Stream Processing Exercise 4 - Consuming from Kafka

January 28, 2021

0.1 Stream Processing Exercise 4 - Consuming from Kafka

Goals:

- Perform different computations on a input stream: read, aggregation, windowed aggregation
- Additional references
  - Spark Streaming
  - Structured Spark Streaming documentation
  - Spark and Kafka integration guide

Let's inspect content of Pageviews topic, showing it every 5 seconds:

[1]:
```
import sys
import os
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

from pyspark.streaming.kafka import KafkaUtils


sc = SparkContext(appName="PageViewsConsumer")

ssc = StreamingContext(sc, 5)


topics = ['pageviews']

kafkaParams = {'bootstrap.servers': 'broker:29092',
               'group.id' : 'test'}

stream = KafkaUtils.createDirectStream(ssc, topics, kafkaParams)

stream.map(lambda record : (record[0], record[1])).pprint()


ssc.start()
ssc.awaitTermination()
```

```
-------------------------------------------
Time: 2021-01-28 18:40:20
```

```
-------------------------------------------
('35261', '1611859214746,User_1,Page_16')
('35271', '1611859214824,User_7,Page_47')
('35281', '1611859215243,User_9,Page_74')
('35291', '1611859215360,User_2,Page_31')
('35301', '1611859215643,User_7,Page_16')
('35311', '1611859215737,User_1,Page_41')
('35321', '1611859215855,User_6,Page_22')
('35331', '1611859215897,User_8,Page_15')
('35341', '1611859216031,User_5,Page_17')
('35351', '1611859216271,User_7,Page_74')
…

-------------------------------------------
Time: 2021-01-28 18:40:25
-------------------------------------------
('35531', '1611859220365,User_7,Page_79')
('35541', '1611859220467,User_6,Page_29')
('35551', '1611859220911,User_8,Page_69')
('35561', '1611859221049,User_8,Page_83')
('35571', '1611859221493,User_4,Page_51')
('35581', '1611859221615,User_2,Page_50')
('35591', '1611859221724,User_2,Page_16')
('35601', '1611859222206,User_7,Page_24')
('35611', '1611859222408,User_8,Page_48')
('35621', '1611859222905,User_9,Page_21')
…

-------------------------------------------
Time: 2021-01-28 18:40:30
-------------------------------------------
('35691', '1611859225220,User_2,Page_29')
('35701', '1611859225521,User_3,Page_18')
('35711', '1611859225776,User_8,Page_99')
('35721', '1611859226246,User_7,Page_19')
('35731', '1611859226529,User_7,Page_44')
('35741', '1611859226806,User_9,Page_43')
('35751', '1611859226856,User_4,Page_24')
('35761', '1611859227272,User_2,Page_71')
('35771', '1611859227716,User_5,Page_59')
('35781', '1611859228180,User_5,Page_39')
…
```

```
KeyboardInterrupt                         Traceback (most recent call␣
↪last)

<ipython-input-1-819892cef919> in <module>
 23
 24 ssc.start()
---> 25 ssc.awaitTermination()


/usr/local/spark/python/pyspark/streaming/context.py in␣
↪awaitTermination(self, timeout)
    190         """
    191         if timeout is None:
--> 192             self._jssc.awaitTermination()
    193         else:
    194             self._jssc.awaitTerminationOrTimeout(int(timeout * 1000))


/usr/local/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py in␣
↪__call__(self, *args)
   1253             proto.END_COMMAND_PART
   1254
-> 1255         answer = self.gateway_client.send_command(command)
   1256         return_value = get_return_value(
   1257             answer, self.gateway_client, self.target_id, self.name)


/usr/local/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py in␣
↪send_command(self, command, retry, binary)
    983         connection = self._get_connection()
    984         try:
--> 985             response = connection.send_command(command)
    986             if binary:
    987                 return response, self.
↪_create_connection_guard(connection)


/usr/local/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py in␣
↪send_command(self, command)
   1150
   1151         try:
-> 1152             answer = smart_decode(self.stream.readline()[:-1])
   1153             logger.debug("Answer received: {0}".format(answer))
   1154             if answer.startswith(proto.RETURN_MESSAGE):


/opt/conda/lib/python3.7/socket.py in readinto(self, b)
```

```
        587         while True:
        588             try:
    --> 589                 return self._sock.recv_into(b)
        590             except timeout:
        591                 self._timeout_occurred = True


        /usr/local/spark/python/pyspark/context.py in signal_handler(signal,␣
    ↪frame)
        268         def signal_handler(signal, frame):
        269             self.cancelAllJobs()
    --> 270             raise KeyboardInterrupt()
        271
        272         # see http://stackoverflow.com/questions/23206787/


        KeyboardInterrupt:
```

Now, inspect also the content of Users topic

[1]:
```
import sys
import os
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

from pyspark.streaming.kafka import KafkaUtils


sc = SparkContext(appName="PageViewsConsumer")

ssc = StreamingContext(sc, 5)


topics = ['users']

kafkaParams = {'bootstrap.servers': 'broker:29092',
               'group.id' : 'test'}

stream = KafkaUtils.createDirectStream(ssc, topics, kafkaParams)

stream.map(lambda record : (record[0], record[1])).pprint()


ssc.start()
ssc.awaitTermination()
```

```
-------------------------------------------
```

```
Time: 2021-01-28 18:42:50
-------------------------------------------
('User_4', '1506345791690,User_4,Region_9,FEMALE')
('User_3', '1504013595481,User_3,Region_2,OTHER')
('User_2', '1501971523525,User_2,Region_7,OTHER')
('User_6', '1503017126471,User_6,Region_3,FEMALE')
('User_7', '1511621557789,User_7,Region_6,FEMALE')
('User_5', '1492140956709,User_5,Region_9,FEMALE')
('User_9', '1513628778048,User_9,Region_4,FEMALE')
('User_4', '1506932701859,User_4,Region_1,FEMALE')
('User_7', '1496203837070,User_7,Region_1,OTHER')
('User_1', '1495348263146,User_1,Region_3,MALE')
…

-------------------------------------------
Time: 2021-01-28 18:42:55
-------------------------------------------
('User_8', '1499414142877,User_8,Region_6,FEMALE')
('User_9', '1496333071756,User_9,Region_8,OTHER')
('User_3', '1512887675139,User_3,Region_9,FEMALE')
('User_7', '1490087080094,User_7,Region_2,MALE')
('User_8', '1503626581573,User_8,Region_6,MALE')
('User_7', '1505151641776,User_7,Region_4,OTHER')
('User_3', '1517475817084,User_3,Region_3,OTHER')
('User_1', '1498225963379,User_1,Region_2,FEMALE')
('User_2', '1513491998422,User_2,Region_9,FEMALE')
('User_7', '1509773240263,User_7,Region_1,MALE')
…



 ␣
↪----------------------------------------------------------------------------

       KeyboardInterrupt                         Traceback (most recent call␣
↪last)

       <ipython-input-1-020058323522> in <module>
        23
        24 ssc.start()
   ---> 25 ssc.awaitTermination()


       /usr/local/spark/python/pyspark/streaming/context.py in␣
↪awaitTermination(self, timeout)
       190         """
       191         if timeout is None:
   --> 192             self._jssc.awaitTermination()
```

```
      193         else:
      194             self._jssc.awaitTerminationOrTimeout(int(timeout * 1000))


      /usr/local/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py in␣
↪__call__(self, *args)
     1253             proto.END_COMMAND_PART
     1254
  -> 1255         answer = self.gateway_client.send_command(command)
     1256         return_value = get_return_value(
     1257             answer, self.gateway_client, self.target_id, self.name)


      /usr/local/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py in␣
↪send_command(self, command, retry, binary)
      983         connection = self._get_connection()
      984         try:
  --> 985             response = connection.send_command(command)
      986             if binary:
      987                 return response, self.
↪_create_connection_guard(connection)


      /usr/local/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py in␣
↪send_command(self, command)
     1150
     1151         try:
  -> 1152             answer = smart_decode(self.stream.readline()[:-1])
     1153             logger.debug("Answer received: {0}".format(answer))
     1154             if answer.startswith(proto.RETURN_MESSAGE):


      /opt/conda/lib/python3.7/socket.py in readinto(self, b)
      587         while True:
      588             try:
  --> 589                 return self._sock.recv_into(b)
      590             except timeout:
      591                 self._timeout_occurred = True


      /usr/local/spark/python/pyspark/context.py in signal_handler(signal,␣
↪frame)
      268         def signal_handler(signal, frame):
      269             self.cancelAllJobs()
  --> 270             raise KeyboardInterrupt()
      271
      272         # see http://stackoverflow.com/questions/23206787/
```

```
        KeyboardInterrupt:
```

Here we will consume streaming data from pageviews kafka topic to count numer of visits per page. First we are going to define input Stream

[1]:
```
from pyspark import sql
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *

spark = SparkSession \
    .builder \
    .appName("PageViewsConsumer") \
    .getOrCreate()


dfPageViewsStream = (
    spark
    .readStream
    .format("kafka")
    .option("kafka.bootstrap.servers", "broker:29092")
    .option("subscribe", "pageviews")
    .load()
)

dfPageViews = (
    dfPageViewsStream
    .selectExpr("CAST(key AS STRING)", "CAST(value AS STRING)", "timestamp")
    .withColumn("_tmp", split(col("value"), "\\,"))
    .select((col("_tmp").getItem(0).cast("long") / lit(1000)).cast("timestamp").
 ↪alias("viewtime"),
            col("_tmp").getItem(1).alias("userid"),
            col("_tmp").getItem(2).alias("pageid"),
            col("timestamp"))
)

dfPageViews.printSchema()
```

```
root
 |-- viewtime: timestamp (nullable = true)
 |-- userid: string (nullable = true)
 |-- pageid: string (nullable = true)
 |-- timestamp: timestamp (nullable = true)
```

Now let's create a table to store query output on memory

[2]:
```
dfPageViews.writeStream.format("memory").outputMode("append").
↪queryName("PageViews").start()
```

[2]:
```
<pyspark.sql.streaming.StreamingQuery at 0x7f2571f95fd0>
```

Here you can see table structure

[3]:
```
spark.sql("describe pageviews").show()
```

```
+---------+---------+-------+
| col_name|data_type|comment|
+---------+---------+-------+
| viewtime|timestamp|   null|
|   userid|   string|   null|
|   pageid|   string|   null|
|timestamp|timestamp|   null|
+---------+---------+-------+
```

Now, select those events happening in odd minutes.

[4]:
```
spark.sql("select * from PageViews where (minute(timestamp)%2) != 0").show()
```

```
+--------------------+------+-------+--------------------+
|            viewtime|userid| pageid|           timestamp|
+--------------------+------+-------+--------------------+
|2021-01-28 18:47:…|User_6|Page_77|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_7|Page_92|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_8|Page_82|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_9|Page_74|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_8|Page_92|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_2|Page_76|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_4|Page_71|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_1|Page_64|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_4|Page_94|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_9|Page_60|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_4|Page_74|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_7|Page_33|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_2|Page_69|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_6|Page_35|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_9|Page_99|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_1|Page_20|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_8|Page_12|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_6|Page_74|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_6|Page_80|2021-01-28 18:47:…|
|2021-01-28 18:47:…|User_1|Page_65|2021-01-28 18:47:…|
+--------------------+------+-------+--------------------+
only showing top 20 rows
```

Try with an order over userid.

[6]:
```
spark.sql("select * from PageViews where (minute(timestamp)%2) != 0 order by␣
 ↪userid").show()
```

```
+--------------------+------+-------+--------------------+
|            viewtime|userid| pageid|           timestamp|
+--------------------+------+-------+--------------------+
|2021-01-28 18:49:…|User_1|Page_39|2021-01-28 18:49:…|
|2021-01-28 18:49:…|User_1|Page_74|2021-01-28 18:49:…|
|2021-01-28 18:47:…|User_1|Page_10|2021-01-28 18:47:…|
|2021-01-28 18:49:…|User_1|Page_79|2021-01-28 18:49:…|
|2021-01-28 18:47:…|User_1|Page_61|2021-01-28 18:47:…|
|2021-01-28 18:49:…|User_1|Page_10|2021-01-28 18:49:…|
|2021-01-28 18:49:…|User_1|Page_82|2021-01-28 18:49:…|
|2021-01-28 18:49:…|User_1|Page_61|2021-01-28 18:49:…|
|2021-01-28 18:49:…|User_1|Page_71|2021-01-28 18:49:…|
|2021-01-28 18:47:…|User_1|Page_55|2021-01-28 18:47:…|
|2021-01-28 18:49:…|User_1|Page_21|2021-01-28 18:49:…|
|2021-01-28 18:49:…|User_1|Page_58|2021-01-28 18:49:…|
|2021-01-28 18:49:…|User_1|Page_79|2021-01-28 18:49:…|
|2021-01-28 18:49:…|User_1|Page_97|2021-01-28 18:49:…|
|2021-01-28 18:49:…|User_1|Page_25|2021-01-28 18:49:…|
|2021-01-28 18:49:…|User_1|Page_17|2021-01-28 18:49:…|
|2021-01-28 18:49:…|User_1|Page_91|2021-01-28 18:49:…|
|2021-01-28 18:49:…|User_1|Page_77|2021-01-28 18:49:…|
|2021-01-28 18:49:…|User_1|Page_14|2021-01-28 18:49:…|
|2021-01-28 18:49:…|User_1|Page_67|2021-01-28 18:49:…|
+--------------------+------+-------+--------------------+
only showing top 20 rows
```

Now count number of visits of each page:

- from the source stream: dfPageViews
- by page means group by pageid
- count as the aggregation operation
- store the output stream as an in-memory table: CountsByPage.

Describe its content and show part of the content

[7]:
```
dfCountsByPage = dfPageViews.groupBy("pageid").count()

dfCountsByPage.printSchema()
```

```
root
 |-- pageid: string (nullable = true)
 |-- count: long (nullable = false)
```

[8]:
```
dfCountsByPage \
    .writeStream \
    .format("memory") \
    .outputMode("Complete") \
    .queryName("CountsByPage") \
    .start()
```

[8]:
```
<pyspark.sql.streaming.StreamingQuery at 0x7f2571f95b10>
```

[11]:
```
spark.sql("select * from CountsByPage").show()
```

```
+-------+-----+
| pageid|count|
+-------+-----+
|Page_85|    6|
|Page_69|    4|
|Page_33|    2|
|Page_70|    1|
|Page_95|    3|
|Page_41|    1|
|Page_37|    2|
|Page_14|    3|
|Page_31|    3|
|Page_86|    3|
|Page_25|    2|
|Page_90|    3|
|Page_45|    1|
|Page_94|    2|
|Page_47|    3|
|Page_99|    4|
|Page_62|    4|
|Page_92|    4|
|Page_73|    3|
|Page_32|    4|
+-------+-----+
only showing top 20 rows
```

Now we want to get number of visits every 5 minutes over last 10 minutes:

- 10 minutes is the window duration
- 5 minutes is the slide duration

Additional references for windowing in Spark can be found here.

[13]:
```
dfWindow = dfPageViews.groupBy(window("timestamp", "10 minutes", "5 minutes"),␣
 ↪"pageid").count()

dfWindow.printSchema()
```

```
root
 |-- window: struct (nullable = true)
 |    |-- start: timestamp (nullable = true)
 |    |-- end: timestamp (nullable = true)
 |-- pageid: string (nullable = true)
 |-- count: long (nullable = false)
```

[14]:
```
dfWindow.writeStream.format("memory").outputMode("complete").
 ↪queryName("pageWindow").start()
```

[14]:
```
<pyspark.sql.streaming.StreamingQuery at 0x7f25708f0810>
```

[16]:
```
spark.sql("select * from pageWindow").show()
```

```
+--------------------+-------+-----+
|              window| pageid|count|
+--------------------+-------+-----+
|[2021-01-28 18:50…|Page_33|    1|
|[2021-01-28 18:50…|Page_17|    1|
|[2021-01-28 18:50…|Page_12|    2|
|[2021-01-28 18:55…|Page_42|    1|
|[2021-01-28 18:55…|Page_83|    1|
|[2021-01-28 18:55…|Page_71|    2|
|[2021-01-28 18:55…|Page_66|    1|
|[2021-01-28 18:55…|Page_34|    1|
|[2021-01-28 18:50…|Page_75|    1|
|[2021-01-28 18:50…|Page_15|    1|
|[2021-01-28 18:50…|Page_21|    1|
|[2021-01-28 18:50…|Page_22|    1|
|[2021-01-28 18:55…|Page_65|    1|
|[2021-01-28 18:50…|Page_71|    2|
|[2021-01-28 18:50…|Page_35|    1|
|[2021-01-28 18:55…|Page_22|    1|
|[2021-01-28 18:55…|Page_21|    1|
|[2021-01-28 18:50…|Page_28|    1|
|[2021-01-28 18:55…|Page_47|    1|
|[2021-01-28 18:55…|Page_69|    1|
+--------------------+-------+-----+
only showing top 20 rows
```

[ ]:
```

```