Построение event-driven приложения с Apache Flink

Даниил Туликов | Sales Technologies, Т-Банк

О чём сегодня поговорим

- Задача

О чём сегодня поговорим

- Задача
- Решение в лоб, его плюсы и минусы

О чём сегодня поговорим

- Задача
- Решение в лоб, его плюсы и минусы
- В поисках нового инструмента

О чём сегодня поговорим

- Задача
- Решение в лоб, его плюсы и минусы
- В поисках нового инструмента
- Решение задачи с помощью Apache Flink

О чём сегодня поговорим

- Задача
- Решение в лоб, его плюсы и минусы
- В поисках нового инструмента
- Решение задачи с помощью Apache Flink
- Как флинк справляется с трудностями обнаруженными в рамках решения в лоб

О чём сегодня поговорим

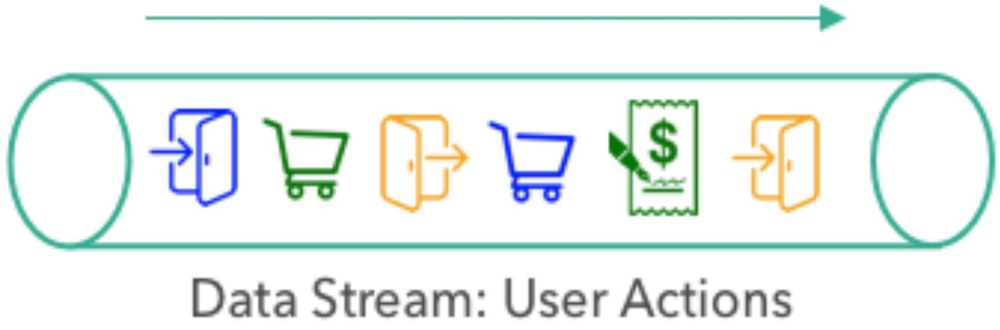
- Задача
- Решение в лоб, его плюсы и минусы
- В поисках нового инструмента
- Решение задачи с помощью Apache Flink
- Как флинк справляется с трудностями обнаруженными в рамках решения в лоб
- Границы применимости

Задача

- Мы e-commerce площадка с **30 млн активных пользователей**

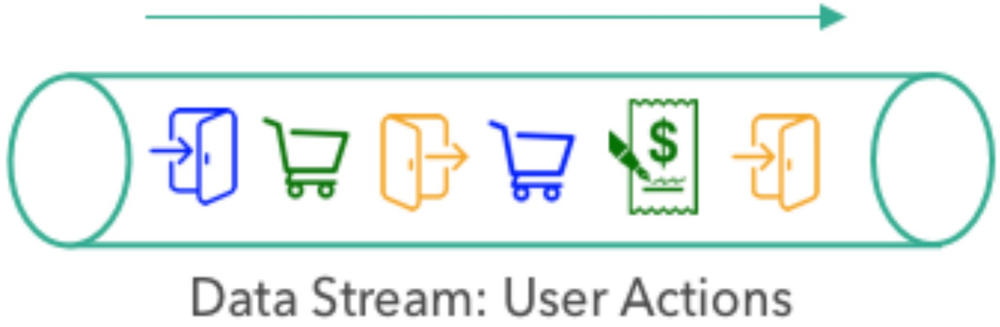
Задача

- Мы e-commerce площадка с **30 млн активных пользователей**
- На вход получаем **поток действий** совершенных пользователями

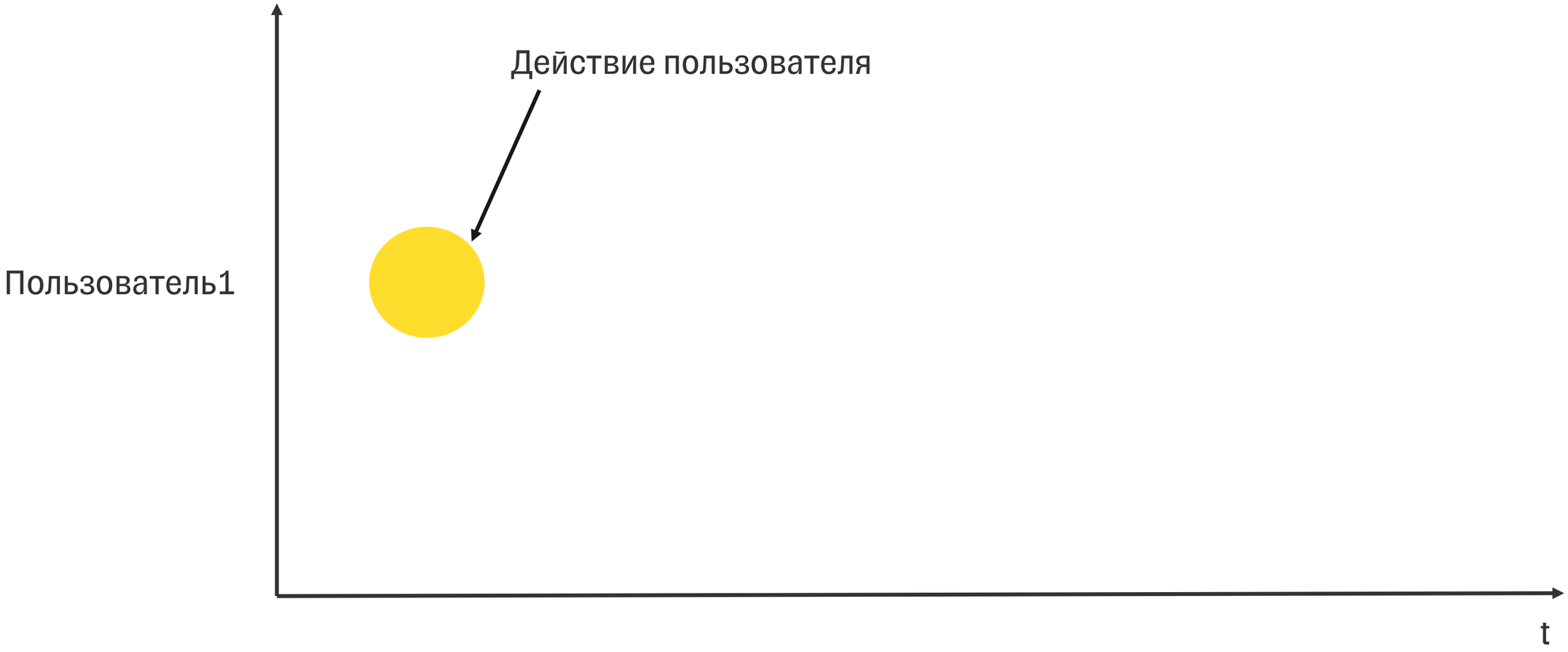


Задача

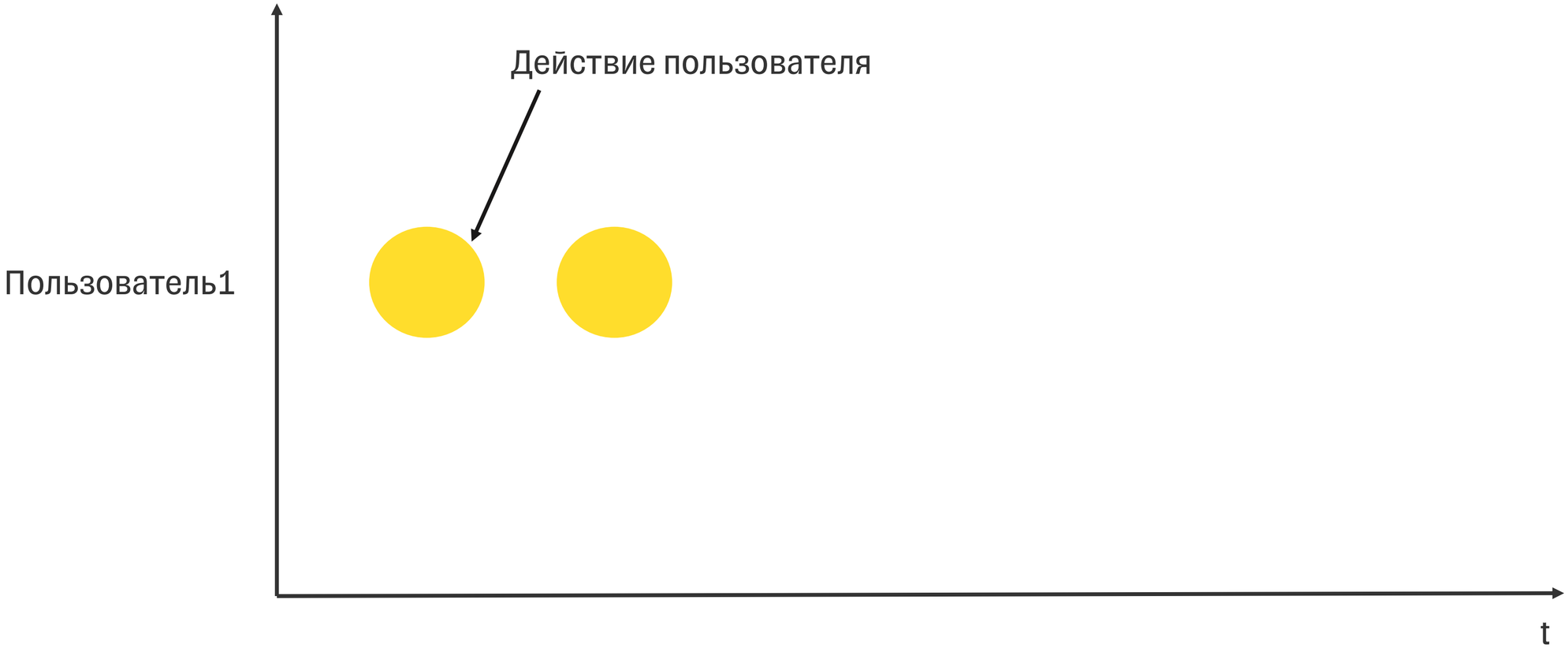
- Мы e-commerce площадка с **30 млн активных пользователей**
- На вход получаем **поток действий** совершенных пользователями
- Хотели бы анализировать пользовательские сессии **в реальном времени** и на основе этого анализа запускать промотирование определенных продуктов для пользователя



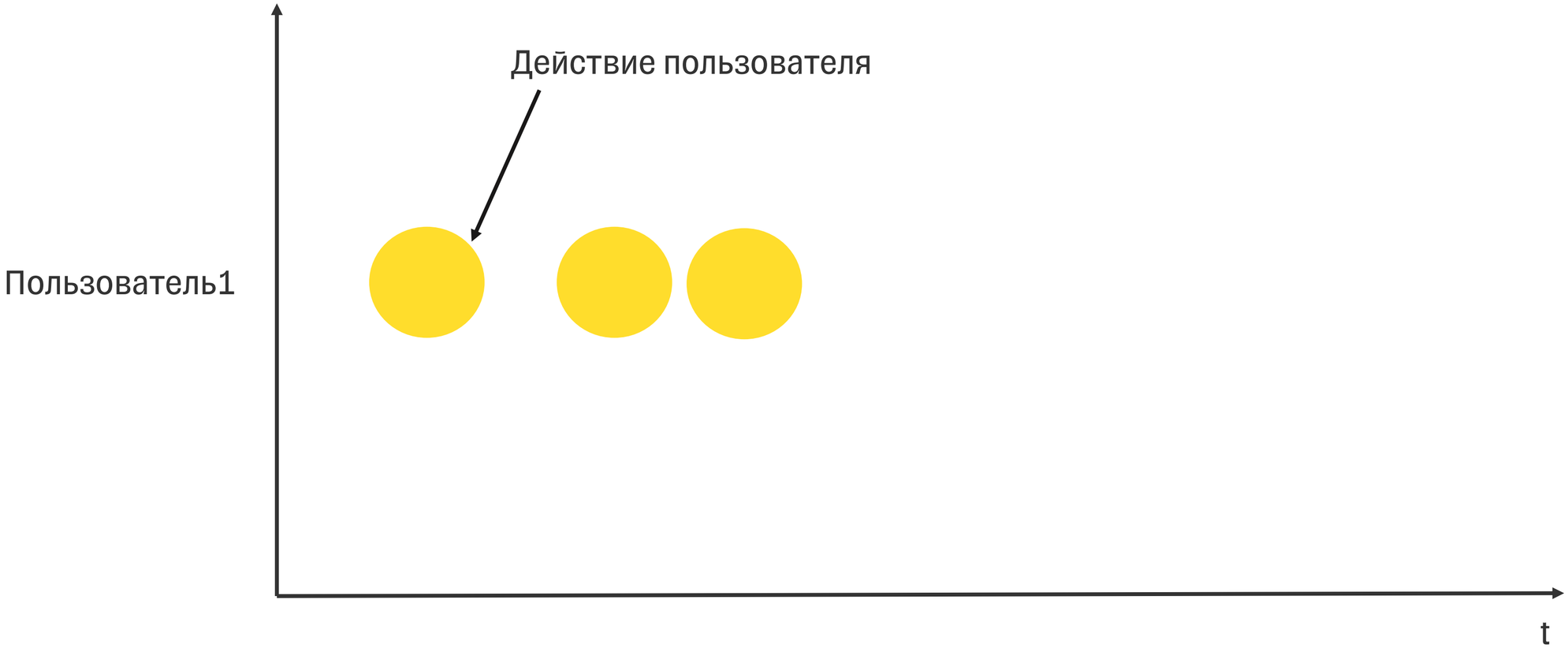
Что будем считать за пользовательскую сессию? – бездействие пользователя



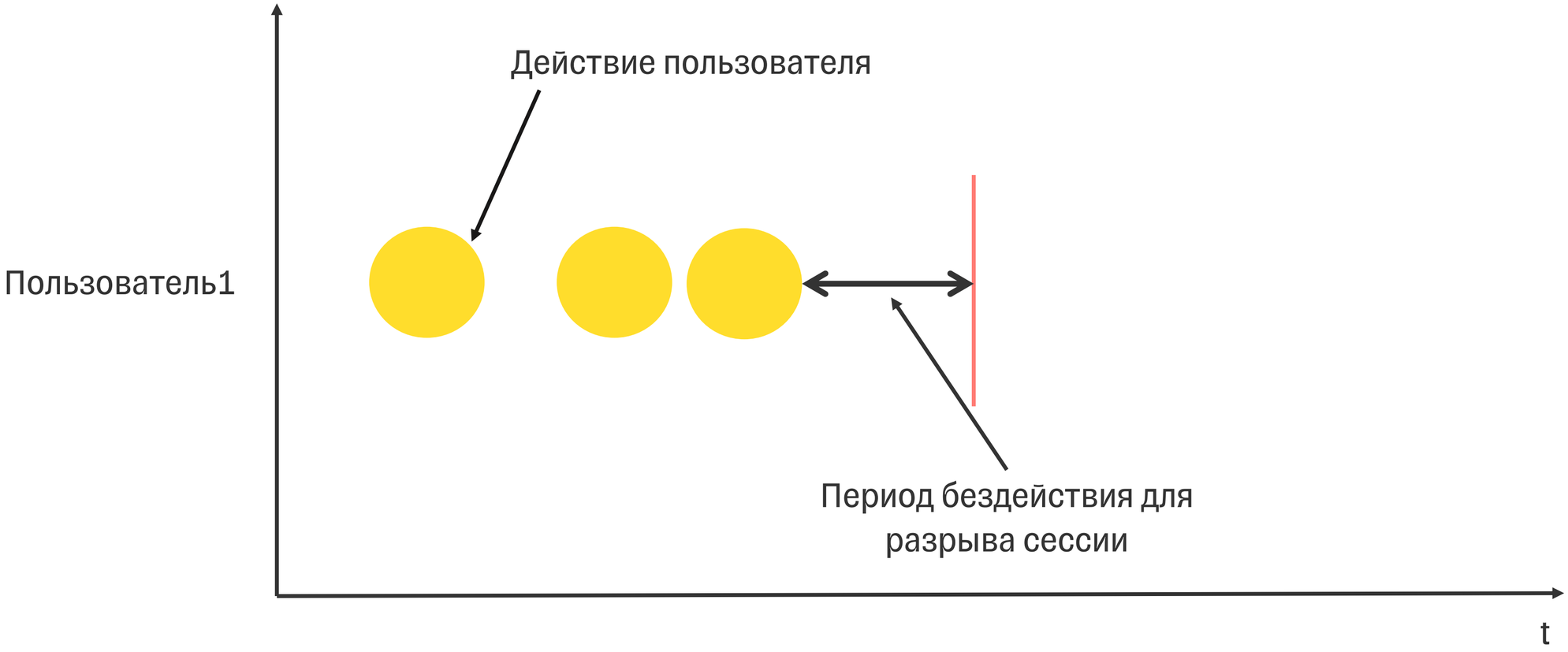
Что будем считать за пользовательскую сессию? – бездействие пользователя



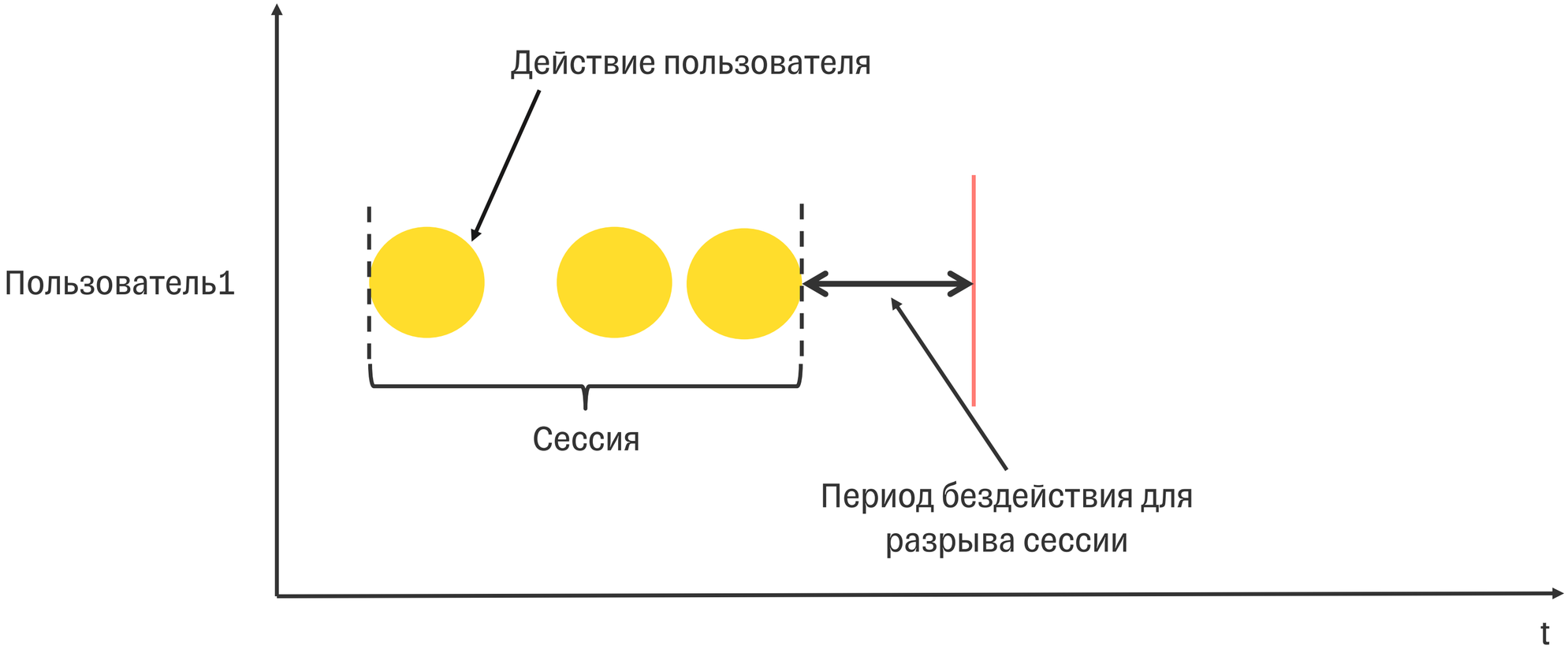
Что будем считать за пользовательскую сессию? – бездействие пользователя



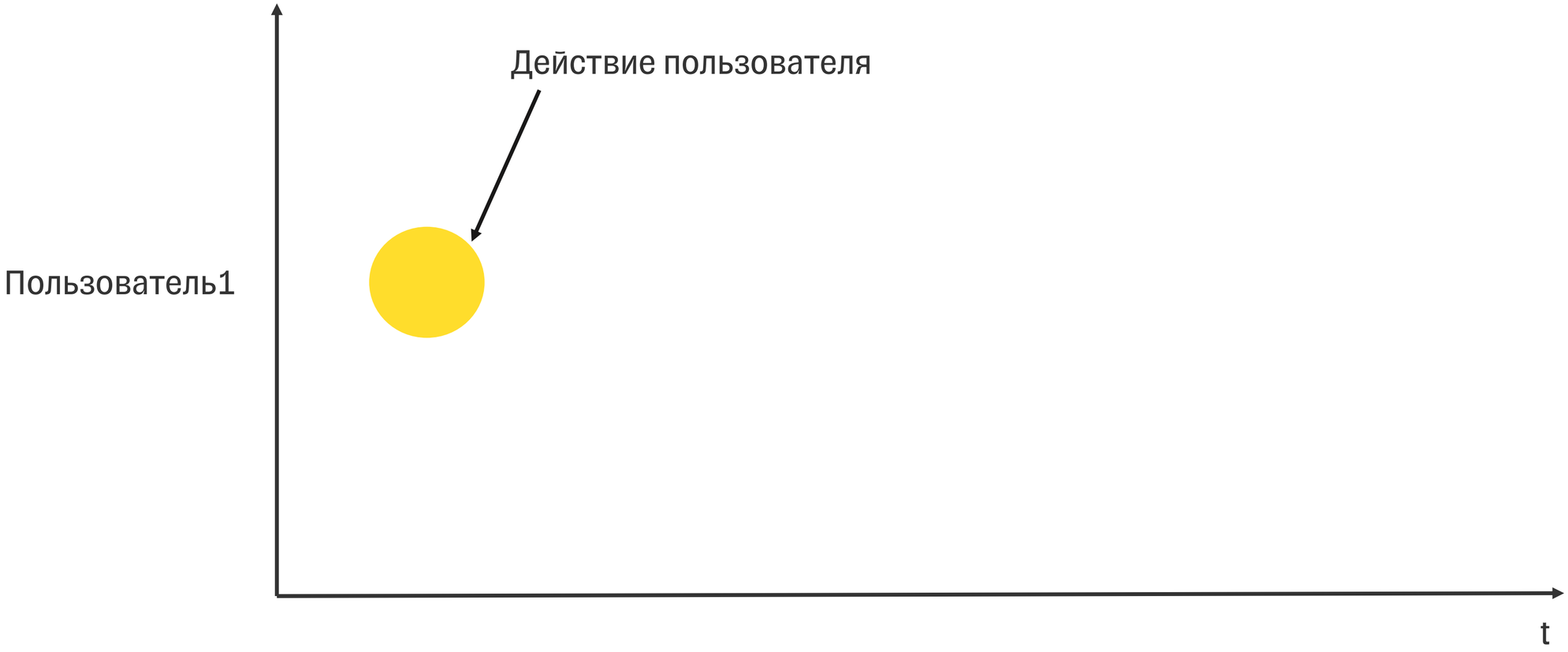
Что будем считать за пользовательскую сессию? – бездействие пользователя



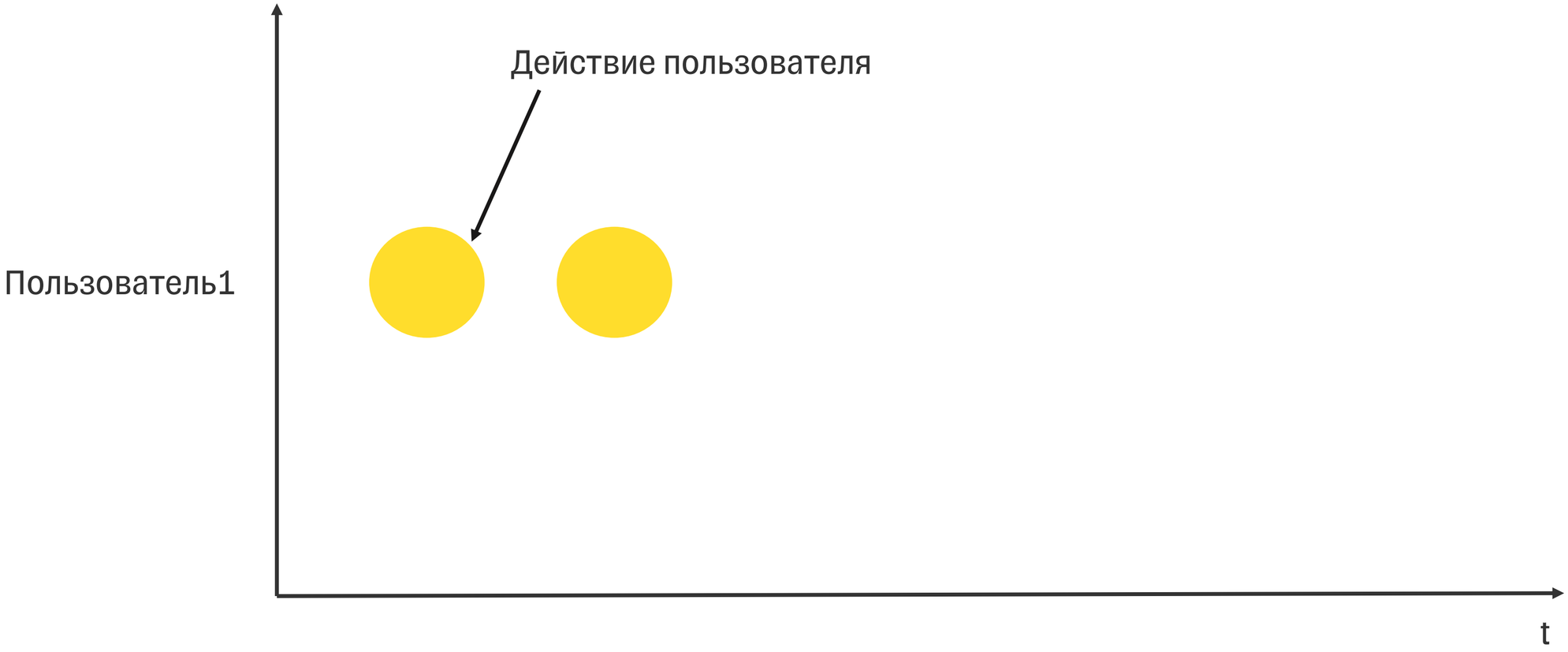
Что будем считать за пользовательскую сессию? – бездействие пользователя



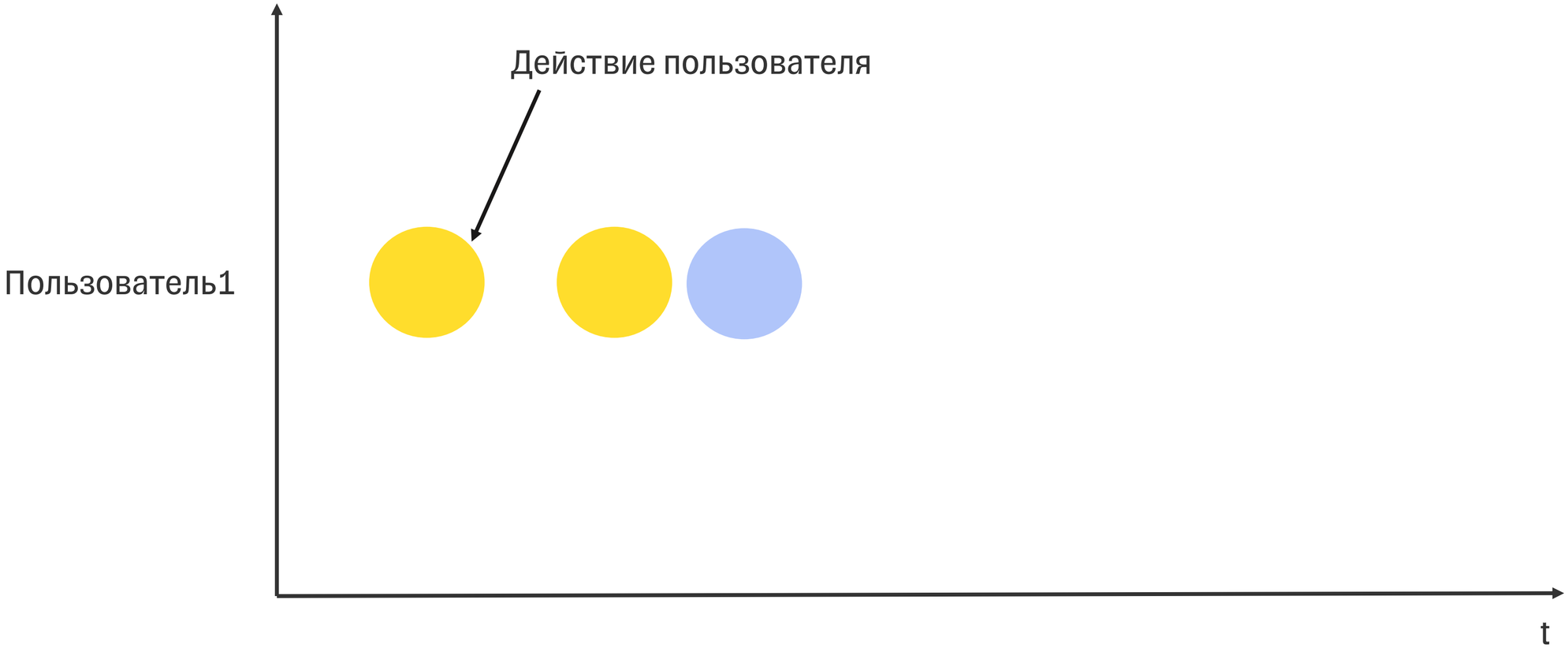
Что будем считать за пользовательскую сессию? – триггерное действие



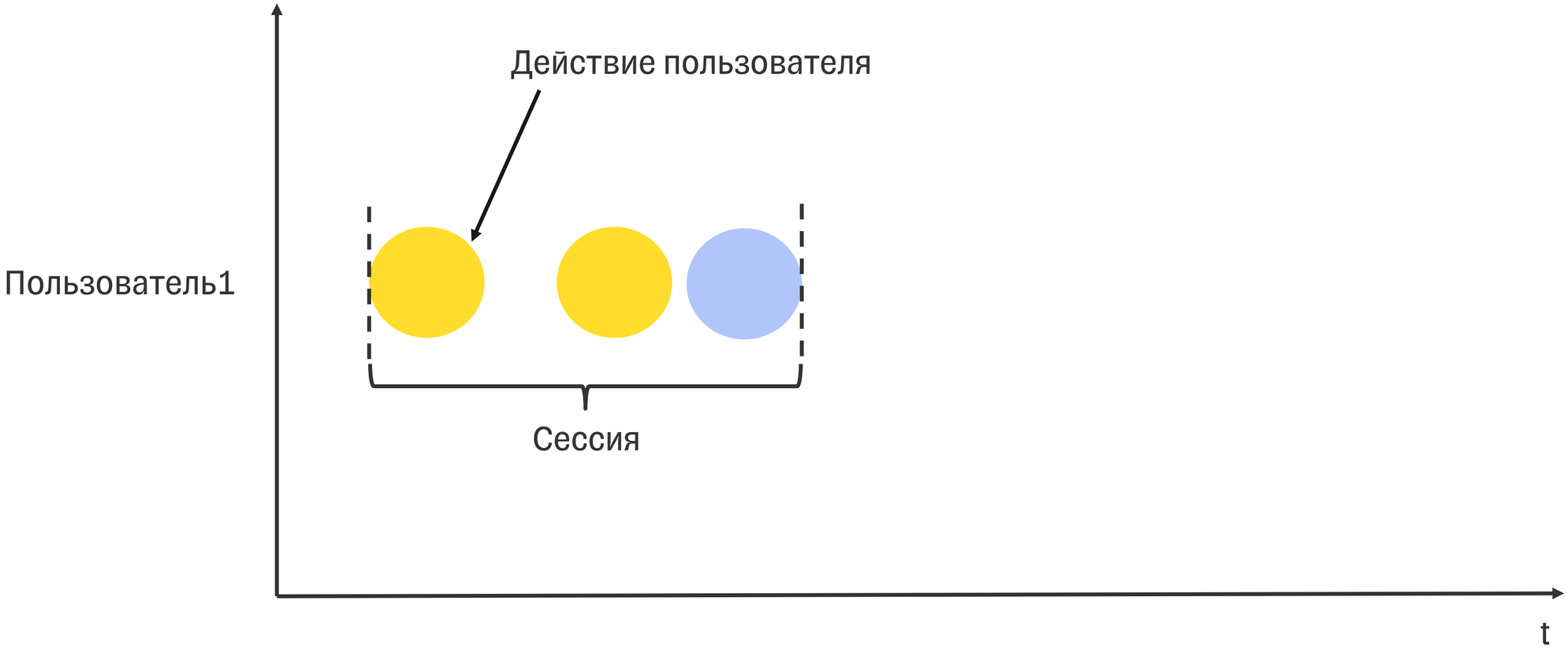
Что будем считать за пользовательскую сессию? – триггерное действие



Что будем считать за пользовательскую сессию? – триггерное действие

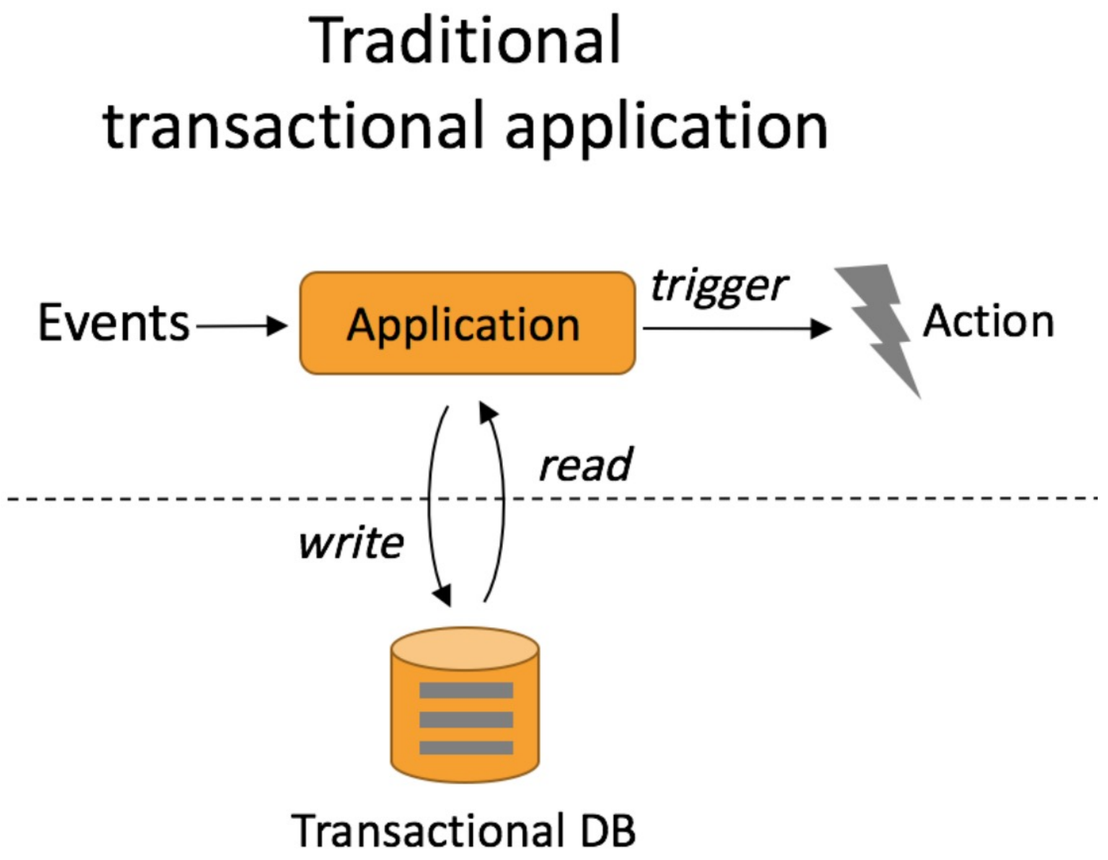


Что будем считать за пользовательскую сессию? – триггерное действие

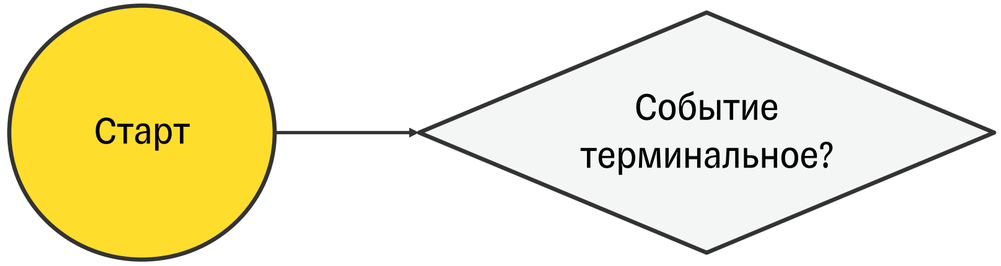


- Задача
- **Решение в лоб, его плюсы и минусы**
- В поисках нового инструмента
- Решение задачи с помощью Apache Flink
- Как флинк справляется с трудностями обнаруженными в рамках решения в лоб
- Границы применимости

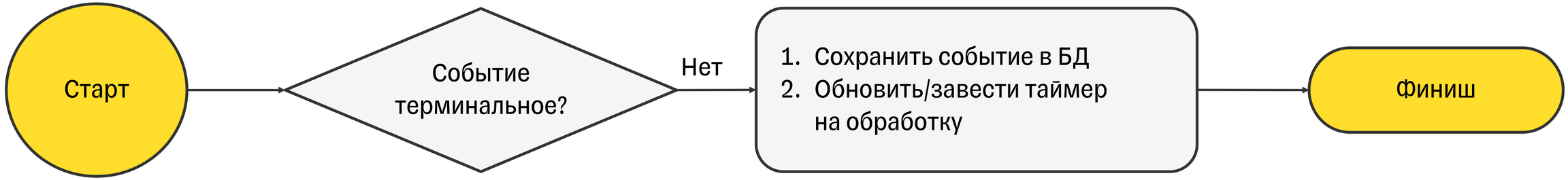
Решение в лоб - реагировать на каждое действие пользователя



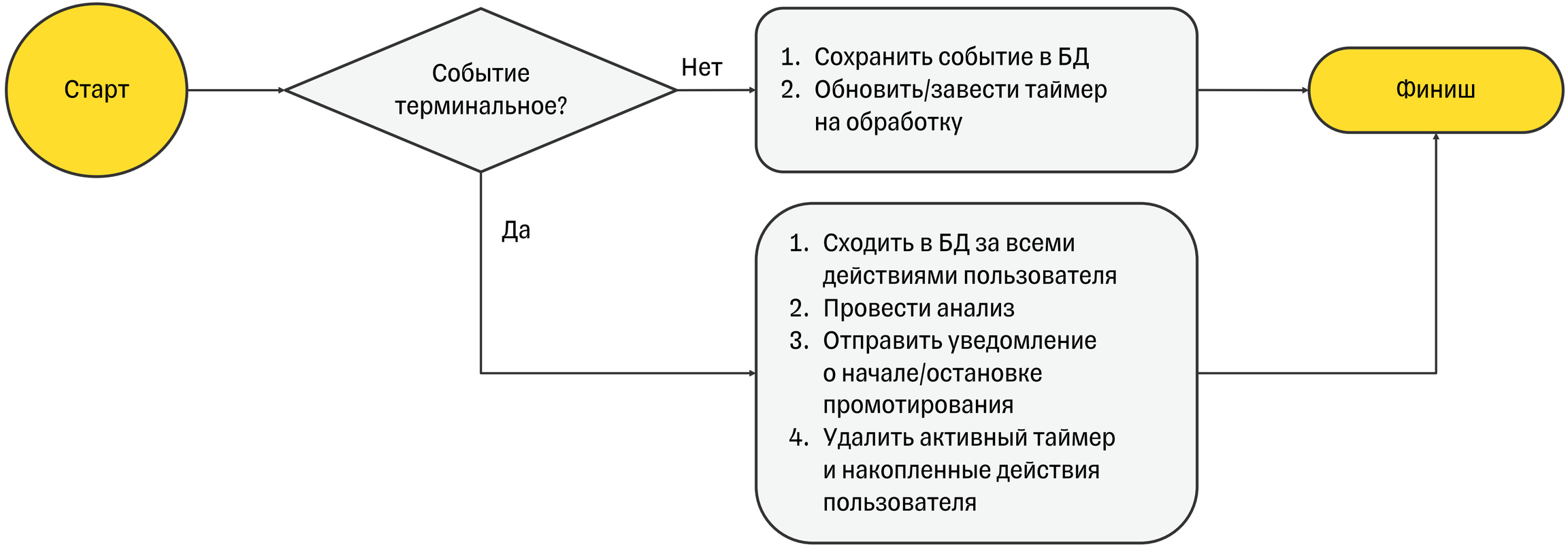
Решение в лоб – алгоритм при поступлении события



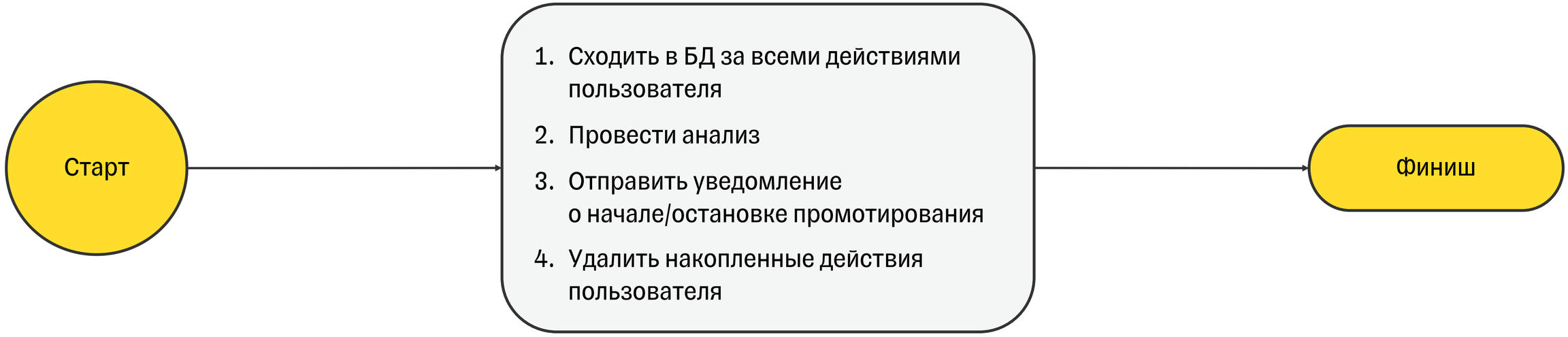
Решение в лоб – алгоритм при поступлении события



Решение в лоб – алгоритм при поступлении события



Решение в лоб – алгоритм при срабатывании таймера



Решение в лоб - мысли

- Рост нагрузки (проблемы масштабирования, нагрузка на БД)

Решение в лоб - мысли

- Рост нагрузки (проблемы масштабирования, нагрузка на БД)
- Эволюционирование бизнес требований

Решение в лоб - мысли

- Рост нагрузки (проблемы масштабирования, нагрузка на БД)
- Эволюционирование бизнес требований
- Сбои (дубликаты/потери данных)

Решение в лоб - мысли

- Рост нагрузки (проблемы масштабирования, нагрузка на БД)
- Эволюционирование бизнес требований
- Сбои (дубликаты/потери данных)
- Out-of-order данные?

- Задача
- Решение в лоб, его плюсы и минусы
- **В поисках нового инструмента**
- Решение задачи с помощью Apache Flink
- Как флинк справляется с трудностями обнаруженными в рамках решения в лоб
- Границы применимости

В поисках нового инструмента

Batch processing

- работаем с конечным набором данных (нет out-of-order событий, легче восстановиться в случае сбоя)

В поисках нового инструмента

Batch processing

- работаем с конечным набором данных (нет out-of-order событий, легче восстановиться в случае сбоя)
- высокая задержка

В поисках нового инструмента

- Добавленные данные сразу доступны для запросов

В поисках нового инструмента

- Добавленные данные сразу доступны для запросов
- Потребуется отдельное приложение, которое будет опрашивать итоговую таблицу

В поисках нового инструмента

- Добавленные данные сразу доступны для запросов
- Потребуется отдельное приложение, которое будет опрашивать итоговую таблицу
- Сможем ли придумать подходящий запрос?

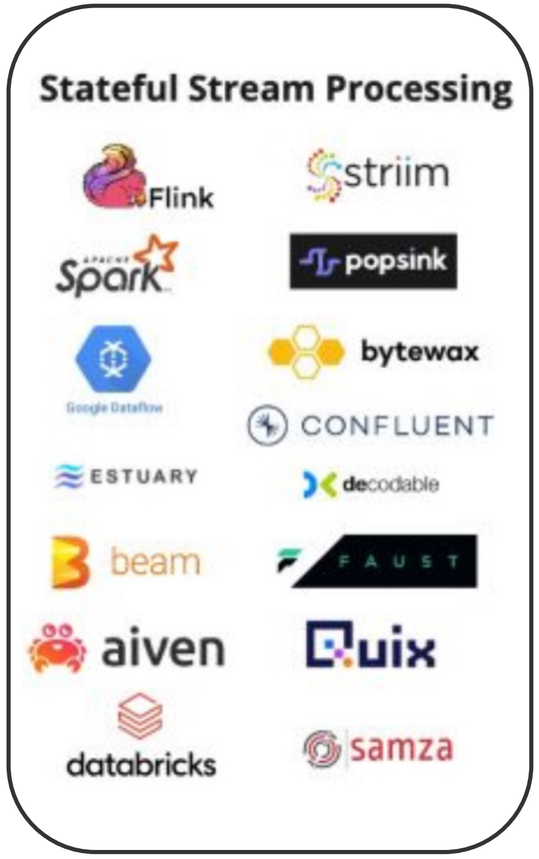
В поисках нового инструмента

- Заточены на обработку потоков данных в реальном времени в отличии от real-time OLAP баз (source-s, sink-s, windows, low latencies)

В поисках нового инструмента

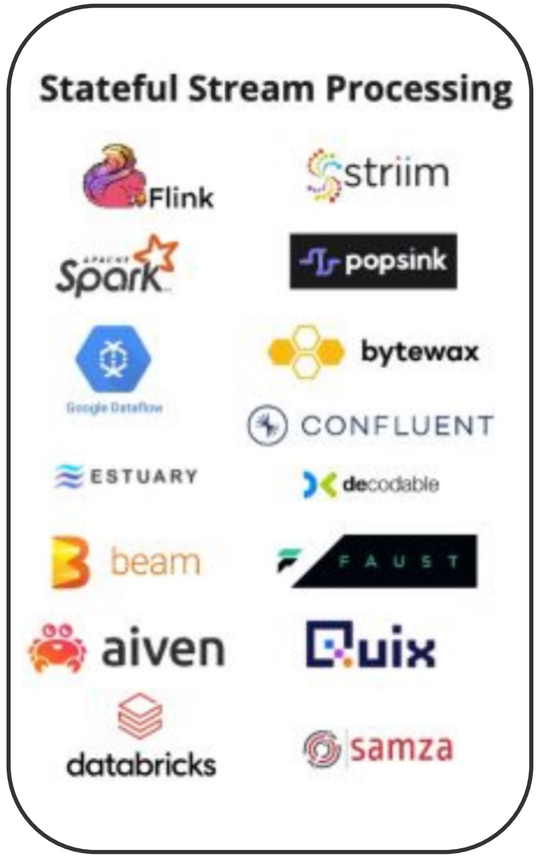
- Заточены на обработку потоков данных в реальном времени в отличии от real-time OLAP баз (source-s, sink-s, windows, low latencies)
- Хватит ли одного SQL?

В поисках нового инструмента



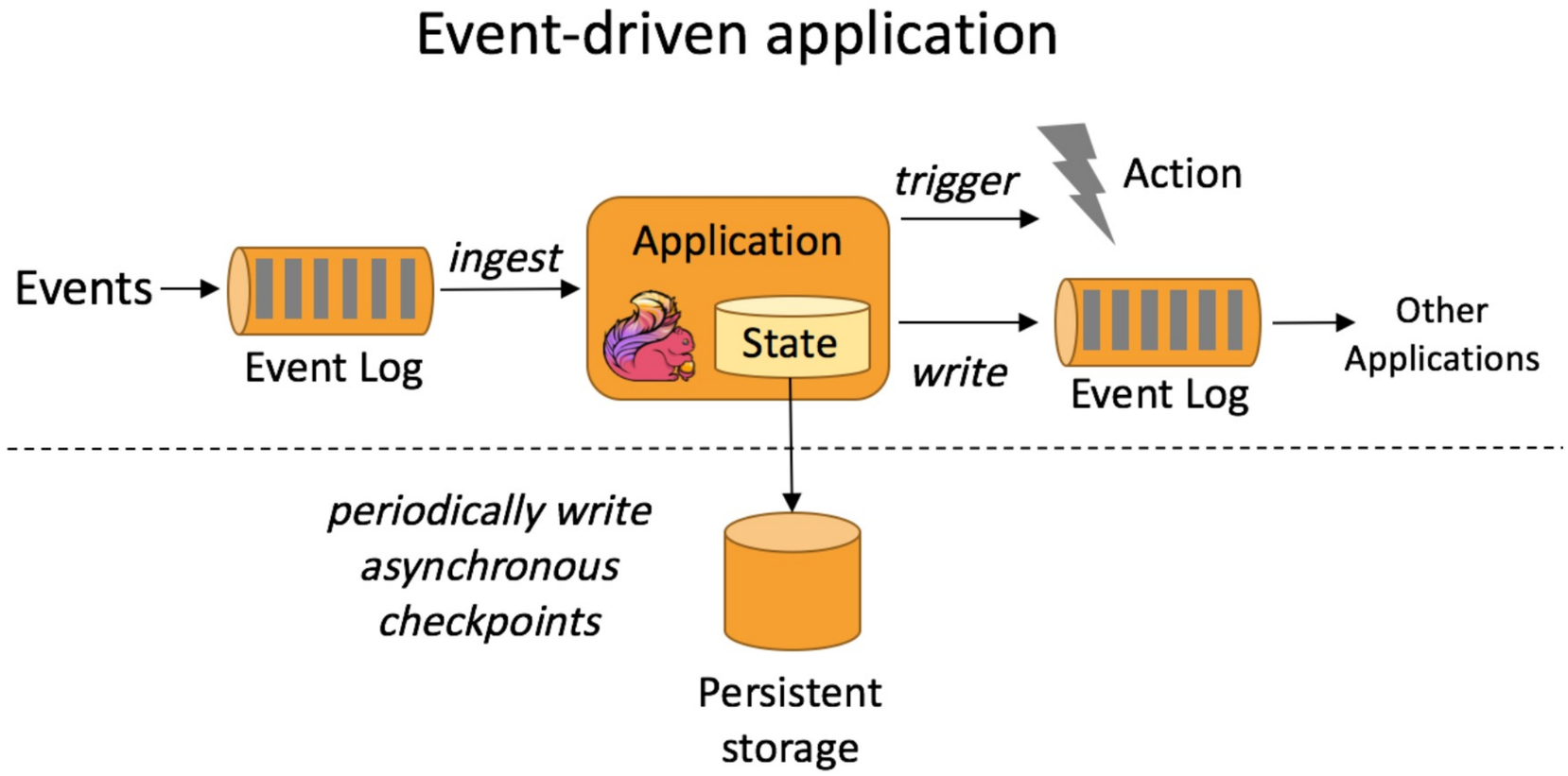
- Заточены на обработку потоков данных в реальном времени (source-s, sink-s, windows, low latencies)

В поисках нового инструмента



- Заточены на обработку потоков данных в реальном времени (source-s, sink-s, windows, low latencies)
- Доступны разные виды API

Event-driven приложение на базе фреймворка потоковой обработки



UBER

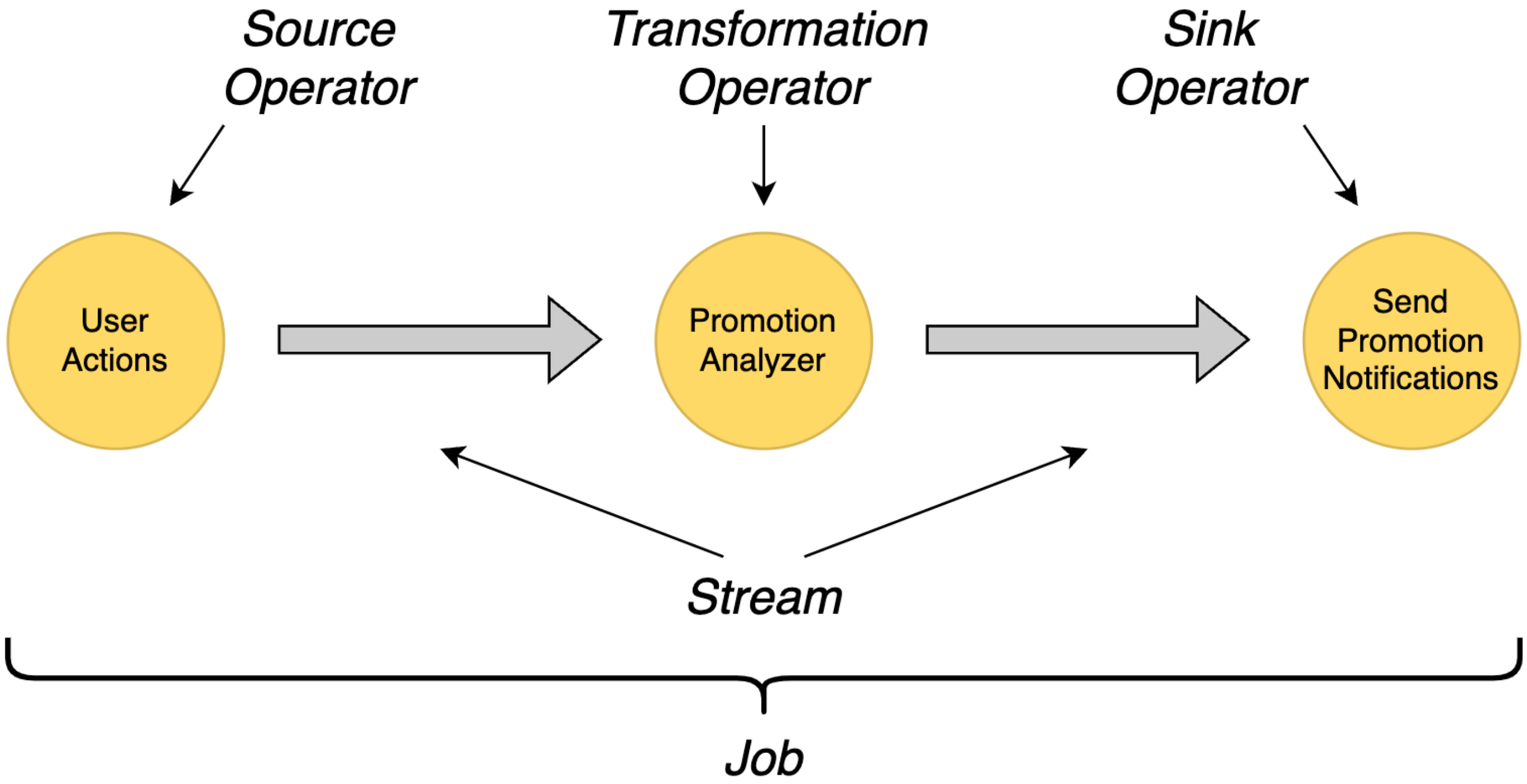
Apache Flink

Компании заявляют о следующих фактах:

- Способен выдерживать нагрузки в **сотни тысяч RPS**
- Кластер успешно работает на **тысячах машин**
- Оперируют **стейтом в десятки терабайт**

- Задача
- Решение в лоб, его плюсы и минусы
- В поисках нового инструмента
- **Решение задачи с помощью Apache Flink**
- Как флинк справляется с трудностями обнаруженными в рамках решения в лоб
- Границы применимости

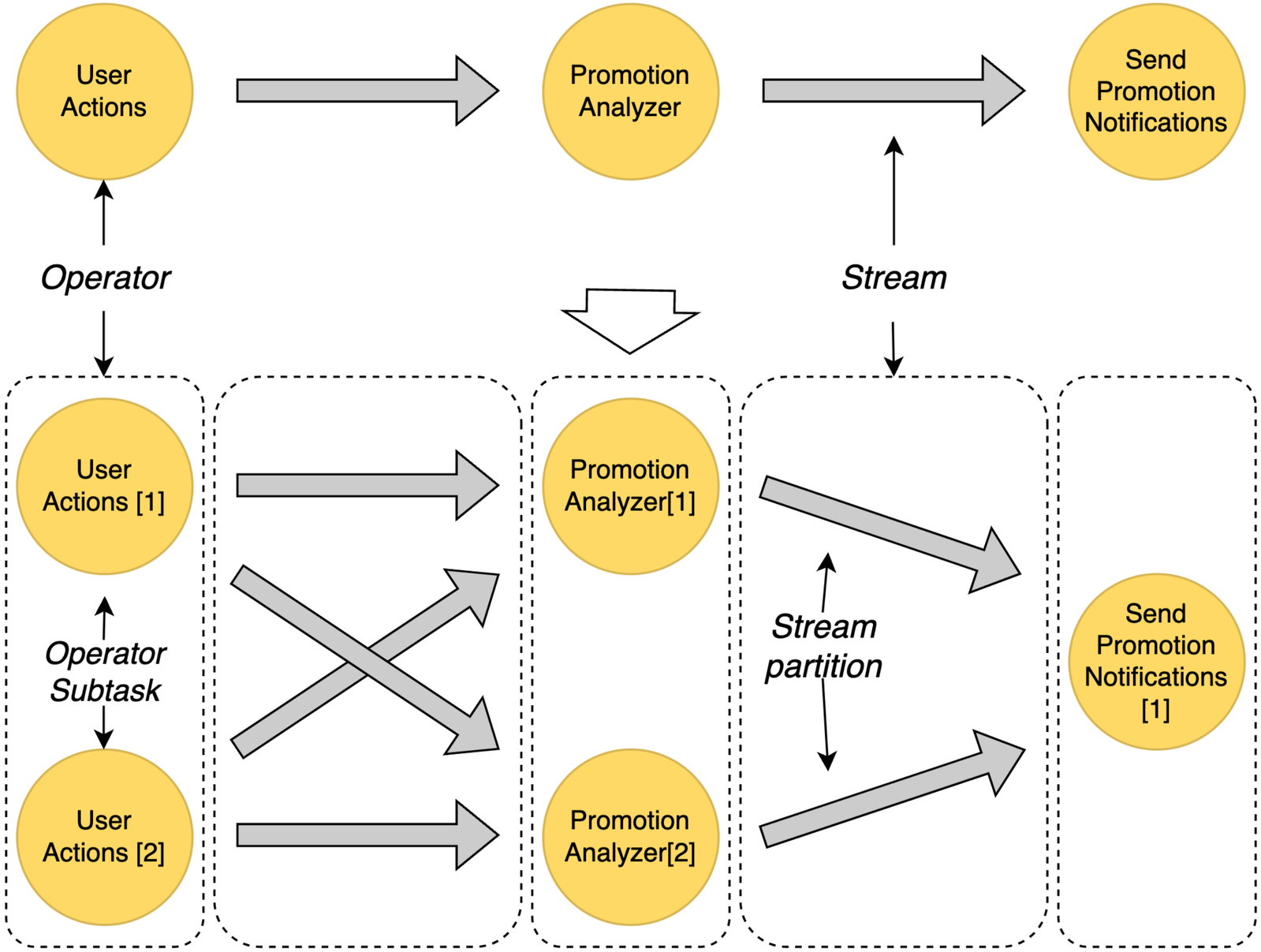
Решение с использованием Apache Flink



```
public static void main(String[] args) throws Exception {
    StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

    DataStream<UserAction> userActions = env
        .addSource(new UserActionSource())
        .name("user-actions");

    DataStream<PromotionNotification> promotionNotifications = userActions
        .keyBy(UserAction::getUserId)
        .window(ProcessingTimeSessionWindows.withGap(Time.minutes(5)))
        .trigger(PromotionAnalysisTrigger.create())
        .process(new PromotionAnalyzer())
        .name("promotion-analyzer");

    promotionNotifications
        .addSink(new PromotionNotificationSink())
        .name("send-promotion-notifications");

    env.execute("Promotion Analysis");
}
```

```
public class PromotionAnalyzer extends
    ProcessWindowFunction<UserAction, PromotionNotification, String, TimeWindow> {

    @Override
    public void process(
        String userId,
        ProcessWindowFunction<UserAction, PromotionNotification, String, TimeWindow>.Context context,
        Iterable<UserAction> userActions,
        Collector<PromotionNotification> out
    ) throws Exception {
        if (shouldStopPromotion(userActions)) {
            out.collect(new PromotionNotification(userId, Type.STOP_PROMOTION));
        } else if (shouldStartPromotion(userActions)) {
            out.collect(new PromotionNotification(userId, Type.START_PROMOTION));
        }
    }
}
```

```
public static void main(String[] args) throws Exception {
    StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

    DataStream<UserAction> userActions = env
        .addSource(new UserActionSource())
        .setParallelism(2)
        .name("user-actions");

    DataStream<PromotionNotification> promotionNotifications = userActions
        .keyBy(UserAction::getUserId)
        .window(ProcessingTimeSessionWindows.withGap(Time.minutes(5)))
        .trigger(PromotionAnalysisTrigger.create())
        .process(new PromotionAnalyzer())
        .setParallelism(2)
        .name("promotion-analyzer");

    promotionNotifications
        .addSink(new PromotionNotificationSink())
        .name("send-promotion-notifications");

    env.execute("Promotion Analysis");
}
```

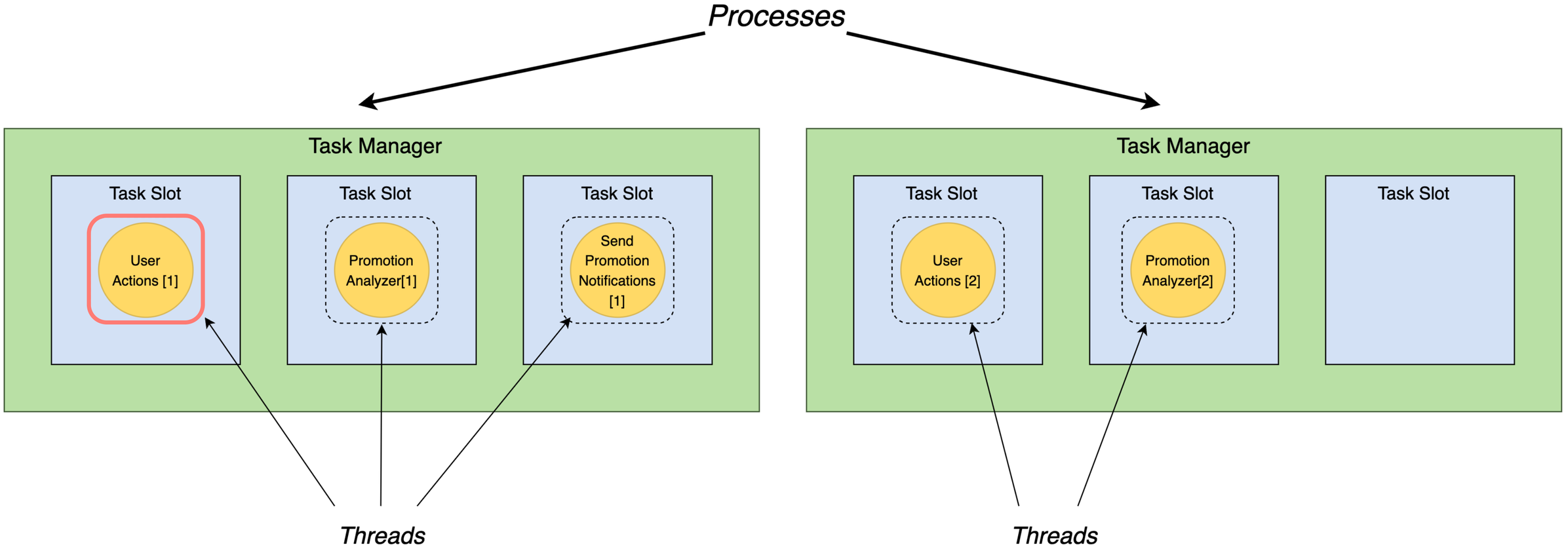

User Actions
Promotion Analyzer
Send Promotion Notifications
Operator
Stream
User Actions [1]
Promotion Analyzer[1]
Operator Subtask
Stream partition
Send Promotion Notifications [1]
User Actions [2]
Promotion Analyzer[2]

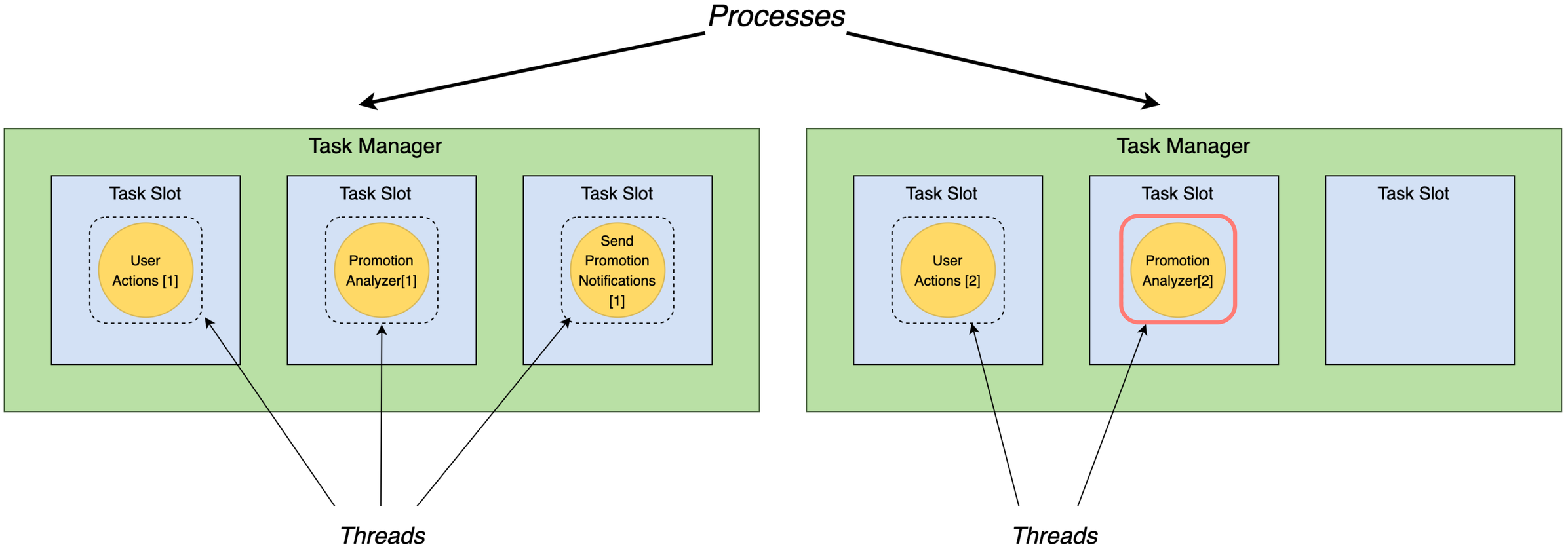
```
public static void main(String[] args) throws Exception {
    StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

    DataStream<UserAction> userActions = env
        .addSource(new UserActionSource())
        .setParallelism(2)
        .name("user-actions");

    DataStream<PromotionNotification> promotionNotifications = userActions
        .keyBy(UserAction::getUserId)
        .window(ProcessingTimeSessionWindows.withGap(Time.minutes(5)))
        .trigger(PromotionAnalysisTrigger.create())
        .process(new PromotionAnalyzer())
        .setParallelism(2)
        .name("promotion-analyzer");

    promotionNotifications
        .addSink(new PromotionNotificationSink())
        .name("send-promotion-notifications");

    env.execute("Promotion Analysis");
}
```

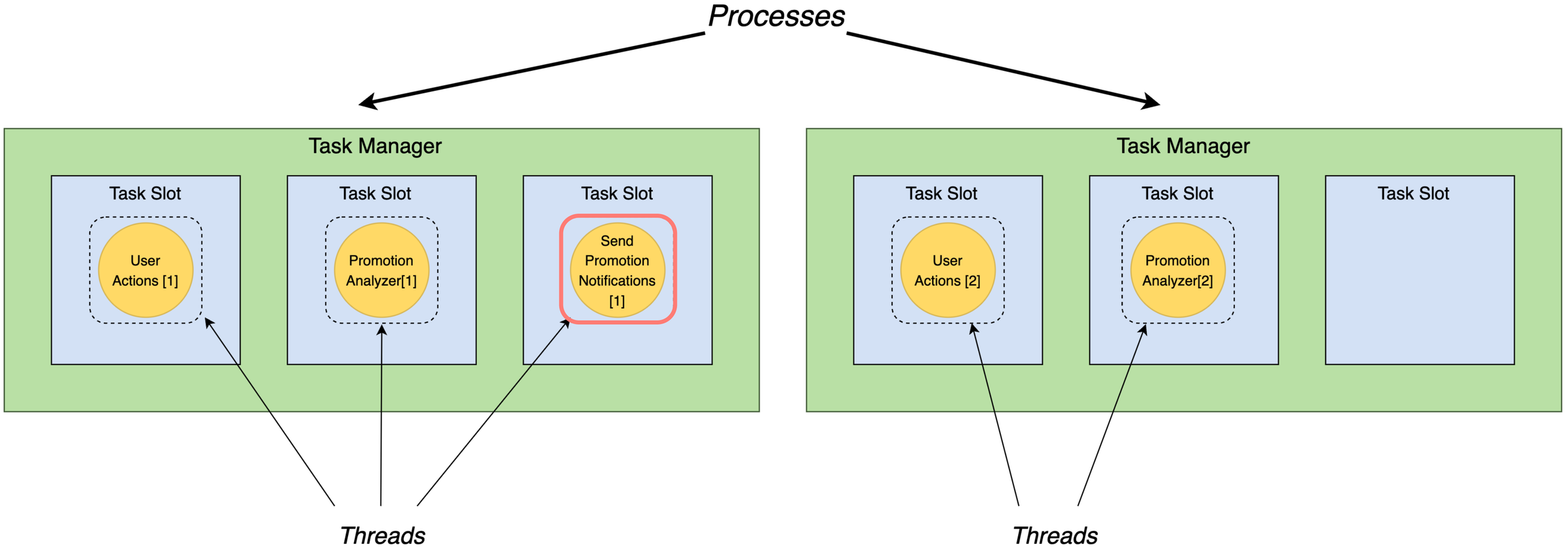
```
public static void main(String[] args) throws Exception {
    StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

    DataStream<UserAction> userActions = env
        .addSource(new UserActionSource())
        .setParallelism(2)
        .name("user-actions");

    DataStream<PromotionNotification> promotionNotifications = userActions
        .keyBy(UserAction::getUserId)
        .window(ProcessingTimeSessionWindows.withGap(Time.minutes(5)))
        .trigger(PromotionAnalysisTrigger.create())
        .process(new PromotionAnalyzer())
        .setParallelism(2)
        .name("promotion-analyzer");

    promotionNotifications
        .addSink(new PromotionNotificationSink())
        .name("send-promotion-notifications");

    env.execute("Promotion Analysis");
}
```

```java
public class PromotionAnalysisTrigger extends Trigger<UserAction, TimeWindow> {

    @Override
    public TriggerResult onElement(
        UserAction userAction, long timestamp, TimeWindow window, TriggerContext ctx) {

        if (userAction.getType() == Type.LOG_OUT) {
            return TriggerResult.FIRE_AND_PURGE;
        }

        ctx.registerProcessingTimeTimer(window.maxTimestamp());

        return TriggerResult.CONTINUE;
    }

    @Override
    public TriggerResult onProcessingTime(long time, TimeWindow window, TriggerContext ctx) {
        return TriggerResult.FIRE_AND_PURGE;
    }

}
```

```
public static void main(String[] args) throws Exception {
    StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

    DataStream<UserAction> userActionsFromAuthorization = env
        .addSource(new UserActionAuthorizationSource())
        .setParallelism(2)
        .name("user-actions-authorization");

    DataStream<UserAction> userActionsFromCart = env
        .addSource(new UserActionCartSource())
        .setParallelism(2)
        .name("user-actions-cart");

    DataStream<UserAction> userActions = userActionsFromAuthorization
        .union(userActionsFromCart);

    DataStream<PromotionNotification> promotionNotifications = userActions
        .keyBy(UserAction::getUserId)
        .window(ProcessingTimeSessionWindows.withGap(Time.minutes(5)))
        .trigger(PromotionAnalysisTrigger.create())
        .process(new PromotionAnalyzer())
        .name("promotion-analyzer");

    promotionNotifications
        .addSink(new PromotionNotificationSink())
        .name("send-promotion-notifications");

    env.execute("Promotion Analysis");
}
```


Processes
Task Manager
Task Slot
User Actions [1]
Task Slot
Promotion Analyzer[1]
Task Slot
Send Promotion Notifications [1]
Threads
Task Manager
Task Slot
User Actions [2]
Task Slot
Promotion Analyzer[2]
Task Slot
Threads


Processes
Task Manager
Task Slot
User Actions [1]
Task Slot
Promotion Analyzer[1]
Task Slot
Send Promotion Notifications [1]
Threads
Task Manager
Task Slot
User Actions [2]
Task Slot
Promotion Analyzer[2]
Task Slot
Threads


Processes
Task Manager
Task Slot
User Actions [1]
Task Slot
Promotion Analyzer[1]
Task Slot
Send Promotion Notifications [1]
Threads
Task Manager
Task Slot
User Actions [2]
Task Slot
Promotion Analyzer[2]
Task Slot
Threads

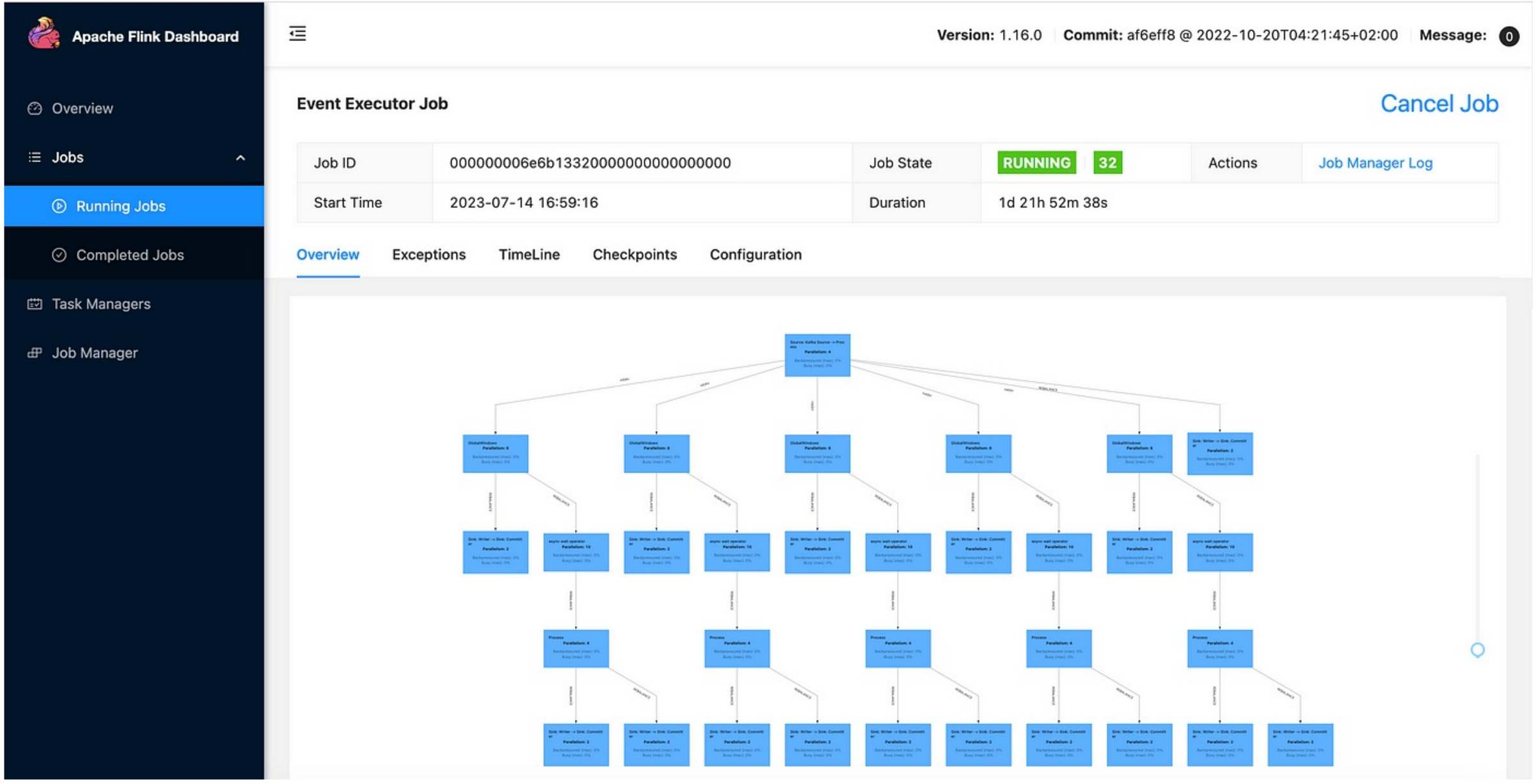
- Задача
- Решение в лоб, его плюсы и минусы
- В поисках нового инструмента
- Решение задачи с помощью Apache Flink
- **Как флинк справляется с трудностями обнаруженными в рамках решения в лоб**
- Границы применимости

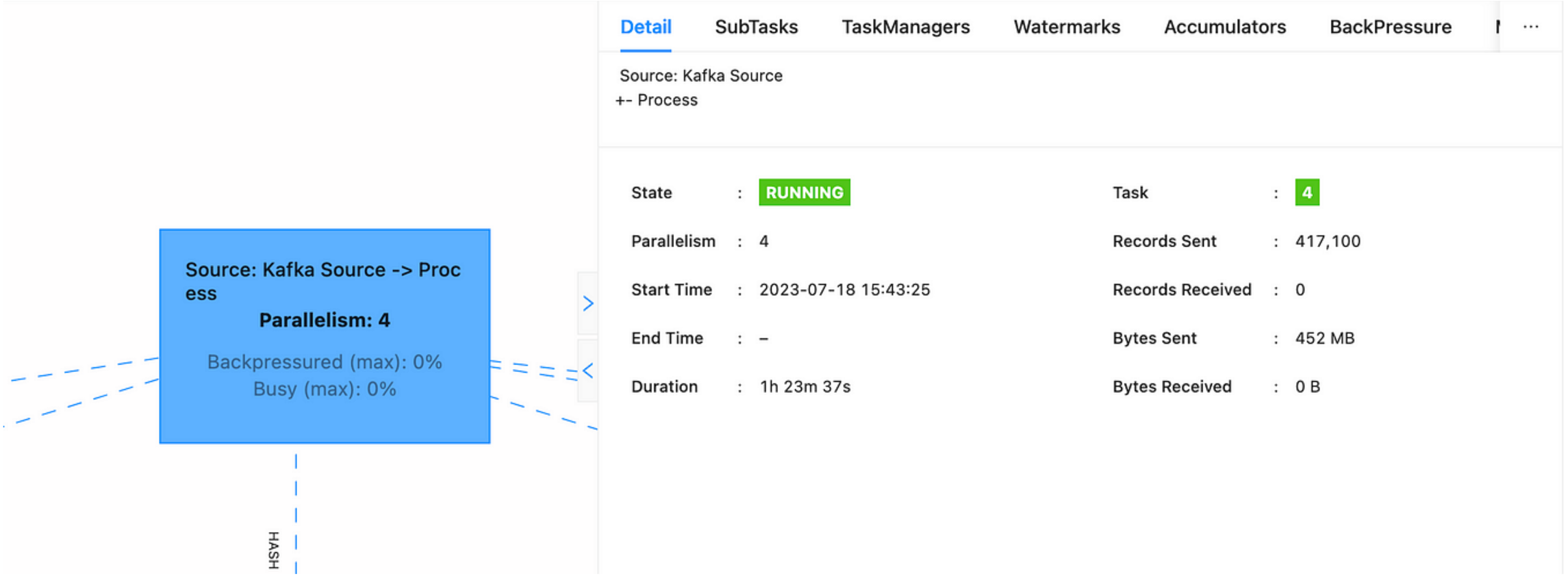
Трудности обнаруженные в рамках решения в лоб

- **Рост нагрузки (проблемы масштабирования, нагрузка на БД)**
- Эволюционирование бизнес требований
- Сбои (дубликаты/потери данных)
- Out-of-order данные?

Трудности обнаруженные в рамках решения в лоб

- Рост нагрузки (проблемы масштабирования, нагрузка на БД)
- **Эволюционирование бизнес требований**
- Сбои (дубликаты/потери данных)
- Out-of-order данные?

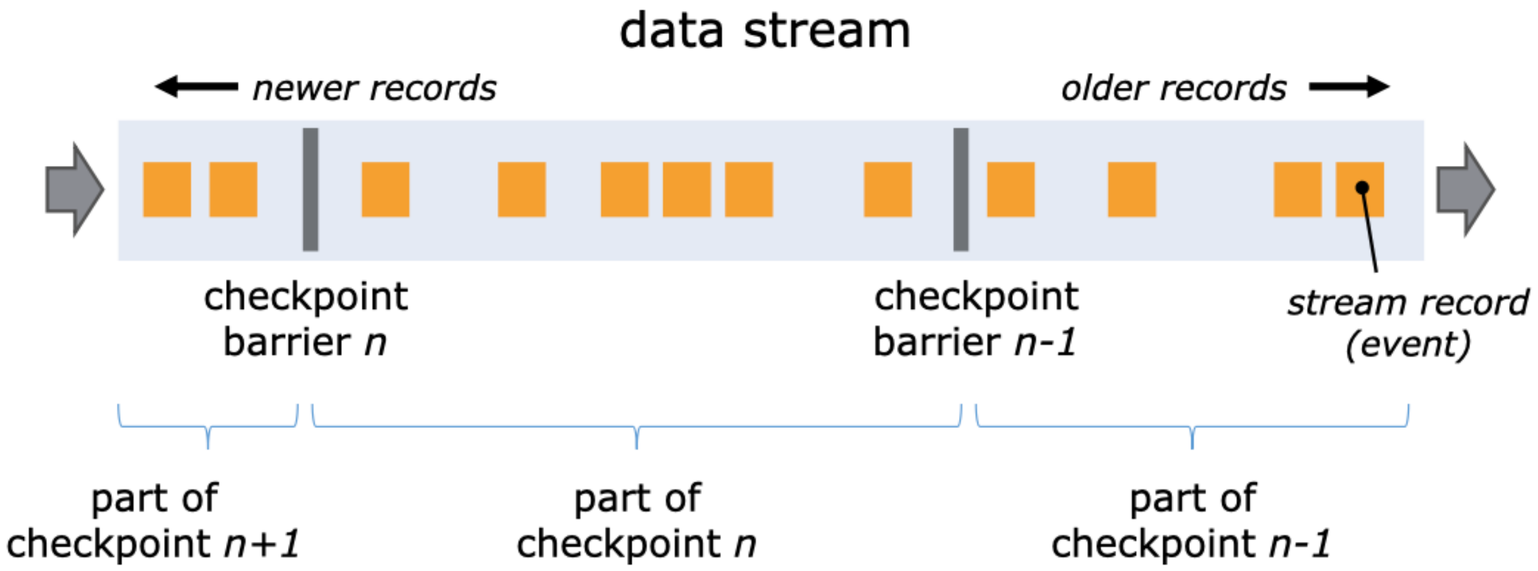

Apache Flink Dashboard
Overview
Jobs
Running Jobs
Completed Jobs
Task Managers
Job Manager
Version: 1.16.0
Commit: af6eff8 @ 2022-10-20T04:21:45+02:00
Message: 0
Event Executor Job
Cancel Job
Job ID
000000006e6b13320000000000000000
Job State
RUNNING
32
Actions
Job Manager Log
Start Time
2023-07-14 16:59:16
Duration
1d 21h 52m 38s
Overview
Exceptions
TimeLine
Checkpoints
Configuration


Source: Kafka Source -> Proc
ess
Parallelism: 4
Backpressured (max): 0%
Busy (max): 0%
HASH
Detail
SubTasks
TaskManagers
Watermarks
Accumulators
BackPressure
Source: Kafka Source
+- Process
State : RUNNING
Parallelism : 4
Start Time : 2023-07-18 15:43:25
End Time : –
Duration : 1h 23m 37s
Task : 4
Records Sent : 417,100
Records Received : 0
Bytes Sent : 452 MB
Bytes Received : 0 B

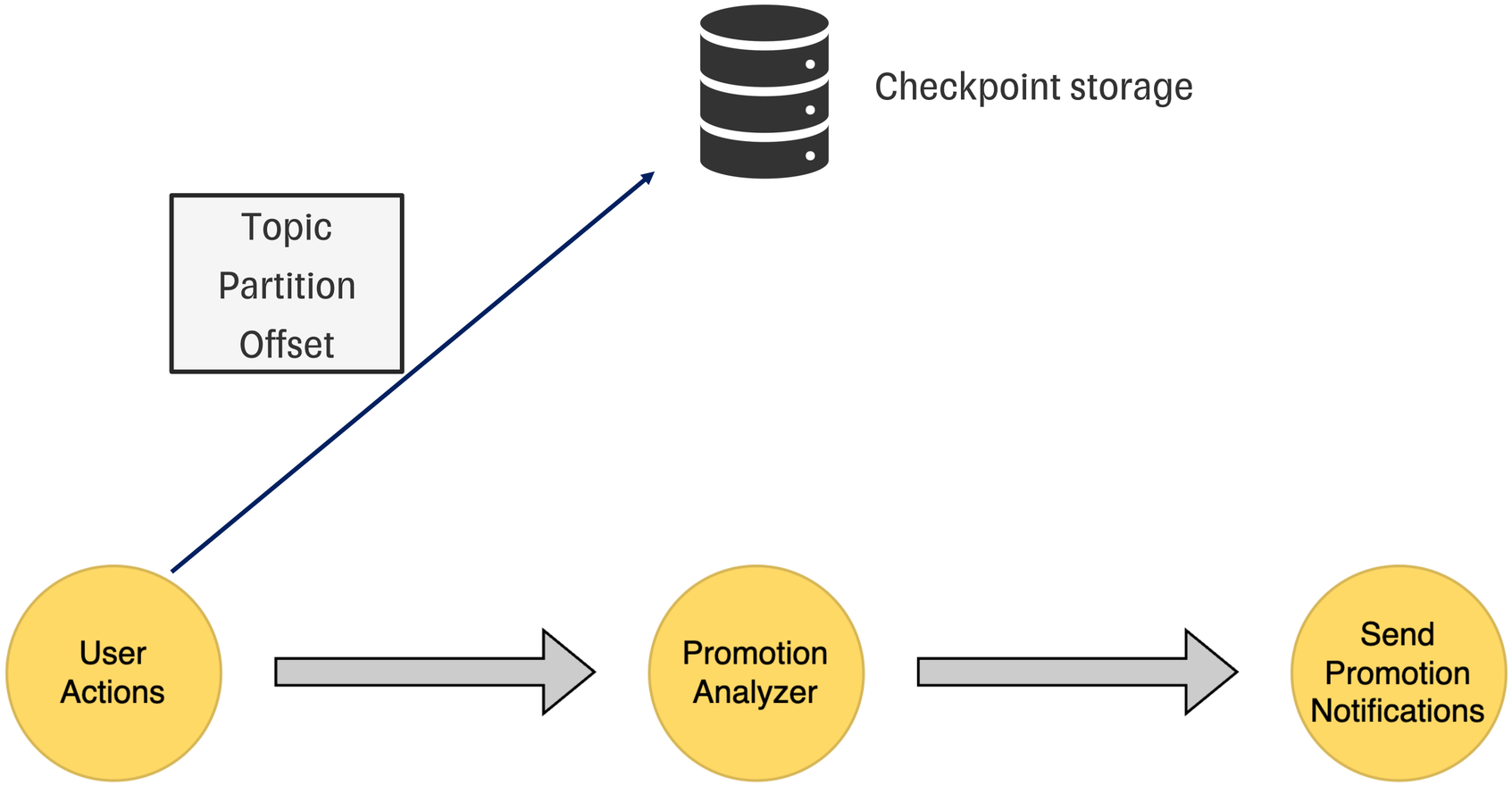
Трудности обнаруженные в рамках решения в лоб

- Рост нагрузки (проблемы масштабирования, нагрузка на БД)
- Эволюционирование бизнес требований
- **Сбои (дубликаты/потери данных)**
- Out-of-order данные?

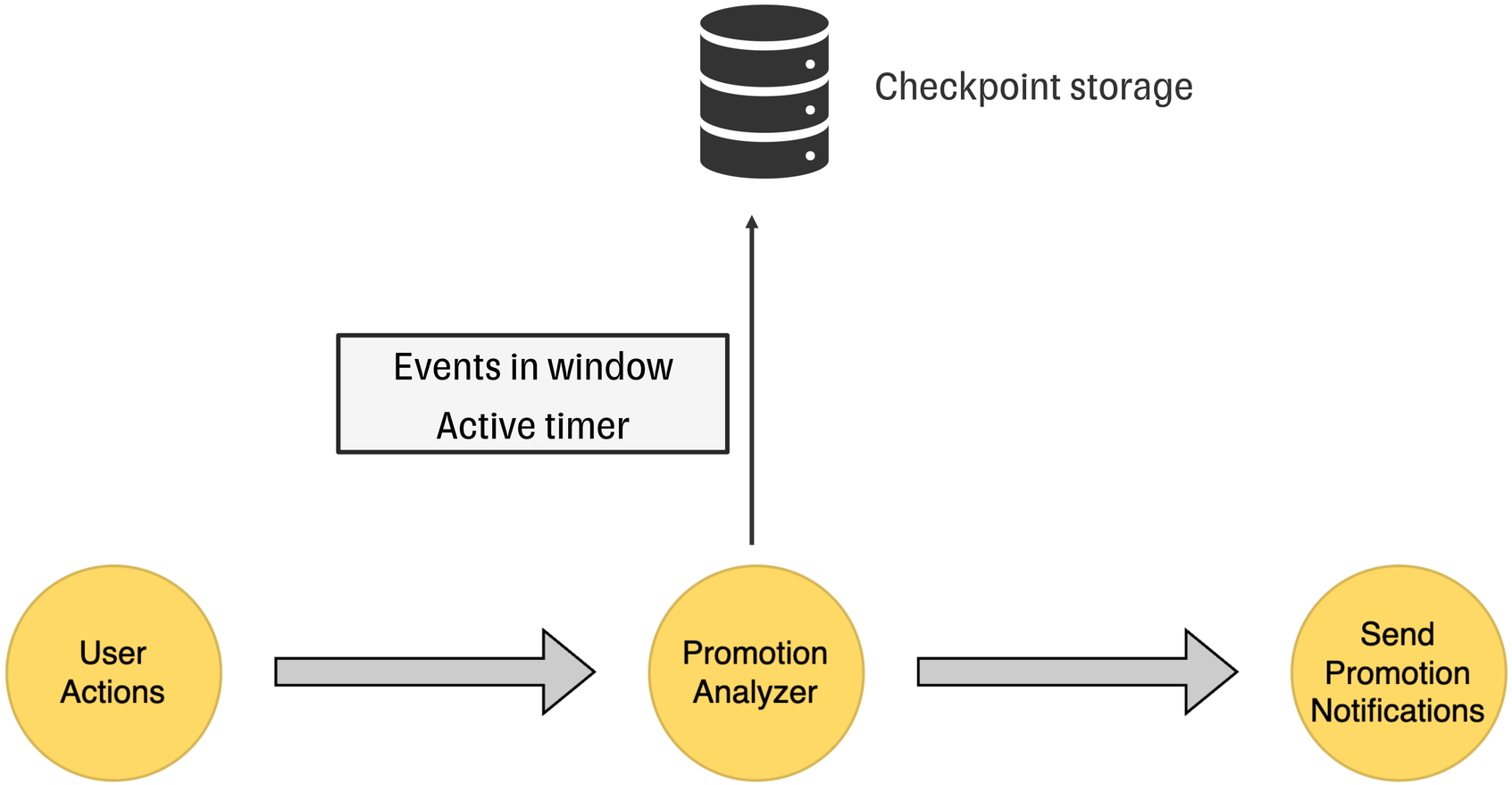
Сбои (дубликаты/потери данных)



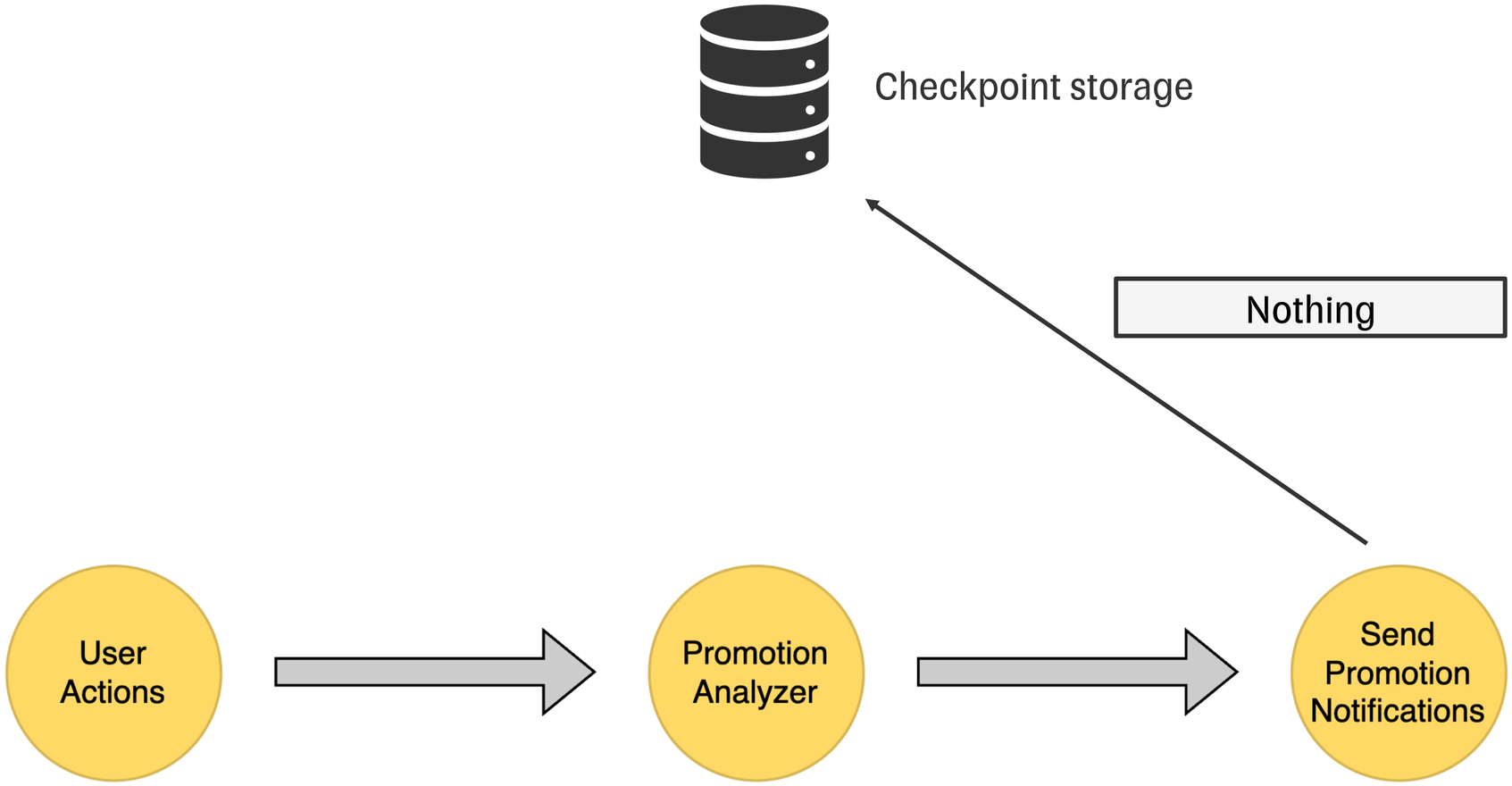
Сбои (дубликаты/потери данных)



Сбои (дубликаты/потери данных)



Сбои (дубликаты/потери данных)



Сбои (дубликаты/потери данных)

- Гарантии доставки зависят от типов Source/Sink

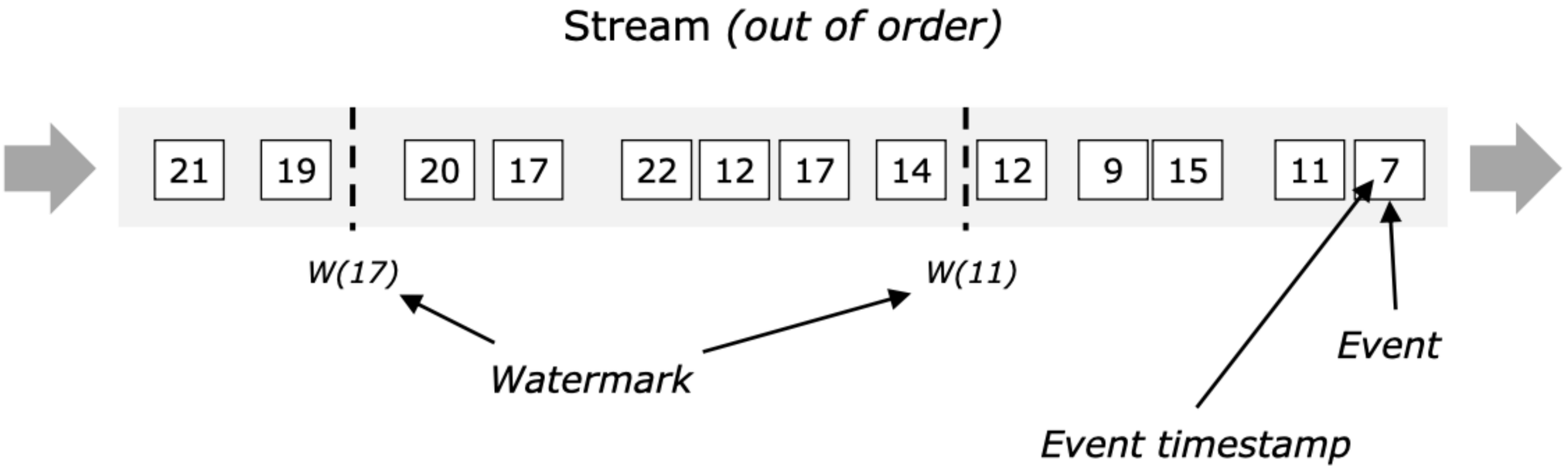
Сбои (дубликаты/потери данных)

- Гарантии доставки зависят от типов Source/Sink
- При гарантии «Exactly Once» возрастают задержки

Трудности обнаруженные в рамках решения в лоб

- Рост нагрузки (проблемы масштабирования, нагрузка на БД)
- Эволюционирование бизнес требований
- Сбои (дубликаты/потери данных)
- **Out-of-order данные?**

Out-of-order события?

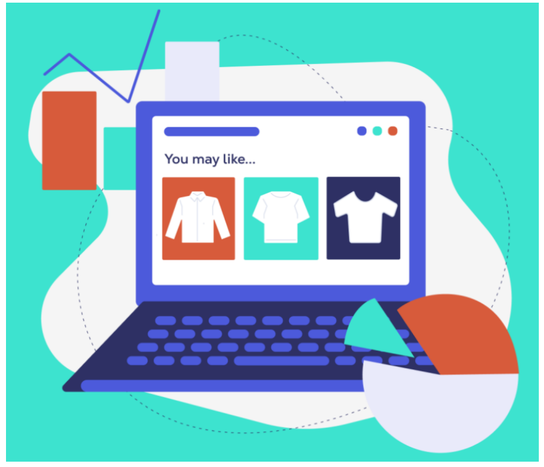


Трудности обнаруженные в рамках решения в лоб

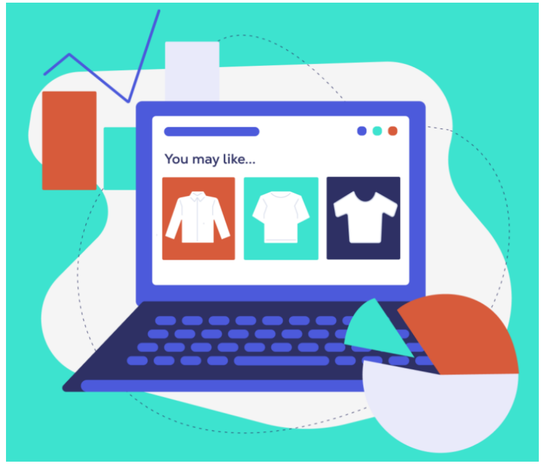
- Рост нагрузки (проблемы масштабирования, нагрузка на БД)
- Эволюционирование бизнес требований
- Сбои (дубликаты/потери данных)
- Out-of-order данные?

- Задача
- Решение в лоб, его плюсы и минусы
- В поисках нового инструмента
- Решение задачи с помощью Apache Flink
- Как флинк справляется с трудностями обнаруженными в рамках решения в лоб
- **Границы применимости**

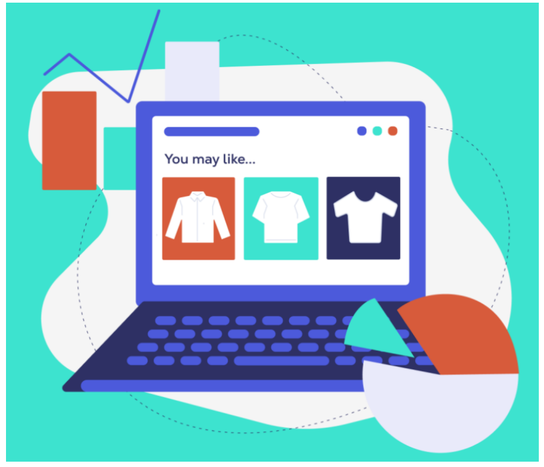
Где применять - Event-driven приложения



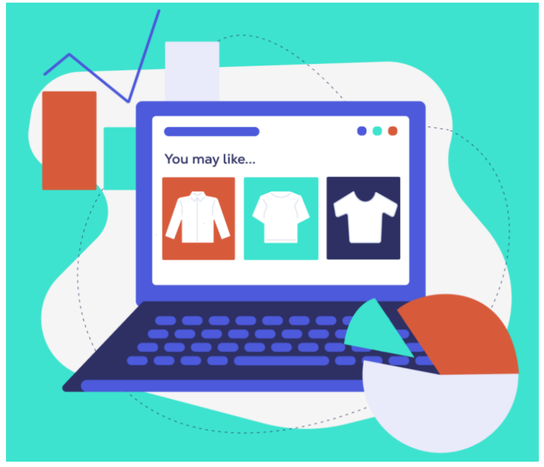
Где применять - Event-driven приложения

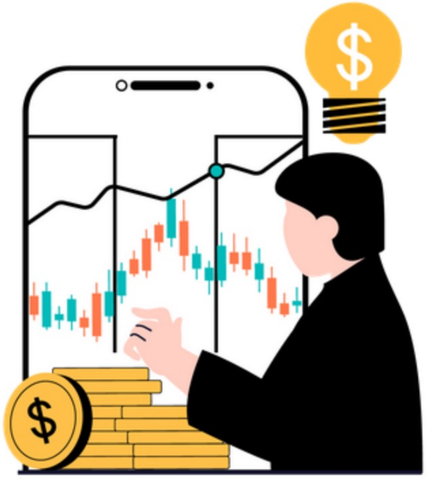


Где применять - Event-driven приложения



Где применять - Event-driven приложения

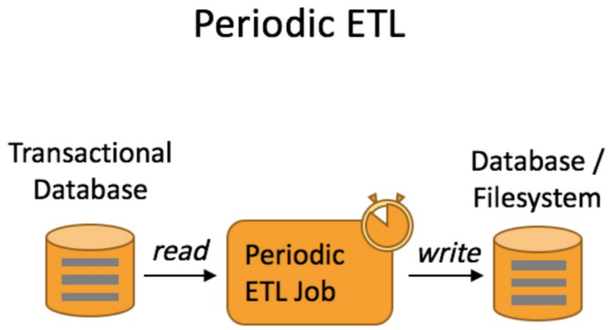


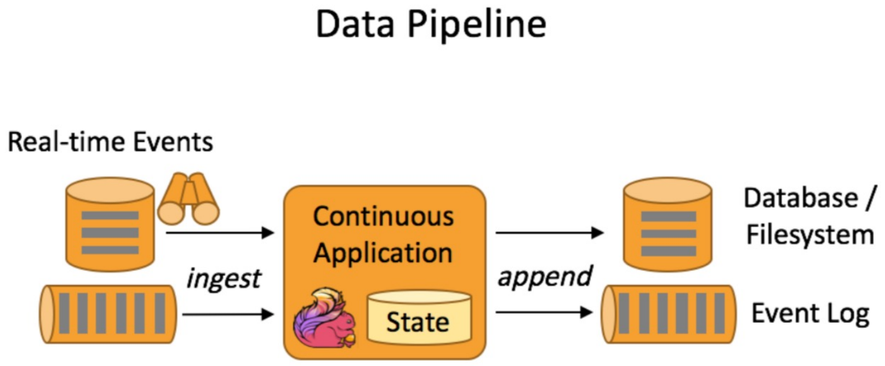


Где применять – Потоковая аналитика

Где применять – Потоковая аналитика

Где применять – Потоковый ETL





Типы задач где не стоит применять фреймворк потоковой обработки

- Обработка больших объёмов исторических данных

Типы задач где не стоит применять фреймворк потоковой обработки

- Обработка больших объёмов исторических данных
- Задачи с фокусом на низкой задержке и высоких гарантиях доставки

Типы задач где не стоит применять фреймворк потоковой обработки

- Обработка больших объёмов исторических данных
- Задачи с фокусом на низкой задержке и высоких гарантиях доставки
- Обработка сложных транзакций

Типы задач где не стоит применять фреймворк потоковой обработки

- Обработка больших объёмов исторических данных
- Задачи с фокусом на низкой задержке и высоких гарантиях доставки
- Обработка сложных транзакций
- Нестабильный поток данных

Т БАНК

Спасибо!

Архитектура