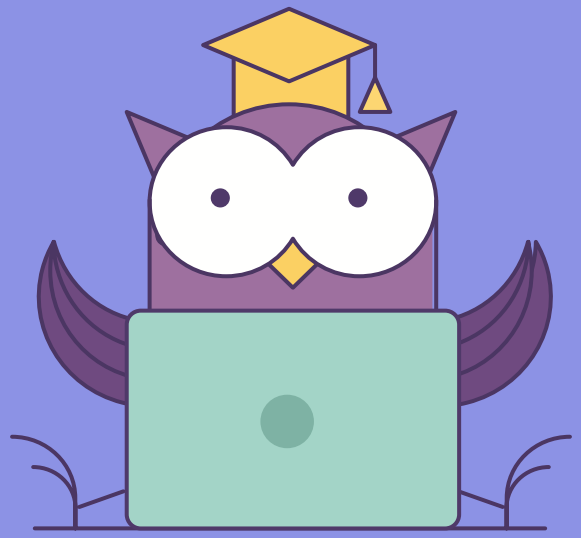
ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

OTUS

Spark Streaming

- Reactive processing
- Micro-batch processing
- Structured Streaming API

Трусов Иван Алексеевич

01

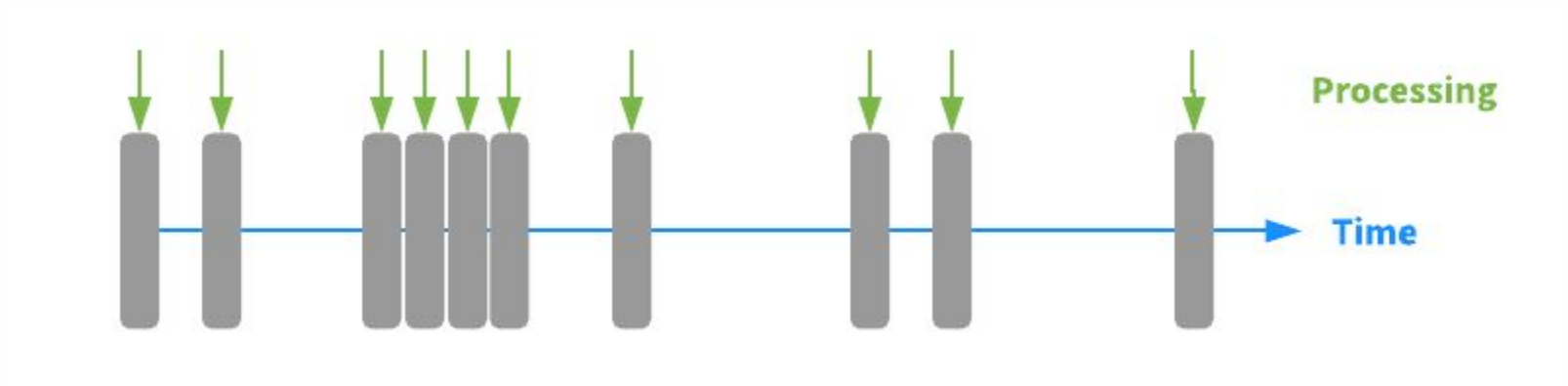
onReceive processing

Теория

Reactive processing

Подход к процессингу данных, основанный на моментальной обработке сообщений

- процессинг происходит по одному событию
- система-приемник реагирует на сообщения
- нет новых сообщений - нет вычислений



Reactive processing - пример

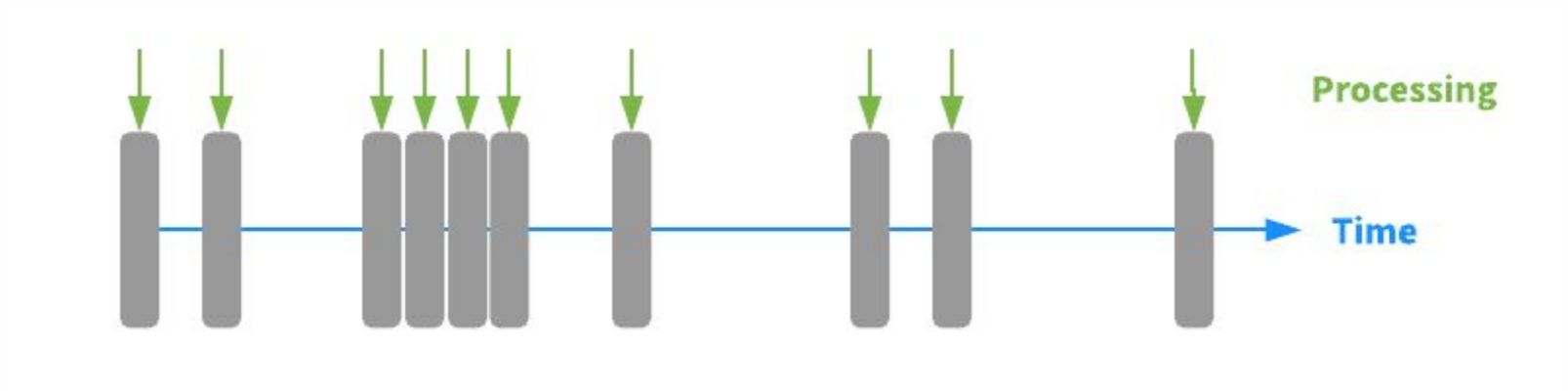
OTUS

```
import akka.actor.Actor
import akka.actor.Props
import akka.event.Logging

class MyActor extends Actor {
  val log = Logging(context.system, this)

  def receive = {
    case "test" => log.info("received test")
    case _      => log.info("received unknown message")
  }
}
```

Reactive processing

Плюсы:
- Subsecond latency
- High scalability
- Easy load-balancing

Минусы:
- Сложно разделять логику обработки событий
- Проблемы при работе со stateful data



Reactive processing - фреймворки

OTUS

02

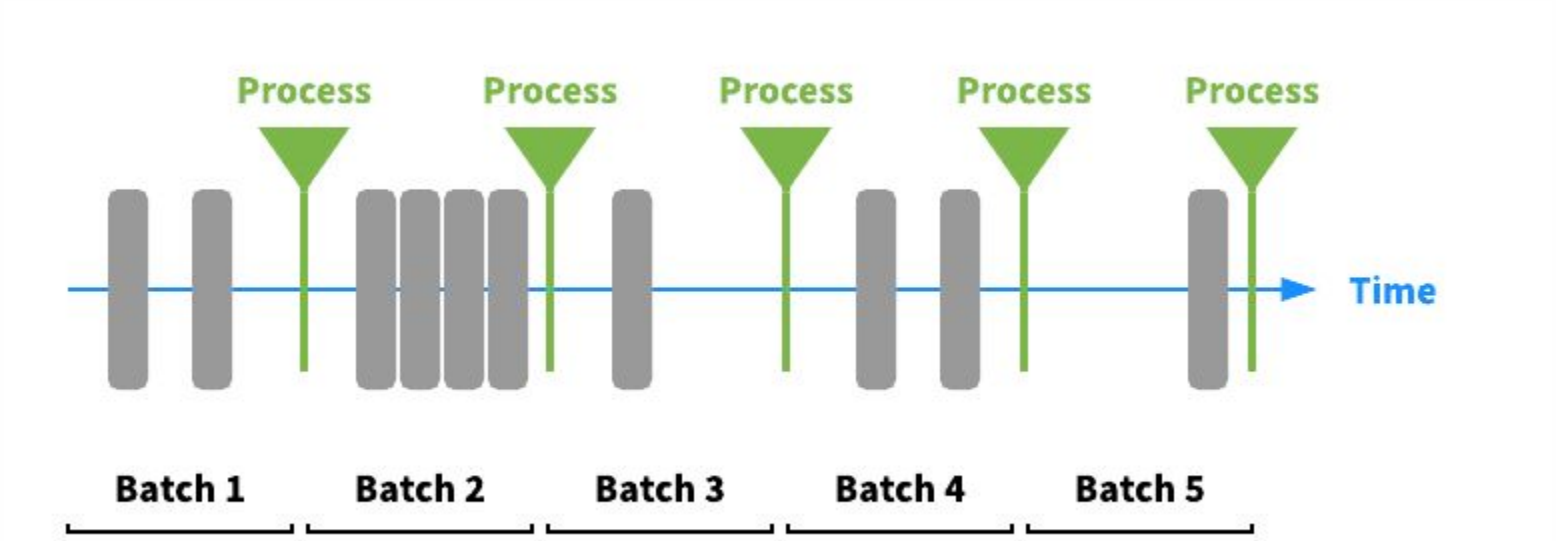
Micro-batch processing

Теория

Micro-batch processing

Подход к процессингу данных, основанный на обработке сообщений небольшими блоками - batches

- процессинг происходит по группе событий
- система-приемник считывает события блоками
- Вычисления могут происходить по пустым данным



Micro-batch processing - пример

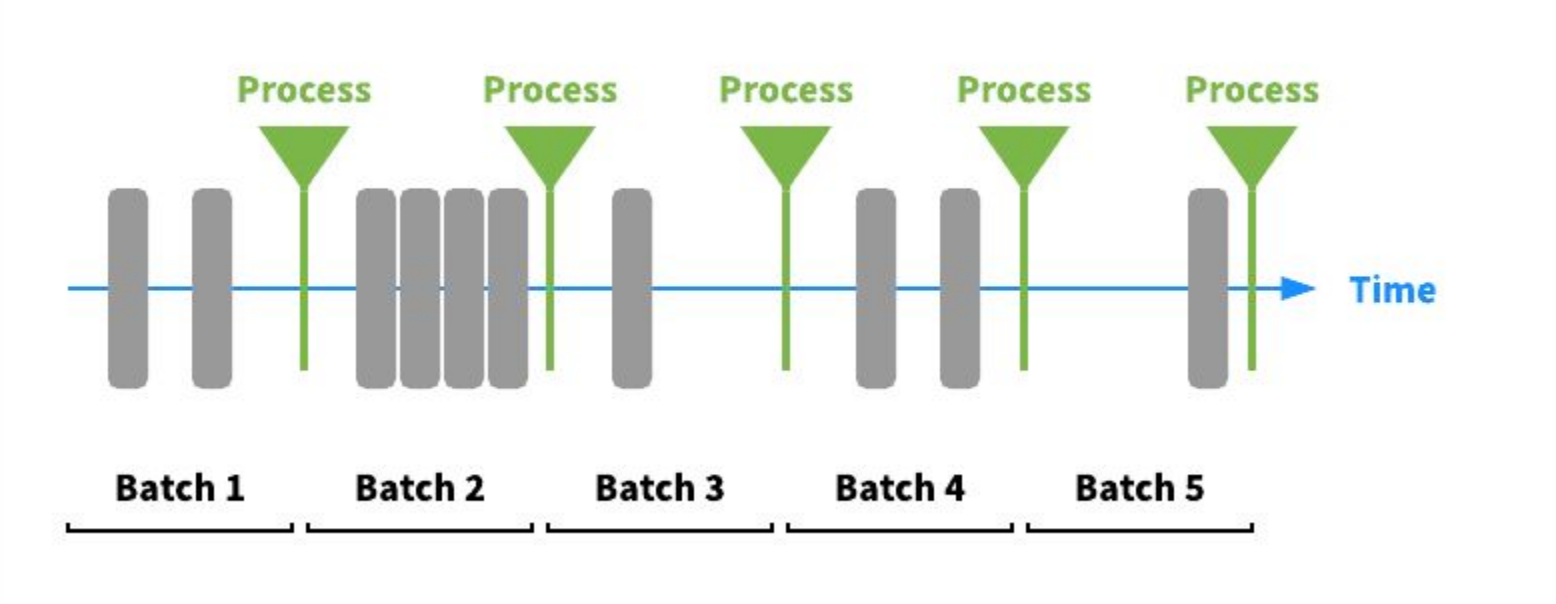
OTUS

```
val df = spark
  .readStream
  .format("kafka")
  .option("kafka.bootstrap.servers", "localhost:9092")
  .option("subscribe", "new-events")
  .load()


val parsed = df.selectExpr("CAST(value AS STRING)", "CAST(timestamp AS
TIMESTAMP)").as[(String, Timestamp)]
  .select(from_json($"value", mySchema).as("data"), $"timestamp")
  .select("data.*", "timestamp")


val grouped = parsed
     .groupBy("type").count()
     .withColumn("processed_at",lit(current_timestamp()))


grouped.writeStream
     .format("parquet")
     .save("s3a://streaming-data-bucket/grouped")
     .mode("append")
     .start()
    .awaitTermination()
```

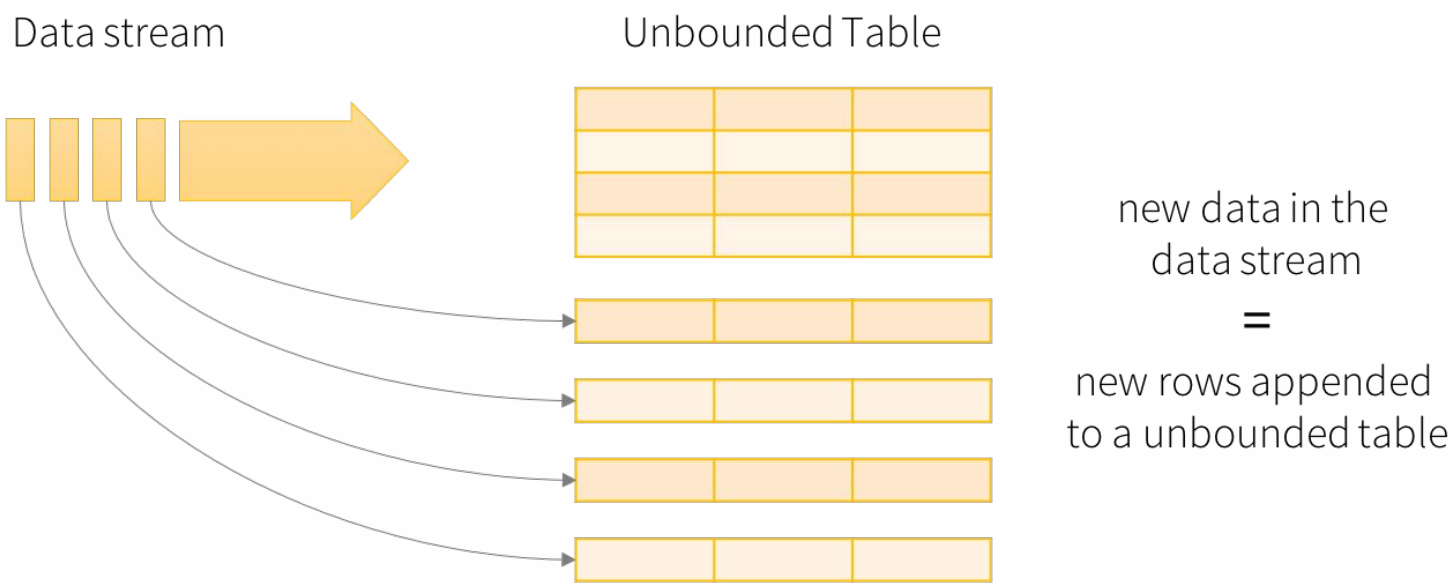
Micro-batch processing

Плюсы:
- Понятный фреймворк обработки батчей
- High scalability
- Легко работать со stateful data

Минусы:
- Сложный дебаг и логгирование
- Stream-to-stram joins
- Сложная обработка burst-load



Micro-batch processing в Apache Spark



Data stream as an unbounded table

04

Практический пример

Трусов Иван

ivan.trusov.contact@gmail.com
https://renardeinside.github.io/

Спасибо
за внимание!