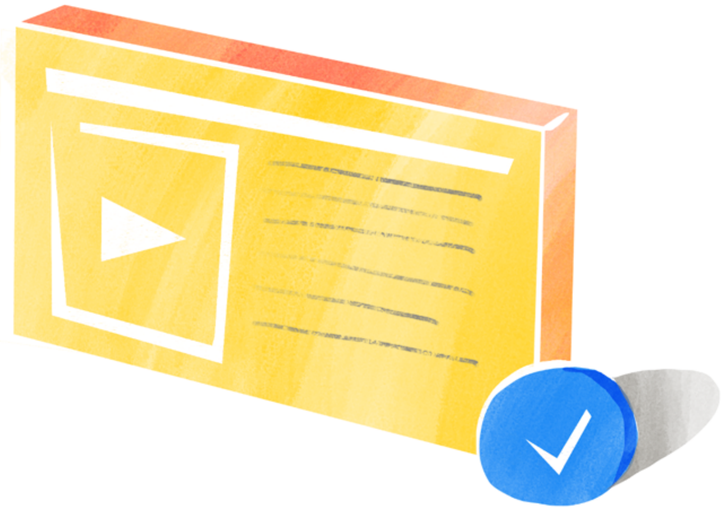

Яндекс

Яндекс

Перевод прямых трансляций в Яндекс Браузере



Арсентий Мельников

Перевод видео и трансляций в Я. Браузере

Перевод видео

- 9 исходных языков (итальянский, английский, испанский, французский, китайский, японский, корейский, немецкий, русский)
- 2 языка перевода (русский, казахский)
- Работает на всех основных платформах (YouTube, vimeo, vk, coursera, bilibili, twitch и прочее)

Перевод трансляций

- 5 исходных языков
- 1 язык перевода (русский)
- Работает на всех YouTube каналах

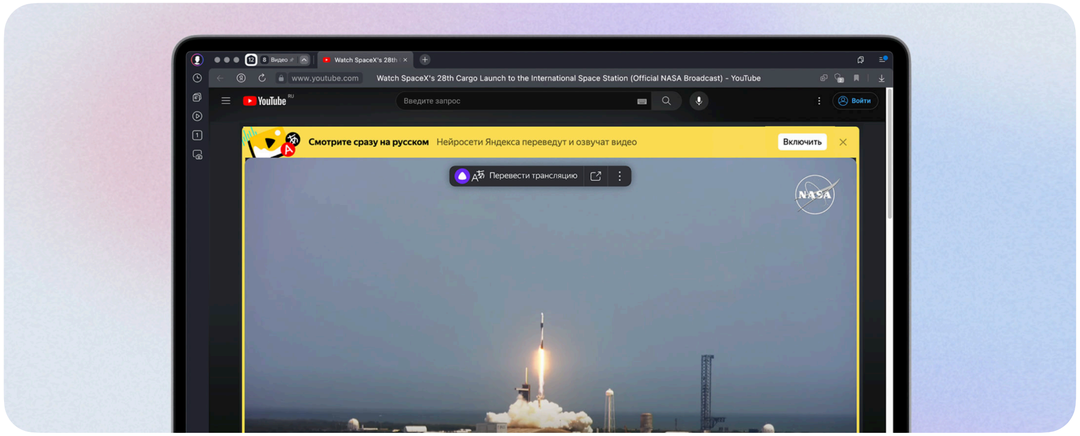
7 сентября 2021
Запуск перевода видео

5 августа 2022
Перевод ограниченного списка трансляций

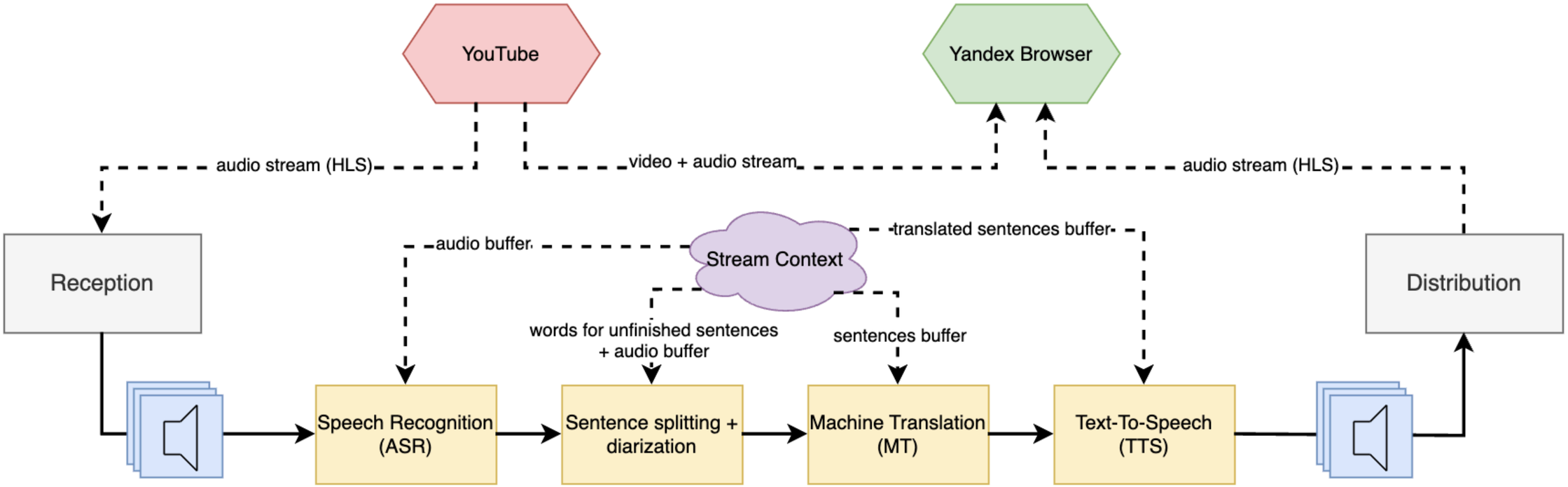
21 сентября 2023
Перевод любых YouTube трансляций

Яндекс Браузер переведёт и озвучит любые YouTube-трансляции с пяти популярных европейских языков

21 сентября 2023

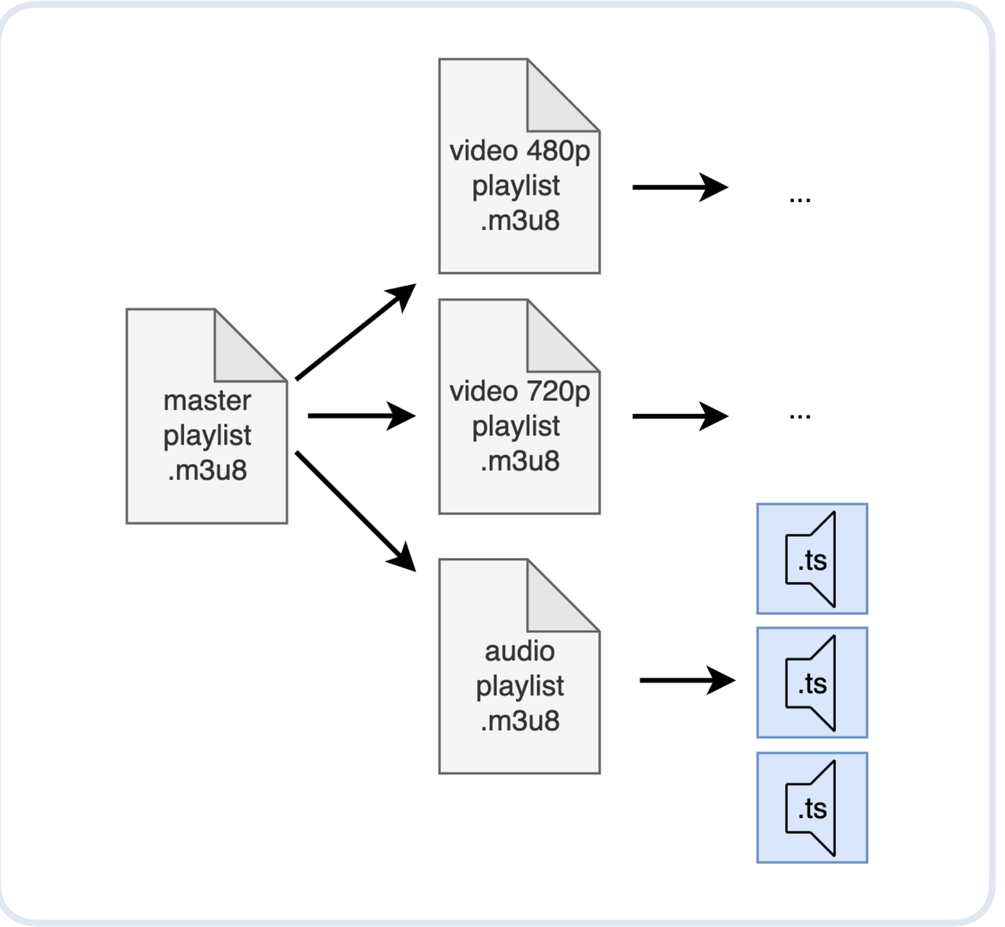


Пайплайн потокового перевода



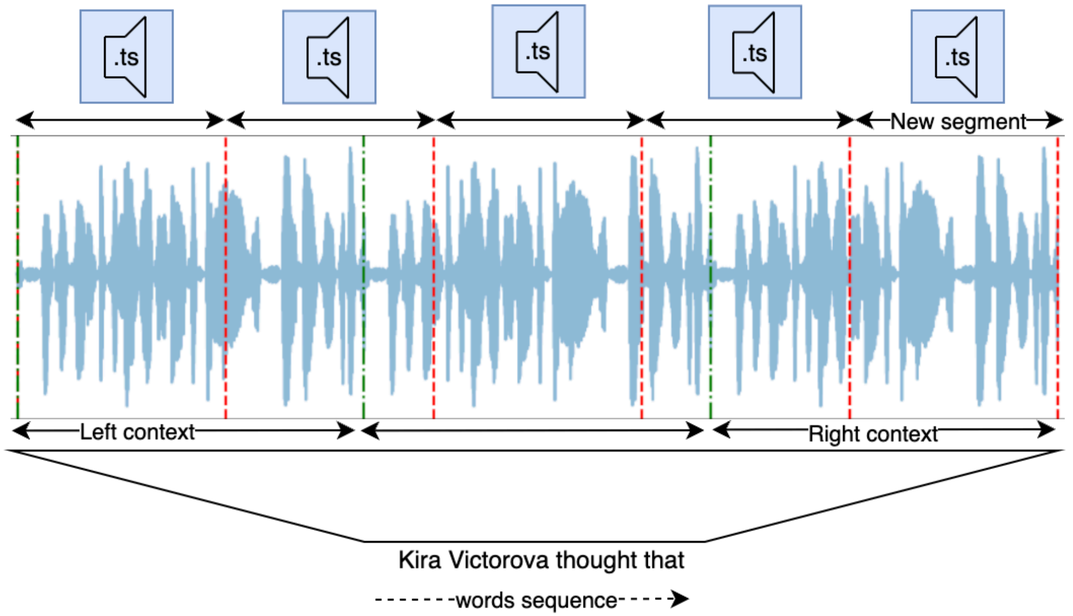
HTTP Live Streaming (HLS)

- Протокол для потоковой передачи медиа на основе HTTP
- Разработан компанией Apple и опубликован в 2009 году
- Серверная часть обычно включает в себя кодировку, сегментирование на фрагменты одинаковой длины и сохранение в .ts файлы
- Клиент переодически скачивает playlist, в котором содержаны ссылки на последние несколько сегментов
- Типичная длина сегментов составляет от 1 до 5 секунд



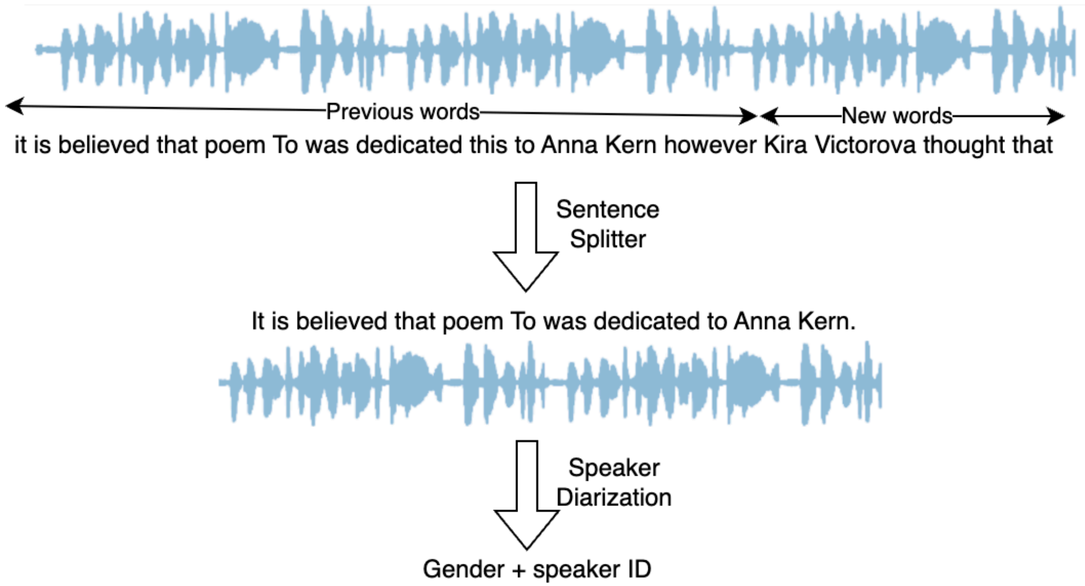
Automatic Speech Recognition (ASR)

- Получаем сообщение с очередным сегментом
- Храним в kv хранилище буфер аудио фрагментов и не оконченное слово (если имеется)
- Если нового сегмента достаточно, чтобы распознать очередной фрагмент аудио, то распознаем речь
- Отправляем дальше по пайплайну распознанную речь (набор оконченных слов)



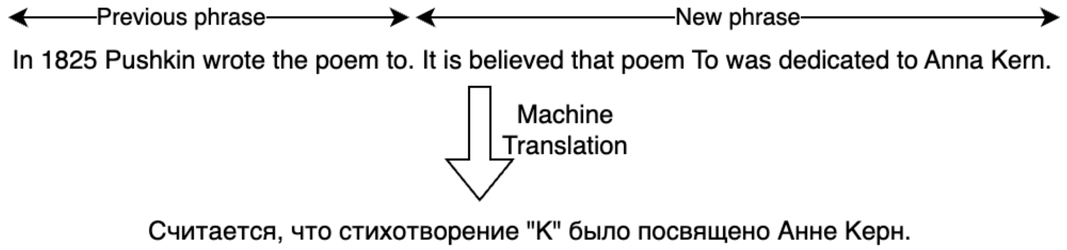
Sentence Splitting & Diarization

- Получаем сообщение с очередным сегментом и распознанными словами
- Храним в kv хранилище буфер аудио фрагментов и не оконченную фразу
- Расставляем знаки препинания
- Определяем пол спикеров для новых фраз
- Отправляем дальше набор оконченных фраз и информацию о спикере



Machine Translation (MT)

- Получаем сообщение с новыми фразами
- Храним в kv хранилище буфер с прошлыми переведенными фразами
- Переводим новые фразы
- Отправляем сообщения с новыми переведенными фразами

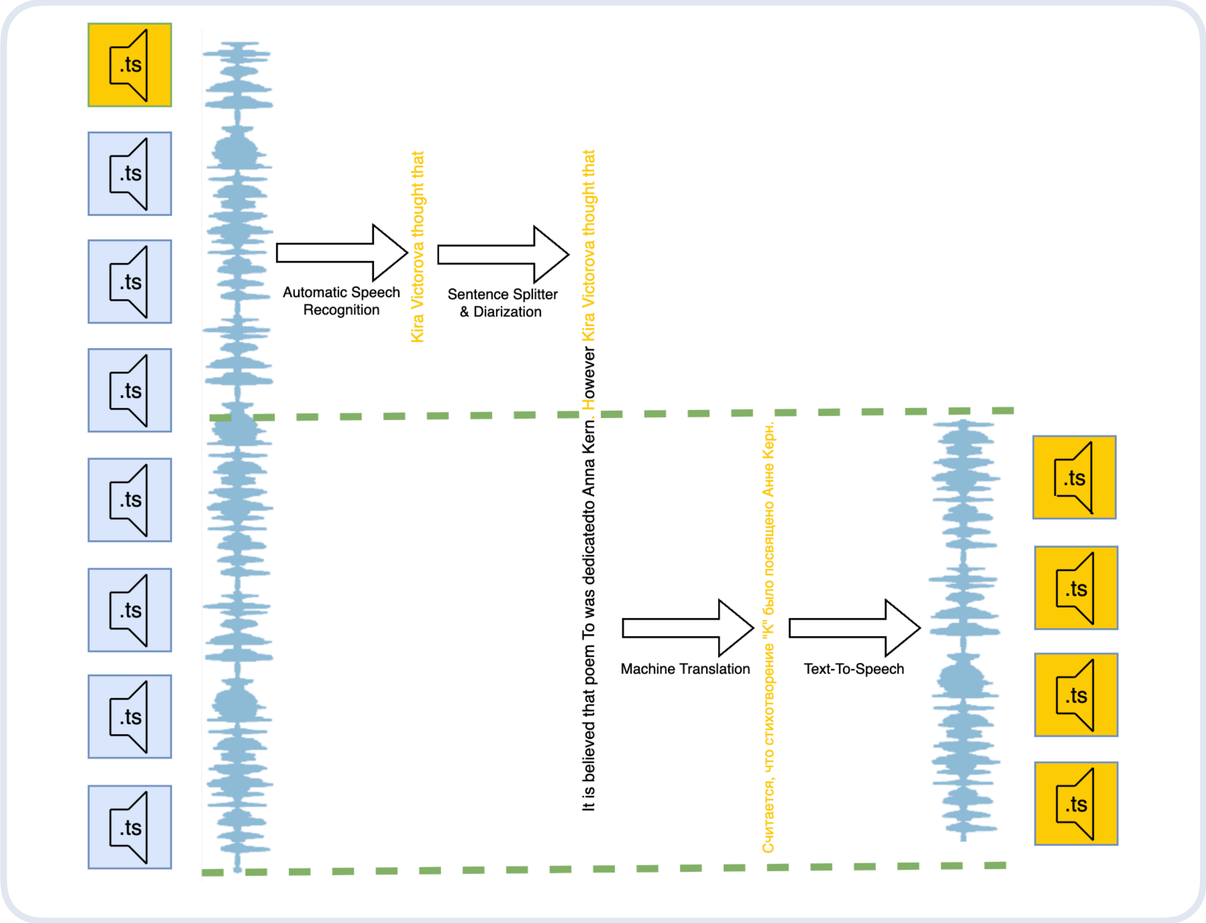


Text-To-Speech (TTS)

- Получаем сообщение с новыми переведенными фразами
- Храним в kv хранилище буфер с прошлыми фразами
- Синтезируем речь
- Отправляем синтезированные фрагменты дальше, там бьем все фрагменты на сегменты равной длины и отдаем в сервис, который уже занимается раздачей

Пайплайн потокового перевода

- Неизбежное отставание перевода на десятки секунд
- Строго последовательная обработка сегментов
- Тяжелый контекст (может быть в разы больше исходного сегмента)

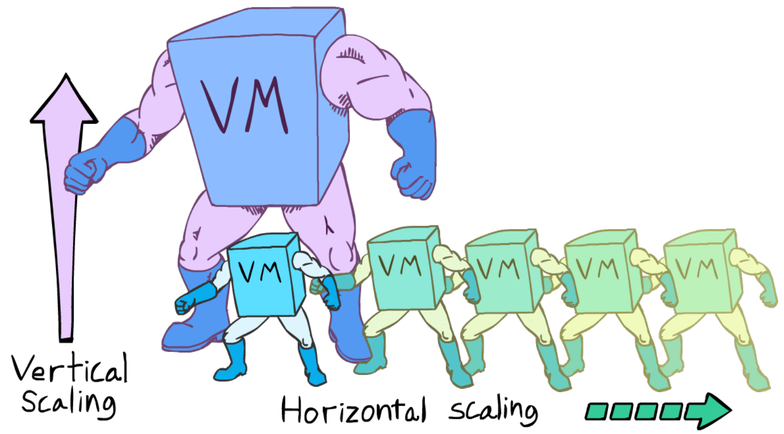


Проблемы масштабирования

YDB

Какие технологии использовали изначально

- Python и C++
- Yandex Message Queue — совместимый с Amazon SQS HTTP API брокер сообщений (FIFO очереди с ограничениями в 30rps)
- Redis для хранения всего контекста, включая аудио буфер



Что поменяли

- YDB — распределённая отказоустойчивая Distributed SQL база данных с открытым исходным кодом, которая поддерживает выполнение потоковых нагрузок
- Redis для хранения контекста, но не аудио буфера
- Аудио буфер храним локально и при необходимости восстанавливаем по ссылкам из контекста
- Существенно переработали кодовую базу (ушли от питоначьих процессов и стали использовать asyncio)

Topics in YDB

YDB

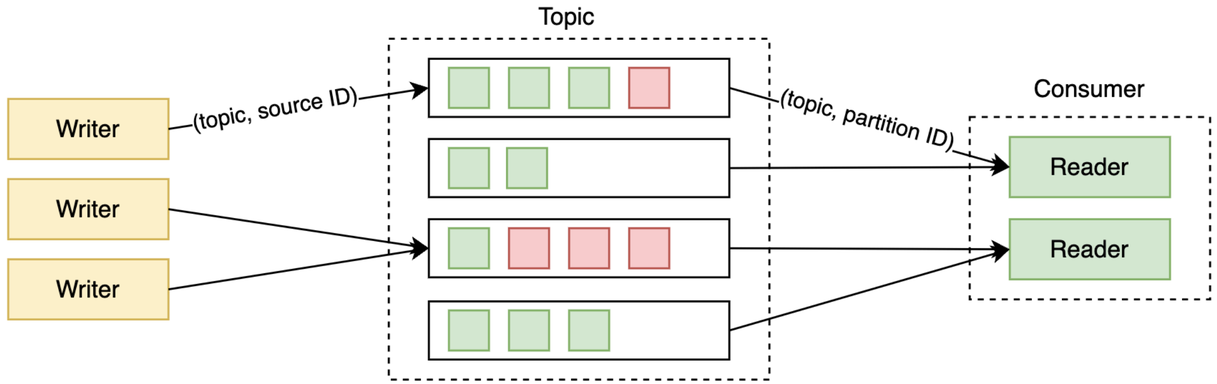
YDB Topics представляет собой механизм pub/sub для доставки сообщений

- At-least-once при чтении сообщений (каждое сообщение будет прочитано подписчиком)
- Exactly-once при публикации сообщений (чтобы исключить дублирование сообщений)
- Гарантия порядка (FIFO) для сообщений с одним source ID
- Простое горизонтальное масштабирование (с помощью патриций)

Апрель 2022
Выход в опенсорс

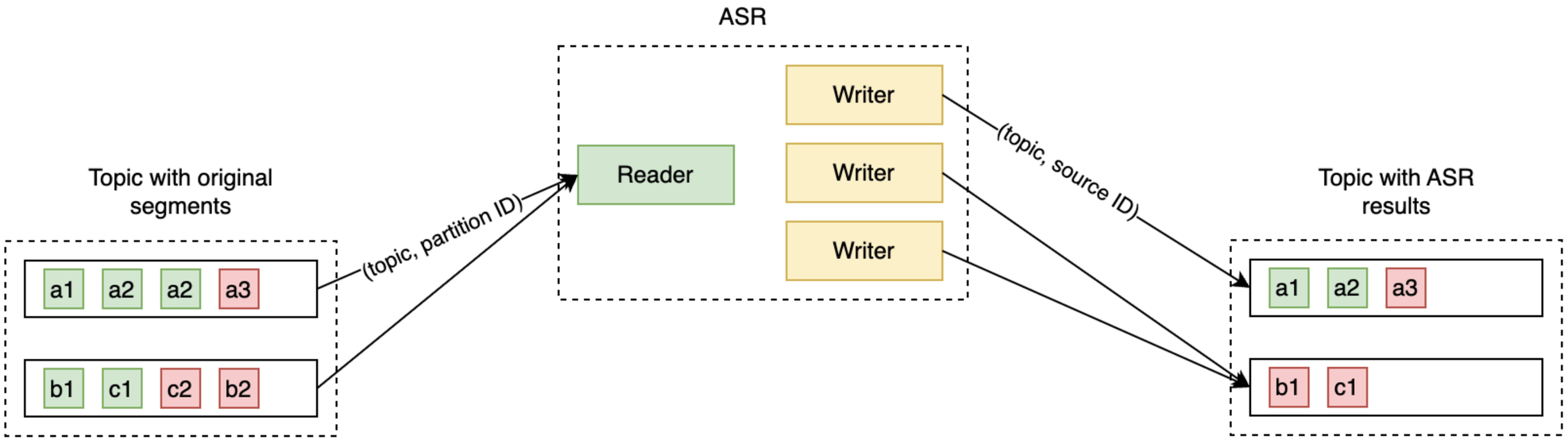
Октябрь 2022
Добавление Topic API

Март 2023
Поддержка Topic API в python ydb sdk



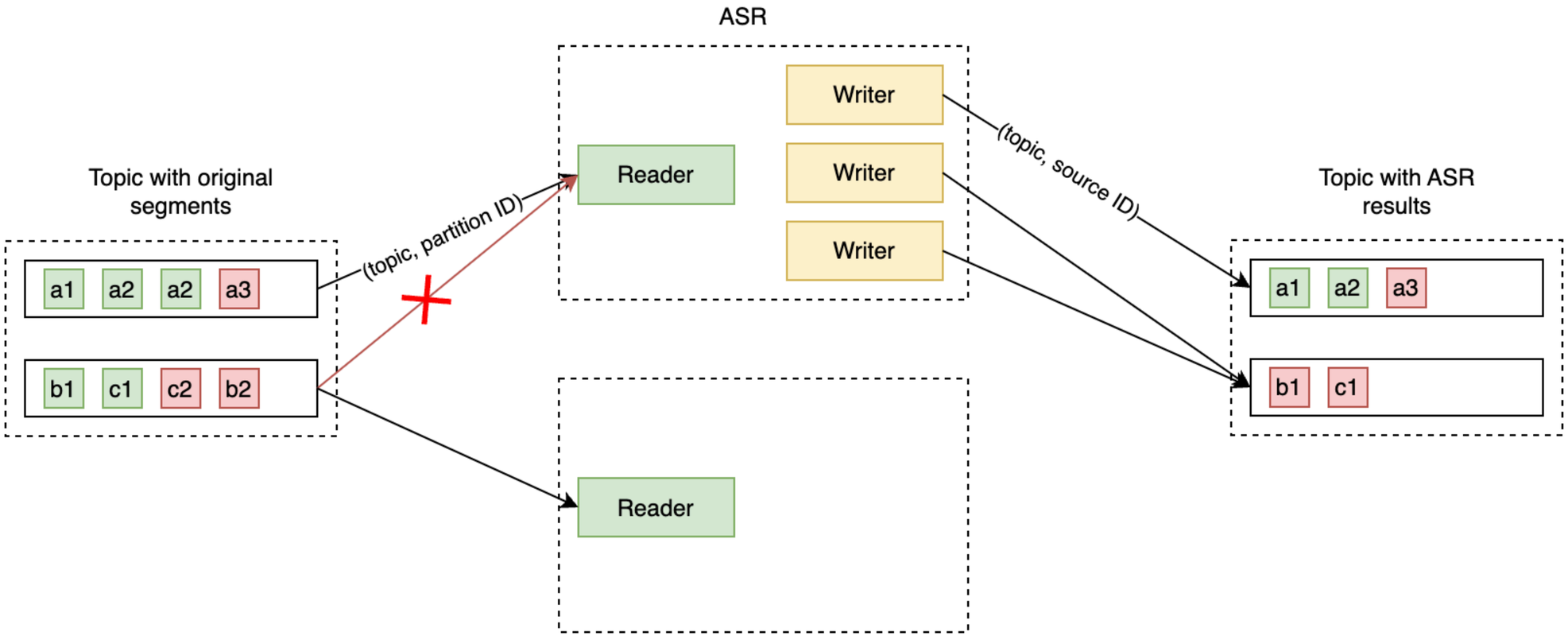
Topics in YDB

YDB



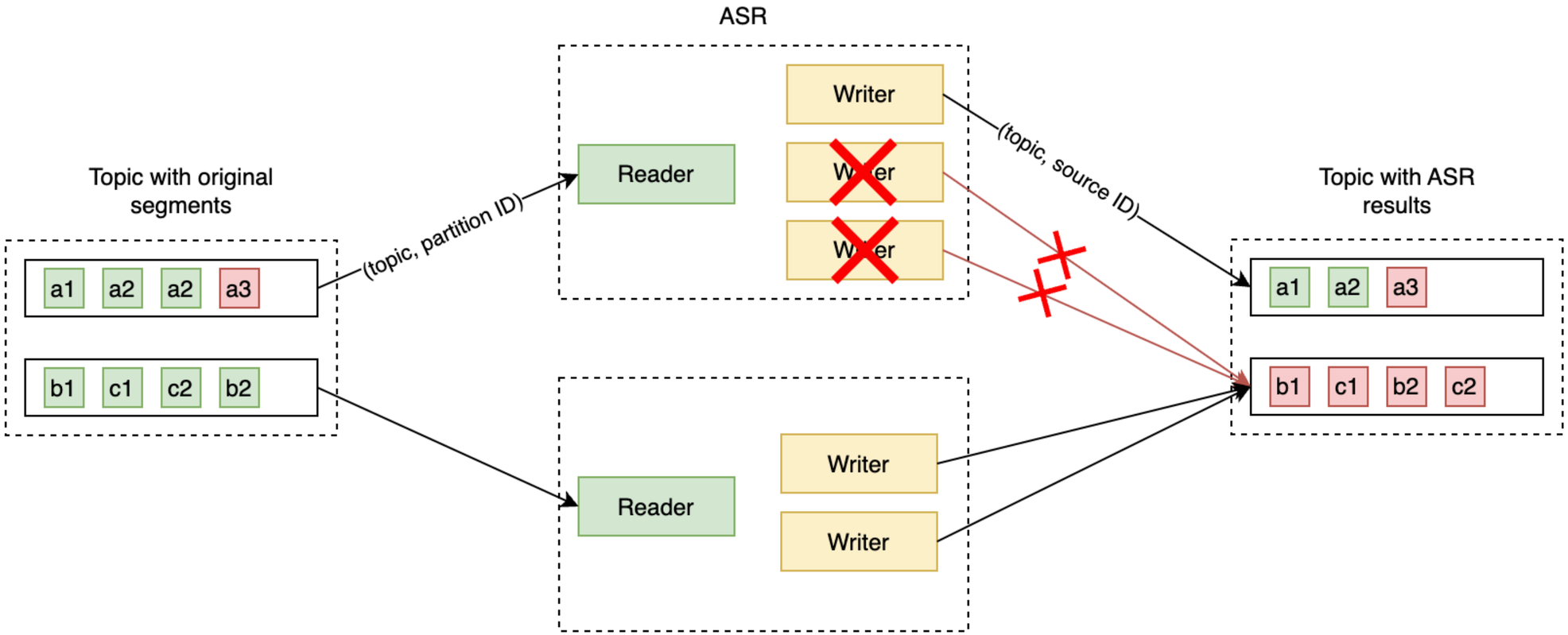
Topics in YDB

YDB



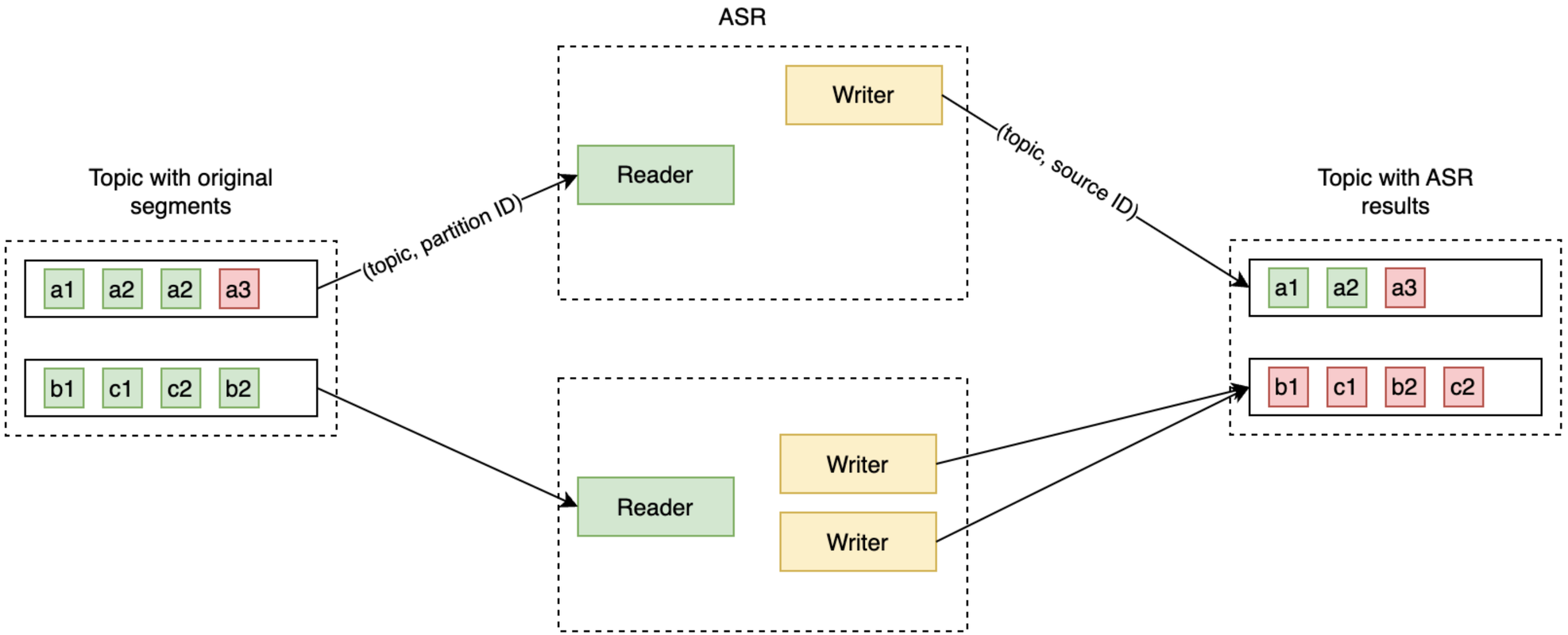
Topics in YDB

YDB



Topics in YDB

YDB



YDB vs Apache Kafka®

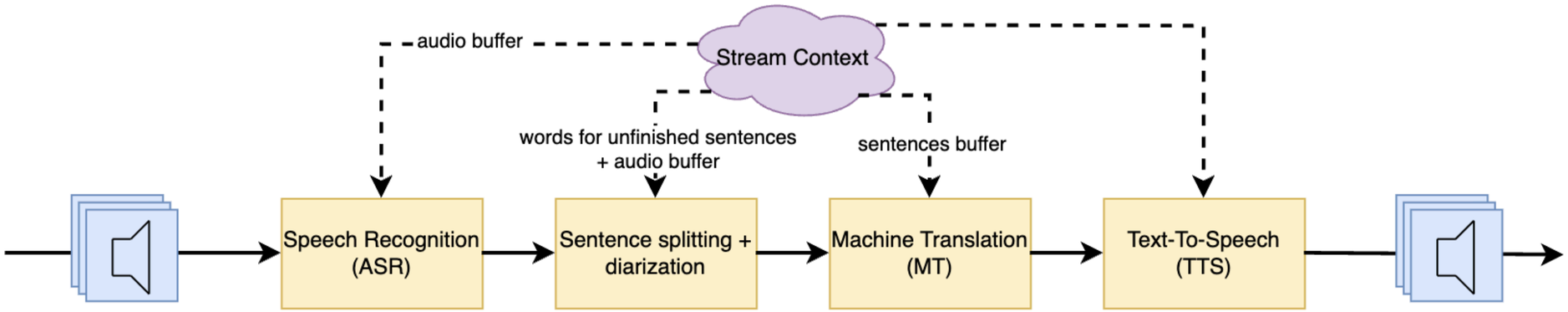
	YDB	Apache Kafka®
At-least-once при чтении	+	+
FIFO в рамках одной партиции/ сообщений записанных по одному ключу	+	+
Exactly-once при записи	+ дедупликация по паре *(source ID, seq no)*	± Idempotent producer (только в пределах одной сессии, т.к. *Producer ID* назначается при подключении)
Конкурентная запись сообщений с одним ключом	– Два продюсера не могут иметь одинаковую пару *(topic, source ID)*	+

YDB vs Apache Kafka®

	YDB	Apache Kafka®
Ребалансировка	Задевает только патриции, которые надо переназначить	По дефолту stop-the-world (есть incremental cooperative rebalancing)
Порядок коммита сообщения	Коммит надо делать для каждого сообщения. Можно делать в произвольном порядке, но подтверждение придет только тогда, когда все сообщения до были закоммичены	Во время коммита сдвигается оффсет на указанный. За порядком коммита надо следить на клиентской стороне
Протокол	gRPC API + Kafka Wire Protocol	Kafka Wire Protocol (binary protocol over TCP)
Транзакции между топиками и таблицами	TBC	–

Пайплайн потокового перевода + YDB Topics

- Сегменты для одного стрима последовательно записываем с одним source ID во входную очередь/топик (source ID = stream ID, seq no = segment number)
- Все последующие компоненты пишут результат в свой выходную очередь топик с тем же source ID и seq no
- Удачно используем нового клиента в python YDB SDK



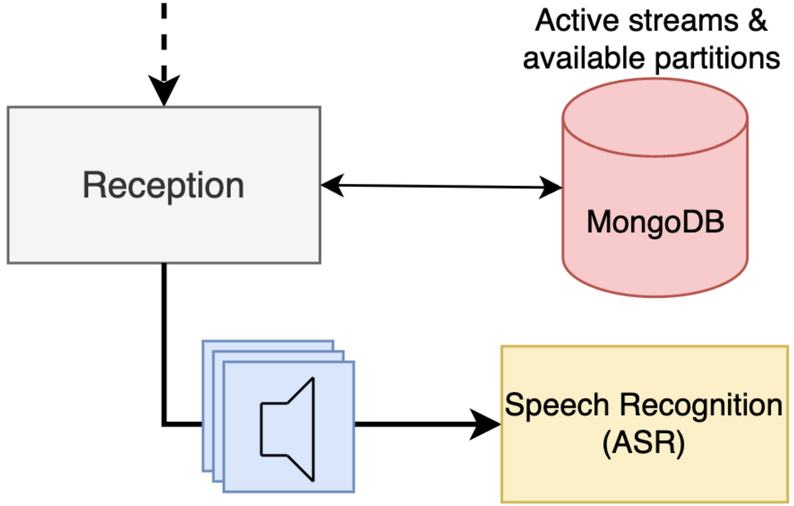
Как гарантировать «равномерную» нагрузку

Проблема

- Партиции назначаются равномерно между активными читателями
- Выбор патриции при запуске перевода новой трансляции происходит случайным образом
- Может произойти ситуация, когда инстанс конкретной компоненты будет не успевать обрабатывать

Решение

- Храним в DB вместе с информацией об активных стримах, еще и информацию о «свободных» патрициях
- Можем спокойно масштабировать максимальное число стримов с шагом равным числу партиций



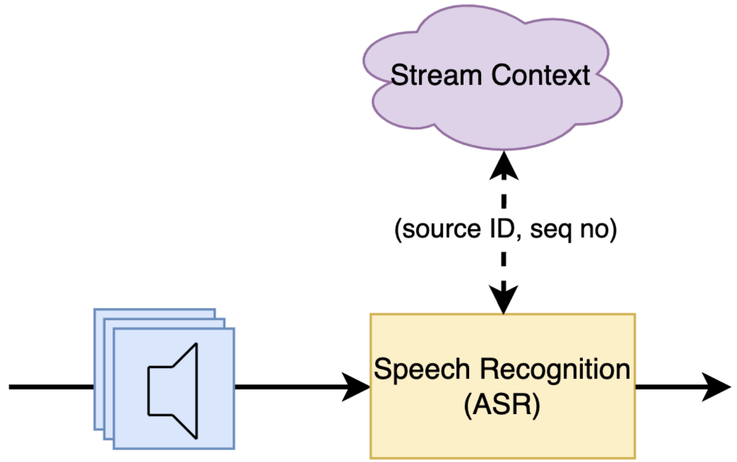
Идемпотентность при обработке сообщений

Проблема

- Одинаковые сообщения могут быть прочитаны и обрабатываться 2 и более инстансами
- Одновременная обработка разными инстансами может привести к несогласованности данных (напр. “подвисший” инстанс может записать в стейт устаревшие данные)

Решение

- Делать все воркеры идемпотентными. Напр. хранить стейт не для трансляции, а для уникальной пары *(source ID, seq no)*



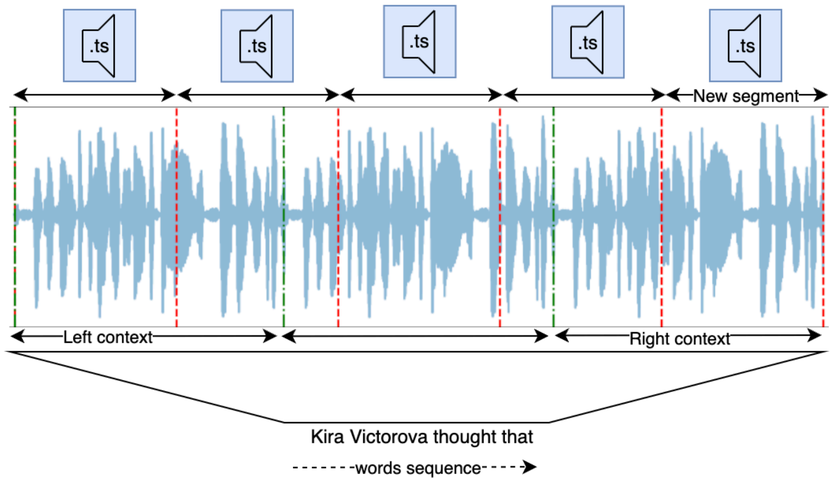
Хранить в редисе буфер с аудио затратно

Проблема

- В **ASR** и **Sentence Splitter + Diarization** нужно иметь правый и левый контекст, которые изначально хранились в редисе
- Для масштабирования всего пайплайна нужно шардировать редис

Решение

- Храним в редисе не сами сегменты, а только ссылки на них и информацию о начале левого контекста
- Т.к. ребалансировка это редкая операция, то локально для каждого стрима храним необходимые предыдущие фрагменты аудио (в несколько раз уменьшаем нагрузку на сеть)
- В случае переназначения партиций скачиваем необходимые фрагменты



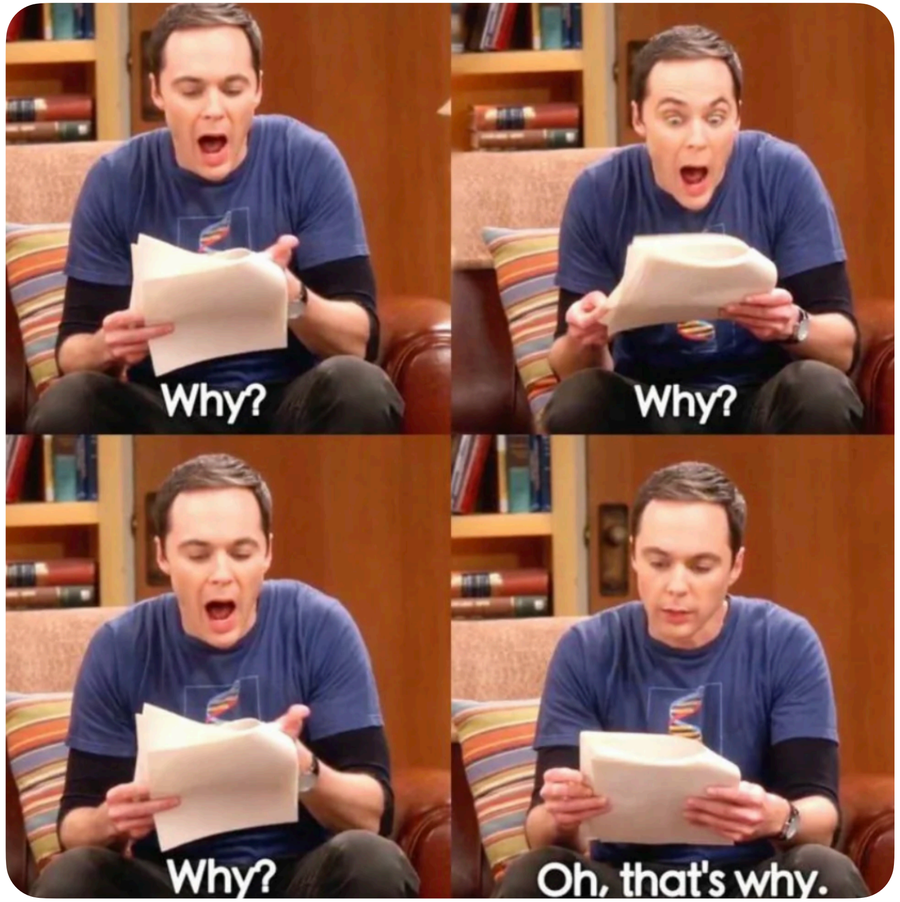
Почему так долго масштабировались?

Первые попытки переезда на YDB Topics (aka Logbroker) были не удачные

- Ретраили все ошибки писателя (включая *ownership session is killed by another session*)
- Создавали сотни читателей на 1 инстансе

А что ещё?

- Пришлось сильно переделать кодовую базу и уйти от запуска питоначьих процессов на asyncio
- Масштабироваться в 100 раз — это всегда тяжело



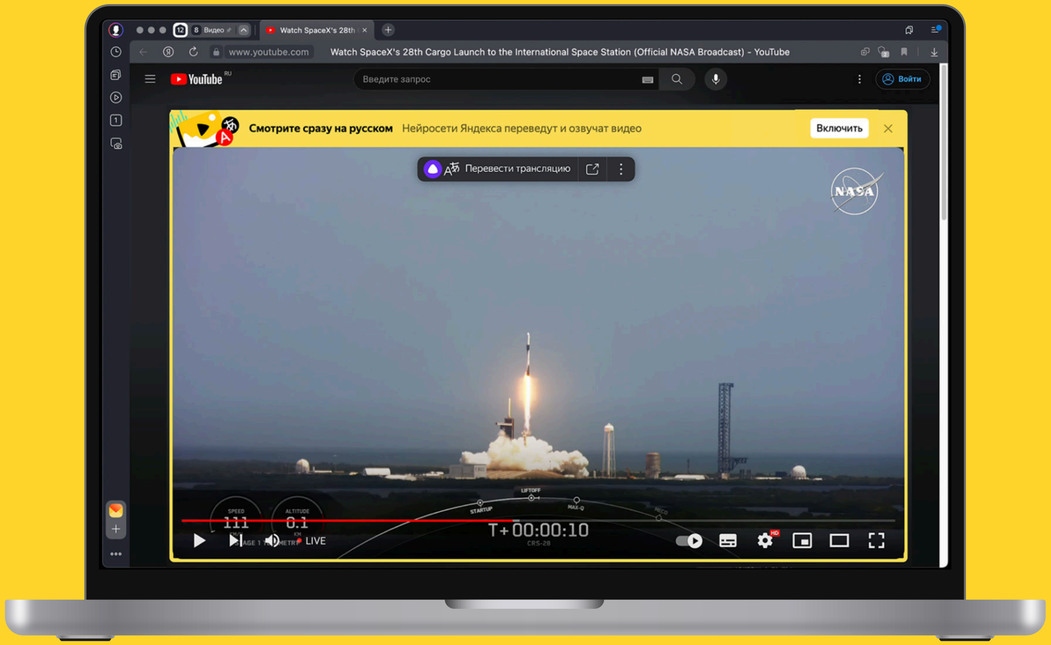
Заключение

Общие слова

- Если задумываетесь о брокере сообщений, то посмотрите на YDB (просто заводится из коробки и очень понятный клиент)
- И не забывайте смотреть любимые трансляции в переводе в Яндекс Браузере

Полезные ссылки

- Статья на хабре про перевод трансляций
- Документация YDB
- Классная детская книжка про Apache Kafka®



Яндекс

Спасибо!

Арсентий Мельников