به‌نام خداوند بخشنده و مهربان

تمرین سوم: زمان اجرا
درس: تحلیل سیستم داده‌های حجیم

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

نام و نام‌خانوادگی: سید عمید اسدالهی مجد
آدرس گیت: https://github.com/amidmajd/HW-3-hadoop-spark-comparison

عملیات Map و Reduce در Hadoop

دو فایل **hadoop_mapper.py** و **hadoop_reducer.py** در پوشه src موجود است که مربوط به این عملیات می‌باشند. فایل ورودی متنی توسط Hadoop از طریق ورودی استاندارد (Standard Input) به هر بخش از عمیلات داده می‌شود و همچنین خروجی هر بخش نیز از خروجی استاندارد (Standard Output) توسط Hadoop دریافت می‌شود. این عمیات، با استفاده از قابلیت streaming در هدوپ انجام می‌شود که خروجی mapper را پس از مرتب‌سازی بر اساس کلمات، به عنوان ورودی به reducer می‌دهد. در هدوپ با بکارگیری streaming به صورت خودکار یک مرحله میانی تحت عنوان shuffle & sort روی خروجی نهایی مرحله map انجام می‌شود و سپس به عنوان ورودی به مرحله reduce داده می‌شود.

در این برنامه، دو تابع **read_input** و **read_mapper_output** برای خواندن ورودی از تولیدکننده‌های (Generators) پایتون استفاده شد که به‌جای اینکه ابتدا تمام عناصر را دریافت کنند و سپس یک لیست یا آرایه را برگردانند، به محض آماده شدن هر عنصر، آن را برمی‌گردانند. با استفاده از این روش به‌جای خواندن معمولی آرایه، می‌توان در مصرف حافظه صرفه‌جویی کرد و نیاز نیست تمام داده‌ها در حافظه بارگذاری شوند و سپس برگردانده شوند. همچنین در زمان اجرا نیز تاثیر بسزایی دارد زیرا نیاز نیست منتظر باشیم حلقه کامل انجام شود و سپس خروجی برگردانده شود و سپس توسط حلقه دیگری عملیات مورد نظر روی داده‌ها انجام شود.

تابع **clean_word** وظیفه تمیزکردن هر کلمه را دارد. این تابع علائم سجاوندی (مانند نقطه، ویرگول، پرانتز و یا علامت نقل قول) را حذف می‌کند و کلمه را به حالت حروف کوچک (lowercase) تبدیل می‌کند.

```python
def clean_word(word):
    for char in string.punctuation + '0123456789':
        word = word.replace(char, "")
    return word.lower().strip().replace('\n', ' ')
```

توابع اصلی mapper و reducer

```python
def main(separator='\t'):
    # input comes from STDIN (standard input)
    data = read_input(sys.stdin)

    for words in data:
        for word in words:
            token = clean_word(word)
            if token:
                print(token, 1, sep=separator)
```

```python
def main(separator='\t'):
    current_word = None
    current_count = 0
    word = None

    # input comes from STDIN (standard input)
    data = read_mapper_output(sys.stdin, separator=separator)

    for word, count in data:
        try:
            count = int(count)
        except ValueError:
            continue

        if current_word == word:
            current_count += count
        else:
            if current_word:
                # write the result to STDOUT
                print(current_word, current_count, sep=separator)
            current_count = count
            current_word = word
```

عملیات Map و Reduce در Spark

فایل spark_map_reduce.py عمیات نگاشت-کاهش را در اسپارک پیاده‌سازی می‌کند. در این فایل تابع tokenize وظیفه شکستن متن ورودی خود به توکن‌ها را برعهده دارد. در این بخش از برنامه پس از خواندن فایل متنی و تبدیل آن به یک RDD، با استفاده از تابع map اسپارک، ابتدا هر جمله‌ی درون RDD را به توکن‌های مختلف آن تبدیل می‌کنیم و سپس با استفاده از تابع flatMap اسپارک، تمام این لیست‌های حاوی توکن را ادغام کرده و RDD را تبدیل به لیستی از توکن‌ها می‌نماییم. در انتها با استفاده از تابع map اسپارک، کلمات (توکن‌ها) را تبدیل به دوتایی‌های حاوی کلمه و عدد یک می‌نماییم (عمل نگاشت) و سپس با استفاده از تابع reduceByKey اسپارک، عملیات کاهش را پیاده‌سازی کرده و تعداد هر کلمه را بدست می‌آوریم.

```python
def tokenize(text):
    tokenized_text = []
    for word in text.split():
        token = clean_word(word)
        if token:
            tokenized_text.append(token)
    return tokenized_text
```

```python
def main():
    # get input file path as an argument from user
    try:
        INPUT_TEXT_PATH = sys.argv[1]
    except IndexError:
        print("1 argument missing: input file path")
        sys.exit()

    # setup a spark session
    conf = SparkConf().setAppName("homework")
    spark = SparkContext(conf=conf)

    # with SparkContext(conf=conf) as spark:

    # read and convert input file to RDD
    rdd = spark.textFile(INPUT_TEXT_PATH)

    # tokenizing the input text
    rdd = rdd.map(lambda line: tokenize(line)).flatMap(lambda x: x)

    # map reduce for counting words
    word_count_list = (
        rdd.map(lambda token: (token, 1)).reduceByKey(lambda count1, count2: count1 + count2)
    ).collect()

    print("10 example of words counts: \n", word_count_list[:10])
```

مقایسه زمان اجرا در هدوپ و اسپارک

برای اجرای برنامه شمارش کلمه با استفاده از هدوپ و بررسی زمان اجرا از دستور زیر استفاده شد:

time hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.10.1.jar
-files /home/amid_asadollahi/src/hadoop_mapper.py,/home/amid_asadollahi/src/hadoop_reducer.py
-mapper hadoop_mapper.py **-reducer** hadoop_reducer.py **-input** file10G.txt **-output** output10G

برای اجرای برنامه شمارش کلمه با استفاده از اسپارک و بررسی زمان اجرا از دستور زیر استفاده شد:

time spark-submit --master yarn **--num-executors** 100 **--deploy-mode client**
~/src/spark_map_reduce.py file10G.txt output10G.txt

برای اجرا در حالت cluster نیاز است تا فقط قسمت client از بخش deploy-mode به cluster تغییر داده‌شود. هم‌چنین قابل ذکر است که بخش num-executors در دستور اسپارک به‌منظور تقسیم فایل و اجرای موازی درنظر گرفته شده‌است و عدد ۱۰۰ به‌معنای تقسیم فایل ورودی به ۱۰۰ بخش و اجرای این بخش‌ها به صورت موازی (تا جای ممکن که سیستم توان داشته‌باشد) می‌باشد. این عمل در هدوپ به‌صورت خودکار انجام می‌شود و split نام دارد. بنابراین استفاده از num-executors برای مقایسه امری ضروری است.

این دستورات با استفاده از هریک از فایل‌های ۱، ۵، ۱۰ و ۱۲ گیگابایتی اجرا شدند. برای اجرای نگاشت-کاهش برروی این فایل‌ها، ابتدا هریک از این فایل‌ها با دستور hdfs fs -put FILE_NAME به فایل‌سیستم هدوپ منتقل شدند.

در هر مجموعه شکل ابتدا زمان اجرای برنامه نشان داده‌شده‌است و سپس برای نمایش خروجی آن دستور، ۱۰ کلمه با بیشترین تعداد تکرار پس از اجرای دستور نمایش داده‌شده‌است.

اجرای نگاشت-کاهش برروی فایل ۱ گیگابایتی

هدوپ :

```
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=1048604672
        File Output Format Counters
                Bytes Written=26292321
22/04/07 21:33:50 INFO streaming.StreamJob: Output directory: output1G

real    5m36.085s
user    0m6.638s
sys     0m0.343s
amid_asadollahi@MasterPC:~$ h -cat output1G/part-00000 | sort -k2 | tail
handheld        998
iq      998
shoes   9981
available       99872
photography     9988
ashamed 999
resigned        999
screened        999
skeptical       999
serving 9998
```

اسپارک:

```
amid_asadollahi@MasterPC:~$ time spark-submit --master yarn --num-executors 100
 --deploy-mode client src/spark_map_reduce.py file1G.txt output1G.txt
real    2m27.220s
user    0m20.187s
sys     0m1.896s
amid_asadollahi@MasterPC:~$
```

```
amid_asadollahi@MasterPC:~$ cat output1G.txt | sort -k2 | tail
handheld        998
iq      998
shoes   9981
available       99872
photography     9988
ashamed 999
resigned        999
screened        999
skeptical       999
serving 9998
```

اجرای نگاشت-کاهش برروی فایل ۵ گیگابایتی

هدوپ :

```
        File Input Format Counters
                Bytes Read=5243666432
        File Output Format Counters
                Bytes Written=40903569
22/04/07 22:14:41 INFO streaming.StreamJob: Output directory: output5G

real    24m32.832s
user    0m8.918s
sys     0m0.727s
```

```
amid_asadollahi@MasterPC:~$ h -cat output5G/part-00000 | sort -k2 | tail
grove   9990
parker  9990
gave    99913
infant  9994
chapel  9995
ho      9997
lace    9997
wrong   99975
drill   9998
theology        9998
```

اسپارک:

```
amid_asadollahi@MasterPC:~$ time spark-submit --master yarn --num-executors 100
 --deploy-mode client src/spark_map_reduce.py file5G.txt output5G.txt
real    7m29.049s
user    0m26.942s
sys     0m2.733s
```

```
amid_asadollahi@MasterPC:~$ cat output5G.txt | sort -k2 | tail
grove   9990
parker  9990
gave    99913
infant  9994
chapel  9995
ho      9997
lace    9997
wrong   99975
drill   9998
theology        9998
```

اجرای نگاشت-کاهش برروی فایل ۱۰ گیگابایتی

هدوپ :

```
        File Input Format Counters
                Bytes Read=10737741824
        File Output Format Counters
                Bytes Written=56099716
22/04/07 23:06:10 INFO streaming.StreamJob: Output directory: output10G

real    47m1.383s
user    0m11.202s
sys     0m1.175s
```

```
amid_asadollahi@MasterPC:~$ h -cat output10G/part-00000 | sort -k2 | tail
thêm    999
untouchable     999
pursued 9991
filtered        9993
protesters      9993
condemned       9995
gibson  9997
kilometres      9998
stripped        9998
repayment       9999
```

اسپارک :

```
amid_asadollahi@MasterPC:~$ time spark-submit --master yarn --num-executors 100
--deploy-mode client src/spark_map_reduce.py file10G.txt output10G.txt

real    13m44.184s
user    0m24.893s
sys     0m2.007s
```

```
amid_asadollahi@MasterPC:~$ cat output10G.txt | sort -k2 | tail
thêm    999
untouchable     999
pursued 9991
filtered 9993
protesters      9993
condemned       9995
gibson  9997
kilometres      9998
stripped 9998
repayment       9999
```

اجرای نگاشت-کاهش برروی فایل ۱۲ گیگابایتی

هدوپ :

```
        File Input Format Counters
                Bytes Read=12449907892
        File Output Format Counters
                Bytes Written=56221250
22/04/08 00:01:19 INFO streaming.StreamJob: Output directory: output12G

real    54m6.752s
user    0m11.479s
sys     0m1.188s
```

```
amid_asadollahi@MasterPC:~$ h -cat output12G/part-00000 | sort -k2 | tail
liquids 9990
axle    9991
perfectly       99925
innocence       9994
drugs   99942
cupcakes        9995
freak   9995
represent       99956
weighted        9997
catastrophic    9999
```

اسپارک:

```
amid_asadollahi@MasterPC:~$ time spark-submit --master yarn --num-executors 100
 --deploy-mode client src/spark_map_reduce.py file12G.txt output12G.txt

real    16m8.636s
user    0m25.750s
sys     0m2.137s
```

```
amid_asadollahi@MasterPC:~$ cat output12G.txt | sort -u -k2 | tail
cheaters 999
liquids 9990
axle    9991
perfectly       99925
innocence       9994
drugs   99942
freak   9995
represent       99956
weighted 9997
catastrophic    9999
```

مقایسه اجرای دستورات اسپارک در دو حالت Client و Cluster

در حالت Client پس از اجرای دستور (Submit)، برنامه برروی همان ماشین اجرا می‌شود و به ماشین دیگری تخصیص داده نمی‌شود؛ بنابراین ماشین اجرا کننده (Client) باید تا زمان پایان اجرا روشن باشد و منابع خود را به اجرا اختصاص دهد. اما در حالت Cluster پس از اجرای دستور، برنامه توسط مدیر منابع اسپارک (Spark Resource Manager) به یکی از ماشین‌های Slave تخصیص داده می‌شود و دیگر نیازی نیست تا سیستم فراخواننده دستور روشن بماند و یا منابع خود را به اجرای دستور اختصاص دهد و فقط پس از پایان اجرا، به سیستم فراخواننده اطلاع داده می‌شود که اجرا خاتمه یافته‌است. درواقع در حالت Cluster مدریت نحوه اجرای دستورات را به مدیر منابع اسپارک داده می‌شود تا دستور را برروی سیستم دارای بار کاری کمتر اجرا کند و اینگونه اجراها را در تمام سیستم‌ها پخش نماید.

در ادامه دو نمونه از این تفاوت نمایش داده شده‌است. همانطور که مشاهده می‌شود در زمان اجرا تفاوتی ایجاد نشده است (به دلیل مشابهت تقریبی سیستم‌ها). تفاوت اصلی در محل اجرای دستور است؛ در حالت Client دستور برروی همان سیستم اجرا می‌شود ولی در حالت Cluster اجرای دستور به یکی از سیستم‌های Slave واگذار شده‌است.

اجرای دستور اسپارک برروی فایل ۱ گیگابایتی در حالت Cluster

```
amid_asadollahi@MasterPC:~$ time spark-submit --master yarn --num-executors 100
 --deploy-mode client src/spark_map_reduce.py file1G.txt output1G.txt
real    2m27.220s
user    0m20.187s
sys     0m1.896s
amid_asadollahi@MasterPC:~$ time spark-submit --master yarn --num-executors 100
 --deploy-mode cluster src/spark_map_reduce.py file1G.txt output1G.txt

real    2m27.030s
user    0m9.756s
sys     0m1.109s
```

```
Application Report :
        Application-Id : application_1649179500517_0262
        Application-Name : homework
        Application-Type : SPARK
        User : amid_asadollahi
        Queue : default
        Application Priority : 0
        Start-Time : 1649390582378
        Finish-Time : 1649390721372
        Progress : 100%
        State : FINISHED
        Final-State : SUCCEEDED
        Tracking-URL : N/A
        RPC Port : -1
        AM Host : 172.16.20.12
        Aggregate Resource Allocation : 3425355 MB-seconds, 1737 vcore-seconds
        Aggregate Resource Preempted : 0 MB-seconds, 0 vcore-seconds
        Log Aggregation Status : DISABLED
        Diagnostics :
        Unmanaged Application : false
        Application Node Label Expression : <Not set>
        AM container Node Label Expression : <DEFAULT_PARTITION>
        TimeoutType : LIFETIME  ExpiryTime : UNLIMITED  RemainingTime : -1seconds
```

```
Application Report :
        Application-Id : application_1649179500517_0263
        Application-Name : spark_map_reduce.py
        Application-Type : SPARK
        User : amid_asadollahi
        Queue : default
        Application Priority : 0
        Start-Time : 1649390953316
        Finish-Time : 1649391083227
        Progress : 100%
        State : FINISHED
        Final-State : SUCCEEDED
        Tracking-URL : N/A
        RPC Port : 34375
        AM Host : slave2
        Aggregate Resource Allocation : 3267464 MB-seconds, 1587 vcore-seconds
        Aggregate Resource Preempted : 0 MB-seconds, 0 vcore-seconds
        Log Aggregation Status : DISABLED
        Diagnostics :
        Unmanaged Application : false
        Application Node Label Expression : <Not set>
        AM container Node Label Expression : <DEFAULT_PARTITION>
        TimeoutType : LIFETIME  ExpiryTime : UNLIMITED  RemainingTime : -1seconds
```

اجرای دستور اسپارک برروی فایل ۱۰ گیگابایتی در حالت Cluster

```
amid_asadollahi@MasterPC:~$ time spark-submit --master yarn --num-executors 100
--deploy-mode client src/spark_map_reduce.py file10G.txt output10G.txt

real    13m44.184s
user    0m24.893s
sys     0m2.007s
amid_asadollahi@MasterPC:~$ time spark-submit --master yarn --num-executors 100 -
-deploy-mode cluster src/spark_map_reduce.py file10G.txt output10G-cluster.tcxt

real    13m45.649s
user    0m9.937s
sys     0m1.067s
```

```
Application Report :
        Application-Id : application_1649179500517_0277
        Application-Name : homework
        Application-Type : SPARK
        User : amid_asadollahi
        Queue : default
        Application Priority : 0
        Start-Time : 1649396152847
        Finish-Time : 1649396955954
        Progress : 100%
        State : FINISHED
        Final-State : SUCCEEDED
        Tracking-URL : N/A
        RPC Port : -1
        AM Host : 172.16.20.11
        Aggregate Resource Allocation : 20384741 MB-seconds, 10347 vcore-seconds
        Aggregate Resource Preempted : 0 MB-seconds, 0 vcore-seconds
        Log Aggregation Status : DISABLED
        Diagnostics :
        Unmanaged Application : false
        Application Node Label Expression : <Not set>
        AM container Node Label Expression : <DEFAULT_PARTITION>
        TimeoutType : LIFETIME  ExpiryTime : UNLIMITED  RemainingTime : -1seconds
```

```
Application Report :
        Application-Id : application_1649179500517_0278
        Application-Name : spark_map_reduce.py
        Application-Type : SPARK
        User : amid_asadollahi
        Queue : default
        Application Priority : 0
        Start-Time : 1649398527754
        Finish-Time : 1649399336550
        Progress : 100%
        State : FINISHED
        Final-State : SUCCEEDED
        Tracking-URL : N/A
        RPC Port : 37687
        AM Host : slave2
        Aggregate Resource Allocation : 21327619 MB-seconds, 10407 vcore-seconds
        Aggregate Resource Preempted : 0 MB-seconds, 0 vcore-seconds
        Log Aggregation Status : DISABLED
        Diagnostics :
        Unmanaged Application : false
        Application Node Label Expression : <Not set>
        AM container Node Label Expression : <DEFAULT_PARTITION>
        TimeoutType : LIFETIME  ExpiryTime : UNLIMITED  RemainingTime : -1seconds
```