Очередь сообщений Kafka. Spark Streaming

LSML #3

Что сегодня изучим?

- Очереди сообщений
- Apache Kafka
- Подход – micro batch streaming
- Spark streaming
- Примеры реализации стримингового ETL

Шина данных Kafka

Очередь сообщений

Решение для передачи, временного хранения и обработки данных в режиме реального времени. Часто используются для распределения данных между различными приложениями или компонентами системы. Широко используется для стриминговой обработки данных, создания централизованных журналов событий и реализации архитектур "событие-ориентированных" систем

Очередь сообщений

Модели очередей

PULL

Инициатива запроса данных: получатель запрашивает данные у отправителя по мере необходимости. Получатель активно инициирует запрос данных.

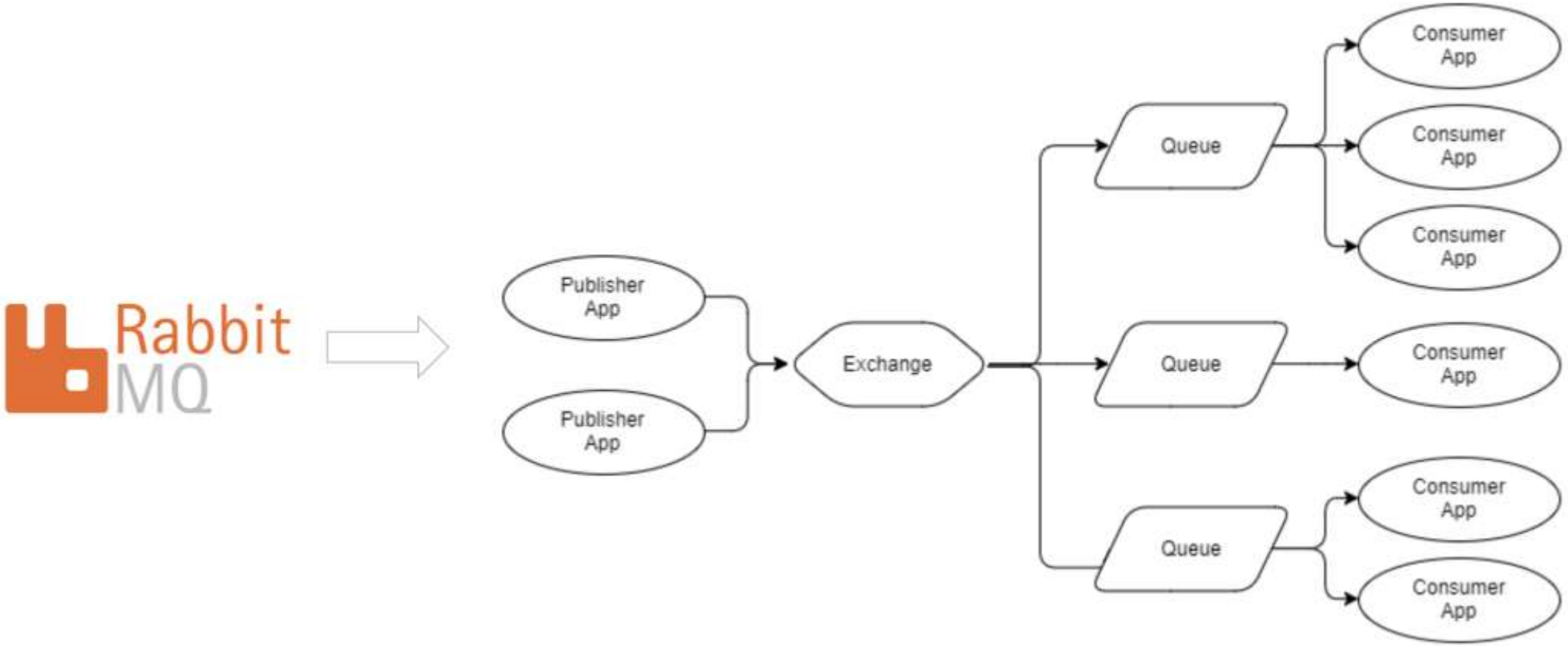
Примеры: запросы к базе данных, загрузка вебстраниц, клиент-серверное взаимодействие, где клиент запрашивает данные у сервера.

PUSH

Инициатива передачи данных: данные передаются от отправителя к получателю без запроса со стороны получателя.

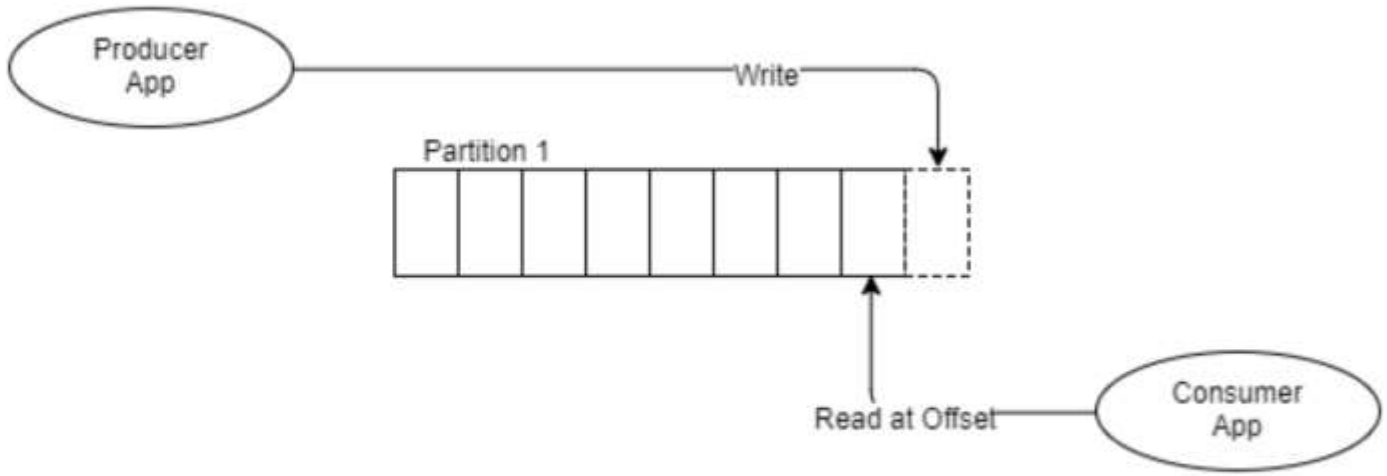
Примеры: уведомления на мобильных устройствах, рассылка электронной почты, стриминговые сервисы, где данные активно отправляются клиенту

PUSH-модель


Rabbit
MQ
Publisher App
Publisher App
Exchange
Queue
Queue
Queue
Consumer App
Consumer App
Consumer App
Consumer App
Consumer App
Consumer App

PULL-модель

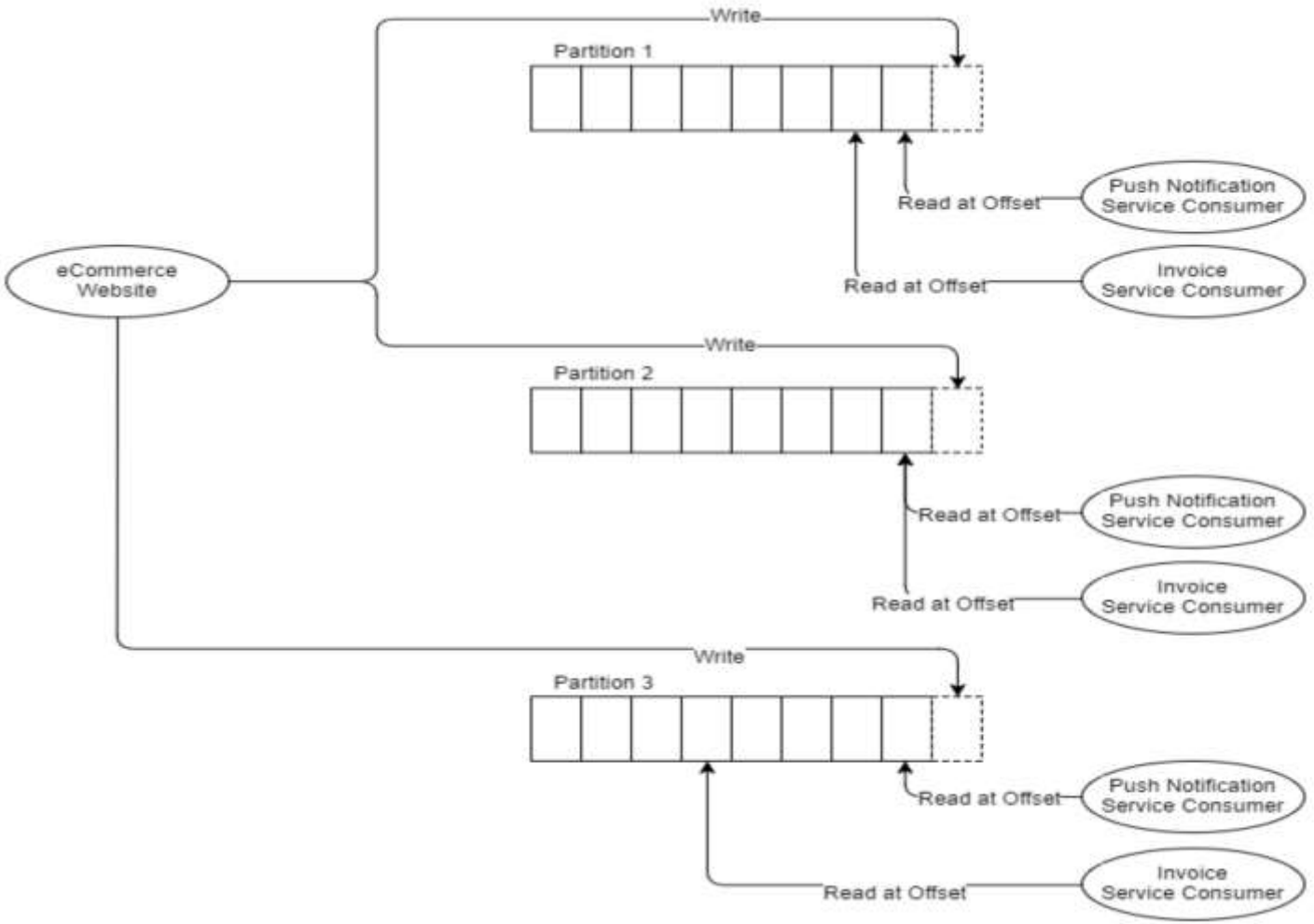
kafka



Определение Apache Kafka

Распределенная платформа, которая предоставляет высокопроизводительное, надежное и масштабируемое решение для передачи, хранения и обработки данных в реальном времени. Шина данных Kafka позволяет управлять потоками данных с использованием тем (topics) для организации информации и разделения ее между различными приложениями или компонентами системы

Принцип работы Kafka

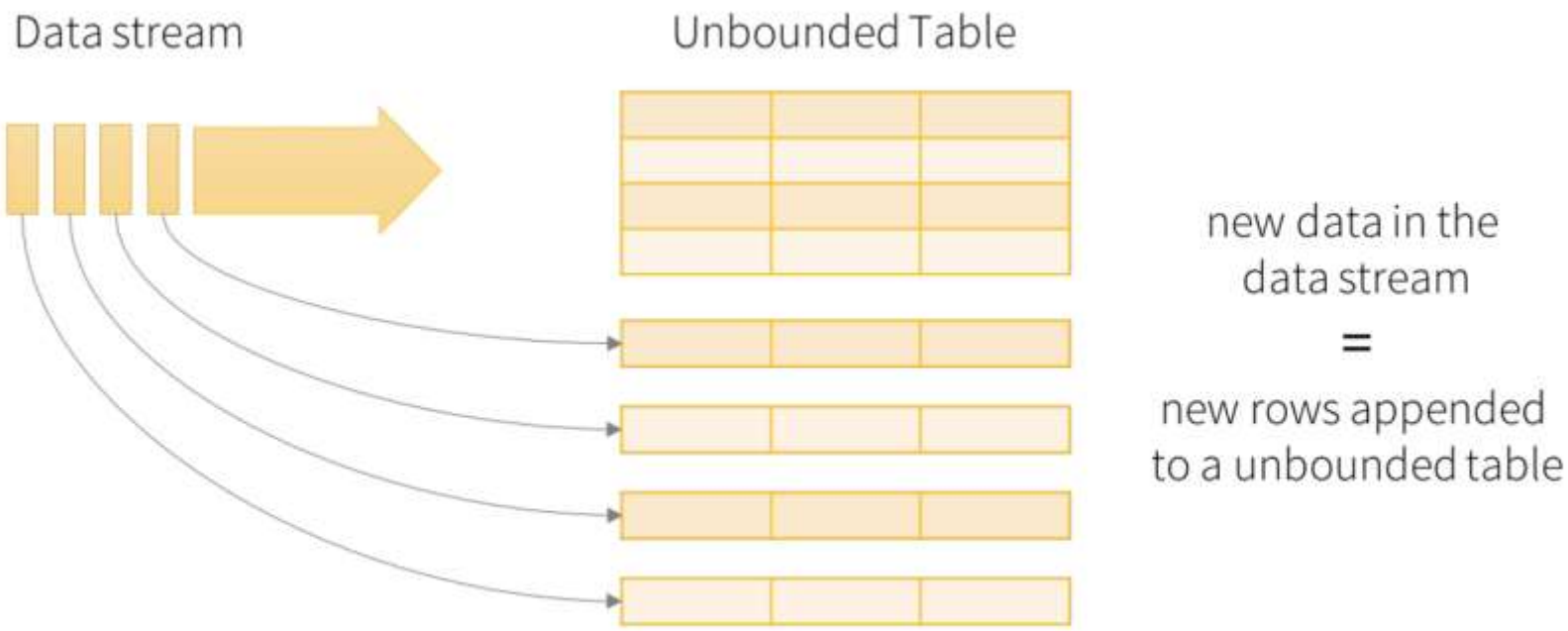


Spark Streaming

Micro batch streaming

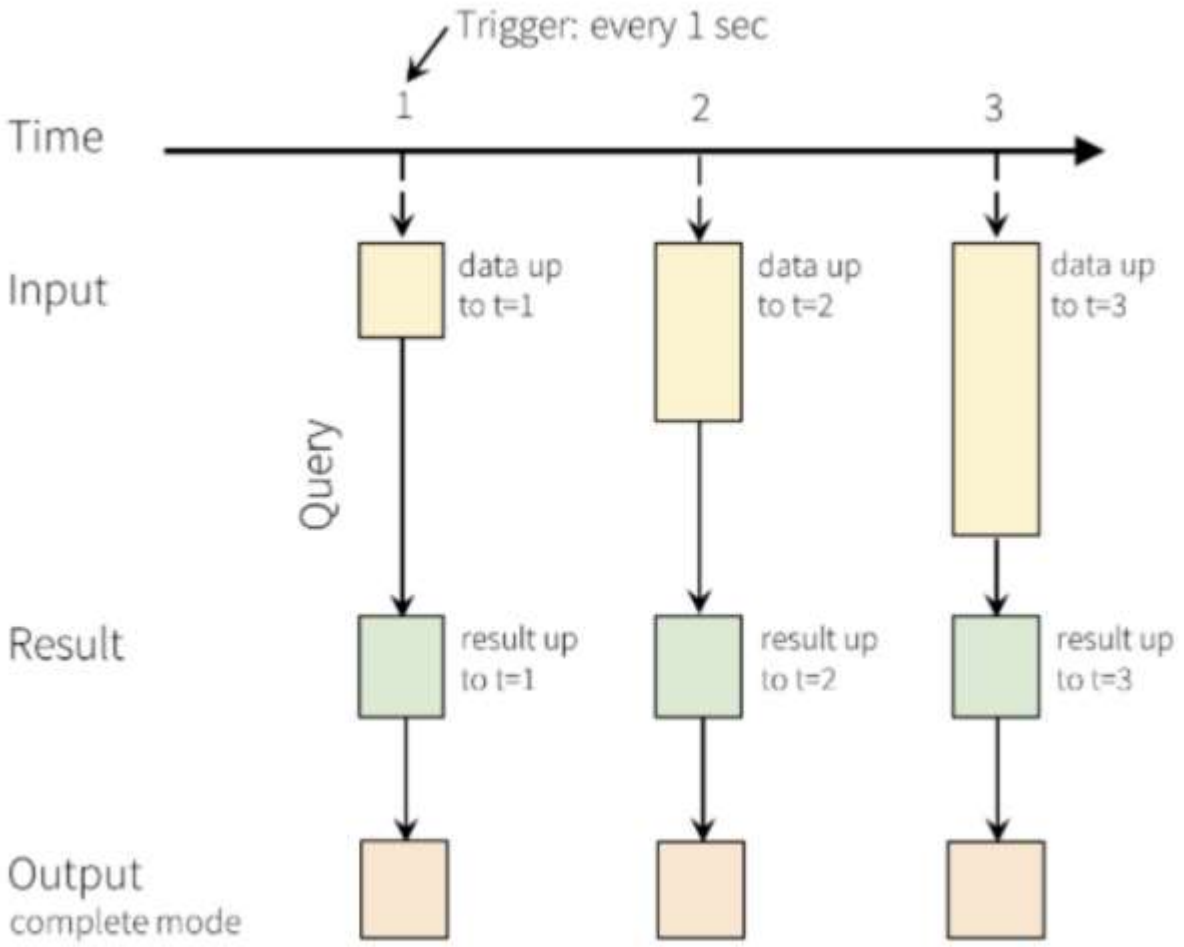
Подход к обработке потоковых данных, при котором данные обрабатываются пакетами фиксированного размера, называемыми micro-batches. Является компромиссом между традиционной пакетной обработкой данных и строгим потоковым (streaming) подходом.

Концепция Spark Streaming

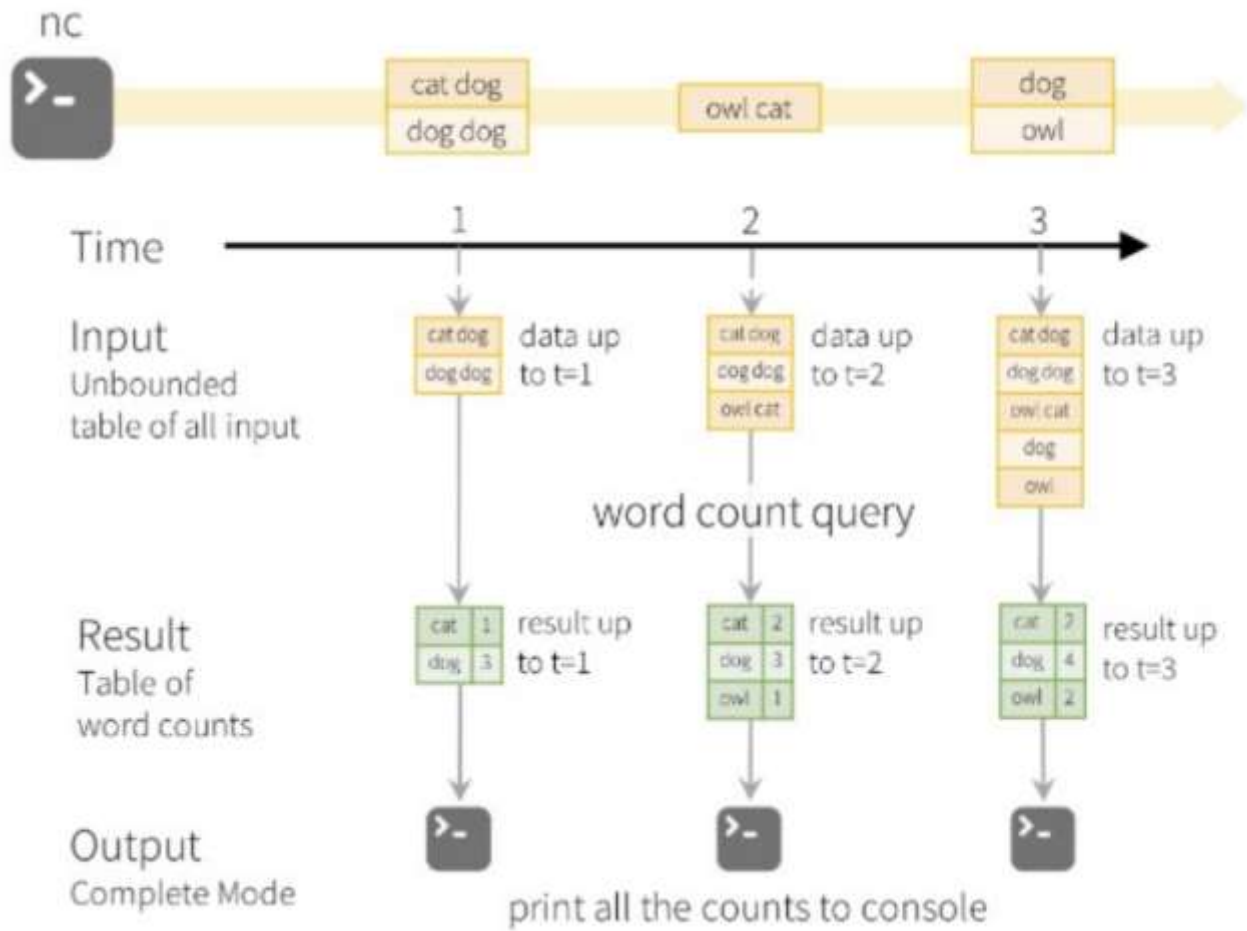


Data stream as an unbounded table

Концепция Spark Streaming - II



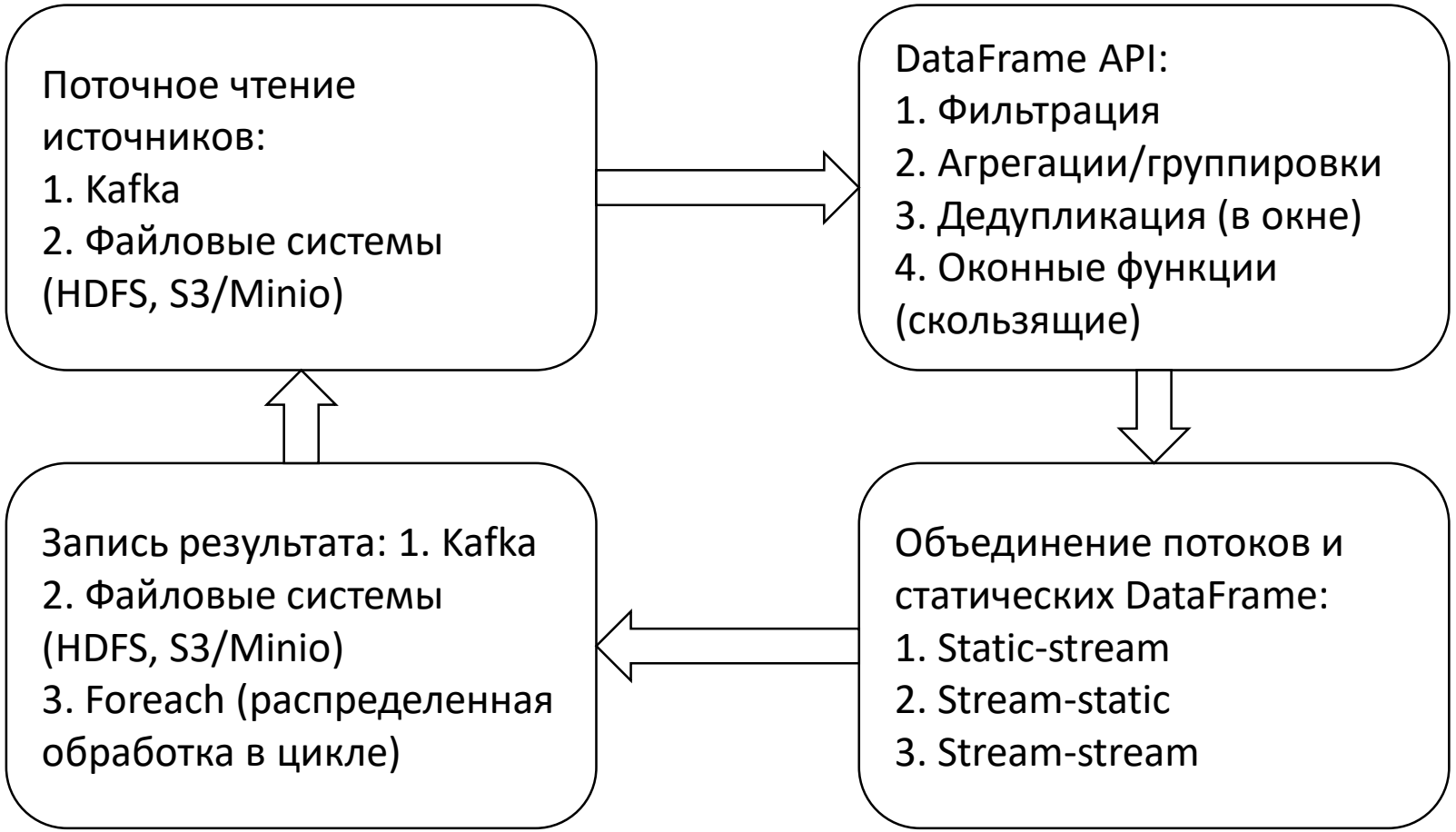
Интуитивный пример



Примеры работы с кодом

```
from pyspark.sql.types import StructType, StringType, IntegerType

schema = StructType().add("name", StringType()).add("age", IntegerType())

df = spark.readStream \
    .schema(schema) \
    .option("maxFilesPerTrigger", 1) \
    .csv("/path/to/incoming/files")

df_filtered = df.filter("age >= 18")

query = df_filtered.writeStream \
    .format("console") \
    .outputMode("append") \
    .start()

query.awaitTermination()
```

Основные возможности



Достоинства

Единый API для пакетной и потоковой обработки

Толерантность к сбоям и обработка точно один раз

Обработка только 1 раз

Интеграция со стандартными источниками данных

Интеграция с экосистемой Spark/Hadoop

Поддержка SQLзапросов

Недостатки

Задержка (Latency)

Сложность масштабирования (обработка огромных micro-batch)

Меньше строк кода, чем SQL

Сложность управления состоянием

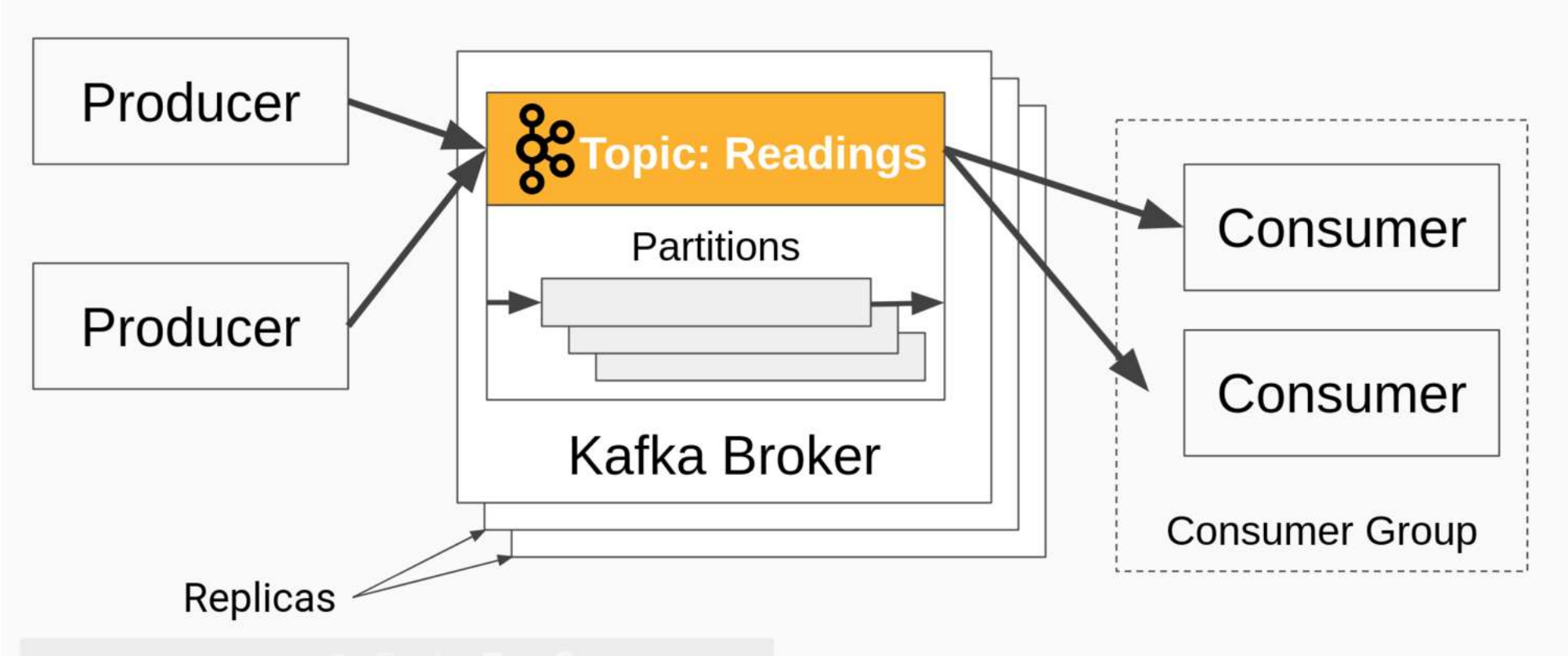
Необходимость знания экосистемы Spark

Ограничения по поддержке источников данных

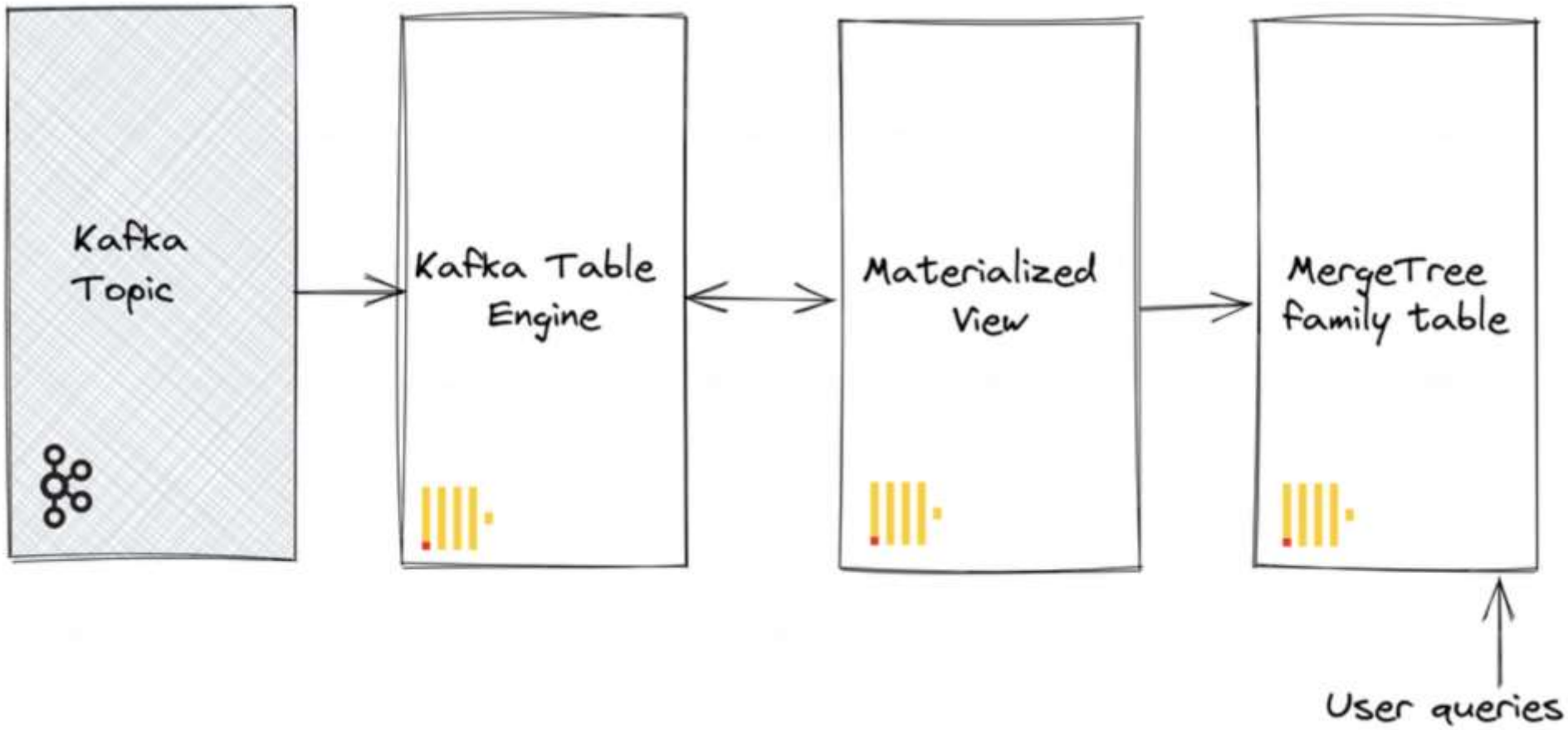
ClickHouse для рилтайм аналитики?

ClickHouse — это высокопроизводительная колоночная система управления базами данных (СУБД) с открытым исходным кодом, предназначенная для онлайн-аналитической обработки (OLAP) данных в реальном времени. Она позволяет быстро обрабатывать миллиарды строк и широкие таблицы, используя SQL-запросы, благодаря хранению данных по столбцам, а не строкам.

RealTime обработка данных через ClickHouse



RealTime обработка данных через ClickHouse



Kafka Engine

```
-- Consumer
CREATE TABLE test.kafka (key UInt64, value UInt64)
    ENGINE = Kafka
    SETTINGS kafka_broker_list = 'kafka1:19092',
             kafka_topic_list = 'my_topic',
             kafka_group_name = 'my_conumber_group_name',
             kafka_format = 'JSONEachRow',
             kafka_max_block_size = 1048576;
```

Destination table

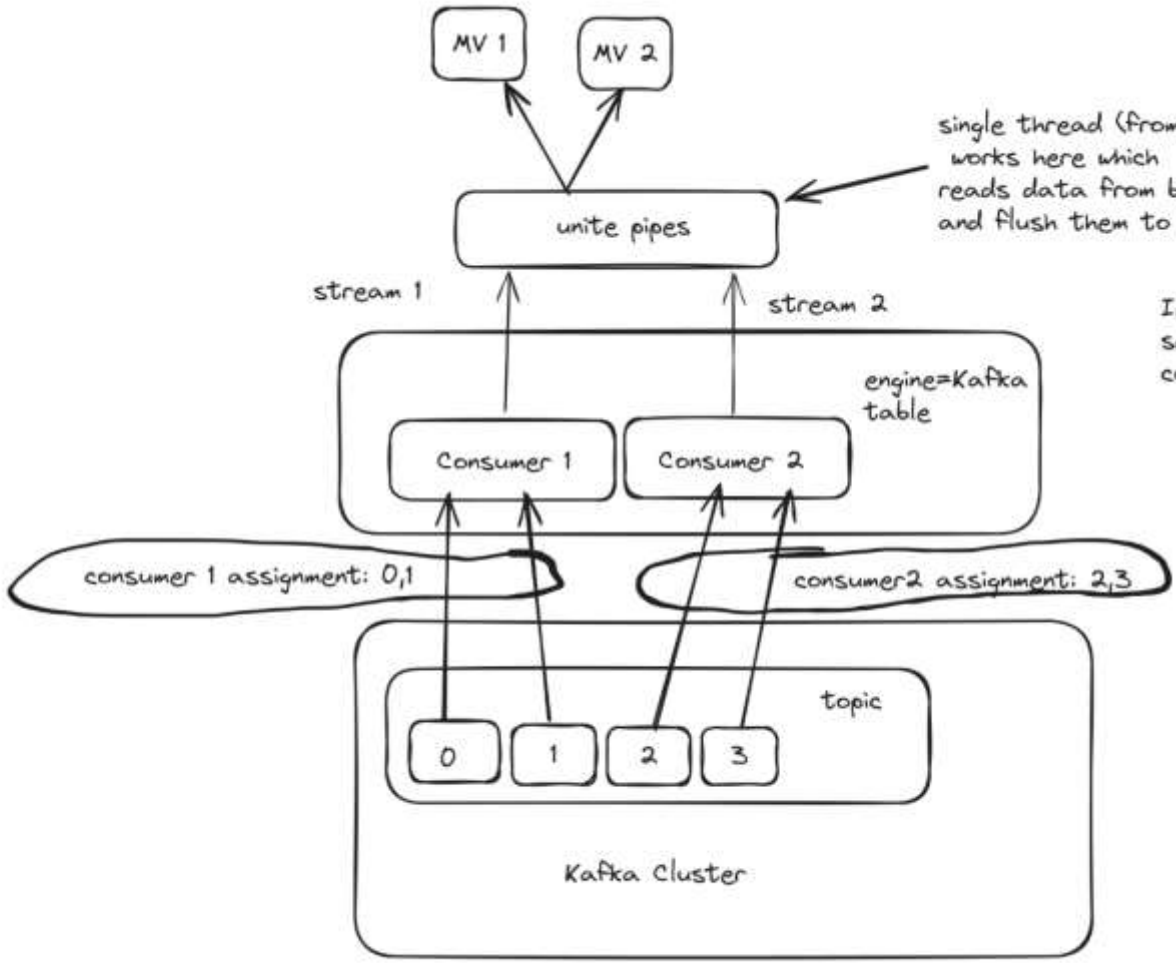
```
-- Destination table
CREATE TABLE test.view (key UInt64, value UInt64)
    ENGINE = MergeTree()
    ORDER BY key;
```

MV

-- Materialized View to move the data from a Kafka topic to a ClickHouse table

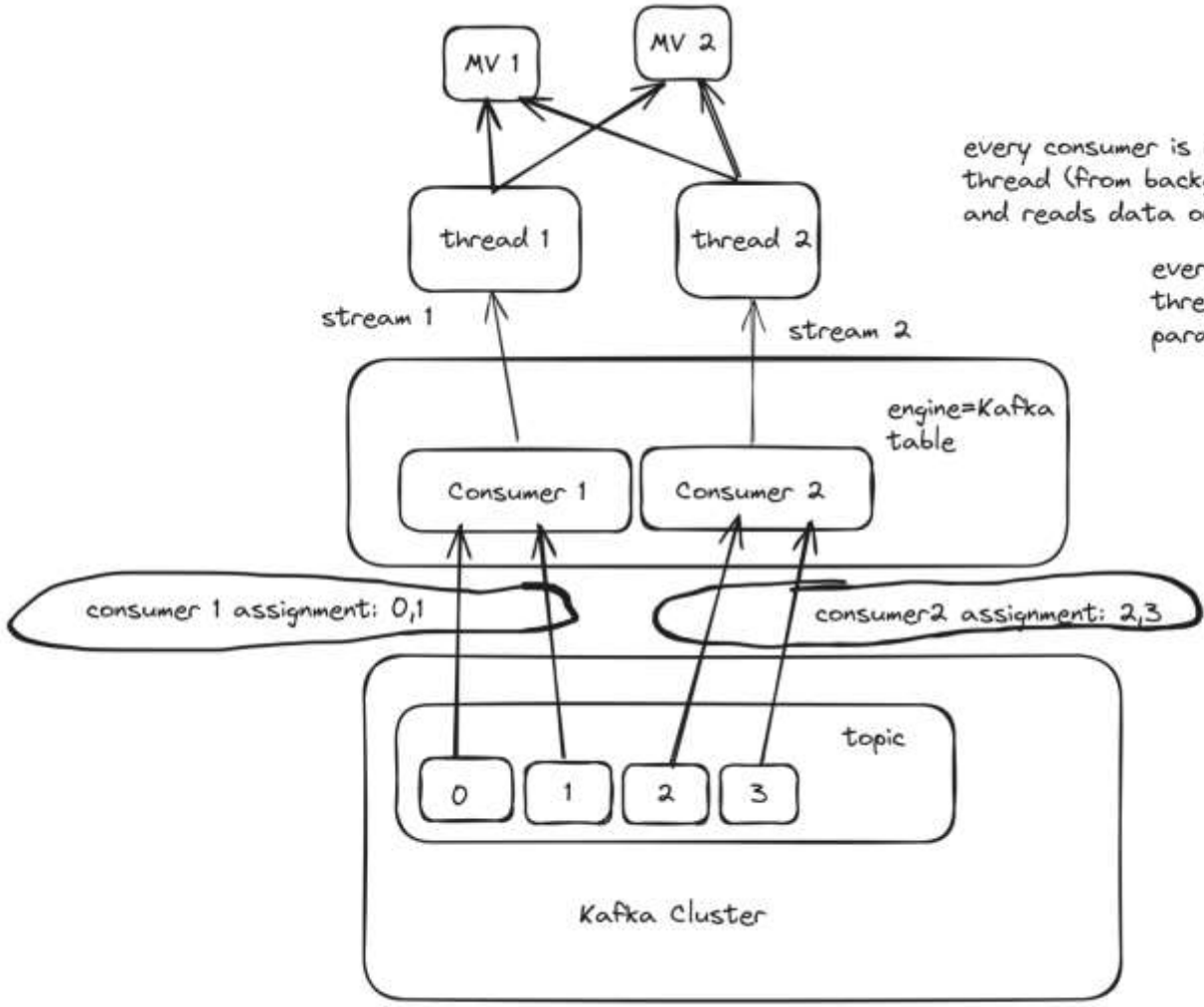
CREATE MATERIALIZED VIEW test.consumer TO test.view AS

SELECT * FROM test.kafka;



INSERTS to all MVs done from one thread sequentially (Linearization of INSERTs). Data from all consumers is mixed together in one insert

thread_per_consumer = 0



every consumer is read by a separate thread (from background_message_broker_schedule_pool_size) and reads data only from one consumer

every thread does INSERTS to all MVs
threads can work in parallel, so you can see parallel inserts into tables happening

thread_per_consumer = 1