*ЗАДАНИЕ:повторить всё, что было на уроке*

Запускаем mobaXterm

Создаем сессию с подключеним к worker-0

Выполняем команды:

*export SPARK\_KAFKA\_VERSION=0.10*

*/spark2.4/bin/pyspark --packages org.apache.spark:spark-sql-kafka-0-10\_2.11:2.4.5 --driver-memory 512m --num-executors 1 --executor-memory 512m --master local[1]*

Здесь делаем ограничение драйвера по памяти в 512Мб

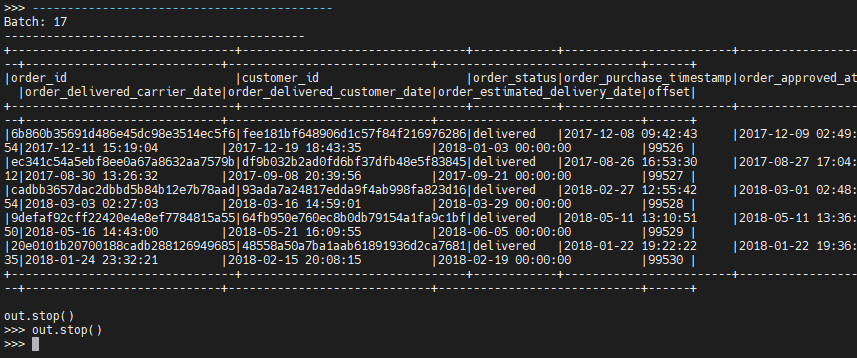


Читаем в консоль:

out = console\_output(parsed\_orders, 5)

Остановим процесс

out.stop()



# **MEMORY SINK**

**def** memory\_sink(df, freq):  
 **return** df.writeStream.format(**"memory"**) \  
 .queryName(**"mmingalov\_memory\_sink\_table"**) \  
 .trigger(processingTime=**'%s seconds'** % freq ) \  
 .start()

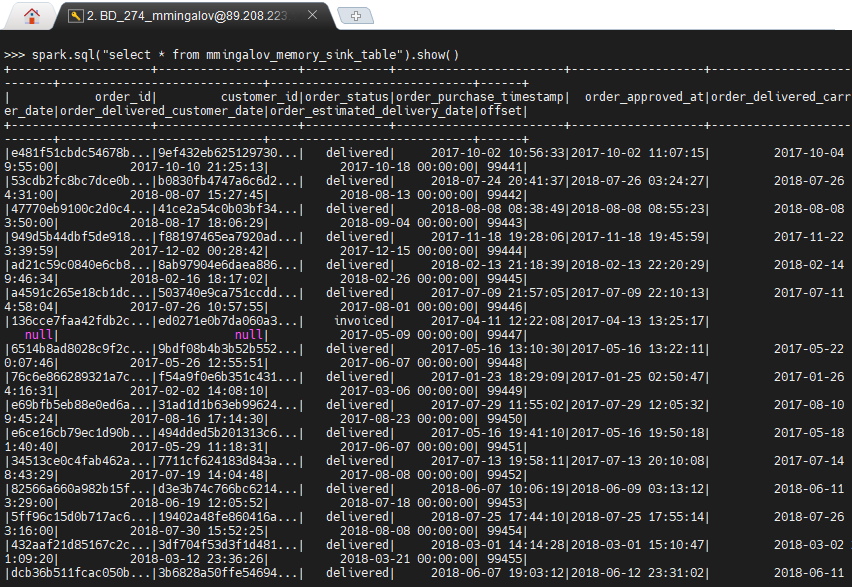
Запускаем stream.

stream = memory\_sink(parsed\_orders,5)

На экране ничего не увидим, потому что используем sink отличный от console.

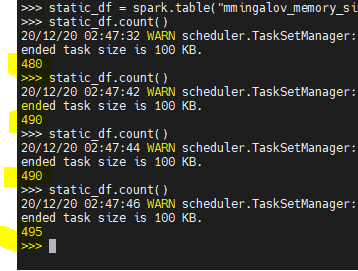
Но записи можно увидеть вот так:

spark.sql(**"select \* from mmingalov\_memory\_sink\_table"**).show()



Работает до тех пор, пока не лопнет память драйвера. Работает не параллельно, а только на драйвере всегда.

Ее можно в статический datafame положить



Останавиливаем

stream.stop()

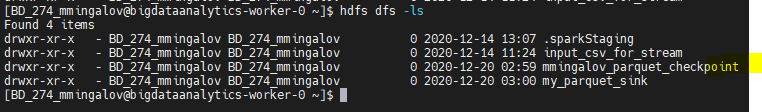
но по-прежнему можем обращаться в static\_df

# **FILE SINK**

**def** file\_sink(df, freq):  
 **return** df.writeStream.format(**"parquet"**) \  
 .trigger(processingTime=**'%s seconds'** % freq ) \  
 .option(**"path"**,**"my\_parquet\_sink"**) \  
 .option(**"checkpointLocation"**, **"mmingalov\_parquet\_checkpoint"**) \  
 .start()

stream = file\_sink(parsed\_orders,1)

Запустим отдельную сессию и прочитаем каталог файлов. Видим, что директории появились



Можно посмотреть содержимое директории командой:

hdfs dfs -ls my\_parquet\_sink

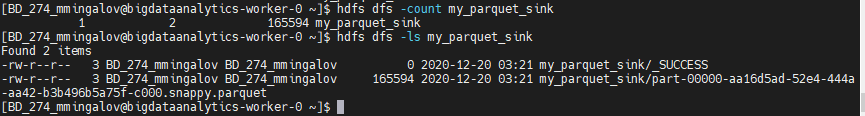
посчитаем кол-во файлов командой:

hdfs dfs -count my\_parquet\_sink



**def** compact\_directory(path):  
 df\_to\_compact = spark.read.parquet(path + **"/\*.parquet"**)  
 df\_to\_compact.persist() *#для сохранения в памяти* df\_to\_compact.count() *#для активации персиста, поскольку он lazy и не выполнится сам собой* df\_to\_compact.repartition(1).write.mode(**"overwrite"**).parquet(path)  
 df\_to\_compact.unpersist()  
  
compact\_directory(**"my\_parquet\_sink"**)

во второй вкладке, где HDFS, проверим содержимое директории:



Было 582, а сейчас 2. Сработало

Этот метод для production не идеален, поскольку всю информацию мы сохранили в

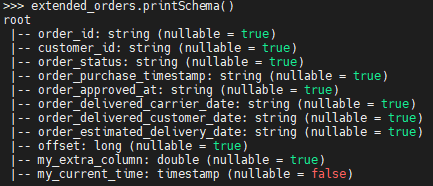
ОЗУ. А что если произойдет сбой в этот момент?

По-правильному надо было через temp директорию реализовать, а не через память.

# **KAFKA SINK**

Добавим столлбцы:

extended\_orders = parsed\_orders \  
 .withColumn(**"my\_extra\_column"**, F.round( F.rand() \* 100)) \  
 .withColumn(**"my\_current\_time"**, F.current\_timestamp())



*#в Kafka пишем KEY и VALUE. Чтобы получить VALUE, воспользуемся функцией STRUCT()*  
**def** kafka\_sink(df, freq):  
 **return** df.selectExpr(**"CAST(null AS STRING) as key"**, **"CAST(struct(\*) AS STRING) as value"**) \  
 .writeStream \  
 .format(**"kafka"**) \  
 .trigger(processingTime=**'%s seconds'** % freq ) \  
 .option(**"topic"**, **"kafka\_sink"**) \  
 .option(**"kafka.bootstrap.servers"**, kafka\_brokers) \  
 .option(**"checkpointLocation"**, **"my\_kafka\_checkpoint"**) \  
 .start()

**Примечание:**

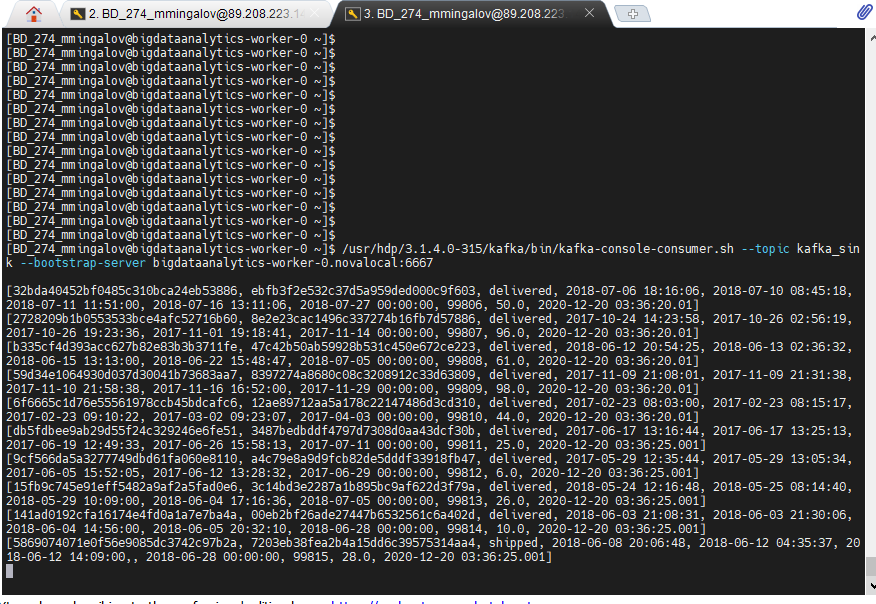
version 2.4.5

в Kafka мы должны писать Key, Value и topic (опционально)

stream = kafka\_sink(extended\_orders,5)

Прочитаем наш topic **kafka\_sink** через **console consumer** из Урок 2

/usr/hdp/3.1.4.0-315/kafka/bin/kafka-console-consumer.sh --topic kafka\_sink --bootstrap-server bigdataanalytics-worker-0.novalocal:6667



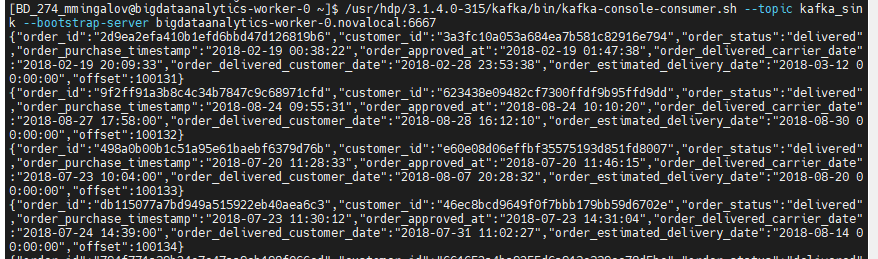
stream.stop()

**JSON**

**def** kafka\_sink\_json(df, freq):  
 *#в Kafka пишем KEY и VALUE. Чтобы получить VALUE, воспользуемся функцией STRUCT()* **return** df.selectExpr(**"CAST(null AS STRING) as key"**, **"CAST(to\_json(struct(\*)) AS STRING) as value"**) \  
 .writeStream \  
 .format(**"kafka"**) \  
 .trigger(processingTime=**'%s seconds'** % freq ) \  
 .option(**"topic"**, **"kafka\_sink"**) \  
 .option(**"kafka.bootstrap.servers"**, kafka\_brokers) \  
 .option(**"checkpointLocation"**, **"my\_kafka\_checkpoint"**) \  
 .start()  
  
stream = kafka\_sink\_json(parsed\_orders,5)

Прочитаем наш topic **kafka\_sink** через **console consumer** из Урок 2

/usr/hdp/3.1.4.0-315/kafka/bin/kafka-console-consumer.sh --topic kafka\_sink --bootstrap-server bigdataanalytics-worker-0.novalocal:6667



stream.stop()