BIGDATA TEAM

Spark Structured Streaming

1. Rate Source

Запускаем jupyter notebook:

```
PYSPARK_PYTHON=python3.6 PYSPARK_DRIVER_PYTHON=jupyter
PYSPARK_DRIVER_PYTHON_OPTS='notebook --ip=0.0.0.0 --port=<port_1>' pyspark
--conf spark.ui.port=<port_2> --driver-memory 512m --master yarn
--num-executors 2 --executor-cores 1 --packages
org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.0
```

Запускаем в jupyter notebook следующий код:

```
spark.sparkContext.setLogLevel("WARN")

rates = spark \
    .readStream \
    .format("rate") \
    .option("rowsPerSecond", 5) \
    .option("numPartitions", 3) \
    .load()

res = rates.groupBy(
    rates.value % 2
).sum()

query = rates \
    .writeStream \
    .outputMode("append") \
    .format("console") \
    .option("truncate", "false") \
    .start()
```

Убеждаемся, что он корректно работает (для этого требуется найти и исправить две ошибки в приведенном коде). В консоли должна начать печататься статистика. Пример правильного вывода:

```
Batch: 28
-------------------------------------------
+-----------+----------+
|(value % 2)|sum(value)|
+-----------+----------+
|0          |19460     |
|1          |19600     |
+-----------+----------+
```

Останавливаем стриминг, выполнив в ноутбуке команду

```
query.stop()
```

Перед каждым новым запуском стриминга следует останавливать предыдущий!

2. Чтение данных из Kafka

Требуется читать данные из Kafka и писать их в консоль (append mode).
Примечания:

- Пример чтения из Kafka есть в Лекции
- Брокеры кафка:
  `brain-node1.bigdatateam.org:9092,brain-node2.bigdatateam.org:9092,brain-node3.bigdatateam.org:9092`
- Топик кафка:
  `page_views`

Пример формата вывода:

```
----------------------------------------
Batch: 7
----------------------------------------
+----+--------------------+----------+---------+--------+--------------------+-------------+
| key|               value|     topic|partition|  offset|           timestamp|timestampType|
+----+--------------------+----------+---------+--------+--------------------+-------------+
|null|[31 35 32 32 35 3...|page_views|        1|13487786|2020-02-12 23:29:...|            0|
|null|[31 35 32 32 35 3...|page_views|        1|13487787|2020-02-12 23:29:...|            0|
|null|[31 35 32 32 35 3...|page_views|        1|13487788|2020-02-12 23:29:...|            0|
|null|[31 35 32 32 35 3...|page_views|        1|13487789|2020-02-12 23:29:...|            0|
|null|[31 35 32 32 35 3...|page_views|        1|13487790|2020-02-12 23:29:...|            0|
+----+--------------------+----------+---------+--------+--------------------+-------------+
```

3. Предобработка информации из Kafka

На основе данных из предыдущей задачи, требуется реализовать преобразование сырых данных kafka в `streaming df` с колонками: `ts, uid, url, title, ua` и вывести результат в консоль (append mode).

Примечания:

- При чтении из kafka мы получаем `streaming df` где интересная нам информация находится в колонке value в виде последовательности `byte`
- Для многих преобразований удобно использовать `spark sql`, пример: `df.selectExpr("cast(value as string)")`

Пример формата вывода:

```
-------------------------------------------
Batch: 1
-------------------------------------------
+--------------+--------------------+--------------------+--------------------+--------------------+
|            ts|                 uid|                 url|               title|                  ua|
+--------------+--------------------+--------------------+--------------------+--------------------+
|1522588842.883|a467606afdee6fd12...|https://brandshop...|%D0%9A%D1%83%D0%B...|Mozilla/5.0 (Wind...|
+--------------+--------------------+--------------------+--------------------+--------------------+
```

4. Stateful aggregation

На основе данных из описанного выше топика Kafka требуется посчитать число page_view и число уников для каждого домена за всё время работы стриминга. Результат отсортировать по убыванию числа просмотров и выводить в консоль раз в 5 секунд (complete mode).

Пример формата вывода:

```
-------------------------------------------
Batch: 10
-------------------------------------------
+---------------------------+----+------+
|domain                     |view|unique|
+---------------------------+----+------+
|news.rambler.ru            |18  |15    |
|m.lenta.ru                 |9   |7     |
|yandex.ru                  |9   |8     |
|www.championat.com         |7   |7     |
|www.yaplakal.com           |7   |7     |
|www.mk.ru                  |7   |7     |
|www.gazeta.ru              |6   |6     |
|www.coins-spb.ru           |6   |1     |
+---------------------------+----+------+
```

5. Window aggregation

На основе данных из описанного выше топика Kafka требуется посчитать окном размера 2 секунды каждую секунду (в терминах ts из лога) следующую статистику: число просмотров страницы и число уников в срезе доменов зоны ru и остальных доменов. Результат вывести в консоль по окончании обработки каждого батча (complete mode).

Пример формата вывода:

```
-------------------------------------------
Batch: 6
-------------------------------------------
+------------------------------------------+------+------+----+
|window                                    |zone  |unique|view|
+------------------------------------------+------+------+----+
|[2018-04-01 16:20:50, 2018-04-01 16:20:52]|not ru|56    |58  |
|[2018-04-01 16:20:50, 2018-04-01 16:20:52]|ru    |143   |151 |
|[2018-04-01 16:20:51, 2018-04-01 16:20:53]|not ru|78    |81  |
|[2018-04-01 16:20:51, 2018-04-01 16:20:53]|ru    |207   |219 |
|[2018-04-01 16:20:52, 2018-04-01 16:20:54]|not ru|24    |23  |
|[2018-04-01 16:20:52, 2018-04-01 16:20:54]|ru    |68    |68  |
+------------------------------------------+------+------+----+
```