BIGDATA TEAM

Spark Structured Streaming

1. Цель занятия

1. Научиться запускать и отлаживать программы на Spark Structured Streaming
2. Научиться читать данные из Kafka
3. Научиться описывать логику обработки данных, которая работает realtime
4. Получить базис практический знаний, без которого будет сложно выполнить домашнюю работу (:

2. RealTime Hello world

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
```

Эта команда аналогична той которую вы использовали на остальных занятиях по spark, за исключением подключения библиотеки для работы с kafka

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()
```

Добиваемся его корректной работы (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     |
+-----------+----------+
```

В рамках задачи стоит дополнительно обратить внимание на два момента:

1) Строчку кода `spark.sparkContext.setLogLevel("WARN")` - она требуется для отсекания бездны INFO логов которые при работе порождает Spark
2) То что такие простые вычисления небольших батчей производятся достаточно долго. Причина - в избыточном числе партиций после шафла данных по умолчанию. Чтобы изменить это число воспользуетесь настройкой `spark.sql.shuffle.partitions`.. Эта настройка вам пригодится как в последующих заданиях, так и в домашней работе

Останавливаем стриминг, выполнив в ноутбуке команду

```
query.stop()
```

Перед каждым новым запуском стриминга следует останавливать предыдущий!

3. Чтение данных из Kafka

В рамках этого задания требуется научиться 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|
+----+--------------------+----------+---------+--------+--------------------+-------------+
```

Как видно из примера вывода мы получили множество столбцов содержащих совершенно разнообразную информацию. По сути это всё технические столбцы которые для каждой записи позволяет получить сама Kafka, а интересующее нас сообщение лежит в колонке `value` в виде массива байт. В последующих упражнениях вам потребуется работать именно с этим столбцом.

4. Предобработка данных

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...|
+-------------+--------------------+--------------------+--------------------+--------------------+
```

При выполнении этого задания и ДЗ рекомендую воспользоваться двумя приёмами: отладка с использованием обычных dataframe и использование spark sql

Начнем с перехода во время отладки кода от streaming df к работе с обычными dataframe. Для этого требуется ключевое слово `readStream` заменить на `read`. Пример ниже:

```
spark.readStream.format("kafka")... ->
spark.read.format("kafka")...
```

По сути это приведет ситуацию к обычному spark dataframe, в который при запуске будет вычитано небольшое число строк из kafka. Это позволит быстро отладиться на dataframe не выходя из интерфейса jupiter notebook и не запуская rt стриминг. Когда получите в результате обработки тот результат, который вы ожидали, самое время вернуть ключевое слово `readStream` и протестировать его уже в условиях real-time.

Следующий приём про который все забывают это собственно использование spark.sql. Напомню что мы можем streaming dataframe преобразовать в view и обрабатывать с помощью sql, а результат всё также будет являться streaming df (пример есть в лекции). Это позволит использовать всю мощь sql и этим:

1. Упростить чтение кода
2. Использовать готовые функции из spark sql (набор функций схож с hive) и этим избежать python udf там где они не требуются.

5. Обратная связь

Обратная связь: http://rebrand.ly/mailbd2021q1_feedback_module

Просьба потратить 1-2 минут Вашего времени, чтобы поделиться впечатлением, описать что было понятно, а что непонятно. Мы учитываем рекомендации и имеем возможность переформатировать учебную программу под Ваши запросы.