گزارش کامل راه‌اندازی کلاستر Hadoop و اجرای پروژه MapReduce با Python

مقدمه

در این پروژه، ما یک کلاستر سه‌نودی Hadoop شامل:

- یک **نود Master** (iut1)
- دو **نود Worker** (iut2, iut3)

راه اندازی کردیم که توانایی اجرای Job های MapReduce به صورت توزیع‌شده روی داده‌های بزرگ را دارا است.

مراحل انجام پروژه

1-اتصال به سرورها و نصب پیش‌نیازها

- اتصال به سرورها با ssh و VPN دانشگاه
- نصب Java با دستور: `apt install openjdk-8-jdk -y`
- دانلود Hadoop: `wget` https://archive.apache.org/dist/hadoop/core/hadoop-3.3.4/hadoop-3.3.4.tar.gz
- استخراج و انتقال به `/usr/local/Hadoop`

2-تنظیم متغیرهای محیطی و فایل‌های پیکربندی

- افزودن مسیرهای Hadoop و Java به فایل: `/etc/environment`
- تنظیم فایل‌های زیر در دایرکتوری:
  `/usr/local/hadoop/etc/hadoop/`

  که شامل: `core-site.xml` و `hdfs-site.xml` و `mapred-site.xml` و `yarn-site.xml`

- فایل `workers` برای مشخص‌کردن نودهای `worker`

3-راه‌اندازی HDFS و YARN

- فرمت کردن NameNode : `hdfs namenode -format`
- اجرای دیمون‌های HDFS : `start-dfs.sh`
- اجرای YARN : `start-yarn.sh`
- بررسی اجرای صحیح با `jps`

4-حل چالش های مهم در مسیر

چالش	راه حل
عدم اتصال نودها به هم	تنظیم دقیق فایل /etc/hosts و حذف آدرس‌های پیش‌فرض لوکال
ارور JAVA_HOME is not set	افزودن به hadoop-env.sh
ارور permission denied یا publickey	فعال‌سازی دسترسی SSH و اشتراک فایل‌ها
ارور Could not find MRAppMaster	تعریف HADOOP_MAPRED_HOME در mapred-site.xml
ارور auxService:mapreduce_shuffle does not exist	افزودن mapreduce_shuffle در yarn-site.xml
ارور Broken pipe یا mapper.py: not found	استفاده از files- برای ارسال فایل‌ها به YARN و مقاوم‌سازی کد

5-پاک‌سازی و آماده‌سازی دیتاست

- فایل اصلی Online_Retail.xlsx را به CSV تبدیل کردیم.
- با Python فایل را تمیز کردم:
- حذف ردیف‌های ناقص
- تبدیل encoding به UTF-8
- مقداردهی به فیلدهای خالی مثل:
  - Country = Unknown
  - CustomerID = 0
- ارسال فایل به HDFS: **`hdfs dfs -put Online_Retail_Clean.csv /input_clean/`**

6-فایل های mapper.py و reducer.py

- Mapper.py:

```
#!/usr/bin/env python3
import sys, csv
reader = csv.reader(sys.stdin)
next(reader, None)  # Skip header
for row in reader:
    if len(row) >= 8 and
row[7].strip():
        print(f"{row[7].strip()}\t1")
```

این کد نقش Mapper در فرآیند MapReduce را ایفا می‌کند. درMapReduce ، فایل ورودی به صورت سطر به سطر به برنامه داده می‌شود. در اینجا ما از ماژول csv برای خواندن داده‌ها به عنوان فایل CSV استفاده کرده‌ایم. ابتدا با ()next سطر اولHeader حذف می‌شود. سپس برای هر سطر از فایل:

- بررسی می‌شود که آیا تعداد ستون‌ها حداقل ۸ عدد هست یا خیر (چون ستون Country ستون هشتم است)
- اگر فیلد Country خالی نباشد، نام کشور را استخراج کرده و همراه با عدد ۱ به خروجی ارسال می‌کند.

درMapReduce ، خروجی Mapper به صورت جفت کلید-مقدار (key\tvalue) است. در اینجا کلید نام کشور و مقدار عدد ۱ است. یعنی هر بار که یک خرید از یک کشور دیده می‌شود، یک عدد ۱ برای آن کشور چاپ می‌شود.

- Reducer.py:

```
#!/usr/bin/env python3
import sys
current_country = None
count = 0
for line in sys.stdin:
    country, c = line.strip().split('\t')
    c = int(c)
    if current_country == country:
        count += c
    else:
        if current_country:

print(f"{current_country}\t{count}")
        current_country = country
        count = c
if current_country:
    print(f"{current_country}\t{count}")
```

این کد نقش Reducer را بر عهده دارد .خروجی Mapper که شامل Country\t1 است، به صورت مرتب شده (Sorted by key) وارد Reducer می‌شود. در این کد:

- ابتدا با استفاده از یک متغیر current_country بررسی می‌شود که آیا کشور فعلی با کشور قبلی یکسان است یا خیر.
- اگر یکسان باشد، مقدار شمارنده (count) افزایش می‌یابد.
- اگر تغییر کند (یعنی کشور جدیدی شروع شده باشد)، مقدار قبلی چاپ شده و شمارنده برای کشور جدید مقداردهی اولیه می‌شود.
- در انتها، آخرین کشور نیز چاپ می‌شود.

در واقع این کد، تعداد کل تراکنش‌ها یا سطرهای مربوط به هر کشور را می‌شمارد و در خروجی به صورت Country\tTotalCount نمایش می‌دهد.

7-اجرای نهایی MapReduce

```
hadoop jar/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-*.jar\
- files mapper.py,reducer.py\
- input /input_clean/Online_Retail_Clean.csv\
- output /output_country_count\
- mapper mapper.py\
- reducer reducer.py
```

8-خروجی نهایی برای قسمت 4 تکلیف

```
root@iut1:/home/iut1# hdfs dfs -cat /output_country_count/part-00000
Australia       1259
Austria 401
Bahrain 19
Belgium 2069
Brazil  32
Canada  151
Channel Islands 758
Cyprus  622
Czech Republic  30
Denmark 389
EIRE    8196
European Community      61
Finland 695
France  8557
Germany 9495
Greece  146
Hong Kong       288
Iceland 182
Israel  297
Italy   803
Japan   358
Lebanon 45
Lithuania       35
Malta   127
Netherlands     2371
Norway  1086
Poland  341
Portugal        1519
RSA     58
Saudi Arabia    10
Singapore       229
Spain   2533
Sweden  462
Switzerland     2002
USA     291
United Arab Emirates    68
United Kingdom  495477
Unspecified     446
```

قسمت 4 تکلیف (مقادیر آماری برای ستون Quantity)

- Quantity_mapper.py

```python
#!/usr/bin/env python3
import sys
import csv

reader = csv.reader(sys.stdin)
next(reader, None)  # Skip header
for row in reader:
    if len(row) >= 8:
        country = row[7].strip()
        quantity = row[3].strip()
        if country and quantity:
            try:
                quantity = int(float(quantity))
                print(f"{country}\t{quantity}")
            except:
                continue
```

این Mapper برای محاسبه آمار Quantity به تفکیک کشور طراحی شده. در ابتدا داده‌ها از ورودی استاندارد (sys.stdin) به صورت CSV خوانده می‌شوند. سطر اول (سربرگ) با next رد می‌شود. سپس برای هر سطر بررسی می‌شود که آیا تعداد ستون‌ها حداقل ۸ هست و فیلدهای Country و Quantity مقدار دارند یا نه. اگر مقدار موجود بود، مقدار Quantity به عدد صحیح تبدیل می‌شود و با کلید Country به صورت Country \t Quantity چاپ می‌شود. این خروجی به Reducer داده می‌شود تا آمار نهایی را محاسبه کند. اگر مقدار عددی نباشد یا تبدیل با خطا مواجه شود، آن سطر نادیده گرفته می‌شود.

- Quantity_reducer.py

```
#!/usr/bin/env python3
import sys
current_country = None
count = 0
total = 0
min_qty = None
max_qty = None

for line in sys.stdin:
    try:
        country, qty = line.strip().split('\t')
        qty = int(qty)
        if current_country == country:
            total += qty
            count += 1
            min_qty = min(min_qty, qty)
            max_qty = max(max_qty, qty)
        else:
            if current_country:
                avg = total / count
                print(f"{current_country}\tMin: {min_qty}, Max: {max_qty}, Avg: {avg:.2f}")
            current_country = country
            total = qty
            count = 1
            min_qty = qty
            max_qty = qty
    except:
        continue

if current_country:
    avg = total / count
    print(f"{current_country}\tMin: {min_qty}, Max: {max_qty}, Avg: {avg:.2f}")
```

این Reducer برای هر کشور، کمترین، بیشترین و میانگین Quantity را محاسبه می‌کند. داده‌ها به صورت مرتب‌شده بر اساس کشور وارد می‌شوند. اگر کشور فعلی با کشور قبلی یکسان باشد، مقدار Quantity به مجموع و تعداد رکوردها افزوده می‌شود و بیشینه و کمینه نیز به‌روزرسانی می‌شوند. اگر کشور تغییر کند، آماره‌های کشور قبلی چاپ شده و متغیرها برای کشور جدید مقداردهی اولیه می‌شوند. در پایان نیز آمار کشور آخر چاپ می‌شود. این کد کاملاً مقاوم در برابر خطاهای ورودی است و خروجی را به شکل ساخت‌یافته Country \t Min: x, Max: y, Avg: z نمایش می‌دهد.

```
root@iut1:/home/iut1# hdfs dfs -cat /output_quantity_stats/part-00000
Australia       Min: -120, Max: 1152, Avg: 66.44
Austria Min: -48, Max: 288, Avg: 12.04
Bahrain Min: -54, Max: 96, Avg: 13.68
Belgium Min: -12, Max: 272, Avg: 11.19
Brazil  Min: 2, Max: 24, Avg: 11.12
Canada  Min: 1, Max: 504, Avg: 18.30
Channel Islands Min: -2, Max: 407, Avg: 12.51
Cyprus  Min: -33, Max: 288, Avg: 10.16
Czech Republic  Min: -24, Max: 72, Avg: 19.73
Denmark Min: -25, Max: 256, Avg: 21.05
EIRE    Min: -288, Max: 1440, Avg: 17.40
European Community      Min: -2, Max: 24, Avg: 8.15
Finland Min: -27, Max: 144, Avg: 15.35
France  Min: -250, Max: 912, Avg: 12.91
Germany Min: -288, Max: 600, Avg: 12.37
Greece  Min: -1, Max: 48, Avg: 10.66
Hong Kong       Min: -1, Max: 144, Avg: 16.56
Iceland Min: 2, Max: 240, Avg: 13.51
Israel  Min: -32, Max: 100, Avg: 14.66
Italy   Min: -12, Max: 200, Avg: 9.96
Japan   Min: -624, Max: 2040, Avg: 70.44
Lebanon Min: 2, Max: 24, Avg: 8.58
Lithuania       Min: 6, Max: 48, Avg: 18.63
Malta   Min: -4, Max: 48, Avg: 7.43
Netherlands     Min: -480, Max: 2400, Avg: 84.41
Norway  Min: -12, Max: 240, Avg: 17.72
Poland  Min: -6, Max: 72, Avg: 10.71
Portugal        Min: -12, Max: 120, Avg: 10.65
RSA     Min: 1, Max: 12, Avg: 6.07
Saudi Arabia    Min: -5, Max: 12, Avg: 7.50
Singapore       Min: -1, Max: 288, Avg: 22.86
Spain   Min: -288, Max: 360, Avg: 10.59
Sweden  Min: -240, Max: 768, Avg: 77.14
Switzerland     Min: -120, Max: 288, Avg: 15.15
USA     Min: -36, Max: 72, Avg: 3.55
United Arab Emirates    Min: 1, Max: 72, Avg: 14.44
United Kingdom  Min: -80995, Max: 80995, Avg: 8.61
Unspecified     Min: 1, Max: 48, Avg: 7.40
```

زمان اجرای این job ها به اینصورت شد:

real 0m19.761s

user 0m3.224s

sys 0m0.266s

مدت زمان اجرای job برای تحلیل quantity تقریبا 19.7 ثانیه طول کشید.

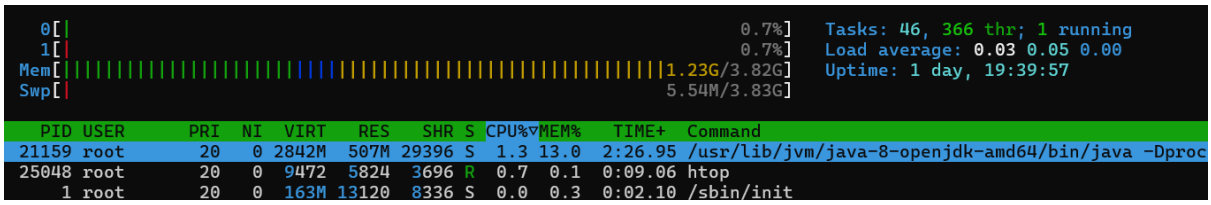
تحلیل با htop:

قبل از اجرا

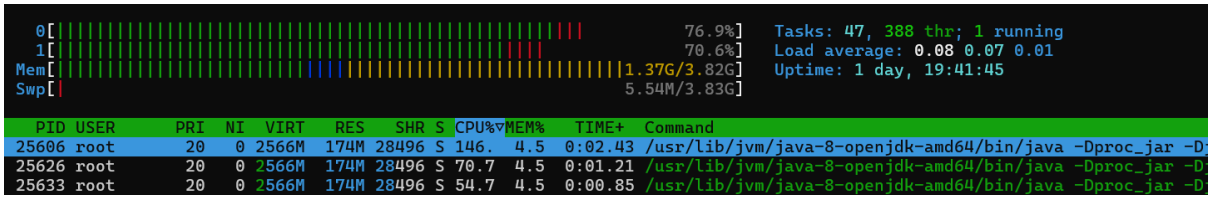
حین اجرا

حالا دیتاست را 10 برابر کردم و به hdfs فرستادم:

قبل از اجرا



حین اجرا



زمان اجرای job:

real 0m32.995s

user 0m3.236s

sys 0m0.227s

که تقریبا 33 ثانیه اجرای آن طول کشید برای مقادیر آماری quantity

خروجی job و محاسبه مقادیر خواسته شده:

```
root@iut1:/home/iut1# hdfs dfs -cat /output_quantity_large/part-00000
Australia       Min: 1, Max: 10956, Avg: 10.07
Austria Min: 1, Max: 4533, Avg: 10.02
Bahrain Min: 1, Max: 4596, Avg: 10.00
Belgium Min: 1, Max: 6125, Avg: 10.01
Brazil  Min: 1, Max: 68194, Avg: 10.57
Canada  Min: 1, Max: 3562, Avg: 10.05
Channel Islands Min: 1, Max: 4939, Avg: 9.95
Cyprus  Min: 1, Max: 11481, Avg: 10.05
Czech Republic  Min: 1, Max: 3798, Avg: 9.93
Denmark Min: 1, Max: 61628, Avg: 10.67
EIRE    Min: 1, Max: 5920, Avg: 10.09
European Community      Min: 1, Max: 82400, Avg: 10.61
Finland Min: 1, Max: 11693, Avg: 10.15
France  Min: 1, Max: 6007, Avg: 9.91
Germany Min: 1, Max: 3533, Avg: 9.79
Greece  Min: 1, Max: 4639, Avg: 10.03
Hong Kong       Min: 1, Max: 2955, Avg: 10.05
Iceland Min: 1, Max: 3167, Avg: 9.95
Israel  Min: 1, Max: 69495, Avg: 10.56
Italy   Min: 1, Max: 63868, Avg: 10.39
Japan   Min: 1, Max: 3147, Avg: 10.16
Lebanon Min: 1, Max: 14860, Avg: 10.11
Lithuania       Min: 1, Max: 12592, Avg: 10.07
Malta   Min: 1, Max: 84758, Avg: 10.65
Netherlands     Min: 1, Max: 86249, Avg: 10.78
Norway  Min: 1, Max: 4319, Avg: 9.87
Poland  Min: 1, Max: 6023, Avg: 9.84
Portugal        Min: 1, Max: 72387, Avg: 10.65
RSA     Min: 1, Max: 68091, Avg: 10.61
Saudi Arabia    Min: 1, Max: 3201, Avg: 10.12
Singapore       Min: 1, Max: 91936, Avg: 10.68
Spain   Min: 1, Max: 85840, Avg: 11.63
Sweden  Min: 1, Max: 5358, Avg: 9.88
Switzerland     Min: 1, Max: 5597, Avg: 9.94
USA     Min: 1, Max: 73854, Avg: 10.43
United Arab Emirates    Min: 1, Max: 87282, Avg: 10.48
United Kingdom  Min: 1, Max: 68701, Avg: 10.41
Unspecified     Min: 1, Max: 94672, Avg: 11.80
```