به‌نام خداوند بخشنده و مهربان

تمرین اول: مقدمه‌ای بر اسپارک
درس: تحلیل‌ها و سیستم‌های داده حجیم

استاد: محمدعلی نعمت‌بخش
دستیاران: فاطمه ابراهیمی، پریسا لطیفی، امیر سرتیپی

نام و نام‌خانوادگی: عطیه نیکبخت (۴۰۰۳۶۱۴۰۴۷)
آدرس گیت: https://github.com/AtiyehNikbakht/SparkPractice.git

- **لطفا پاسخ تمارین حتما در سامانه‌ی کوئرا ارسال شود.**
- **لطفا پاسخ‌های خود را در خود سند سوال نوشته و در قالب یک فایل PDF ارسال کنید.**
- **نام سند ارسالی HW-{homework number}-{Name Family}-{student number}**
- **تمامی فایل‌های مورد نیاز این تمرین در این لینک قابل دسترس است.**
- **خروجی از هر مرحله‌ی تمرین را در سند خود بارگذاری کنید.**
- **کد + سند را در گیت بارگذاری کرده و لینک آن را در سند قرار دهید.**

در این تمرین، هدف ما آشنایی Action و Transformation در موتور تحلیلی Spark است.

۱. منظور از Lazy Evaluation در Spark چیست؟ این مفهوم را همراه با یک مثال توضیح دهید.

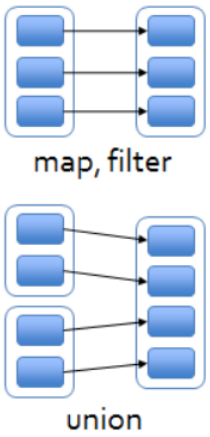
در اسپارک عملیات به دو دسته Transformation و Action تقسیم می‌شوند [۱]. عملیات‌های Transformation به‌صورت Lazy Evaluation هستند یعنی پس از فراخوانی، بلافاصله اجرا نمی‌شوند بلکه در صورتی عملیات‌های Transformation انجام می‌شوند که یک Action فراخوانی شود درنتیجه داده‌ها تا زمانی که نیاز نباشد بارگذاری نمی‌شوند. به عنوان مثال عملیات Map Reduce را در نظر بگیرید. همانطور که در شکل ۱ مشاهده می‌کنید ابتدا یک فایل متنی به RDD تبدیل می‌شود. به دلیل اینکه تابع flatMap()، map() و reduceByKey() از نوع عملیات Transformation هستند اجرا نمی‌شوند. هنگامی که برنامه به تابع collect() می‌رسد چون یک تابع از نوع عملیات Action فراخوانی شده است، اکنون عملیات Transformation به ترتیب اجرا می‌شوند.

```
textRdd = sc.textFile("C:/Users/User/Desktop/Data.txt")
textRddM = textRdd.flatMap(lambda x: x.split(' '))
textRddM = textRddM.map(lambda x: (x,1))
textRddM = textRddM.reduceByKey(lambda x,y: x+y)
textRddM = textRddM.collect()
```

شکل۱- مثال عملیات Action و Transformation

۲. منظور از Narrow Transmission (NT) و Wide Transmission (WT) را در Spark همراه با یک مثال بیان کنید. تفاوت اصلی این دو مفهوم چیست؟

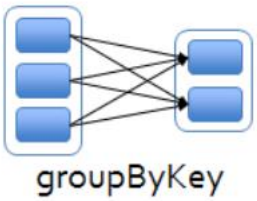
هر قسمت[۱] از والد[۲] RDD (گره والد) که حداکثر توسط یک فرزند[۳] RDD (گره فرزند) استفاده شود، یک Narrow Transmission است [۲]. همانطور که در شکل ۲ مشاهده می‌کنید به‌عنوان مثال در تابع[۴] Map() هر قسمت خروجی از یک قسمت ورودی محاسبه می‌شود [۱].



شکل۲- Narrow Transmission [۳]

[1] Partition
[2] Parent
[3] Child
[4] Function

زمانی که هر قسمت از والد RDD به چندین قسمت از فرزند RDD وابسته است (و به داده‌های بیشتر والدها نیاز است)، یک Wide Transmission داریم [۲]. به‌عنوان مثال تابع groupByKey() را درنظر بگیرید. همانطور که در شکل ۳ مشاهده می‌کنید groupByKey() باعث حرکت داده[1] در هر قسمت از Executorها می‌شود و به خروجی دیگر قسمت‌ها برای محاسبه نتیجه نهایی احتیاج دارد [۱].



شکل۳- Wide Transmission [۳]

سرعت Narrow Transmission بیشتر از Wide Transmission است زیرا در Narrow Transmission نیازی به حرکت[2] داده‌ها در شبکه خوشه‌ای نیست و هم‌چنین بهینه‌سازی‌هایی مثل Pipline امکان‌پذیر است اما در Wide Transmission نیاز به جابه‌جایی همه یا بخشی از داده‌ها در شبکه است بنابراین سرعت محاسبات را تحت تأثیر قرار می‌دهد [۲].

۳. با توجه به سوال پیشین، ۴ مورد از NT، WT و Action هایی که در اسپارک وجود دارند نام ببرید.

۴ مورد Narrow Transmission عبارتند از [۴]:

- Map (func)
- FlatMap (func)
- Filter (func)
- Union()

۴ مورد Wide Transmission عبارتند از [۴]:

- reduceByKey (func)
- groupByKey ()
- Join ()
- Distinct ()

[1] Data Shuffle
[2] Shuffle

۴ مورد Action عبارتند از [۵]:

- Take (n)
- Count ()
- Reduce (func)
- Collect ()

۴. برای آشنایی بیشتر با مفاهیم بیان شده و مقدمه‌ای بر توابع عملیات‌های زیر را انجام داده و خروجی هریک به همراه بلاک کد آن را گزارش دهید. مثالی از خروجی برای هر بخش نمایش داده شده است.

برای حل ادامه تمرینات از منابع [۶] و [۷] استفاده شده است.

- برای کار با اسپارک، کتاب‌خانه‌ای با نام pyspark وجود دارد.
- نوت‌بوکی بر روی گوگل کولب ایجاد کرده و این کتاب‌خانه را فراخوانی کنید.
- سپس یک لیست ۵۰ تایی از یک موضوع را برای خود درست کنید. برای مثال لیستی از (کتاب‌ها، نرم‌افزارها و ...)

پس از ایجاد نوت‌بوک، با دستور import کتابخانه‌های مورد نیاز را به برنامه اضافه می‌کنیم. طبق شکل ۴ برای ایجاد یک SparkSession از کتابخانه pyspark استفاده می‌کنیم. پس از آن یک لیست ۵۰تایی از فیلم‌ها می‌سازیم.

```
import pyspark
from pyspark.sql import SparkSession

myList = ['Kigdom', 'Squid Game', 'See', 'Game of Thrones', 'Vikings', 'Peaky Blinders', 'The Queens Gambit', 'Delhi Crime',
          'Breathe(hindi)', 'Breaking Bad', 'Chernobyl', 'The Night Of', 'Patrick Melrose', 'Walking Dead', 'Rick & Morty',
          'Mortal Wound', 'Moana', 'Solar Opposites', 'Soul', 'Monsters University', 'Money Heist', 'Goblin', 'Wish Dragon',
          'Minions', 'The Mitchells vs. the Machines', 'Mr. Queen', 'Dexter', 'Shameless', 'Mr. Robot','Monsters, Inc.',
          'Hannibal', 'The Witcher', 'Lucifer', 'Dark', 'Better call Saul', 'The 100', 'The Maze Runner', 'Shrek', 'Tangled',
          'Pirates of the Caribbean', 'Coco', 'Zootropolis', 'Sing', 'Hotel Transylvania', 'Harry Potter', 'Pasta', 'Heirs',
          'Twenty', 'The Twilight Saga', 'Chicken Little']
```

شکل۴- فراخوانی کتابخانه‌ها و تعریف لیست

- لیست خود را به RDD تبدیل کنید.

همانطور که در شکل ۵ مشاهده می‌کنید با استفاده از دستور SparkSession.builder، SparkSession را مقداردهی اولیه می‌کنیم و با دستور appName نام برنامه را SparkPractice می‌گذاریم. در تابع getOrCreate() اگر شی SparkSession وجود داشته باشد آن را برمی‌گرداند در غیر این‌صورت یک شی جدید می‌سازد. تابع parallelize() در sparkContext قرار دارد و برای تبدیل لیست به RDD استفاده می‌شود. با استفاده از این دستور لیست خود را به RDD تبدیل می‌کنیم.

```
spark = SparkSession.builder.appName('SparkPractice').getOrCreate()
sc = spark.sparkContext
rdd = sc.parallelize(myList)
```

شکل۵- ایجاد SparkSession و تبدیل لیست به RDD

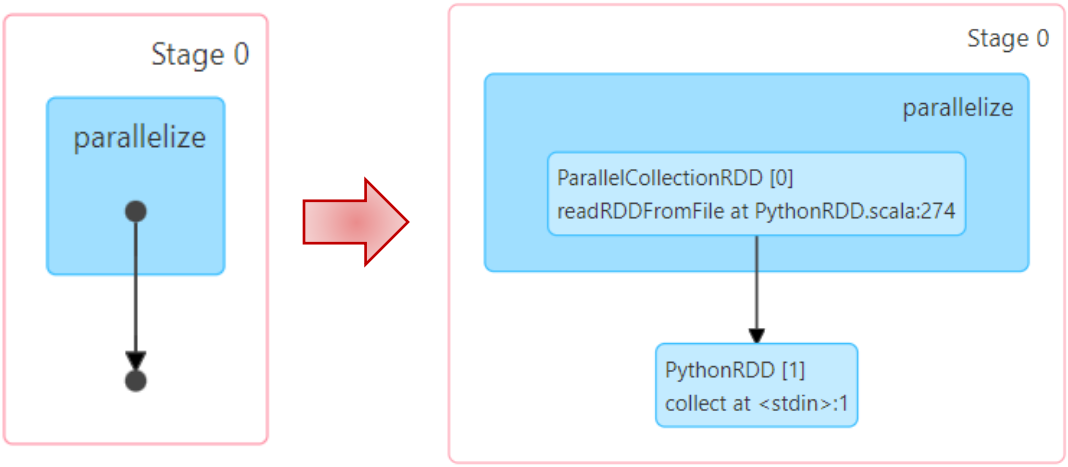
- با کمک دستور filter بر روی RDD، از آن برای بازیابی عنصر ۲۰ام لیست خود استفاده کنید. (برابر با عنصر ۲۰ام باشد)

طبق شکل ۶ با استفاده از تابع filter() عنصر ۲۰ام لیست را بازیابی می‌کنیم. ورودی تابع filter() یک تابع است بنابراین با استفاده از lambda یک تابع یک بار مصرف می‌سازیم و براساس عنصر ۲۰ام لیستی که قبلا ساخته بودیم، عناصر را فیلتر می‌کنیم تا عنصر مورد نظر را برگرداند سپس با استفاده از تابع collect() آن را به‌صورت لیست برمی‌گرداند و در متغیر twentiethItem ذخیره کرده و درنهایت با دستور print آن را چاپ می‌کنیم. خروجی نیز در شکل ۶ آمده است. در شکل ۷ نیز توالی انجام عملیات نشان داده شده است.

```
twentiethItem = rdd.filter(lambda x: myList[19] in x).collect()
print("Result is:" , twentiethItem)
```

```
Result is: ['Monsters University']
```

شکل۶- بازیابی عنصر ۲۰ام به کمک دستور filter



شکل۷- توالی انجام عملیات

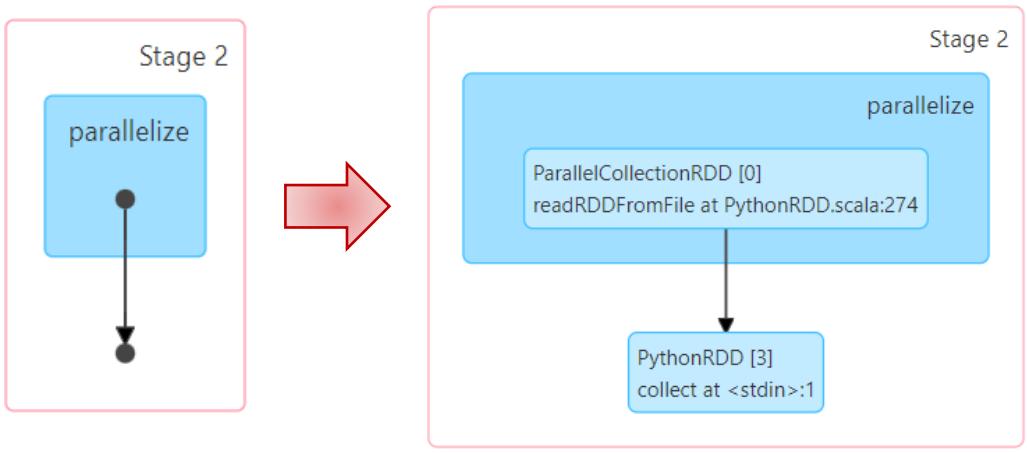
- با کمک map تمامی عناصر لیست خود را به حروف بزرگ تبدیل و آن را بازیابی کنید.

همانطور که در شکل ۸ مشاهده می‌کنید با استفاده از دستور map تابع ()upper روی تمام عناصر RDD اعمال می‌کند. تابع ()upper حروف کوچک را به بزرگ تبدیل می‌کند. هم‌چنین در شکل ۸ خروجی را مشاهده می‌کنید. در شکل ۹ نیز توالی انجام عملیات نشان داده شده است.

```
capitalLetterItems = rdd.map(lambda x: x.upper()).collect()
print("Result is:\n", capitalLetterItems)
```

```
Result is:
 ['KIGDOM', 'SQUID GAME', 'SEE', 'GAME OF THRONES', 'VIKINGS', 'PEAKY BLINDERS', 'THE QUEENS GAMBIT', 'DELHI CRIME', 'BREATHE(H
INDI)', 'BREAKING BAD', 'CHERNOBYL', 'THE NIGHT OF', 'PATRICK MELROSE', 'WALKING DEAD', 'RICK & MORTY', 'MORTAL WOUND', 'MOAN
A', 'SOLAR OPPOSITES', 'SOUL', 'MONSTERS UNIVERSITY', 'MONEY HEIST', 'GOBLIN', 'WISH DRAGON', 'MINIONS', 'THE MITCHELLS VS. THE
MACHINES', 'MR. QUEEN', 'DEXTER', 'SHAMELESS', 'MR. ROBOT', 'MONSTERS, INC.', 'HANNIBAL', 'THE WITCHER', 'LUCIFER', 'DARK', 'BE
TTER CALL SAUL', 'THE 100', 'THE MAZE RUNNER', 'SHREK', 'TANGLED', 'PIRATES OF THE CARIBBEAN', 'COCO', 'ZOOTROPOLIS', 'SING',
'HOTEL TRANSYLVANIA', 'HARRY POTTER', 'PASTA', 'HEIRS', 'TWENTY', 'THE TWILIGHT SAGA', 'CHICKEN LITTLE']
```

شکل۸- تبدیل حروف کوچک به بزرگ به کمک دستور map



شکل۹- توالی انجام عملیات

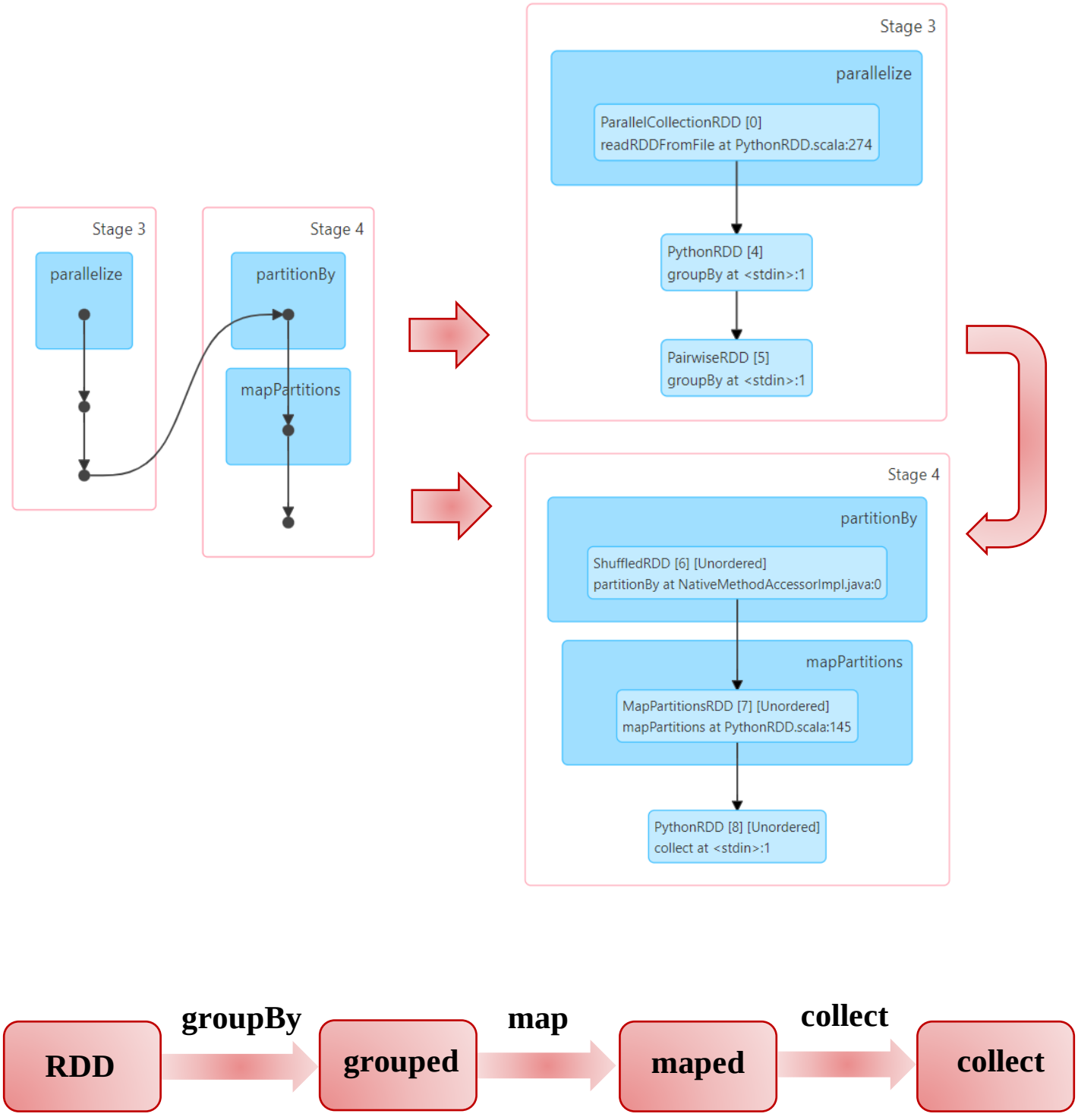
- با کمک دستور groupby و map، لیست خود را بر اساس اولین کاراکتر آن دسته بندی کنید.

همانطور که در شکل ۱۰ مشاهده می‌کنید با استفاده از دستور groupBy() تمام عناصر براساس حروف اولشان دسته‌بندی می‌شوند. برای دسترسی به حروف اول از x[0] استفاده کردیم. هم‌چنین برای نمایش عناصر دسته‌بندی شده از دستور map() استفاده شده است و به صورت تاپل است. اولین عنصر هر تاپل، حرف اول گروه و عنصر دوم که با دستور list به‌صورت لیست درآمده است، عناصر مربوط به آن دسته‌بندی می‌باشد. هم‌چنین در شکل ۱۰ خروجی را نیز مشاهده می‌کنید. در شکل ۱۱ نیز توالی انجام عملیات نشان داده شده است.

```
groupedItems = rdd.groupBy(lambda x: x[0]).map(lambda x: (x[0],list(x[1]))).collect()
print("Result is:\n", groupedItems)
```

```
Result is:
 [('B', ['Breathe(hindi)', 'Breaking Bad', 'Better call Saul']), ('H', ['Hannibal', 'Hotel Transylvania', 'Harry Potter', 'Heir
s']), ('C', ['Chernobyl', 'Coco', 'Chicken Little']), ('W', ['Walking Dead', 'Wish Dragon']), ('R', ['Rick & Morty']), ('V',
['Vikings']), ('Z', ['Zootropolis']), ('K', ['Kigdom']), ('L', ['Lucifer']), ('T', ['The Queens Gambit', 'The Night Of', 'The M
itchells vs. the Machines', 'The Witcher', 'The 100', 'The Maze Runner', 'Tangled', 'Twenty', 'The Twilight Saga']), ('M', ['Mo
rtal Wound', 'Moana', 'Monsters University', 'Money Heist', 'Minions', 'Mr. Queen', 'Mr. Robot', 'Monsters, Inc.']), ('S', ['Sq
uid Game', 'See', 'Solar Opposites', 'Soul', 'Shameless', 'Shrek', 'Sing']), ('G', ['Game of Thrones', 'Goblin']), ('P', ['Peak
y Blinders', 'Patrick Melrose', 'Pirates of the Caribbean', 'Pasta']), ('D', ['Delhi Crime', 'Dexter', 'Dark'])]
```

شکل۱۰- گروه‌بندی عناصر براساس حرف اولشان به کمک groupBy و map



شکل ۱۱- توالی انجام عملیات

- عملیات map و reduce را بر روی یک متن نسبتا بلند پس از تبدیل توکن‌های آن به rdd، انجام دهید.

طبق شکل ۱۲ در ابتدا یک فایل متنی را با دستور textFile() به RDD تبدیل می‌کنیم. سپس با دستور flatMap() و split() متن را به لیستی از کلمات تبدیل می‌کنیم سپس با دستور map() کلمات را به‌صورت تاپل‌های کلید-مقدار درمی‌آوریم و با دستور reduceByKey() براساس کلید (هر کلمه) تعداد آن‌ها را محاسبه می‌کنیم و در نهایت در خروجی نمایش می‌دهیم. هم‌چنین در شکل ۱۲ خروجی نمایش داده شده است. در شکل ۱۳ نیز توالی انجام عملیات نشان داده شده است.

```
textRdd = sc.textFile("Data.txt")
textRddM = textRdd.flatMap(lambda x: x.split(' '))
textRddM = textRddM.map(lambda x: (x,1))
textRddM = textRddM.reduceByKey(lambda x,y: x+y)
textRddM = textRddM.collect()
print("Result is:\n", textRddM)
```

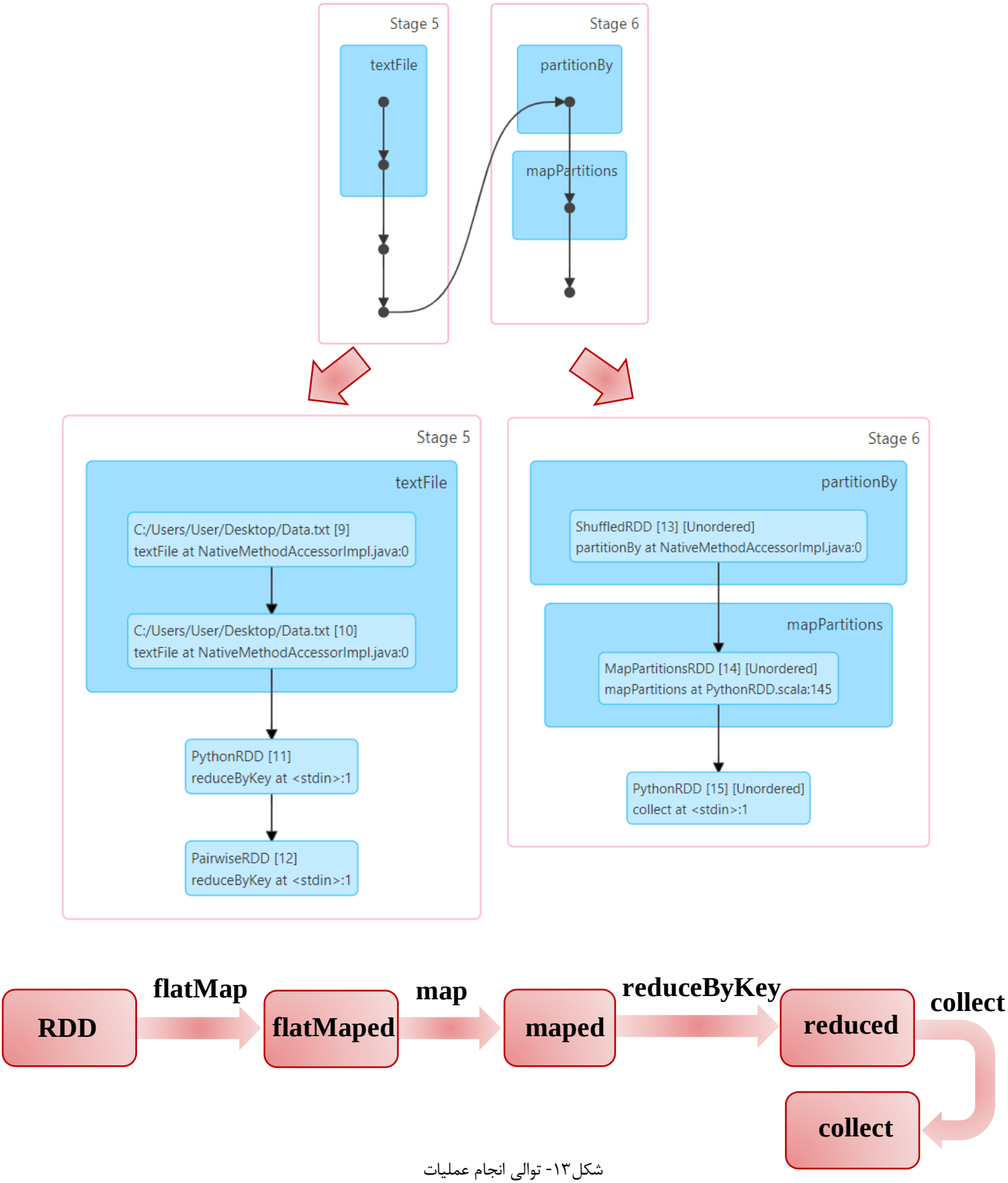
```
Result is:
 [('The', 16), ('an', 1), ('airplane', 2), ('crashes', 1), ('in', 10), ('badly', 1), ('his', 18), ('leaves', 1), ('narrator',
9), ('very', 3), ('water.', 1), ('he', 26), ('is', 10), ('blond', 1), ('boy', 1), ('draw', 1), ('sheep.', 1), ('obliges,', 1),
('two', 3), ('become', 1), ('learns', 2), ('prince', 18), ('calls', 1), ('325', 1), ('but', 2), ('call', 1), ('B-612.', 1), ('t
ook', 1), ('of', 12), ('this', 2), ('planet,', 1), ('preventing', 1), ('making', 2), ('sure', 1), ('was', 2), ('never', 1), ('o
verrun', 1), ('trees.', 1), ('One', 1), ('mysterious', 1), ('rose', 8), ('sprouted', 1), ('love', 3), ('it.', 1), ('But', 1),
('when', 2), ('caught', 1), ('lie', 1), ('decided', 2), ('her', 3), ('lonely', 2), ('leave.', 1), ('Despite', 2), ('reconciliat
ion', 1), ('set', 2), ('out', 2), ('other', 1), ('planets', 2), ('cure', 1), ('loneliness.', 1), ('journeying,', 1), ('tells',
1), ('asteroids', 1), ('encounters', 2), ('world', 1), ('grown-ups.', 1), ('meets', 2), ('king,', 1), ('businessman,', 1), ('li
ve', 1), ('are', 4), ('overly', 1), ('consumed', 1), ('occupations.', 1), ('Such', 1), ('strange', 1), ('behavior', 1), ('bot
h', 1), ('perturbs', 1), ('prince.', 1), ('understand', 1), ('own', 1), ('everything.', 1), ('exception', 1), ('whose', 1), ('d
ogged', 1), ('admires,', 1), ('think', 1), ('anything', 1), ('useful.', 1), ('However,', 1), ('geographer', 1), ('flowers', 1),
('do', 2), ('last', 1), ('begins', 2), ('behind.', 1), ('At', 1), ('middle', 1), ('cannot', 2), ('speaks', 1), ('riddles', 1),
```

شکل۱۲- عملیات map reduce

- چه تفاوتی بین Actionهای take و collect وجود دارد؟ تابع take(n) در ابتدا یک پارتیشن را اسکن می‌کند و از نتایج آن برای تخمین تعداد پارتیشن‌های مورد نیاز استفاده می‌کند و n تعداد عناصری است که برمی‌گرداند.
تابع collect() همه عناصر RDD را به‌صورت یک لیست برمی‌گرداند. هردو تابع باید بر روی تعداد کمی داده استفاده شوند زیرا تمام داده‌ها در حافظه درایور بارگذاری می‌شوند.

- در صورتی که بتوانید توالی انجام هریک از عملیات‌ها در اسپارک که برای هر دستور انجام می‌دهد را برای هریک از دستورات بالا نمایش دهید و باتوجه به مفاهیم سوالات قبل آن را تصویر سازی کنید، نمره اضافه‌ای دریافت خواهید کرد. (به کمک ngrok و UI Spark)

در طی هر سوال نمایش داده شد.



شکل۱۳- توالی انجام عملیات

منابع

[۱] میریعقوب‌زاده، مجتبی (۱۴۰۰). درک مفاهیم در یک برنامه Apache Spark. https://virgool.io/@TabaMojj/apche-spark-efqg2jnqledl

[2] H. Miller. "Wide vs Narrow Dependencies." https://www.coursera.org/lecture/scala-spark-big-data/wide-vs-narrow-dependencies-shGAX (accessed March 1, 2022).

[3] D. Canton. "Wide and Narrow dependencies in Apache Spark." https://medium.com/@dvcanton/wide-and-narrow-dependencies-in-apache-spark-21acf2faf031 (accessed March 1, 2022).

[4] S. NAMBIAR. "Features of RDD & its Operations." https://aiwiz.com/rdd-features-operations/ (accessed March 1, 2022).

[5] bogotobogo."RDDOperations." https://www.bogotobogo.com/Hadoop/images/Spark_RDD/RDD_Operations.png (accessed March 1, 2022).

[6] SparkApache. "RDD Programming Guide." spark.apache.org. https://spark.apache.org/docs/2.2.1/rdd-programming-guide.html (accessed March 1, ۲۰۲۲).

[7] SparkByExamples. "PySpark RDD Tutorial | Learn with Examples." SparkByExamples.Com. https://sparkbyexamples.com/pyspark-rdd/ (accessed March 1, 2022).