درس : Big Data

استاد : خانم دکتر صفری

دانشجو : رقیه یزدانی

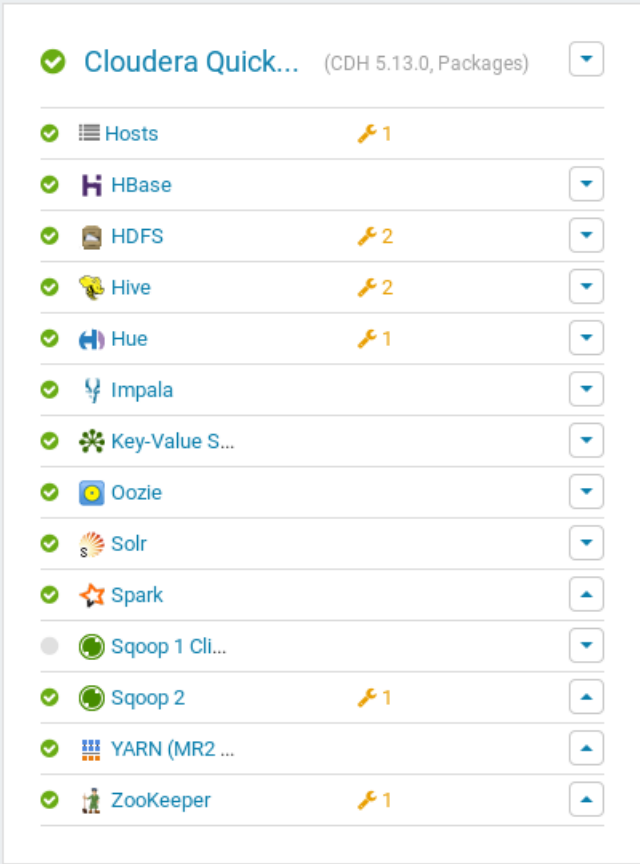
تمرین : Spark

Spark

ایجاد بستر لازم برای امکان کار با Spark

قبلاً فایل cloudera-quickstart-vm-5.13.0-0-vmware که در واقع ماشین مجازی cloudera می‌باشد، دانلود شده و با استفاده از نرم افزار VMWare Player 15.5 امکان کار با آن فراهم شده است.

برای این تمرین، لازم است که cloudera manager(CM) را launch کرده باشیم که قبلاً در انجام تمرین Hbase این کار انجام شد. همچنین لازم است که سرویس spark بالا باشد که اگر اینگونه نیست میتوان سرویس آن را بصورت دستی از از طریق اینترفیس، start کرد.



برای انجام تمرینها، قرار است با دیتاست هایی که برای بخش join از تمرینهای MapReduce استفاده شد، کار شود.

تمرین اول و انجام Simple Join

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 می‌باشد 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 یک خروجی ترکیبی ایجاد شود. مراحل کار بصورت زیر است :

- انتقال این فایلها به hdfs در ماشین cloudera که برای اینکار می‌توانیم ابتدا این دو فایل را از طریق ابزاری مثل Bitwise SSH client به ماشین مجازی cloudera منتقل کرده و سپس مشابه روشی که در تمرین Hadoop استفاده می‌شد، فایلها را به hdfs کپی کرد.

```
[cloudera@quickstart ~]$ ll spark-hw/
total 8
-rw-rw-r-- 1 cloudera cloudera  37 Jul 29  2016 join1_FileA.txt
-rw-rw-r-- 1 cloudera cloudera 122 Jul 29  2016 join1_FileB.txt
[cloudera@quickstart ~]$ hdfs dfs -mkdir spark-join1
[cloudera@quickstart ~]$ hdfs dfs -copyFromLocal spark-hw/join1_File*.txt spark-join1
[cloudera@quickstart ~]$ hdfs dfs -ls spark-join1/
Found 2 items
-rw-r--r--   1 cloudera cloudera         37 2020-06-19 08:12 spark-join1/join1_FileA.txt
-rw-r--r--   1 cloudera cloudera        122 2020-06-19 08:12 spark-join1/join1_FileB.txt
[cloudera@quickstart ~]$
```

- ورود به spark shell و ایجاد RDD بر اساس محتویات فایل join1_FileA.txt و سپس اجرای collect

```
fileA = sc.textFile("spark-join1/join1_FileA.txt")
fileA.collect()
```

- ایجاد RDD بر اساس محتویات فایل join1_FileB.txt و سپس اجرای collect

```
fileB = sc.textFile("spark-join1/join1_FileB.txt")
fileB.collect()
```

- ایجاد یک تابع به زبان پایتون با نام split_fileA که یک سطر از فایل join1_FileA.txt را به عنوان ورودی دریافت کرده و زوج (مقدار،کلید) را بر اساس آن بر می گرداند :

```
def split_fileA(line):
  line = line.split(",")
  word = line[0]
  count = line[1]
  return (word, count)
```

- درستی عملکرد تابع بالا را با یک عبارت تستی مشابه "yazdani,123" بررسی می کنیم.

```
test_line = " yazdani,123"
split_fileA(test_line)
```

- اجرای عمل map بر اساس تابع بالا (split_fileA) روی داده های RDD با نام fileA که در مراحل قبل ایجاد شد.

```
fileA_data = fileA.map(split_fileA)
fileA_data.collect()
```

- ایجاد یک تابع به زبان پایتون با نام split_fileB که یک سطر از فایل join1_FileB.txt را به عنوان ورودی دریافت کرده و زوج (مقدار،کلید) را بر اساس آن بر می‌گرداند بصورتی که کلید، "کلمه" می‌باشد و مقدار، ترکیب "عدد تاریخ" می‌باشد.

```
def split_fileB(line):
  line = line.split(",")
  key_in = line[0].split(" ")
  count = line[1]
  date = key_in[0]
  word = key_in[1]
  return (word, date + " " + count)
```

- درستی عملکرد تابع بالا را با یک عبارت تستی بررسی می کنیم.

```
test_line = "Jan-01 yazdani,123"
split_fileB(test_line)
```

- اجرای عمل map بر اساس تابع بالا (split_fileB) روی داده های RDD با نام fileB

```
fileB_data = fileB.map(split_fileB)
fileB_data.collect()
```

- اجرای عمل join روی دو RDD با نامهای fileA_data و fileB_data و سپس اجرای collect

```
fileB_joined_fileA = fileB_data.join(fileA_data)
fileB_joined_fileA.collect()
```

نتیجه اجرای تمام مراحل بالا در شکلهای زیر آورده شده است :

```
[cloudera@quickstart ~]$ pyspark
Python 2.6.6 (r266:84292, Jul 23 2015, 15:22:56)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/

Using Python version 2.6.6 (r266:84292, Jul 23 2015 15:22:56)
SparkContext available as sc, HiveContext available as sqlContext.
>>> fileA = sc.textFile("spark-join1/join1_FileA.txt")
>>> fileA.collect()
[Stage 0:                                                       (0 + 0) / 2]20/06/19 08:16:14 WARN cluster.YarnSc
heduler: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and
have sufficient resources
[u'able,991', u'about,11', u'burger,15', u'actor,22']
>>> fileB = sc.textFile("spark-join1/join1_FileB.txt")
>>> fileB.collect()
[u'Jan-01 able,5', u'Feb-02 about,3', u'Mar-03 about,8', u'Apr-04 able,13', u'Feb-22 actor,3', u'Feb-23 burger,5', u'
Mar-08 burger,2', u'Dec-15 able,100']
>>> def split_fileA(line):
...     line = line.split(",")
...     word = line[0]
...     count = line[1]
...     return (word, count)
...
>>> test_line="yazdani,123"
>>> split_fileA(test_line)
('yazdani', '123')
>>>
>>> fileA_data = fileA.map(split_fileA)
>>> fileA_data.collect()
[(u'able', u'991'), (u'about', u'11'), (u'burger', u'15'), (u'actor', u'22')]
>>>
```

```
>>> def split_fileB(line):
...     line = line.split(",")
...     key_in = line[0].split(" ")
...     count = line[1]
...     date = key_in[0]
...     word = key_in[1]
...     return (word, date + " " + count)
...
>>>
>>> test_line = "Jan-01 yazdani,123"
>>>
>>> split_fileB(test_line)
('yazdani', 'Jan-01 123')
>>>
>>> fileB_data = fileB.map(split_fileB)
>>> fileB_data.collect()
[(u'able', u'Jan-01 5'), (u'about', u'Feb-02 3'), (u'about', u'Mar-03 8'), (u'able', u'Apr-04 13'), (u'actor', u'Feb-
22 3'), (u'burger', u'Feb-23 5'), (u'burger', u'Mar-08 2'), (u'able', u'Dec-15 100')]
>>>
>>> fileB_joined_fileA = fileB_data.join(fileA_data)
>>>
>>> fileB_joined_fileA.collect()
[(u'able', (u'Jan-01 5', u'991')), (u'able', (u'Apr-04 13', u'991')), (u'able', (u'Dec-15 100', u'991')), (u'burger',
 (u'Feb-23 5', u'15')), (u'burger', (u'Mar-08 2', u'15')), (u'about', (u'Feb-02 3', u'11')), (u'about', (u'Mar-03 8',
 u'11')), (u'actor', (u'Feb-22 3', u'22'))]
>>> _
```

بر اساس نتیجه join در تصویر بالا، برای کلمه actor، زوج (('Feb-22 3','22'),'actor') تولید شده است.

تمرین دوم و انجام Advanced Join

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 داده‌هایی تصادفی ایجاد می‌کند که با شش بار اجرای آن بصورت زیر، شش فایل تولید خواهد شد :

```
python make_join2data.py y 1000 13 > join2_gennumA.txt
python make_join2data.py y 2000 17 > join2_gennumB.txt
python make_join2data.py y 3000 19 > join2_gennumC.txt
python make_join2data.py n 100  23 > join2_genchanA.txt
python make_join2data.py n 200  19 > join2_genchanB.txt
python make_join2data.py n 300  37 > join2_genchanC.txt
```

هر سطر از فایلهای gennum حاوی نام یک برنامه تلویزیونی و تعداد دفعات تماشای آن (views) است و هر سطر از فایلهای genchan شامل نام برنامه تلویزیونی و channel پخش آن می‌باشد. هدف این است که مجموع تماشاگران کلیه برنامه های تلویزیونی کانال BAT را بدست آوریم. مراحل کار بصورت زیر است :

- انتقال فایلهای داده تولید شده به hdfs در ماشین cloudera مشابه تمرین 1

```
[cloudera@quickstart ~]$ hdfs dfs -ls spark-join2
Found 6 items
-rw-r--r--   1 cloudera cloudera       1714 2020-06-19 10:43 spark-join2/join2_genchanA.txt
-rw-r--r--   1 cloudera cloudera       3430 2020-06-19 10:43 spark-join2/join2_genchanB.txt
-rw-r--r--   1 cloudera cloudera       5152 2020-06-19 10:43 spark-join2/join2_genchanC.txt
-rw-r--r--   1 cloudera cloudera      17114 2020-06-19 10:43 spark-join2/join2_gennumA.txt
-rw-r--r--   1 cloudera cloudera      34245 2020-06-19 10:43 spark-join2/join2_gennumB.txt
-rw-r--r--   1 cloudera cloudera      51400 2020-06-19 10:43 spark-join2/join2_gennumC.txt
[cloudera@quickstart ~]$
```

- ایجاد RDD بر اساس محتویات فایلهای gennum و سپس اجرای take برای رویت نمونه هایی از آن

```
show_views_file = sc.textFile("spark-join2/join2_gennum?.txt")
show_views_file.take(3)
```

- نوشتن تابعی به زبان پایتون برای تبدیل هر سطر از فایلهای gennum به زوج (show,views)

```
def split_show_views(line):
  line = line.split(",")
  show = line[0]
  views = line[1]
  return (show,int(views))
```

- اجرای map روی show_views_file

```
show_views = show_views_file.map(split_show_views)
show_views.take(3)
```

- ایجاد RDD بر اساس محتویات فایلهای genchan و سپس اجرای take برای رویت نمونه هایی از آن

```
show_channel_file = sc.textFile("spark-join2/join2_genchan?.txt")
show_channel_file.take(3)
```

- نوشتن تابعی برای تبدیل هر سطر از فایلهای genchan به زوج (show,(channel,views))

```
def split_show_channel(line):
  line = line.split(",")
  show = line[0]
  channel = line[1]
  return (show,channel)
```

- اجرای map روی show_channel_file

```
show_channel = show_channel_file.map(split_show_channel)
show_channel.collect()
```

- اجرای join روی دو RDD

joined_dataset = show_views.join(show_channel)

- نوشتن تابعی برای استخراج زوج (channel,view) از هر عضو دیتاست نتیجه join

```
def extract_channel_views(show_views_channel):
    show = show_views_channel[0]
    views = show_views_channel[1][0]
    channel = show_views_channel[1][1]
    return (channel,int(views))
```

- اجرای map

channel_views = joined_dataset.map(extract_channel_views)

- اجرای reduceByKey

channel_views.reduceByKey(lambda x,y:x+y).collect()

اجرای کلیه مراحل در تصویر زیر آمده و بر اساس نتیجه اجرای آخرین دستور، آمار تماشاگران کانال BAT تعداد 5099141 بدست آمده است.

```
>>> show_views_file = sc.textFile("spark-join2/join2_gennum?.txt")
>>> show_views_file.take(3)
[u'Hourly_Sports,21', u'PostModern_Show,38', u'Surreal_News,73']
>>>
>>> def split_show_views(line):
...     line = line.split(",")
...     show = line[0]
...     views = line[1]
...     return (show,int(views))
...
>>> show_views = show_views_file.map(split_show_views)
>>> show_views.take(3)
[(u'Hourly_Sports', 21), (u'PostModern_Show', 38), (u'Surreal_News', 73)]
>>>
>>> show_channel_file = sc.textFile("spark-join2/join2_genchan?.txt")
>>> show_channel_file.take(3)
[u'Hourly_Sports,DEF', u'Baked_News,BAT', u'PostModern_Talking,XYZ']
>>>
>>> def split_show_channel(line):
...     line = line.split(",")
...     show = line[0]
...     channel = line[1]
...     return (show,channel)
...
>>> show_channel = show_channel_file.map(split_show_channel)
>>> show_channel.take(3)
[(u'Hourly_Sports', u'DEF'), (u'Baked_News', u'BAT'), (u'PostModern_Talking', u'XYZ')]
>>>
>>> joined_dataset = show_views.join(show_channel)
>>> joined_dataset.take(3)
[(u'PostModern_Cooking', (1038, u'DEF')), (u'PostModern_Cooking', (1038, u'CNO')), (u'PostModern_Cooking', (1038, u'CNO
>>>
>>> def extract_channel_views(show_views_channel):
...     show = show_views_channel[0]
...     views = show_views_channel[1][0]
...     channel = show_views_channel[1][1]
...     return (channel,int(views))
...
>>> channel_views = joined_dataset.map(extract_channel_views)
>>> channel_views.take(3)
[(u'DEF', 1038), (u'CNO', 1038), (u'CNO', 1038)]
>>>
>>> channel_views.reduceByKey(lambda x,y:x+y).collect()
[(u'XYZ', 5208016), (u'DEF', 8032799), (u'CNO', 3941177), (u'BAT', 5099141), (u'NOX', 2583583), (u'CAB', 3940862), (u'E
, (u'ABC', 1115974), (u'MAN', 6566187)]
>>>
```