ОНЛАЙН-ОБРАЗОВАНИЕ

Не забыть включить запись!

Меня хорошо слышно && видно?

Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

Преподаватели

Архитектор и разработчик в области аналитических решений

15+ лет опыта в хранилищах, потоках, озерах и других емкостях для данных

Principal Engineer в структуре Teradata Global Delivery Center

Антон Задорожный

anton.zadorozhniy@gmail.com

Правила вебинара

Активно участвуем

Задаем вопрос в чат или голосом

slack

Off-topic обсуждаем в Slack #канал группы или #general

Вопросы вижу в чате, могу ответить не сразу

OTUS

Занятие 9.
Очереди сообщений

Задорожный Антон

Цели вебинара

После занятия вы сможете:

1. определить компонент «очередь сообщений»
2. рассмотреть требования когда очередь сообщений нужна и как ее можно использовать
3. логически проектировать стриминг приложения с использованием основных паттернов

Краткое содержание

- Kafka, RabbitMQ и другие очереди сообщений
- Потоковая обработка
- Модель Dataflow
- Паттерны stream processing: Joins, enricher, router. Event-sourcing.

Определения

Задача

- Приложению А нужно передать данные приложению Б, как это можно реализовать?

- Пример: бэкенд интернет-магазина передает информацию о новом заказе складской системе

Решение

1. Установить прямое соединение между приложениями и передать заказ

2. Записать заказ в файл или базу данных, и ждать когда другая система его прочитает

3. Использовать специальный компонент, который будет принимать заказы от бэкенда, и при появлении нового – нотифицировать склад и хранить сообщение до момента когда склад подтвердит его обработку

Определения

- **Компонент распределенной системы** который обеспечивает **асинхронный обмен сообщениями**

- Строго говоря, мы описываем брокер сообщений, но как в случае с базами данных (БД / СУБД используются взаимозаменяемо), мы будем говорить про очередь сообщений и брокеры как одном и том же

Определения

- **Компонент распределенной системы** который обеспечивает **асинхронный обмен сообщениями**

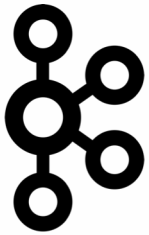
- Строго говоря, мы описываем брокер сообщений, но как в случае с базами данных (БД / СУБД используются взаимозаменяемо), мы будем говорить про очередь сообщений и брокеры как одном и том же

- Примеры компонентов из мира ПО с открытым кодом:

- Apache ActiveMQ

- RabbitMQ

- Apache Kafka



- NATS

Требования

Как понять что очередь нужна? Требования!

OTUS

1. Задержка доставки сообщения / вносимая задержка
   a. Сколько может пройти времени между отправкой сообщения и получением / обработкой с другой стороны?
   b. Часто диктуется бизнес-требованиями

1. Семантика / логика интеграция
   a. Запрос / ответ (Request/Response)
   b. Подписка на сообщения (Publish/Subscribe)
   c. Выгрузка / загрузка данных

1. Количество писателей и читателей
   a. Один писатель – один читатель
   b. Один писатель – много читателей

Примеры критериев

Требования	Прямое соединение	Файл/база данных	Очередь сообщений
Допустимая задержка	0	От ~ 1 секунды	От ~50 мс до 1 часа
Семантика / логика интеграции	Запрос / ответ	Выгрузка / загрузка данных	Подписка на сообщения
Количество читателей и писателей	* – один читатель	Один писатель – *	Любые

Практика

Практические примеры

Требования	Бэкенд магазина – склад	Сайт otus.ru – аналитика (рекомендации)	Онлайн игра
Допустимая задержка	Секунды	Часы	~ 10 мс
Семантика / логика интеграции	Подписка на сообщения	Выгрузка / загрузка данных	Запрос / ответ
Количество читателей и писателей	Много писателей – много читателей	Один писатель – много читателей	Много писателей - один читатель

Практические примеры

Требования	Бэкенд магазина – склад	Сайт otus.ru – аналитика (рекомендации)	Онлайн игра
Допустимая задержка	Секунды	Часы	~ 10 мс
Семантика / логика интеграции	Подписка на сообщения	Выгрузка / загрузка данных	Запрос / ответ
Количество читателей и писателей	Много писателей – много читателей	Один писатель – много читателей	Много писателей - один читатель

Задание на закрепление

- Взаимодействие Сайт Госуслуги – МВД
- Кассовый аппарат в гипермаркете – сервер магазина
- Биллинговая платформа – бухгалтерская система мобильного оператора

Домашнее задание 1

1. Рассмотреть пример взаимодействия приложений близкий вам

2. Предложить требования по таблице и обосновать применение / неприменение очереди сообщений

Срок – 2019.07.15 09:00

Потоковая обработка

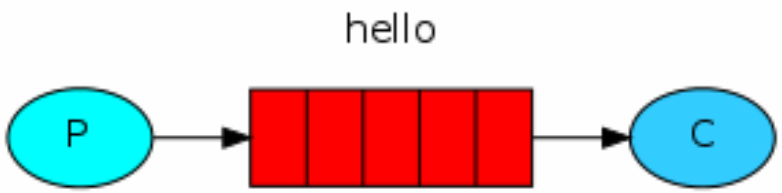
Рассмотрим виды потоковой обработки

- Producer-consumer problem: задача передачи данных между двумя процессами/приложениями через буфер ограниченного размера
- Возможные решения внутри приложения: используя семафор / мутекс, либо используя монитор
- Для внешних очередей характерны следующие виды:

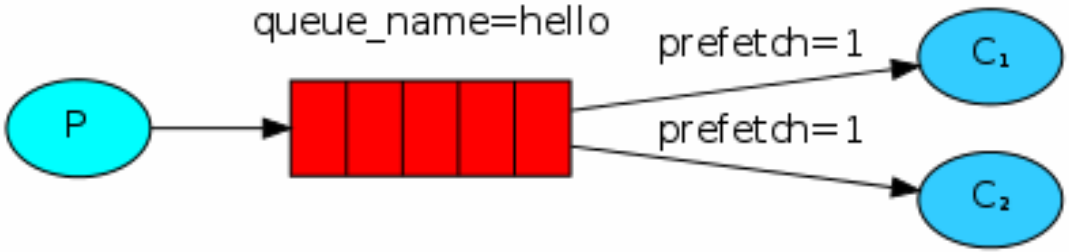
  - Point-to-point
  - Publish / Subscribe
  - Routing / filtering
  - Request / response
  - Transformation (ESB)
  - Commit Log

Point-to-point

- Передача сообщения через очередь с известным именем

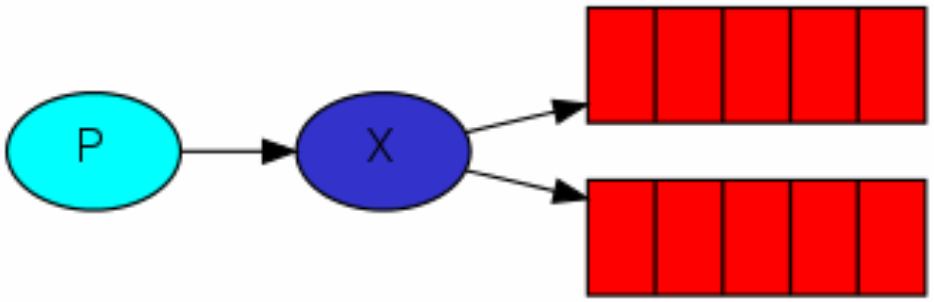


- Может использоваться для распределения работы между несколькими консьюмерами



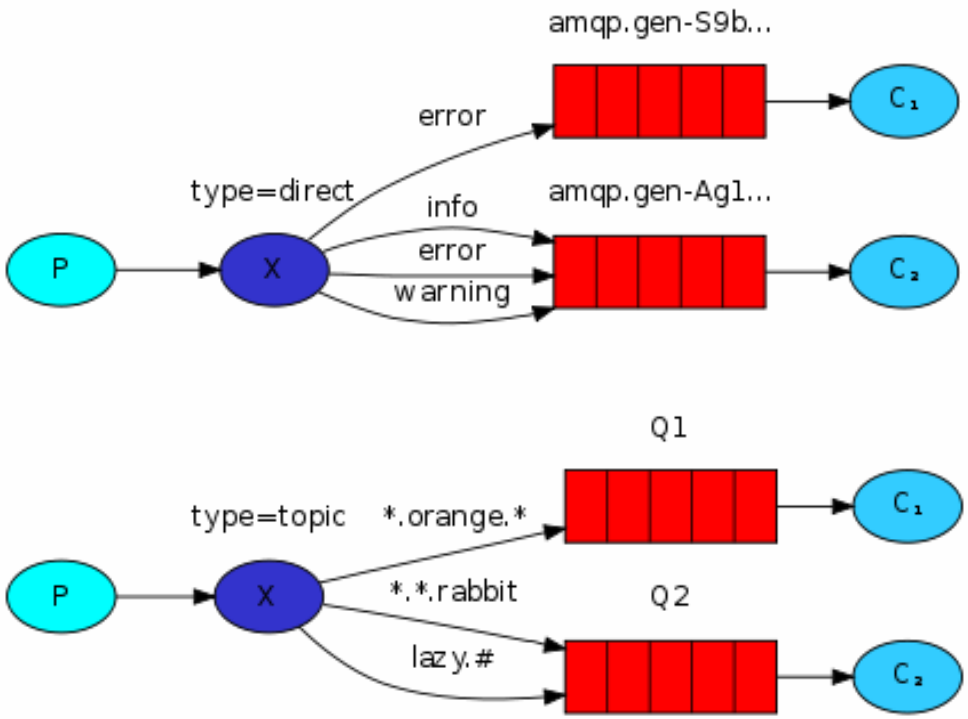
Publish / Subscribe

- Возможность консьюмера выразить интерес к определенным публикуемым данным
- Брокер контролирует процесс потребления (JMS, RabbitMQ) / консьюмер сам контролирует процесс (Kafka, NATS Streaming)



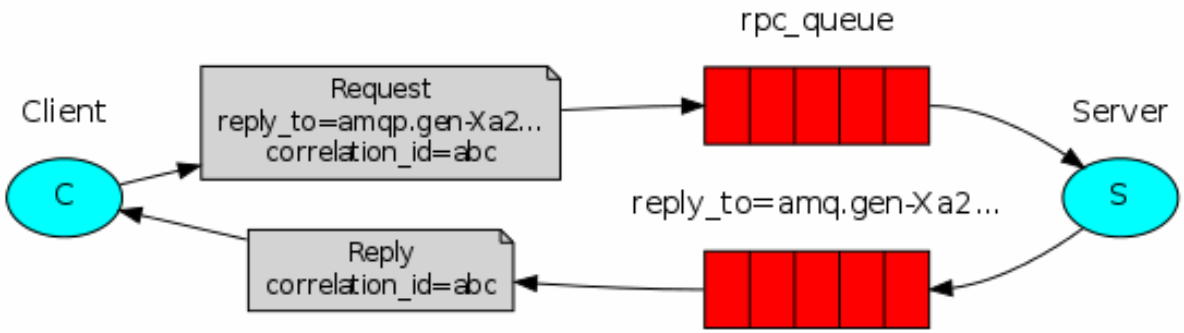
Routing / filtering

- Приложение может получать сообщения в зависимости от логики / конфигурации на брокере



Request / response

- Приложения обмениваются запросами и ответами через брокер

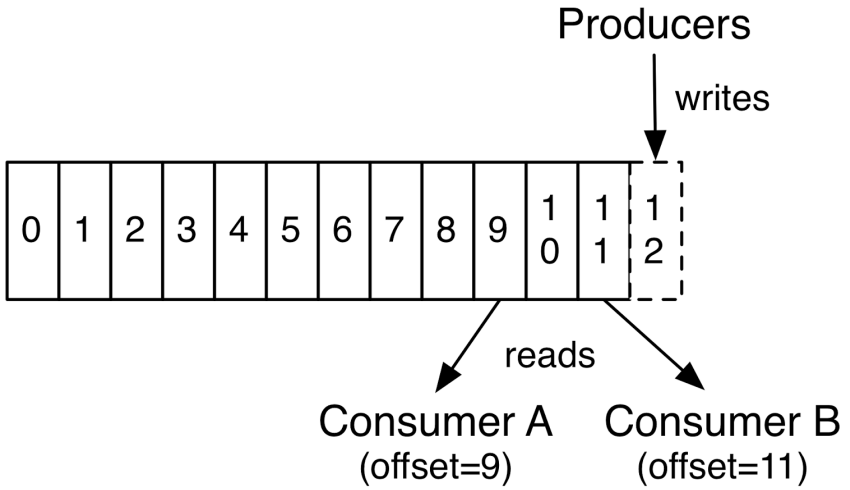


Commit Log

- Строго говоря commit log это не вид обработки, а организация брокера внутри
- Стоит рассмотреть отдельно из-за большой популярности

- Commit log брокер хранит сообщения в виду условно бесконечной ленты
- Потребители могут подключаться для чтения сообщений когда им удобно
- Прогресс отслеживается через ссылки на ленту

- Представители дизайна:
  - Apache Kafka
  - NATS Streaming
  - Amazon Kinesis, Azure Event Hub
  - Apache Pulsar



Фреймворки потоковой обработки данных

- Сказав про очередь мы не можем оставить вопрос «а что за приложения обрабатывают данные в очередях?»
- Этот класс приложений называется **фреймворки потоковой обработки данных**, и мы посвятим некоторым из них отдельные занятия
- Apache Storm
- Spark Streaming *
- Apache Flink / Apache Beam
- Kafka Streams, Confluent KSQL *
- Предложения от облачных провайдеров, например Google Cloud Dataflow, Azure Stream Analytics
- И многие другие

*) Будут освещаться в рамках курса

Примеры архитектурных решений

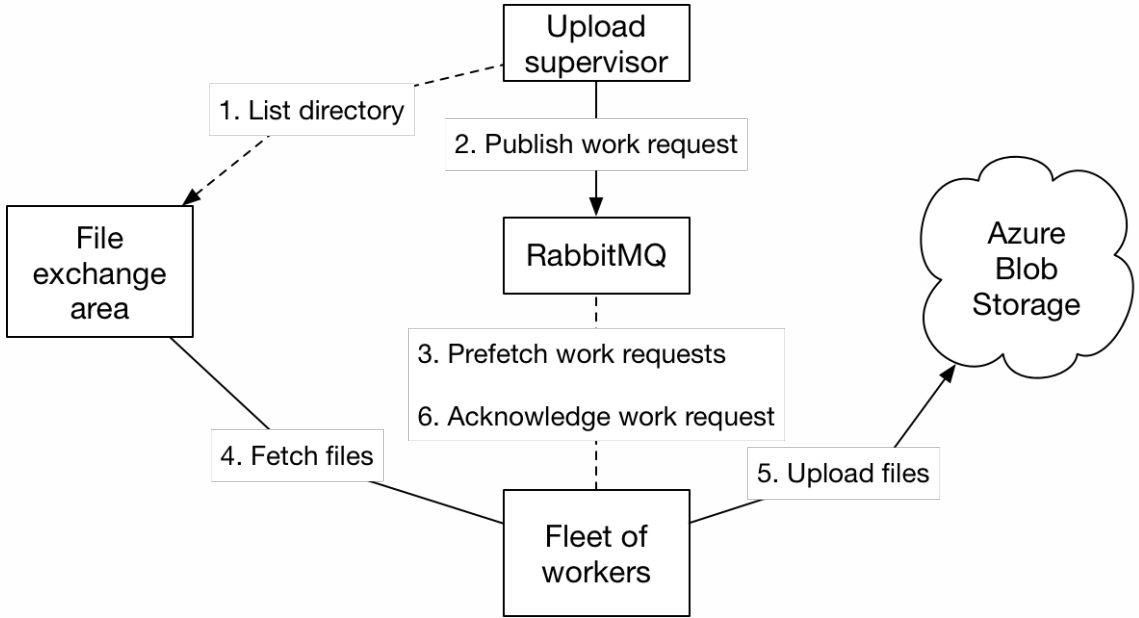
Примеры архитектурных решений

Рассмотрим следующие примеры:

- RabbitMQ: загрузка файлов от приложений-источников в даталейк
- Kafka: потоковая аналитика в мобильном операторе
- NATS Streaming: система по борьбе с отмыванием денег в банковской сфере

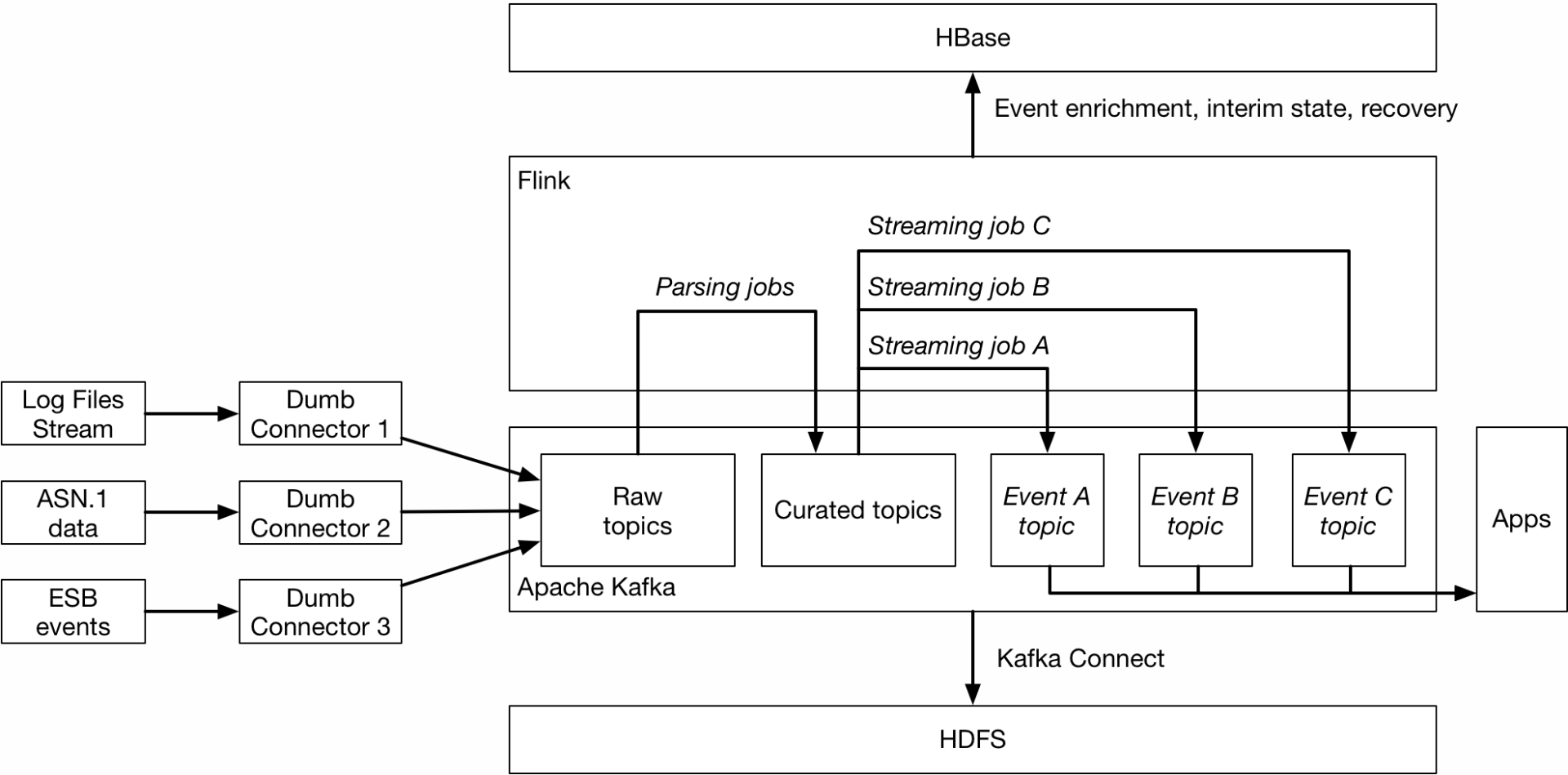
Решение на базе RabbitMQ

- Необходимо организовать доставку большого количества маленьких файлов в облачный сторадж для последующей аналитики
- Решение должно быть надежным и масштабируемым
- Воркеры работают независимо друг от друга, их флот может изменяться динамически, в зависимости от нагрузки
- Средства мониторинга RabbitMQ дают представление о скорости работы и прогрессе воркеров, и дают возможность реализовать TTL работы с поздними данными (через deadletter очередь)



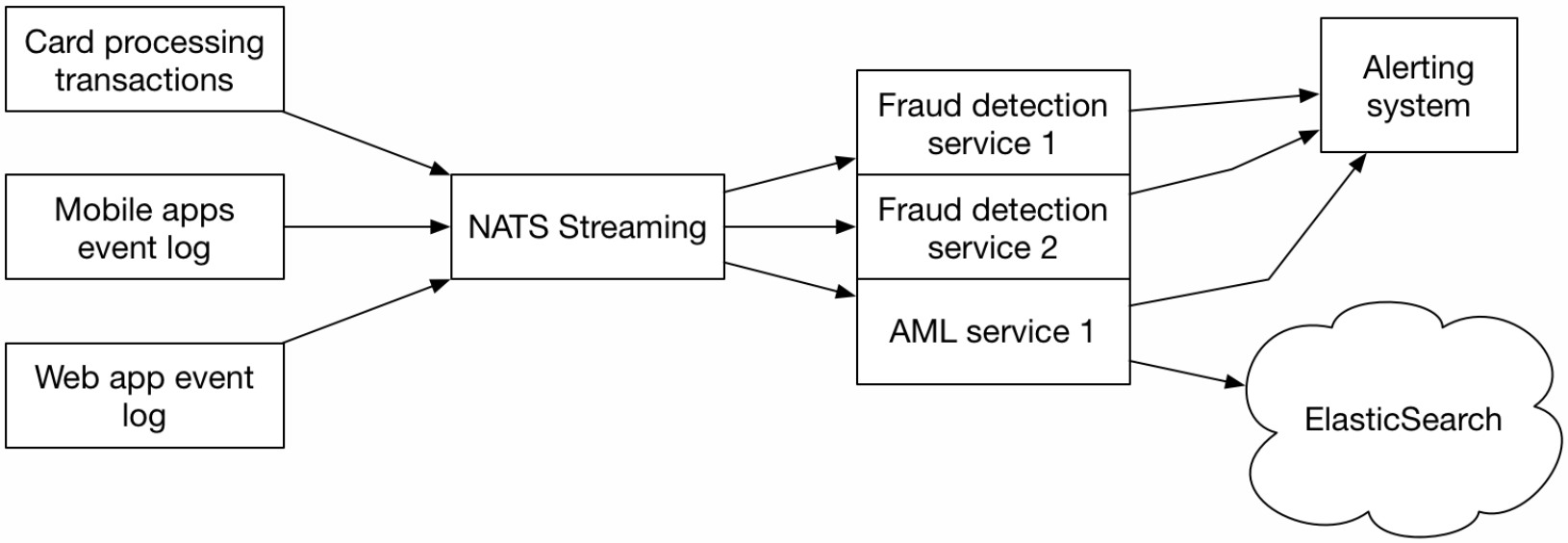
Решение на базе Apache Kafka

- На базе потоков данных в мобильном операторе необходимо рассчитывать ряд событий для применения в маркетинге и управлении сетью
- Архитектура очень похожа на обычный даталейк, но вместо директорий в объектном сторадже – топики в Kafka



Решение на базе NATS Streaming

- На базе потоков данных в финансовой организации необходимо реализовать систему раннего предупреждения для борьбы с мошенничеством и отмыванием средств
- Основные требования: низкие задержки и поддержка большого количества приложений с правилами
- Сервисы написаны на golang и использование NATS Streaming было органичной миграцией с обычного NATS



Модель Dataflow

Модель Dataflow

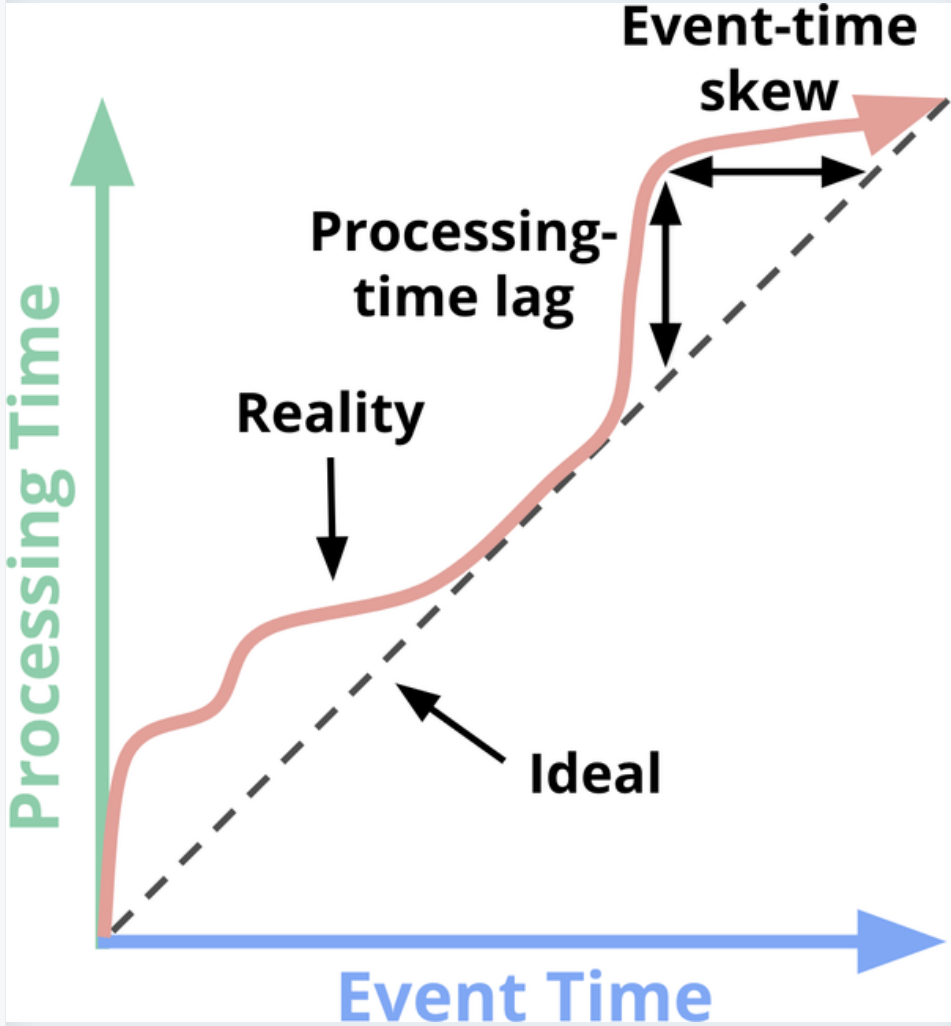
- 2015 год, The Dataflow Model paper https://ai.google/research/pubs/pub43864
- Попытка построить логическую модель для различных ситуаций потоковой обработки, без привязки к физической реализации
- Последующие реализации модели:

  - Google Cloud Dataflow

  - Apache Beam (SDK который может использовать Spark, Flink, Cloud Dataflow и другие платформы для запуска)

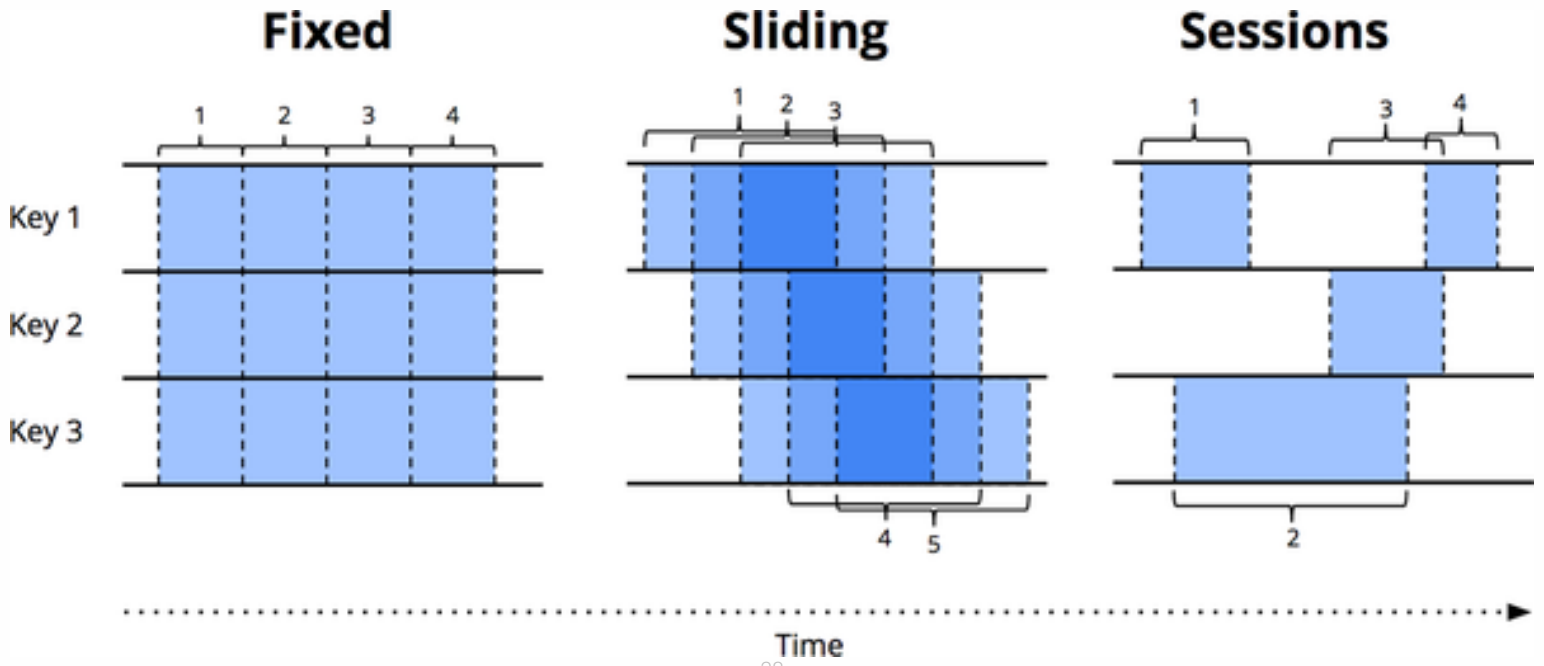
Основные понятия

- **Время события (event time)**: время, когда событие на самом деле произошло
- **Время обработки (processing time)**: время, когда событие обрабатывается
- **Задержка обработки (processing-time lag / event-time skew)**: задержка между временем события и временем обработки
- **Ограниченные данные (bounded data)**: ограниченные данные, когда известно что в какой-то момент данные в потоке закончатся (то есть батч)
- **Неограниченные данные (unbounded data)**: неограниченные данные, то есть поток



Основные понятия часть 2

- **Фиксированное (fixed) окно**: зафиксирована частота и длина окна, и они равны
- **Скользящее (sliding) окно**: зафиксированы частота и длина окна, но частота меньше чем длина
- **Сессии (sessions)**: динамическое окно, где окно определяется для отдельного значения ключа, а длина зависит от бизнес-логики (обычно определяется таймаутом/отсутствием активности за определенное время)



Определения часть 3

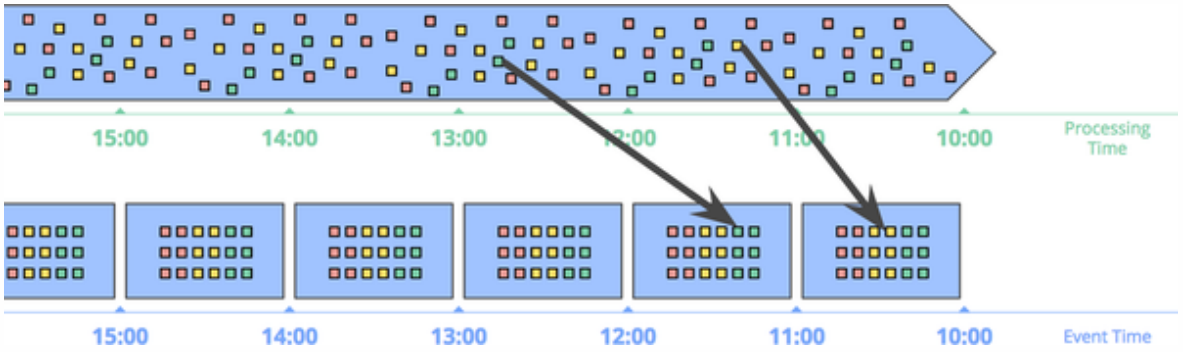
- **Водяной знак (watermark)**: монотонно возрастающий таймстемп самого старого события, которое мы еще не обработали
- Необходимо для оценки «полноты» работы, которое делает приложение для конкретного окна
- Идеальный водяной знак – когда нам известен критерий полноты, в таком случае у нас нет проблемы «позднего прилета» или «поздних данных» (late arrival / late data)
- Чаще всего берется эвристически («предположим что сообщения долетают не более чем за 20 секунд»), в таком случае нам нужно явно обрабатывать «поздние данные»

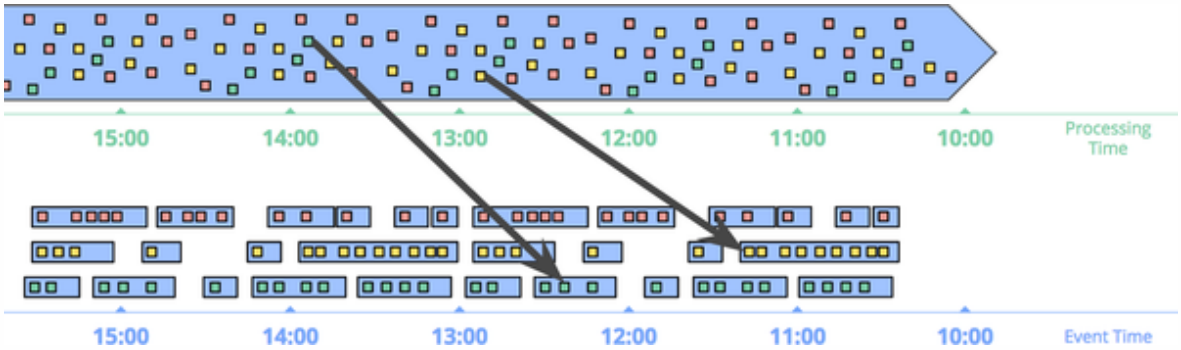
- **Триггер (trigger)**: условие при котором стриминг приложение выдает сообщение наружу, например:
- На каждом входящем сообщении
- В фиксированные интервалы
- Когда мы считаем что вся нужна группа была обработана (по водяному знаку)

Четыре вопроса к стриминг приложению

- **Что считаем? (What?)** Что считает приложение? Описание трансформации над событием
- **Где по времени события? (Where?)** Где во времени (над какими событиями на линии времени) происходят трансформации? Описание используемых окон
- **Когда по времени обработки? (When?)** Когда по времени обработки материализовать результат трансформации? Описание водяных знаков и триггеров
- **Как связаны? (How?)** Как связаны последующие результаты трансформации с предыдущими? Описание агрегации

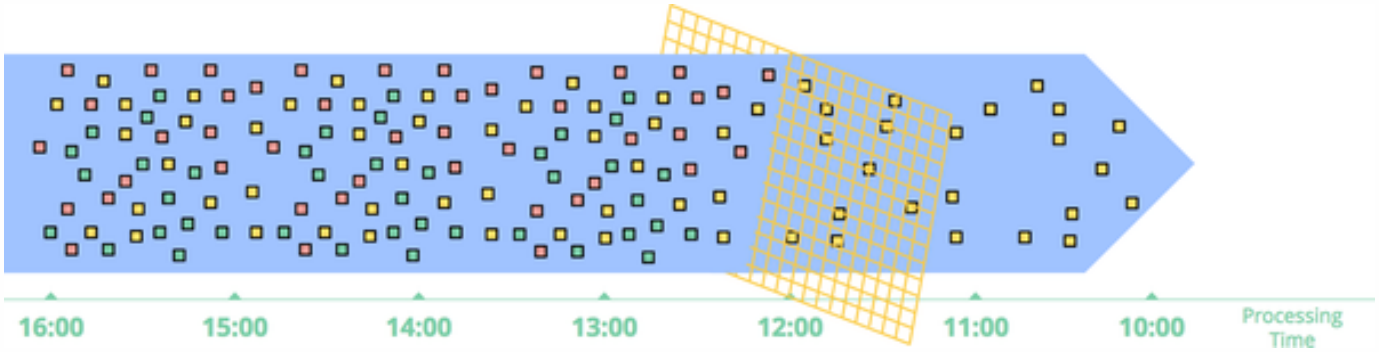
Фиксированные окна и сессии



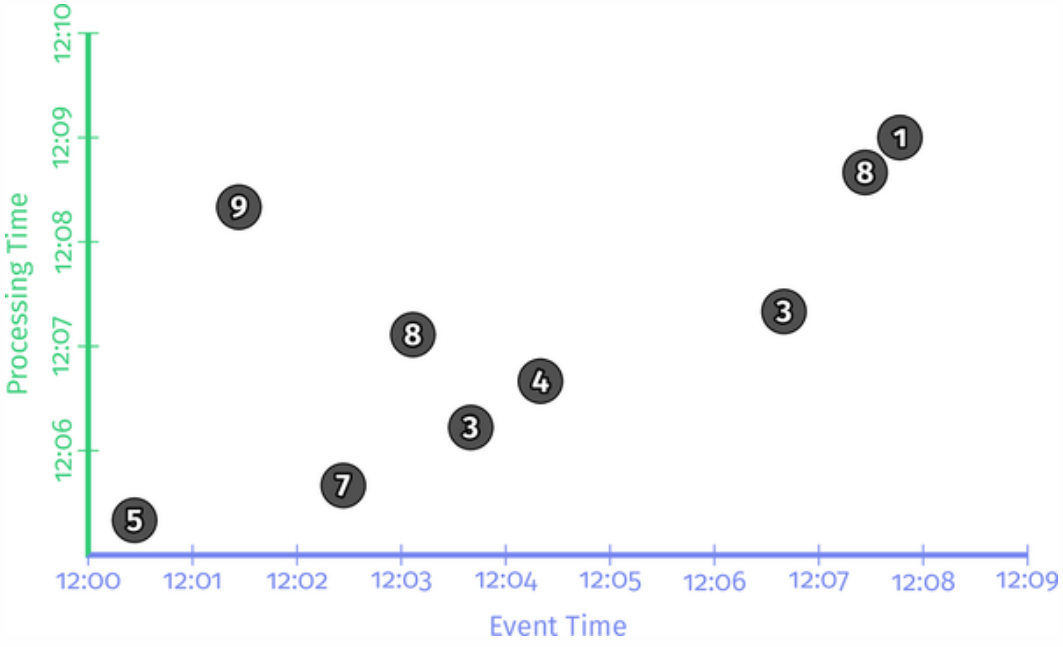


Фильтрация

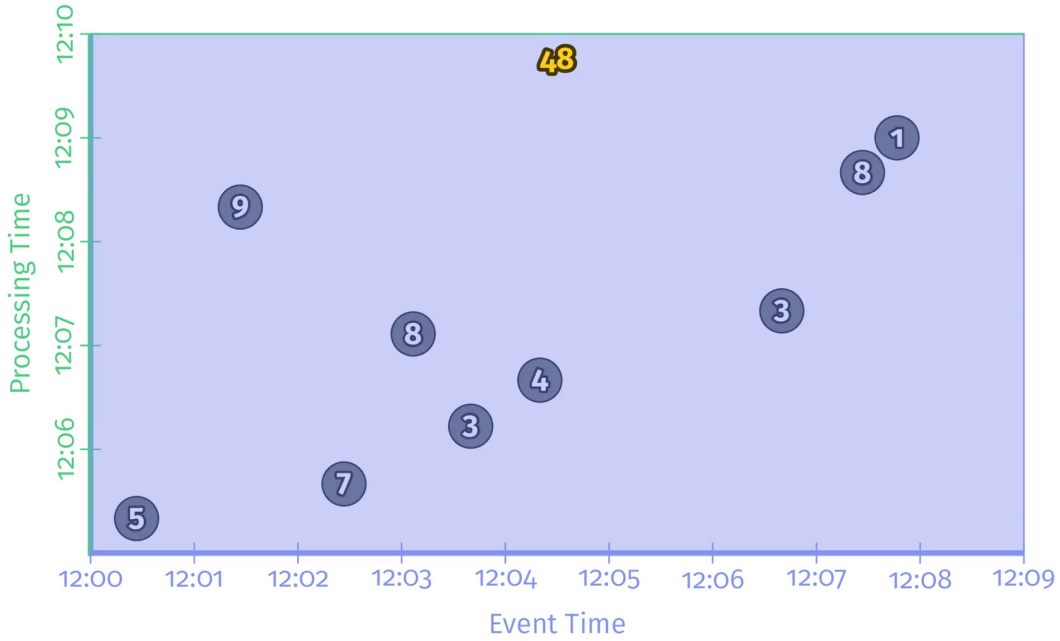
Ограничения сообщений по каким-то критериям



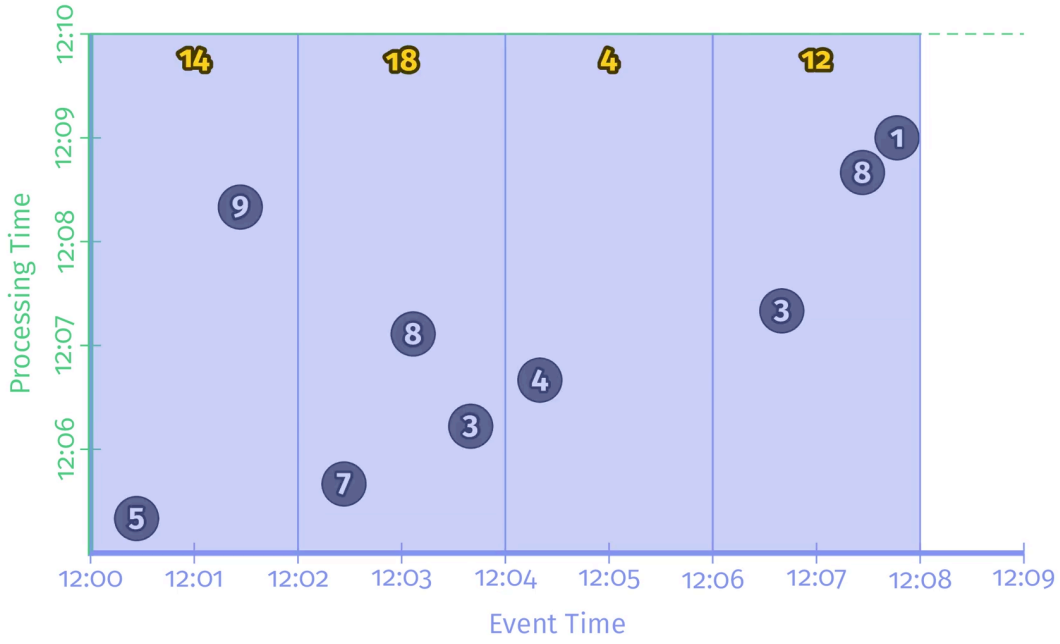
Рассмотрим несколько простых примеров



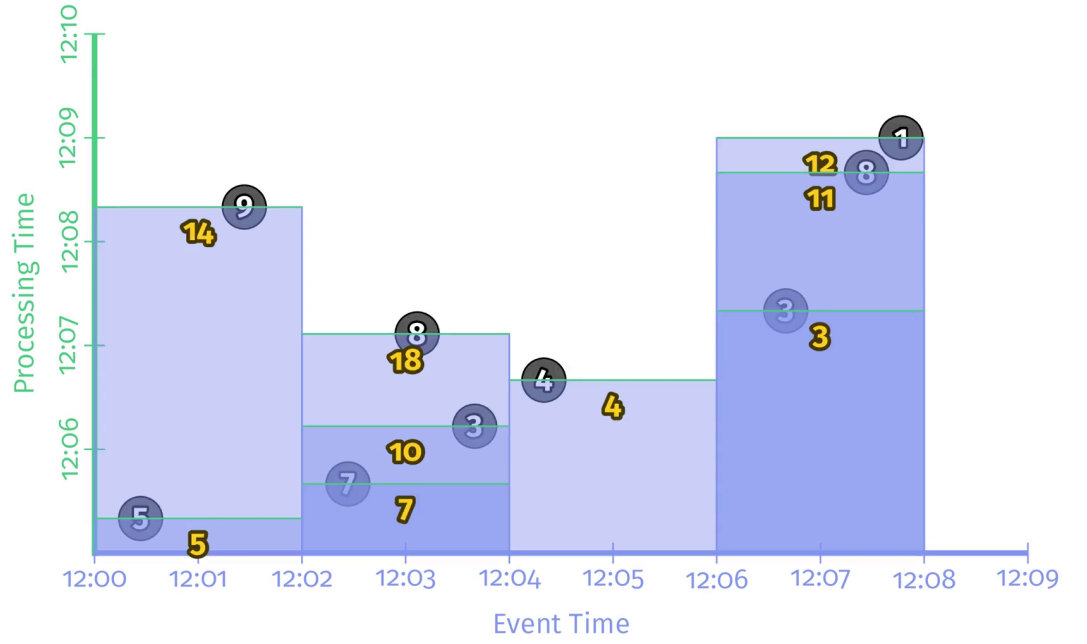
Батч: нет окон, один результат



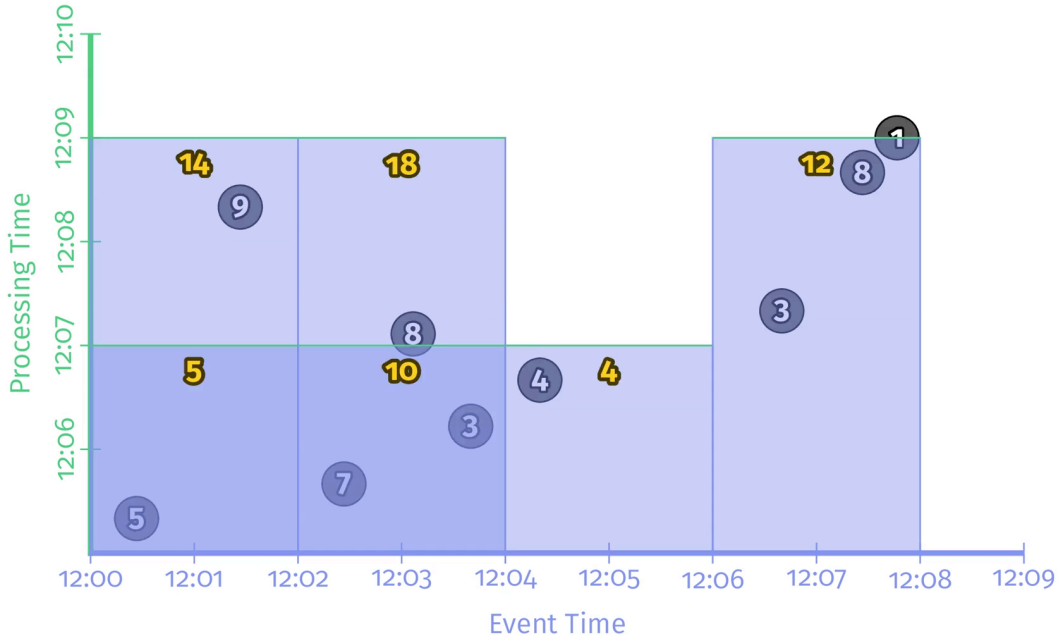
Батч: двухминутные окна, один результат



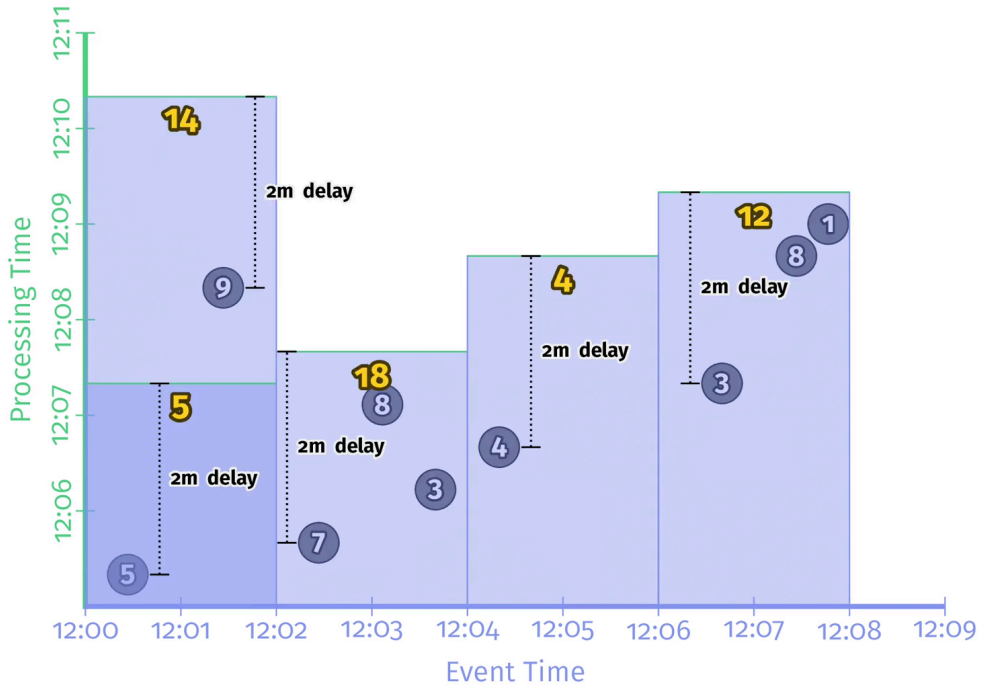
Стриминг: двухминутные окна, результат с каждым сообщением



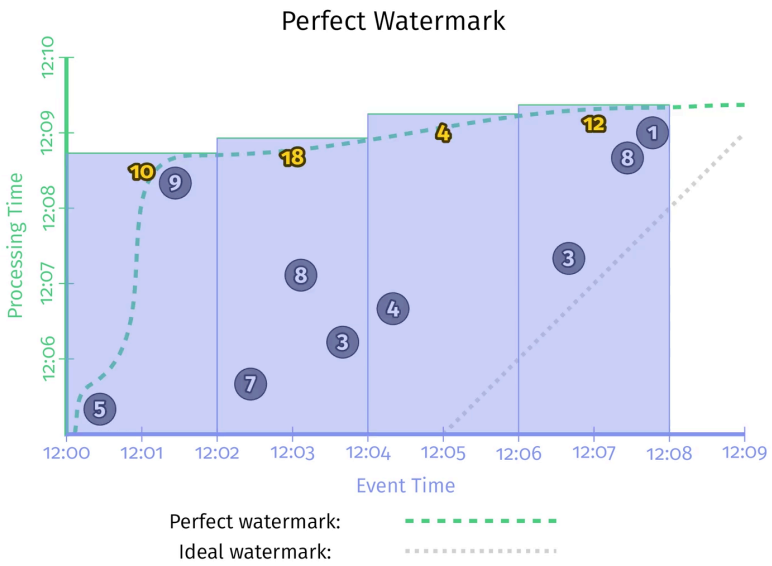
Стриминг: двухминутные окна, результат каждые 2 минуты

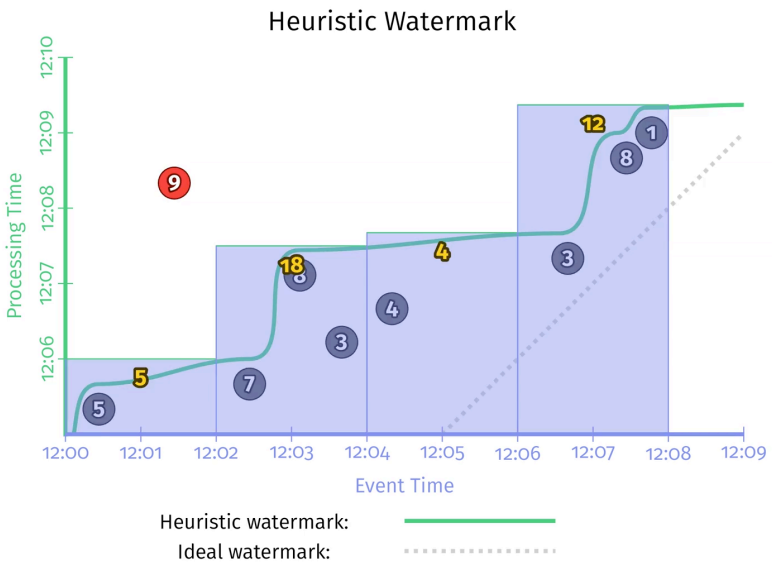


Стриминг: двухминутные окна, результат с задержкой в две минуты



Два разных типа водяных знаков





Паттерны потоковой обработки данных

Рассмотрим несколько паттернов потоковой обработки данных

- Enrich
- Join
- Интеграция
- Event sourcing

Enrich

Обогащение события при помощи внешнего источника данных, примеры:

- Получить текущий баланс счета чтобы валидировать проводку
- Получить вектор предикторов по ключу чтобы выполнить скоринг

Функциональные особенности:

- Можно представить себе как джоин, но не с потоком, а со стабильным датасетом
- Лучше всего когда внешний источник выставляет доменный и легковесный API: сильно упрощает развитие и сопровождение
- Нужно иметь тесты (и логику) на ситуацию когда по ключу не найдено значение

Нефункциональные особенности:

- Внешний источник должен учитывать не только стандартную, но и догоняющую нагрузку (проводить нагрузочное тестирование)
- Обязательно реализовать логику отключения по таймауту
- Необходимо иметь метрики на энрич-клиенте: тайминги, счетчики по ненайденным ключам
- Часто имеет смысл задуматься о кэшировании в стриминг приложении

Join

Объединение двух потоков данных по определенным условиям, чаще всего реализуется объединение окон по потокам данных, примеры:

- Объединение событий клиента с информацией о текущей сессии
- Объединение потока визуальных данных с данными сенсоров

Функциональные особенности:

- Тщательно прописать и обложить тестами пограничные условия и логику джоина (inner/left/right/full outer, late arrival)
- Если требуется нормализация: выделить нормализацию в отдельный модуль, согласовать логику
- Проверить наличия перекоса ключей, предложить решение

Нефункциональные особенности:

- Мониторинг перекоса ключей
- Мониторинг late arrival

Интеграция

Доставить события из очереди в другую систему, например:

- Для каждого сообщения из топика в Kafka вызвать RESTful API
- Организовать реплику данных из Kafka в S3 для долгосрочного хранения

Функциональные особенности:

- Максимально разделить части реализующие бизнес-логику, и часть которая делает интеграцию (оптимально – интеграция отдельным компонентом)
- При наличии большого числа интеграций – использовать commit log брокер (позволит избежать дублирования сообщений для каждой интеграции)

Event sourcing

- Паттерн представления всех изменений внутри приложения как последовательности событий
- Часто путают с CQRS (Command Query Responsibility Segregation), хотя формально это разные вещи
- Нам, как дата инженерам, предоставляет лучший источник данных: все события выражены в доменной модели и любое изменение состояния объекта всегда есть в потоке событий (несколько усложняет жизнь пакетной аналитике)
- Должен исходить от приложения-источника, навязать источнику event sourcing почти невозможно
- CDC это не event sourcing

Домашнее задание 1

1. Рассмотреть пример взаимодействия приложений близкий вам

2. Предложить требования по таблице требований и обосновать применение / неприменение очереди сообщений

Срок – 2019.07.15 09:00

Домашнее задание 2

1. Рассмотрим стриминг приложение которое считает остаток на счете дебетовой карты:
   - Получает события вида “timestamp, account, amount”, где amount может быть положительным или отрицательным
   - Максимальное время задержки доставки событий: 2 секунды
   - Если на счете недостаточно средств – приложение выдает алерт и прежний остаток, если достаточно – обновленный остаток
2. Прочитайте Dataflow paper и развернуто ответьте на четыре вопроса описывающее это стриминговое приложение: what? where? when? how?

   (дополнительное задание: опишите как бы вы поступили в случае поздней доставки данных, если задержка более двух секунд)

Срок – 2019.07.15 09:00

Рефлексия

Отметьте 3 пункта,
которые вам
запомнились с вебинара

- Что вы будете применять в работе из сегодняшнего вебинара?

Следующий вебинар

Тема: DWH. Хранилища данных - 1 часть

Понедельник 2019.07.08 в 20.00

Ссылка на вебинар будет в ЛК за 15 минут

Заполните, пожалуйста, опрос о занятии

Спасибо
за внимание!

ОНЛАЙН-ОБРАЗОВАНИЕ