Infraestruturas Avançadas para Ciência dos Dados

José Pedro Ferreira Dinis e Silva nº2020237724

Diogo Morais Fontes nº2023187110

Duarte Nuno Vieira Almeida Patrício Alves nº2023138348

Assignment 6:

Inicialmente, em dois terminais separados corremos os seguintes códigos para corre o Kafka no nosso computador:

- $KAFKA_HOME/bin/zookeeper-server-start.sh $KAFKA_HOME/config/zookeeper.properties
- $KAFKA_HOME/bin/kafka-server-start.sh $KAFKA_HOME/config/server.properties

Estes dois códigos servem para as tarefas que fazemos de seguida.

Task #1 - Producer and Consumer with Topic 1 Partition:

Criámos um ficheiro consumer.py e um producer.py. O producer gera o id e a temperatura enquanto que o consumer obtém as temperaturas:

$KAFKA_HOME/bin/kafka-topics.sh --create --topic task1-topic --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

corre producer.py e consumer.py em dois terminais diferentes
python3 producer.py
python3 consumer.py

as temperaturas geradas pelo producer.py são enviadas para o consumer.py

Task #2 - Multi-Partition Topic with Multiple Consumers and Consumer Groups:

Criámos um ficheiro consumer.py e um producer.py. Estes foram os códigos utilizados e as conclusões retiradas:

$KAFKA_HOME/bin/kafka-topics.sh --create --topic task2-topic --bootstrap-server localhost:9092 --partitions 3 --replication-factor 1

python3 producer.py

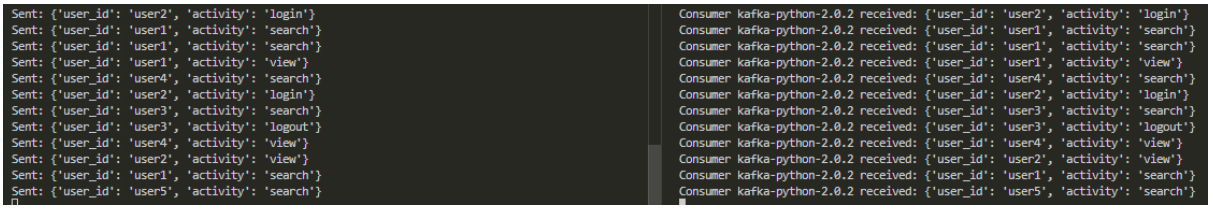
No código do consumer,py adicionamos o grupo de consumidores "activity group" que irá ter n consumidores em que n é o número de terminais abertos com o script a correr: pyhton3 consumer.py

Com um consumer, ele recebe todas as mensagens do producer

Com dois consumer, dividem a carga enviada pela producer de forma "aleatória" (primeiro a ficar livre recebe)

Com três consumer, dividem a carga enviada pela producer de forma "aleatória"

Com quatro consumer, três consumer dividem a carga enviada pela producer de forma "aleatória", e o último a conectar fica sem resposta. Uma vez que o producer apenas tem 3 partições



Task #3 - Multi-Topic producer and consumer with consumer groups:

Criámos um ficheiro activity_consumer.py, multi_producer.py e um purchase_consumer.py. Estes foram os códigos utilizados e as conclusões retiradas:

$KAFKA_HOME/bin/kafka-topics.sh --create --topic purchase-topic --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1
topico criado para compras

$KAFKA_HOME/bin/kafka-topics.sh --create --topic user-activity-topic --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1
topico criado para atividades

python3 multi_producer.py -> para começar a correr quem vai gerar e enviar as mensagens

python3 purchase_consumer.py -> para receber apenas as compras e incrementar o valor total

python3 activity_consumer.py -> para receber apenas atividades e incrementar para cada user o seu valor total

Assignment 7:

Task #1 - Average Number of Friends by Age:

Criámos um ficheiro mapper.py e reducer.py. Os códigos utilizados foram os seguintes:

Upload the File to HDFS:

- hadoop fs -mkdir -p /home/hadoop/input
- hadoop fs -put fakefriends.csv /home/hadoop/input/
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar\
  -input /home/hadoop/input/fakefriends.csv \
  -output /home/hadoop/output/friends_avg_by_age \
  -mapper /home/hadoop/scripts/mapper.py \
  -reducer /home/hadoop/scripts/reducer.py \
  -file /home/hadoop/scripts/mapper.py \
  -file /home/hadoop/scripts/reducer.py
- hadoop fs -get /home/hadoop/output/friends_avg_by_age/part-00000 ./friends_avg_by_age_output.txt

```
hadoop@dudas-laptop:~/hadoop/output/lab7$ cat friends_avg_by_age_output.txt
18      343.375
19      213.27272727272728
20      165.0
21      350.875
22      206.42857142857142
23      246.3
24      233.8
25      197.45454545454547
26      242.05882352941177
27      228.125
28      209.1
29      215.91666666666666
30      235.8181818181818
31      267.25
32      207.9090909090909
33      325.3333333333333
34      245.5
35      211.625
36      246.6
37      249.33333333333334
38      193.53333333333333
39      169.28571428571428
40      250.8235294117647
```

Task #2 - Minimum Temperature Per Capital:

Criámos um ficheiro mapper.py e reducer.py. Os códigos utilizados foram os seguintes:

Upload the File to HDFS:

- hadoop fs -put 1800.csv /home/hadoop/input/
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar\
  -input /home/hadoop/input/1800.csv \
  -output /home/hadoop/output/min_temperature_per_capital \
  -mapper /home/hadoop/scripts/mapper.py \
  -reducer /home/hadoop/scripts/reducer.py \
  -file /home/hadoop/scripts/mapper.py \
  -file /home/hadoop/scripts/reducer.py
- hadoop fs -get /home/hadoop/output/min_temperature_per_capital/part-00000 ./min_temperature_per_capital_output.txt

```
hadoop@dudas-laptop:~/hadoop/output/lab7$ cat min_temperature_per_capital_output.txt
EZE00100082     -135
ITE00100554     -148
```

Task #3 - Sort the Word Frequency in a Book:

Criámos um ficheiro mapper.py e reducer.py. Os códigos utilizados foram os seguintes:

Upload the File to HDFS:

- hadoop fs -put Book.txt /home/hadoop/input/
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar\
  -input /user/hadoop/input/Book.txt \
  -output /user/hadoop/output/word_frequency_sorted \
  -mapper /home/hadoop/scripts/mapper.py \
  -reducer /home/hadoop/scripts/reducer.py \
  -file /home/hadoop/scripts/mapper.py \
  -file /home/hadoop/scripts/reducer.py
- hadoop fs -get /user/hadoop/output/word_frequency_sorted/part-00000 ./word_frequency_sorted.txt

```
hadoop@dudas-laptop:~/hadoop/output/lab7$ cat word_frequency_sorted.txt
{'the': 230, 'and': 161, 'a': 108, 'to': 91, 'he': 89, 'his': 84, 'she': 79, 'of': 78, 'was': 71, 'in': 57
as': 26, 'him': 24, 'mrs': 23, 'for': 22, 'up': 22, 'but': 22, 'sir': 20, 'so': 19, 'you': 16, 'on': 16, '
, 'be': 13, 'then': 13, 'my': 13, 'man': 12, 'by': 12, 'there': 12, 'an': 12, 'visitor': 12, 'have': 12, '
```

Task #4 - Sort theTotal Amount Spent by Costumer:

Criámos um ficheiro mapper.py e reducer.py. Os códigos utilizados foram os seguintes:

Upload the File to HDFS:

- hadoop fs -put customer-orders.csv /home/hadoop/input/
- hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar\
  -input /user/hadoop/input/customer-orders.csv \
  -output /user/hadoop/output/total_amount_by_customer \
  -mapper /home/hadoop/scripts/mapper.py \
  -reducer /home/hadoop/scripts/reducer.py \
  -file /home/hadoop/scripts/mapper.py \

-file /home/hadoop/scripts/reducer.py

- hadoop fs -get /user/hadoop/output/total_amount_by_customer/part-00000 ./total_amount_by_customer.txt

```
hadoop@dudas-laptop:~/hadoop/output/lab7$ cat total_amount_by_customer.txt
68      6375.449999999997
73      6206.200000000001
39      6193.110000000001
54      6065.39
71      5995.659999999997
2       5994.590000000002
97      5977.1900000000005
46      5963.109999999998
42      5696.839999999998
59      5642.889999999999
41      5637.619999999999
```

Assignment 8:

Task #1 - Minimum Temperature Per Capital:

Criámos um ficheiro min_temperature.py. Os códigos utilizados foram os seguintes:

- hdfs dfs -put /home/user/data/1800.csv /user/hadoop/input/
- spark-submit --master local min_temperature.py
- hdfs dfs -cat /user/hadoop/output/min_temperature_per_city/part-*

```
hadoop@dudas-laptop:~/hadoop/output/lab7$ hdfs dfs -cat /user/hadoop/output/min_temperature_per_city/part-*
ITE00100554,-148
EZE00100082,-135
```

Task #2_3 - Obtain/Sort the word Frequency in a Book:

Criámos um ficheiro sort_words.py. Os códigos utilizados foram os seguintes:

- hdfs dfs -put /home/user/data/Book.txt /user/hadoop/input/
- spark-submit --master local sort_words.py
- hdfs dfs -cat /user/hadoop/output/sorted_word_counts/part-*

```
hadoop@dudas-laptop:~/hadoop/output/lab7$ hdfs dfs -cat /user/hadoop/output/sorted_word_counts/part-*
('the', 230)
('and', 161)
('a', 108)
('to', 91)
('he', 89)
('his', 84)
('she', 79)
('of', 78)
('was', 71)
('in', 57)
('said', 52)
('that', 49)
('her', 45)
('at', 44)
('it', 42)
```

Task #4_5 - Obtain/Sort the Total Amount Spent by Costumer:

Criámos um ficheiro sort_total_costumer.py. Os códigos utilizados foram os seguintes:

- hdfs dfs -put customer-orders.csv /user/hadoop/input/

- spark-submit --master local sort_total_customers.py
- hdfs dfs -cat /user/hadoop/output/sorted_customer_spending/part-*

```
hadoop@dudas-laptop:~$ hdfs dfs -cat /user/hadoop/output/sorted_customer_spending/part-*
(68, 6375.449999999997)
(73, 6206.199999999999)
(39, 6193.109999999999)
(54, 6065.389999999999)
(71, 5995.660000000003)
(2, 5994.59)
(97, 5977.189999999995)
(46, 5963.109999999999)
(42, 5696.840000000003)
(59, 5642.89)
(41, 5637.62)
```

Task #6_7 - Most/Least Popular Superhero:

Criámos um ficheiro marvel_superheroes.py. Os códigos utilizados foram os seguintes:

- hdfs dfs -put Marvel+Graph /user/hadoop/input/
- hdfs dfs -put Marvel+Names /user/hadoop/input/
- spark-submit --master local marvel_superheroes.py
- hdfs dfs -cat /user/hadoop/output/superhero_popularity/part-*

```
hadoop@dudas-laptop:~$ hdfs dfs -cat /user/hadoop/output/superhero_popularity/part-*
Most popular superheroes (with 1937 occurrences):
CAPTAIN AMERICA

Least popular superheroes (with 1 occurrences):
RED WOLF II
ZANTOR
DEATHCHARGE
BLARE/
RANDAK
MARVEL BOY II/MARTIN
SEA LEOPARD
GERVASE, LADY ALYSSA
GIURESCU, RADU
JOHNSON, LYNDON BAIN
CALLAHAN, DANNY
SHARKSKIN
CLUMSY FOULUP
BERSERKER II
RUNE
MARVEL BOY/MARTIN BU
FENRIS
LUNATIK II
KULL
```

Assignment 9:

Task #1 - Minimum Temperature Per Capital:

Criámos um ficheiro min_temperature.py. Os códigos utilizados foram os seguintes:

- spark-submit --master local min_temperature.py
- hdfs dfs -cat /user/hadoop/output/min_temperature_per_city9/part-*

```
hadoop@dudas-laptop:~$ hdfs dfs -cat /user/hadoop/output/min_temperature_per_city9/part-*
city,min_temperature
ITE00100554,-148
EZE00100082,-135
```

Task #2_3 - Obtain/Sort the word Frequency in a Book:

Criámos um ficheiro sort_words.py. Os códigos utilizados foram os seguintes:

- spark-submit --master local sort_words.py
- hdfs dfs -cat /user/hadoop/output/sorted_word_counts9/part-*

```
hadoop@dudas-laptop:~$ hdfs dfs -cat /user/hadoop/output/sorted_word_counts9/part-*
word,count
the,230
and,161
a,108
to,91
he,89
his,84
she,79
of,78
was,71
in,57
said,52
that,49
her,45
at,44
it,42
hall,39
had,38
with,38
i,30
```

Task #4_5 - Obtain/Sort the Total Amount Spent by Costumer:

Criámos um ficheiro sort_total_costumers.py. Os códigos utilizados foram os seguintes:

- spark-submit --master local sort_total_customers.py
- hdfs dfs -cat /user/hadoop/output/sorted_customer_spending9/part-*

```
hadoop@dudas-laptop:~$ hdfs dfs -cat /user/hadoop/output/sorted_customer_spending9/part-*
CustomerID,TotalSpending
68,6375.449999999997
73,6206.199999999999
39,6193.109999999999
54,6065.389999999999
71,5995.660000000003
2,5994.59
97,5977.189999999995
46,5963.109999999999
42,5696.840000000003
59,5642.89
41,5637.62
```

Task #6_7 - Most/Least Popular Superhero:

- spark-submit --master local marvel_superheroes.py

- hdfs dfs -cat /user/hadoop/output/superhero_popularity9/part-*