennumC.txt

python3 make\_join2data.py n 100 23 > join2\_genchanA.txt

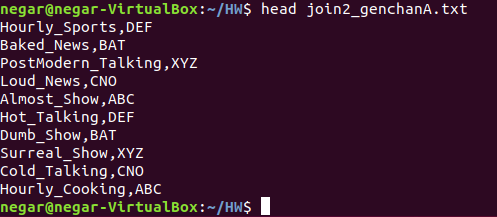
python3 make\_join2data.py n 200 19 > join2\_genchanB.txt

python3 make\_join2data.py n 300 37 > join2\_genchanC.txt

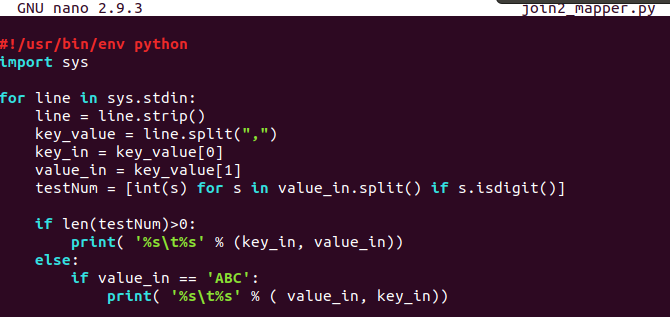
در ادامه نمونه ای از چند خط ابتدایی داده های فایل join2\_gennumA.txt آورده شده است که شامل نام یک برنامه تلویزیونی و تعداد دفعات تماشای آن می‌باشد.



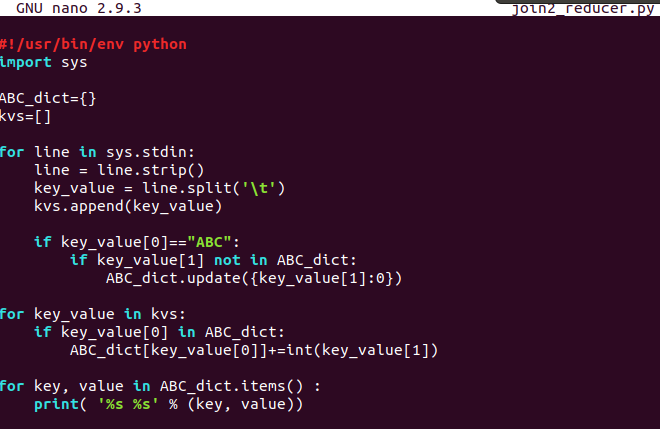
در تصویر زیر، چند خط ابتدایی اطلاعات فایل join2\_genchanA.txt آمده است که شامل نام برنامه تلویزیونی و نام channel پخش آن برنامه است :



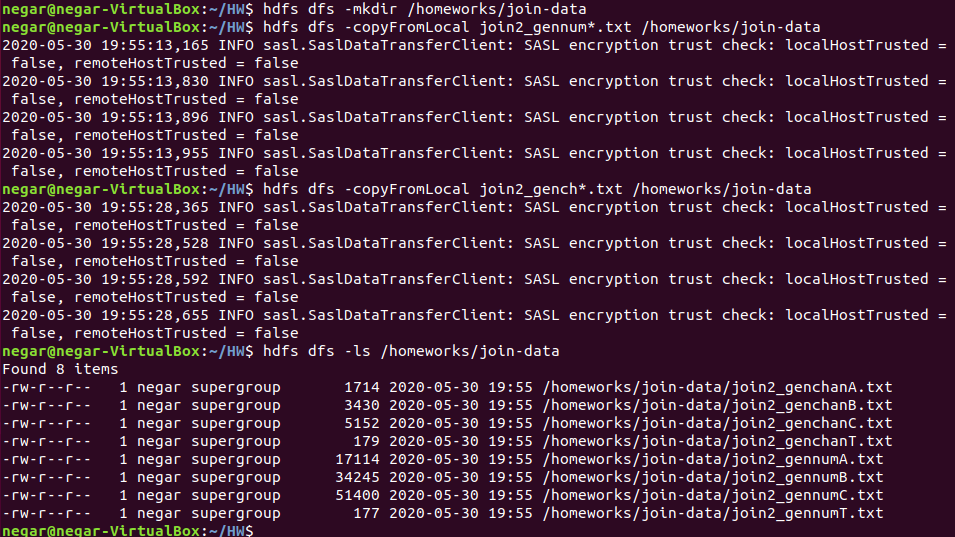
می خواهیم با استفاده از MapReduce بدست آوریم که شبکه ABC، چند تماشاگر داشته است. برای اینکار لازم است که این دو نوع فایل را روی نام برنامه تلویزیونی، join کنیم و سپس تعداد تماشاگرهای برنامه‌های مختلف این channel را با هم جمع کنیم. برنامه mapper بصورت زیر نوشته می‌شود:



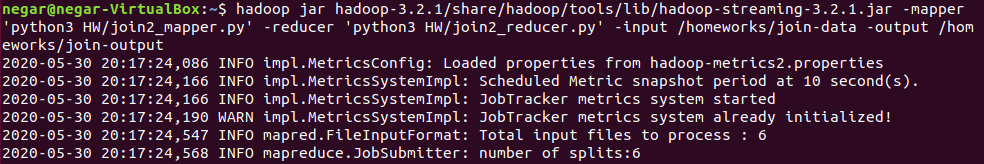
برنامه reducer نیز بصورت زیر نوشته می‌شود :



پس از تست برنامه های mapper و reducer روی یک نمونه فایل کوچک و اطمینان از صحت عملکرد آن، حال برای اجرا روی hdfs، ابتدا باید فایلهای داده تولید شده را به hdfs منتقل کنیم که این کار را با روشهایی که در تمرین hadoop آموختیم، انجام می دهیم. مراحل اینکار در شکل زیر نشان داده شده است :



سپس بصورت زیر برنامه MapReduce را با معرفی برنامه mapper، برنامه reducer، فایلهای ورودی و محل ذخیره خروجی اجرا می کنیم :



پس از پایان کار، محتوای فایل خروجی تولید شده را مشاهده می‌کنیم، که در واقع، لیست برنامه های تلویزیونی ای که در شبکه ABC پخش شده‌اند را به همراه آمار تماشاگر آنها نشان می‌دهد.

