МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра информационной безопасности

ОТЧЕТ

по лабораторной работе № 4
по дисциплине «Распределенные системы обработки данных»
Тема: Чтение и запись данных в кафку помощью spark

Студентка гр. 1361 _____ Токарева У.В.

Преподаватель _____ Троценко В.В.

Санкт-Петербург
2024

Цель работы.

Целью данной работы является выполнение базовых операций с Kafka. Необходимо изучить и применить на практике технологии Apache Spark и Apache Kafka для обработки и передачи данных. В ходе работы требовалось доработать код из предыдущей лабораторной работы, обеспечив возможность записи обработанных данных в Kafka. Разработать процедуры для записи агрегированных данных, полученных в Spark, в отдельные топики Kafka. Разработать процедуры для чтения данных из созданных топиков Kafka обратно в Spark. Собрать и представить полученные данные в виде информативных таблиц.

Основные теоретические положения.

Термины Apache Kafka:

- **Брокер (Broker):** Сервер, на котором работает Kafka. Брокеры хранят данные, обрабатывают запросы на чтение и запись, и управляют топиками и партициями.
- **Топик (Topic):** Категория или канал, в который продюсеры отправляют данные и из которого консьюмеры их читают. Топики представляют собой потоки данных и делятся на партиции для масштабируемости.
- **Партиция (Partition):** Раздел топика. Каждый топик может быть разбит на несколько партиций, что позволяет распределять данные и обрабатывать их параллельно на разных брокерах.
- **Продюсер (Producer):** Приложение или процесс, отправляющий данные (сообщения) в один или несколько топиков Kafka.
- **Консьюмер (Consumer):** Приложение или процесс, читающий данные из одного или нескольких топиков Kafka.
- **Группа консьюмеров (Consumer Group):** Набор консьюмеров, которые совместно читают данные из топика. Kafka распределяет партиции между консьюмерами в группе, чтобы избежать дублирующего чтения.

- **Сообщение (Message):** Единица данных, отправляемая продюсером в топик Kafka. Сообщение состоит из ключа, значения и метаданных (например, таймстемпа).
- **Офсет (Offset):** Позиция сообщения внутри партиции. Каждый офсет уникален для партиции и помогает консьюмерам отслеживать, какие данные были прочитаны.
- **Кластер (Cluster):** Группа брокеров, работающих вместе. Кластер Kafka обеспечивает распределенное хранение данных и управляет репликацией для отказоустойчивости.
- **Репликация (Replication):** Копирование данных между брокерами для повышения отказоустойчивости. Каждый партиция может иметь несколько копий (реплик) на разных брокерах.
- **Schema Registry:** Компонент, позволяющий управлять схемами данных для Avro, Protobuf и других форматов. Обеспечивает согласованность данных и поддержку эволюции схем.

Ход работы.

В данной работе необходимо внести изменения в код лабораторной работы № 3. В начале при создании SparkSessoin нам необходимо подключить кафку:

```
spark = SparkSession.builder \
    .appName("PySpark") \
    .config("spark.jars.packages",
"org.apache.spark:spark-sql-kafka-0-
10_2.12:3.5.3,org.apache.spark:spark-avro_2.12:3.5.3") \
    .getOrCreate()
```

Далее разберем новую часть кода. Во-первых, добавляем схему для всех данных:

```
total_schema = """
{
  "type": "record",
```

```
  "name": "TotalStats",
  "fields": [
    {"name": "YM", "type": "string"},
    {"name": "total_likes", "type": "long"},
    {"name": "total_comment", "type": "double"},
    {"name": "total_view", "type": "long"},
    {"name": "total_CLR", "type": "double"}
  ]
}
"""
```

Во-вторых, добавляем схему для категорий:

```
cats_schema = """
{
  "type": "record",
  "name": "CategoryStats",
  "fields": [
    {"name": "YM", "type": "string"},
    {"name": "cats_likes", "type": "long"},
    {"name": "cats_comment", "type": "double"},
    {"name": "cats_view", "type": "long"},
    {"name": "cat_id", "type": "int"},
    {"name": "cat_name", "type": "string"}
  ]
}
"""
```

В-третьих, добавляем схему для бинарных признаков:

```
bin_schema = """
{
  "type": "record",
  "name": "BinaryStats",
```

```
  "fields":                                                    [
    {"name":        "YM",        "type":        "string"},
    {"name":      "bins_view",      "type":      "long"},
    {"name":    "bins_comment",    "type":    "double"},
    {"name":      "bins_likes",      "type":      "long"}
  ]
}
"""
```

Далее осуществляем запись данных в кафку:

```
def    write_to_kafka(df:    DataFrame,    topic:    str,
avro_schema:                                          str):
    df_avro    =    df.select(to_avro(struct(*df.columns),
avro_schema).alias("value")) # преобразование дата фрейма
в                                                    строку
    df_avro.write                                          \
        .format("kafka")                                   \
        .option("kafka.bootstrap.servers",
kafka_bootstrap_servers)                                   \
        .option("topic",             topic)                \
        .save()

write_to_kafka(df_total,   "total-topic",   total_schema)
write_to_kafka(df_cats,    "cats-topic",    cats_schema)
write_to_kafka(df_challenge,           "challenge-topic",
bin_schema)
write_to_kafka(df_asmr,    "asmr-topic",    bin_schema)
write_to_kafka(df_chandas, "chandas-topic", bin_schema)
```

Проверяем создались ли топики. И читаем данные из топиков кафки:

```
def read_from_kafka(topic: str, avro_schema: str):
    df_avro_read = spark.read \
        .format("kafka") \
        .option("kafka.bootstrap.servers",
```

```
kafka_bootstrap_servers) \
        .option("subscribe", topic) \
        .load()
    df =
df_avro_read.select(from_avro(df_avro_read.value,
avro_schema, options={"mode":
"PERMISSIVE"}).alias("data")).select("data.*")     #
превращаем в дата фрейм
    return df

df_total_kafka = read_from_kafka("total-topic",
total_schema)
df_cats_kafka = read_from_kafka("cats-topic",
cats_schema)
df_challenge_kafka = read_from_kafka("challenge-topic",
bin_schema)
df_asrm_kafka = read_from_kafka( "asmr-topic",
bin_schema)
df_chandas_kafka = read_from_kafka("chandas-topic",
bin_schema)
```

Создаем временные таблицы для сборки финальной витрины:

```
df_total_kafka.createOrReplaceTempView("df_total_kafka")
df_cats_kafka.createOrReplaceTempView("df_cats_kafka")
df_challenge_kafka.createOrReplaceTempView("df_challenge_
kafka")
df_asrm_kafka.createOrReplaceTempView("df_asrm_kafka")
df_chandas_kafka.createOrReplaceTempView("df_chandas_kafk
a")
```

Результаты работы представлены на рисунках 1, 2 и 3.

YM	total_comment	total_likes	total_view	total_CLR	cat_name	cats_view	cats_comment	cats_likes	cats_CLR	diff_CLR_abs	diff_CLR_relative
2024-03	9218794.0	351500982	8676586316	0.03	Илон Маск	13791788	56576.0	579172	0.1	-0.07	0.3
2024-03	9218794.0	351500982	8676586316	0.03	Илон Маск	13791788	56576.0	579172	0.1	-0.07	0.3
2024-03	9218794.0	351500982	8676586316	0.03	Илон Маск	13791788	56576.0	579172	0.1	-0.07	0.3
2024-03	9218794.0	351500982	8676586316	0.03	Илон Маск	13791788	56576.0	579172	0.1	-0.07	0.3
2024-02	1.1091736E7	292969924	6897478524	0.04	Илон Маск	10456901	38302.0	514000	0.07	-0.03	0.57
2024-02	1.1091736E7	292969924	6897478524	0.04	Илон Маск	10456901	38302.0	514000	0.07	-0.03	0.57
2024-02	1.1091736E7	292969924	6897478524	0.04	Илон Маск	10456901	38302.0	514000	0.07	-0.03	0.57
2024-02	1.1091736E7	292969924	6897478524	0.04	Илон Маск	10456901	38302.0	514000	0.07	-0.03	0.57
2024-01	9150624.0	271780911	6060824937	0.03	Илон Маск	2122229	1537.0	121483	0.01	0.02	3.0
2024-01	9150624.0	271780911	6060824937	0.03	Илон Маск	2122229	1537.0	121483	0.01	0.02	3.0
2024-01	9150624.0	271780911	6060824937	0.03	Илон Маск	2122229	1537.0	121483	0.01	0.02	3.0
2024-01	9150624.0	271780911	6060824937	0.03	Илон Маск	2122229	1537.0	121483	0.01	0.02	3.0
2023-12	1.2821919E7	364242277	6742291412	0.04	Илон Маск	13865888	21399.0	818103	0.03	0.01	1.33
2023-12	1.2821919E7	364242277	6742291412	0.04	Илон Маск	13865888	21399.0	818103	0.03	0.01	1.33
2023-12	1.2821919E7	364242277	6742291412	0.04	Илон Маск	13865888	21399.0	818103	0.03	0.01	1.33
2023-12	1.2821919E7	364242277	6742291412	0.04	Илон Маск	13865888	21399.0	818103	0.03	0.01	1.33
2023-11	1.2382188E7	314392817	6084022728	0.04	Илон Маск	1033966	3609.0	50384	0.07	-0.03	0.57
2023-11	1.2382188E7	314392817	6084022728	0.04	Илон Маск	1033966	3609.0	50384	0.07	-0.03	0.57
2023-11	1.2382188E7	314392817	6084022728	0.04	Илон Маск	1033966	3609.0	50384	0.07	-0.03	0.57
2023-11	1.2382188E7	314392817	6084022728	0.04	Илон Маск	1033966	3609.0	50384	0.07	-0.03	0.57

Рисунок 1 – Результат работы по категориям.

YM	total_comment	total_likes	total_view	total_CLR	challenge_likes	challenge_comment	challenge_view	challenge_CLR	diff_CLR_abs_ch	diff_CLR_relative_ch
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL
2024-04	4534493.0	158554991	3629734886	0.03	7007256	10636.0	259298410	0.0	0.03	NULL

Рисунок 2 – Результат работы для бинарных признаков

asmr_likes	asmr_comment	asmr_view	asmr_CLR	diff_CLR_abs_as	diff_CLR_relative_as	chandas_likes	chandas_comment	chandas_view	chandas_CLR	diff_CLR_abs_chas	diff_CLR_relative_chas
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL
1150379	10375.0	48465618	0.01	0.02	3.0	936580	2361.0	45754954	0.0	0.03	NULL

Рисунок 3 – Результат работы для бинарных признаков.

Выводы

Работа подтвердила возможность использования Spark и Kafka как мощного средства для обработки и передачи потоковых данных в реальном времени. Были получены навыки, необходимые для построения распределенных систем обработки данных, основанных на Spark и Kafka.