DevConf

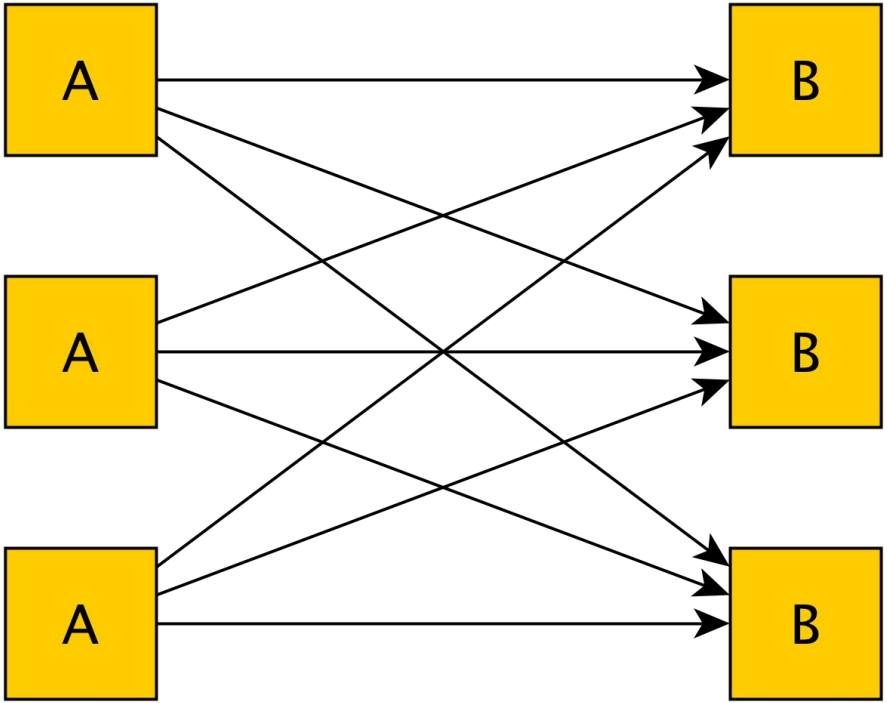
www.devconf.ru

Как мы построили сервис распределённых очередей в Яндексе

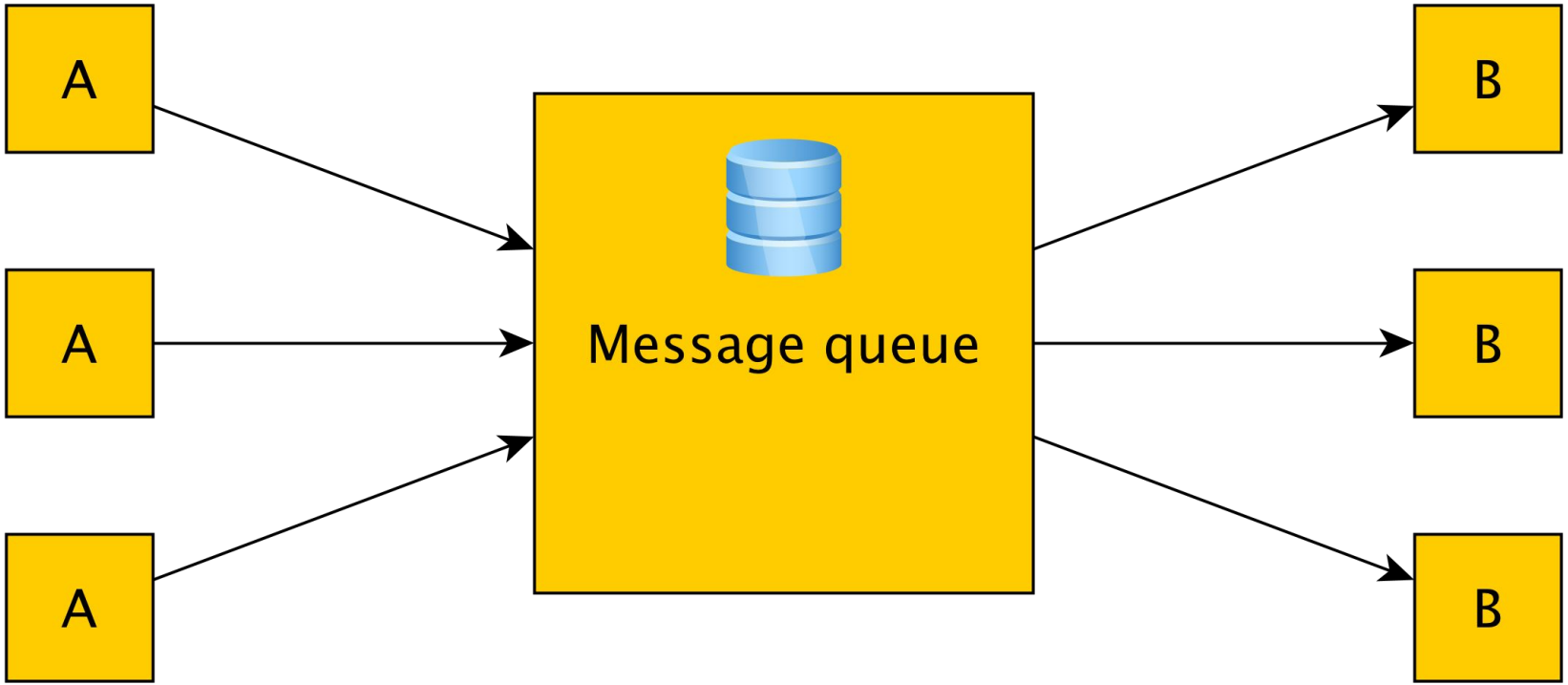
Василий Богонатов, разработчик сервиса

Часть 1, вводная

Коммуникация сервисов



А теперь с помощью очереди



Очередь — компонент для надёжной передачи сообщений

Что дают нам очереди сообщений

- Развязка компонент
- Масштабируемость
- Отказоустойчивость
- Эластичность
- Контроль обработки

Например: проверка решений

- Сниппеты кода складываются в очередь
- Код из очереди запускается и тестируется, результат пишется в базу

Примеры. Celery

Celery — фреймворк для асинхронных задач

- Очередь используется для доставки асинхронных задач
- Воркеры берут задачи из очереди и выполняют их

Основные свойства очередей YMQ

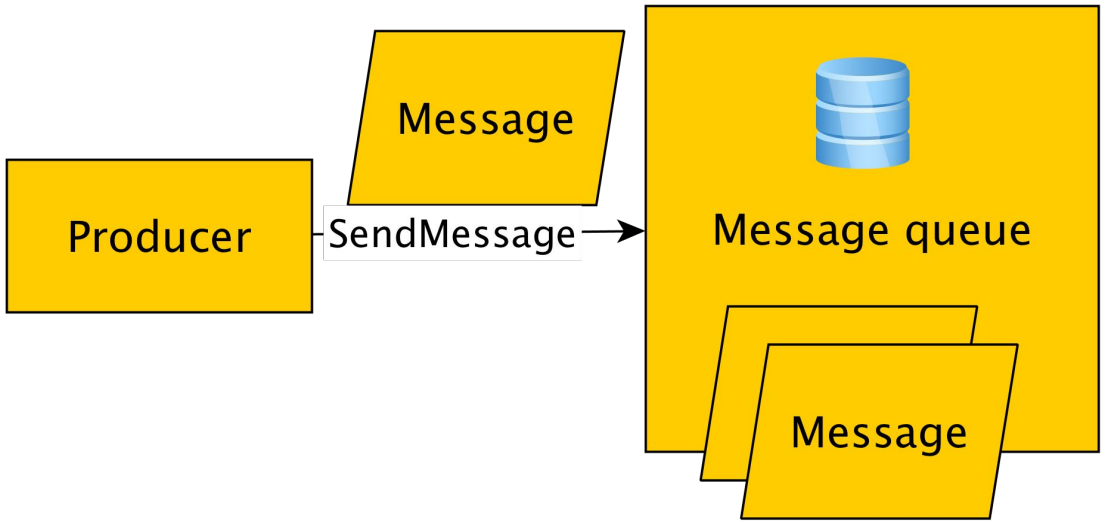
- Http-API, совместимый с Amazon SQS
- Персистентные (переживают потерю датацентра и ещё одной машины)
- Полностью управляемая (“managed”)
- Паттерн “competing consumers”
- Неупорядоченные (но есть и FIFO-очереди)
- Низкие лейтенси операций (порядка 100 мс)

Интерфейс YMQ

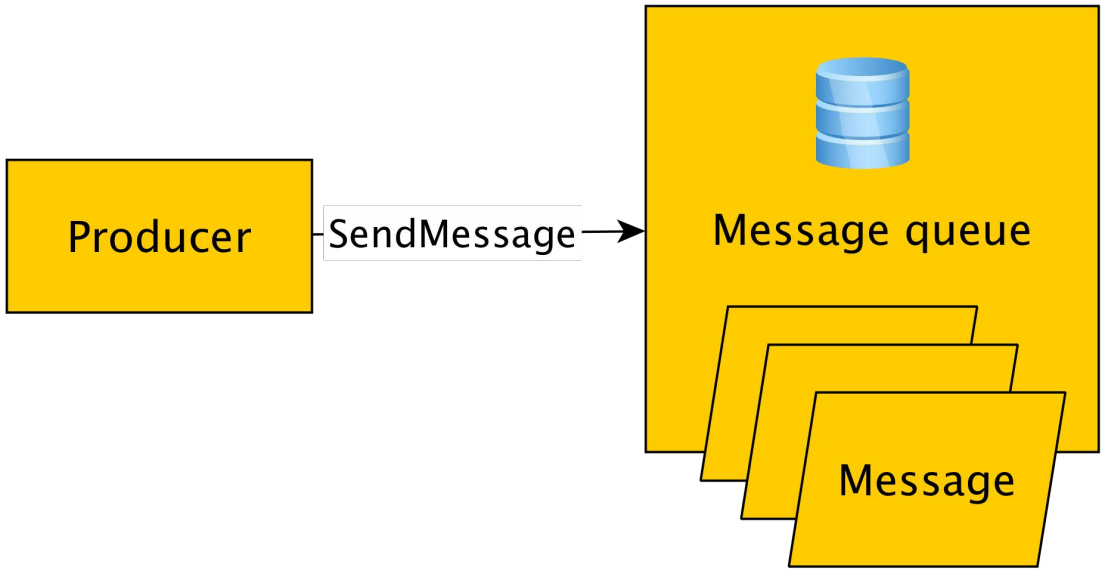
Основные методы:

- SendMessage
- ReceiveMessage
- DeleteMessage

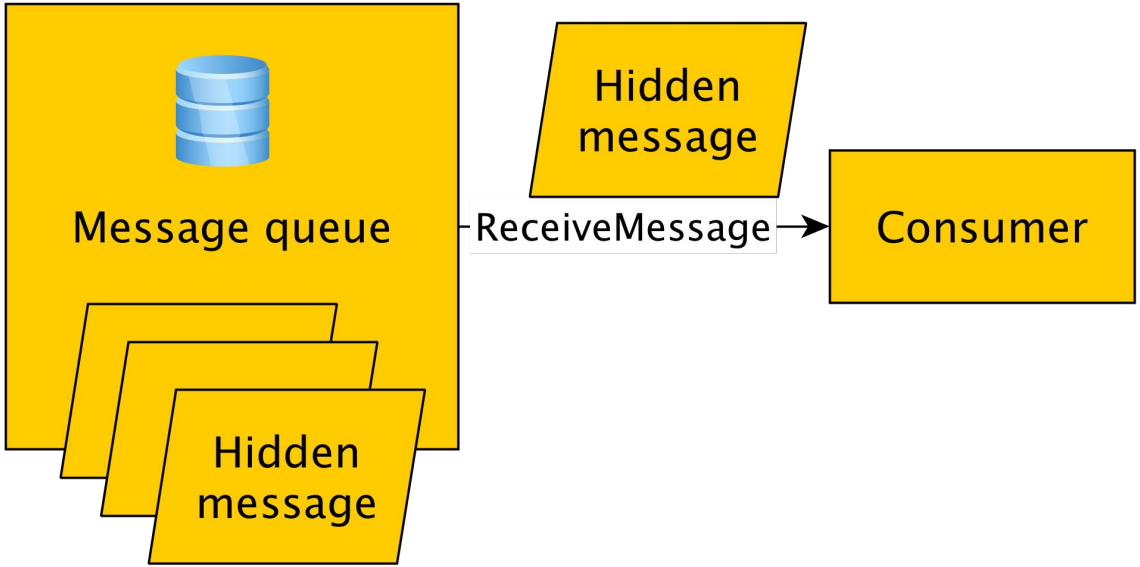
Интерфейс YMQ



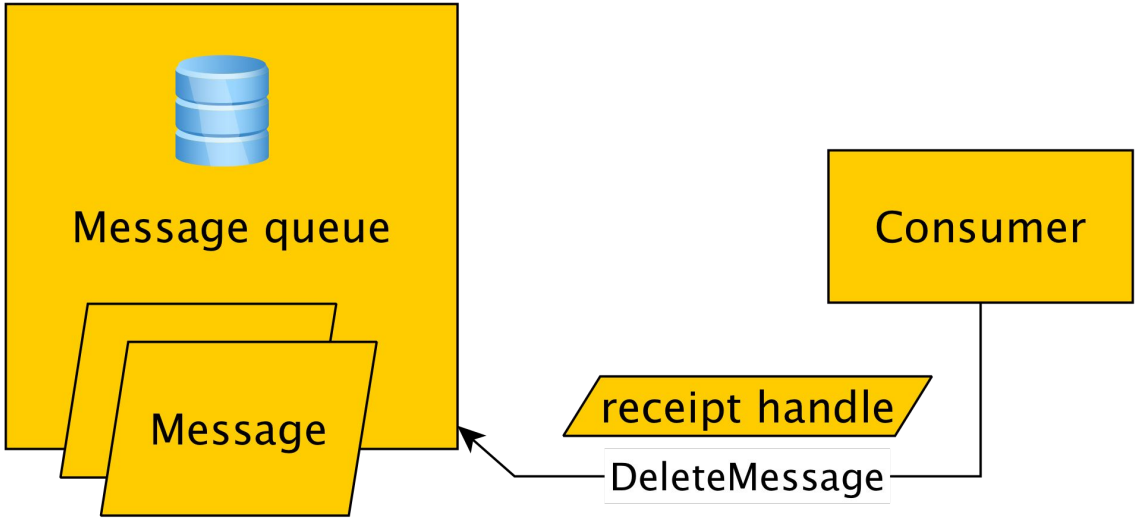
Интерфейс YMQ



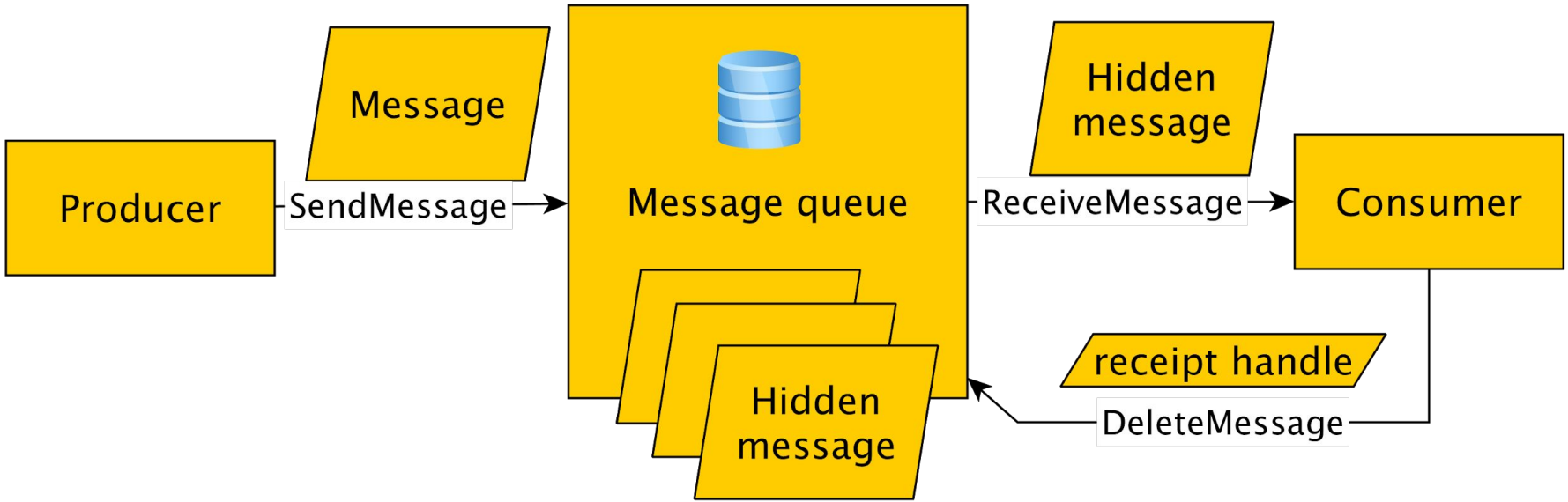
Интерфейс YMQ



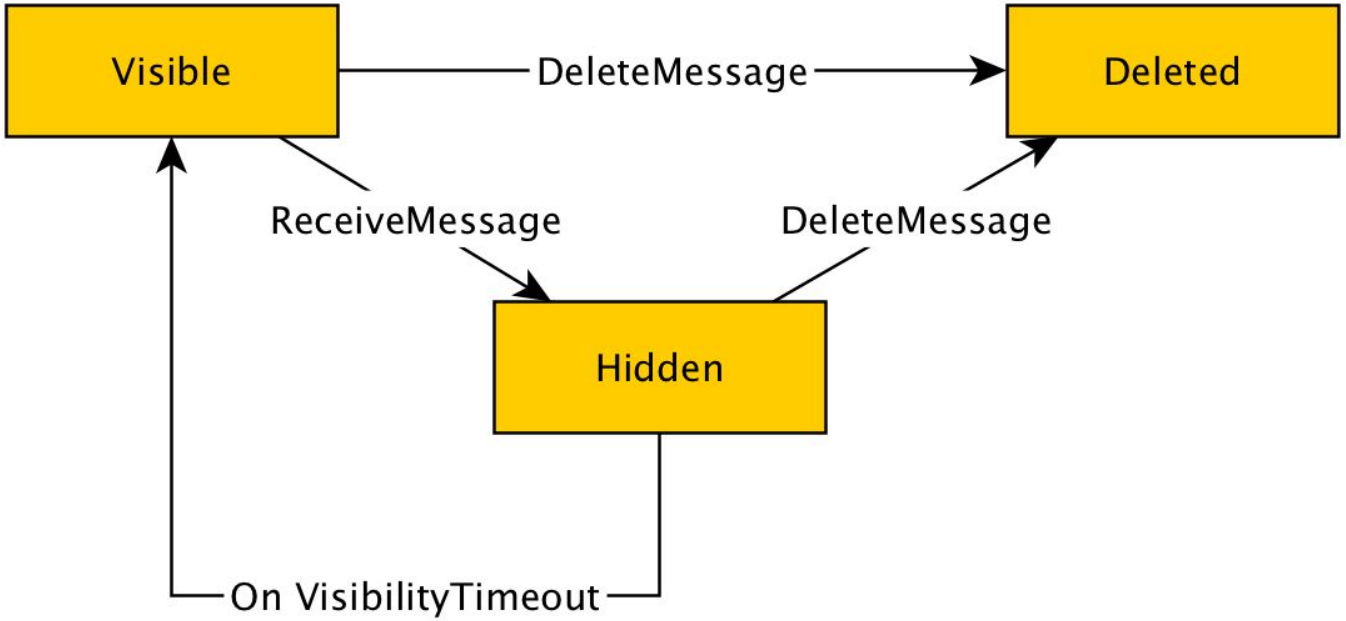
Интерфейс YMQ



Интерфейс YMQ



Жизненный цикл сообщения



Часть 2, архитектурная

Yandex Message Queue построена поверх Yandex Database

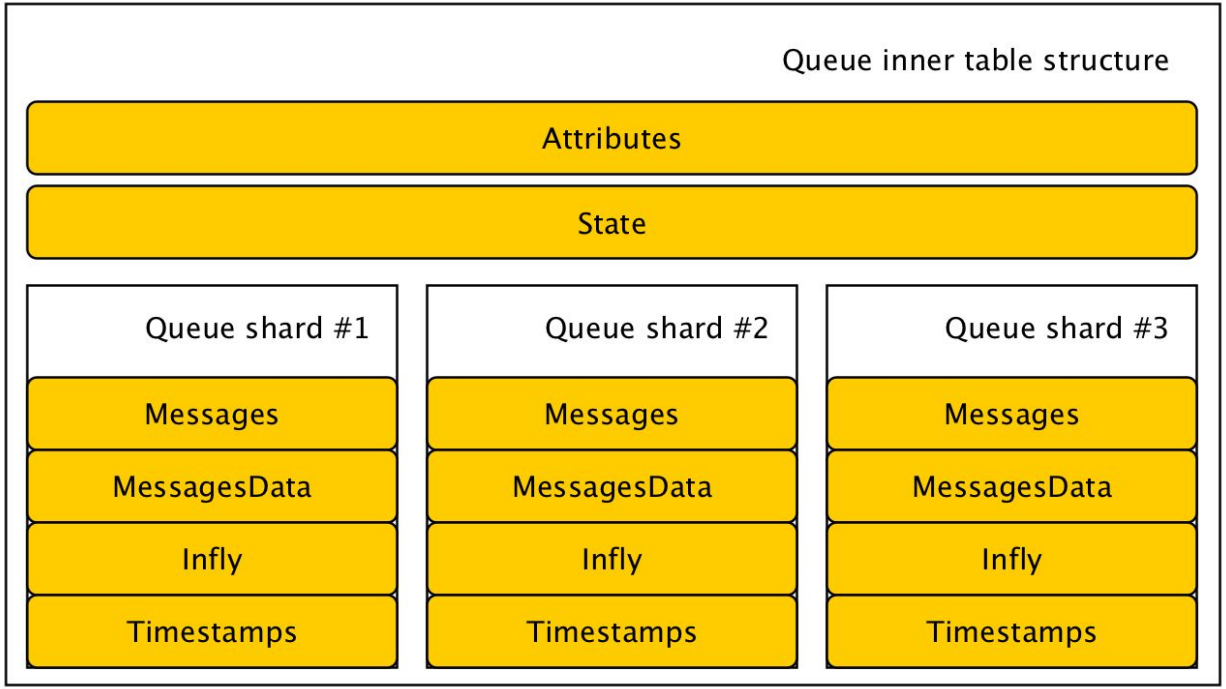
Важные свойства Yandex Database

- Распределённая
- Горизонтально масштабируемая
- Отказоустойчивая
- Поддерживает ACID-транзакции
- Поддерживает диалект SQL (YQL)
- Строго консистентная

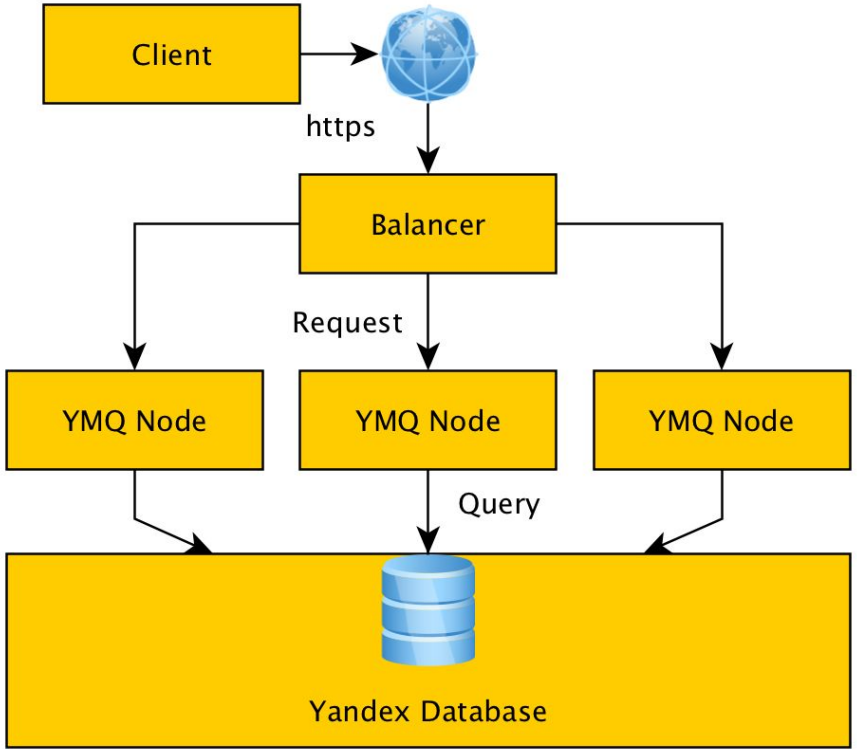
Архитектура YMQ

- Очередь — набор шардов
- Шард — набор таблиц в базе данных
- Операции — запросы к таблицам шарда

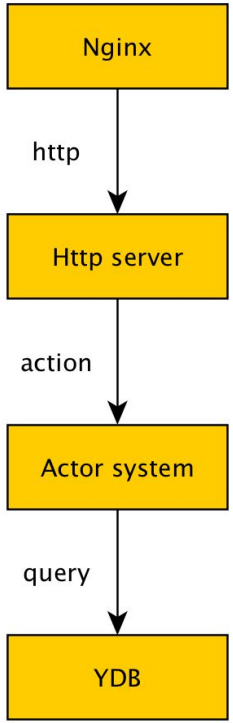
Анатомия очереди



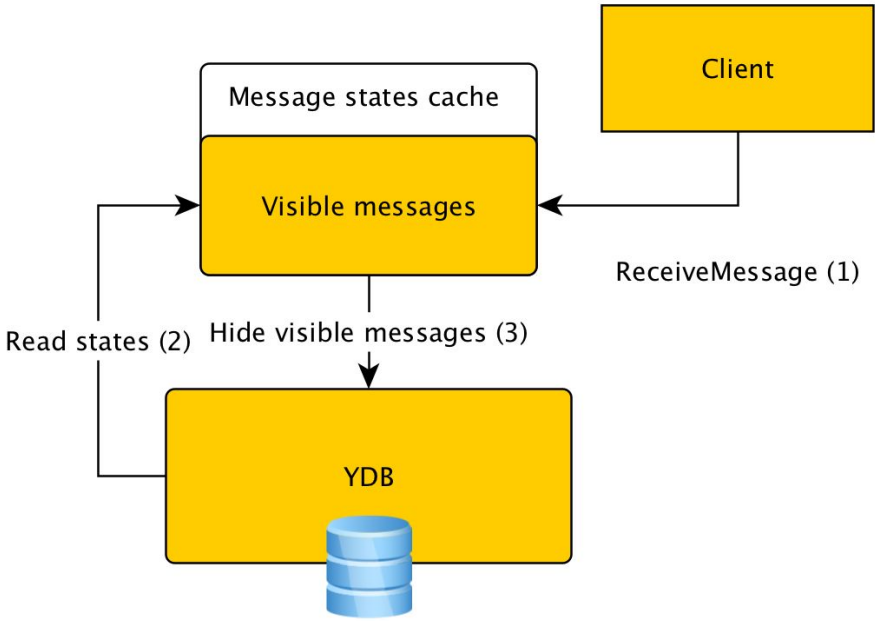
Архитектура YMQ



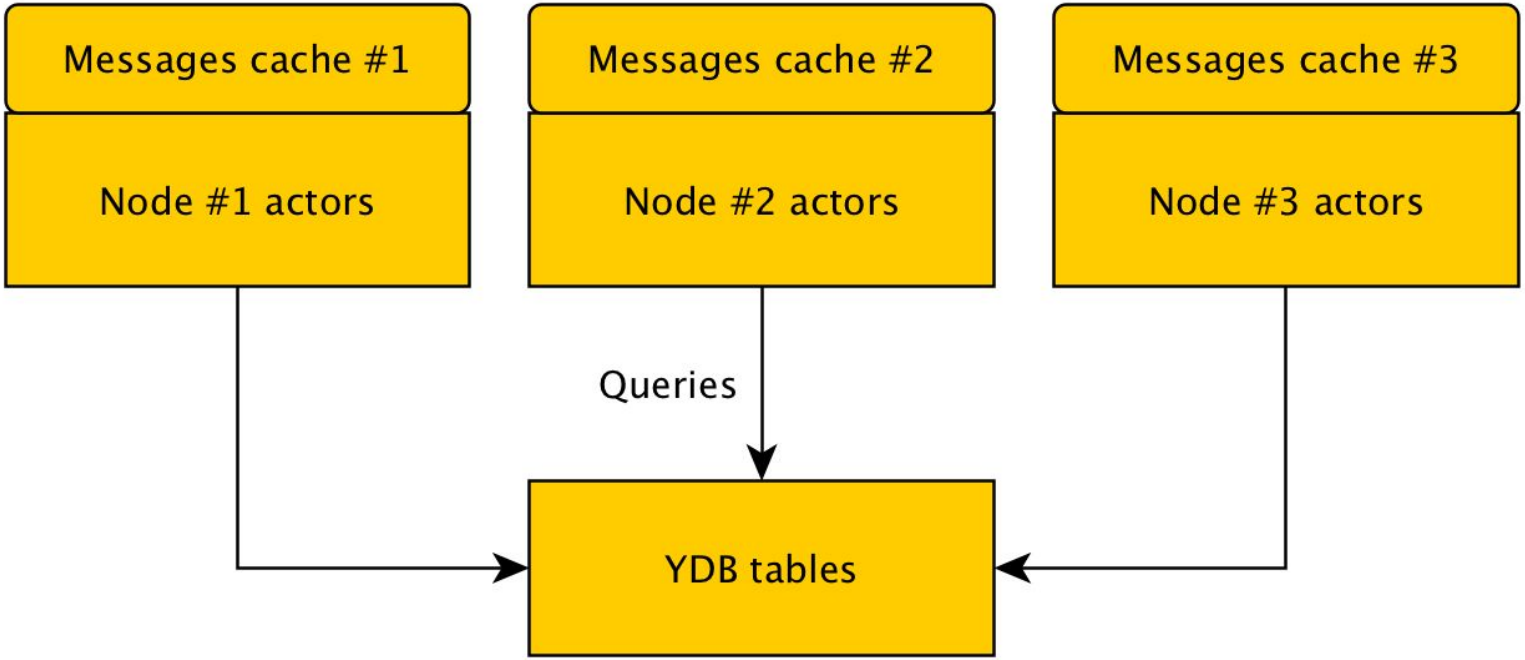
Внутри YMQ Node



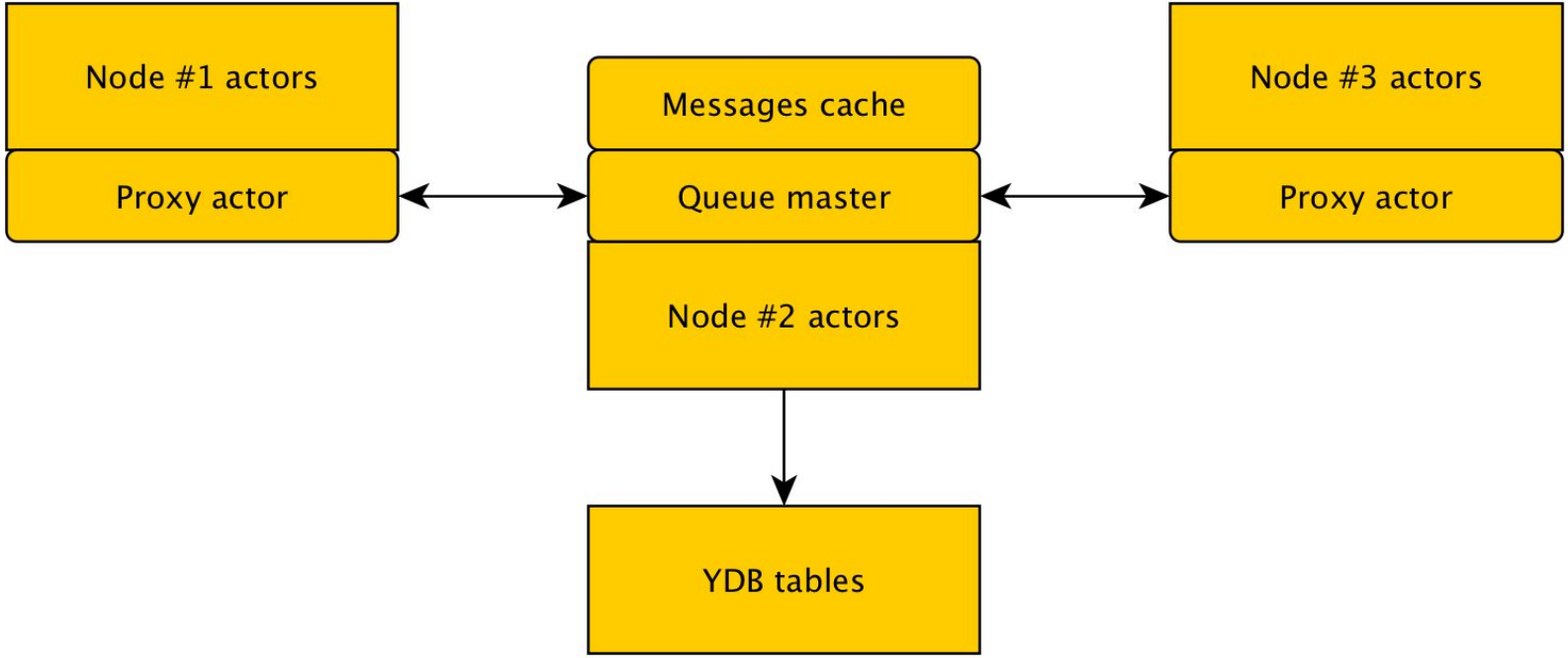
Кэш состояний сообщений



Несогласованный кэш



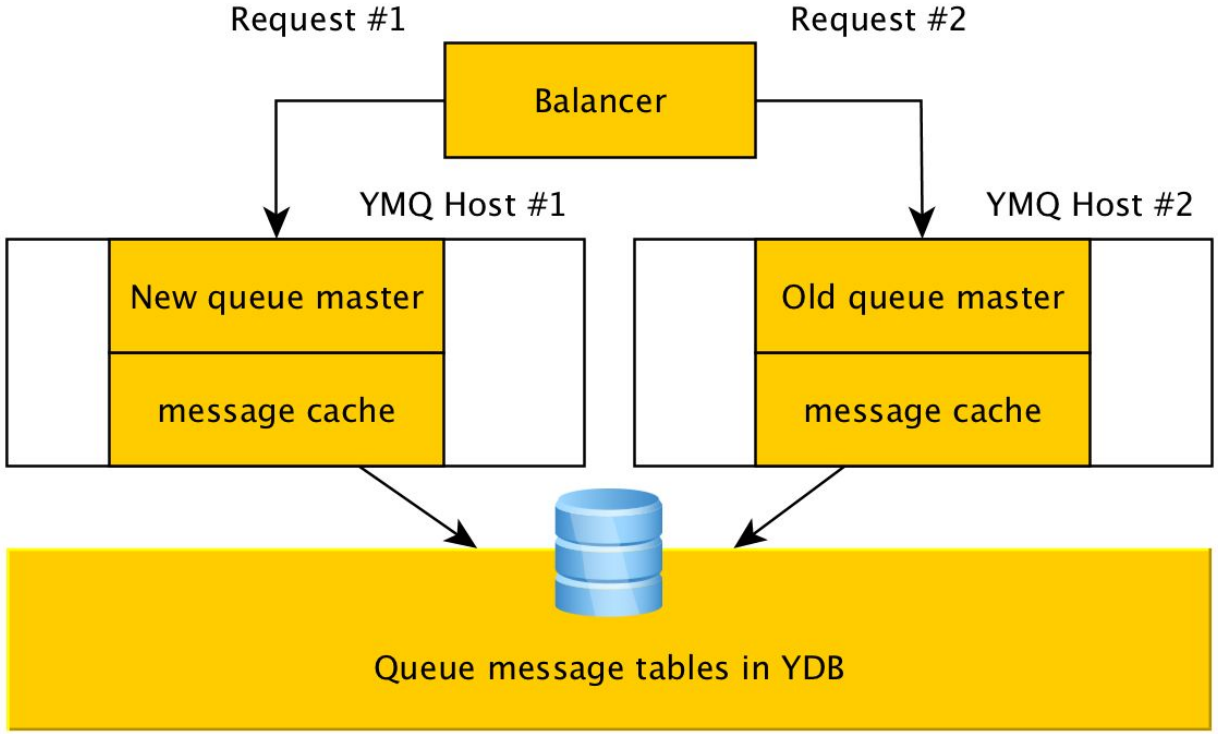
Решение проблемы: мастер очереди



Мастер

- Согласованное кэширование информации
- Единая точка управления выполнением запросов
- Место сбора пользовательских метрик
- Оптимизация производительности

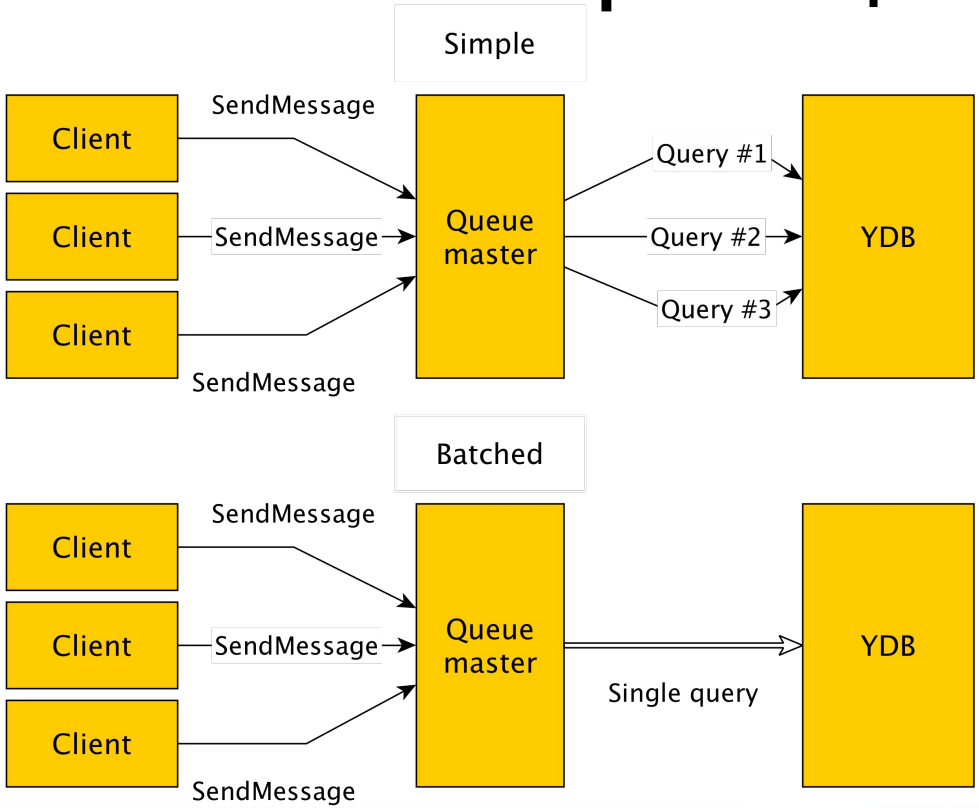
Проблема нескольких мастеров



Проблема нескольких мастеров

- Состояние очередей в базе данных считаем истинным

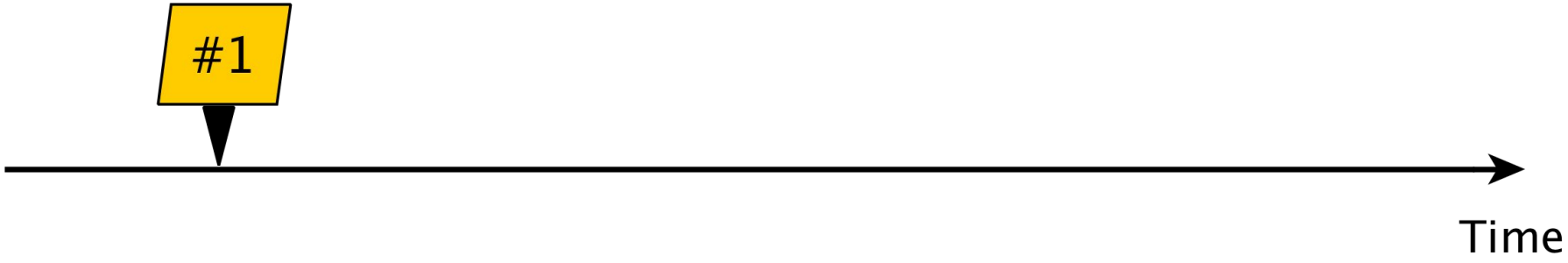
Уменьшение количества транзакций



Уменьшение количества транзакций

- Группировка однотипных транзакций
- Похоже на алгоритм Нейгла

Как работает группировка запросов

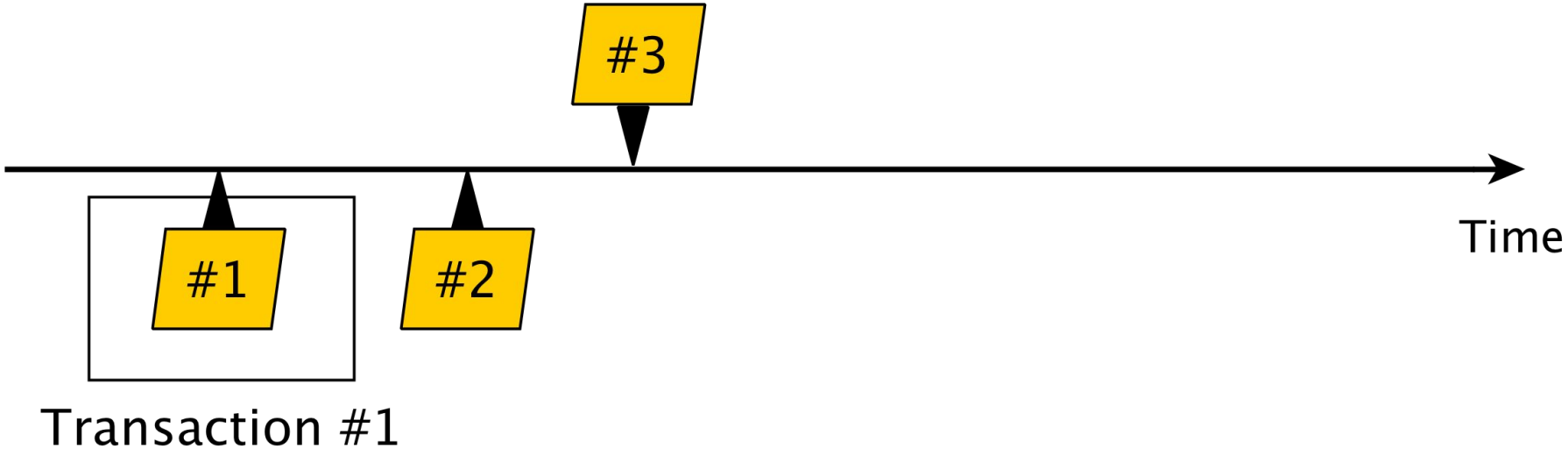


Как работает группировка запросов

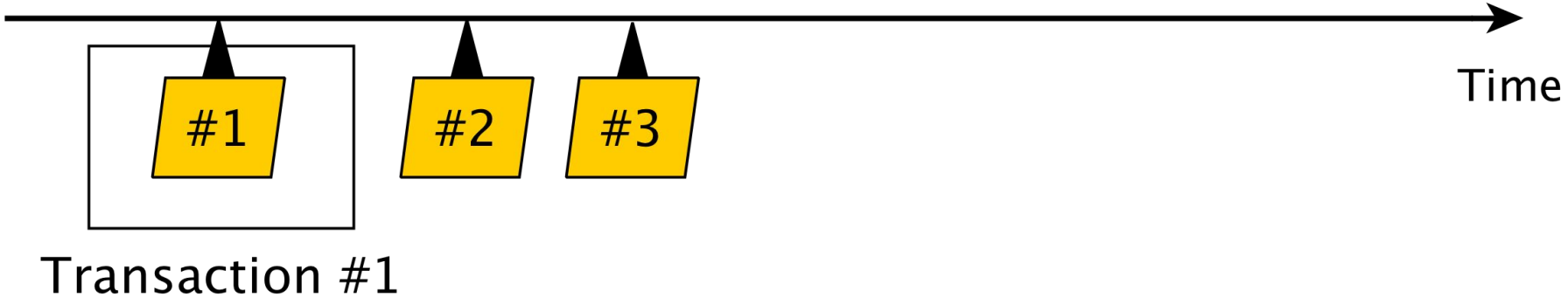
Как работает группировка запросов

Как работает группировка запросов

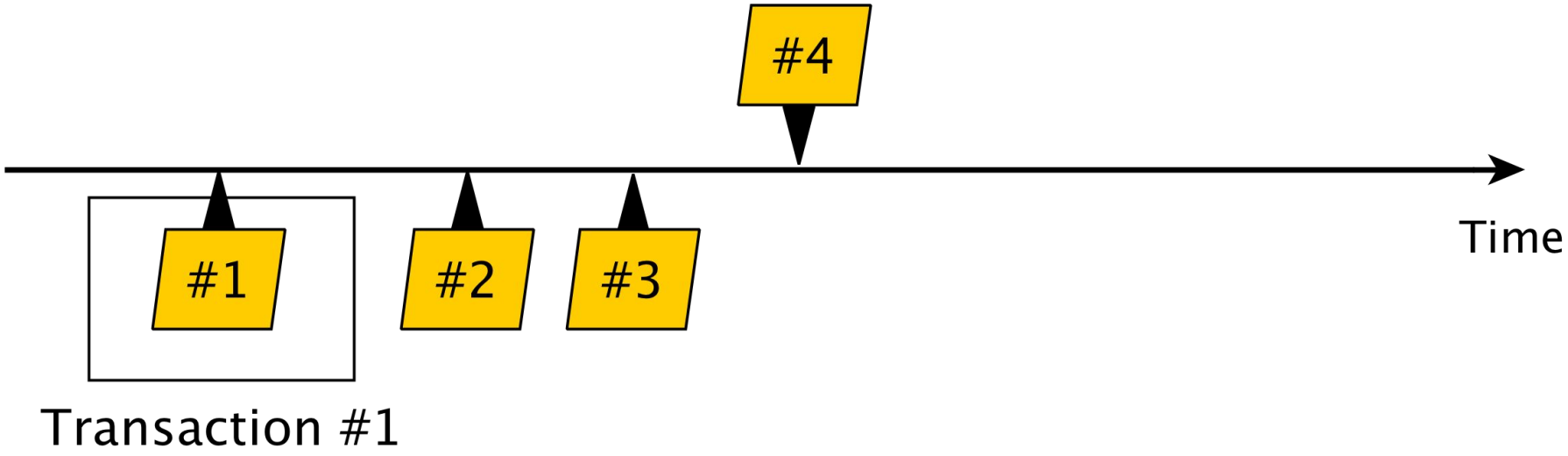
Как работает группировка запросов



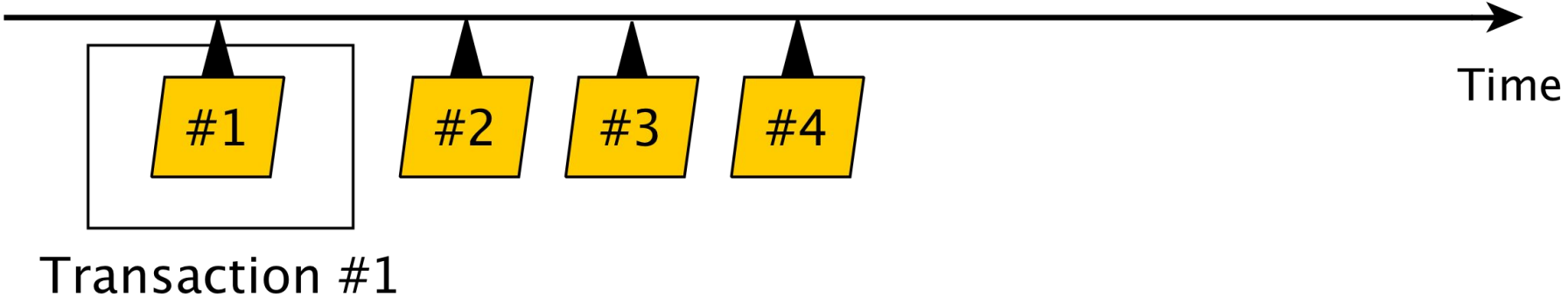
Как работает группировка запросов



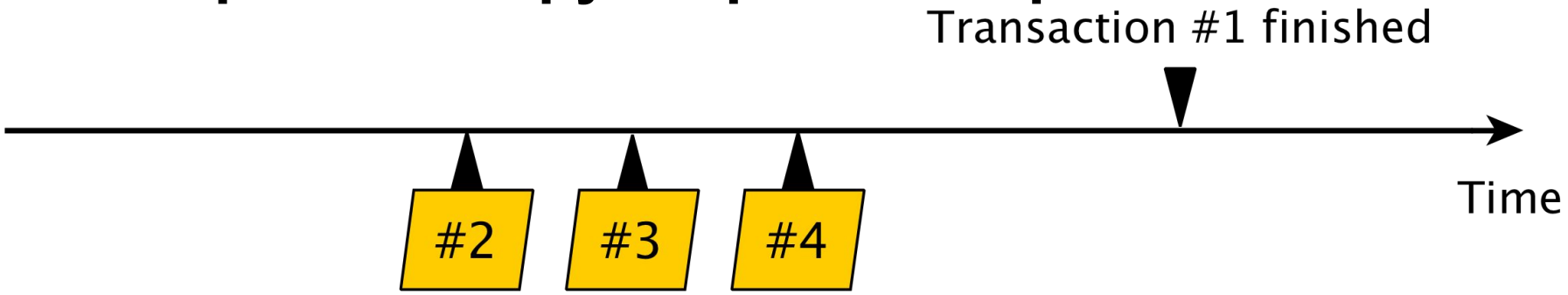
Как работает группировка запросов



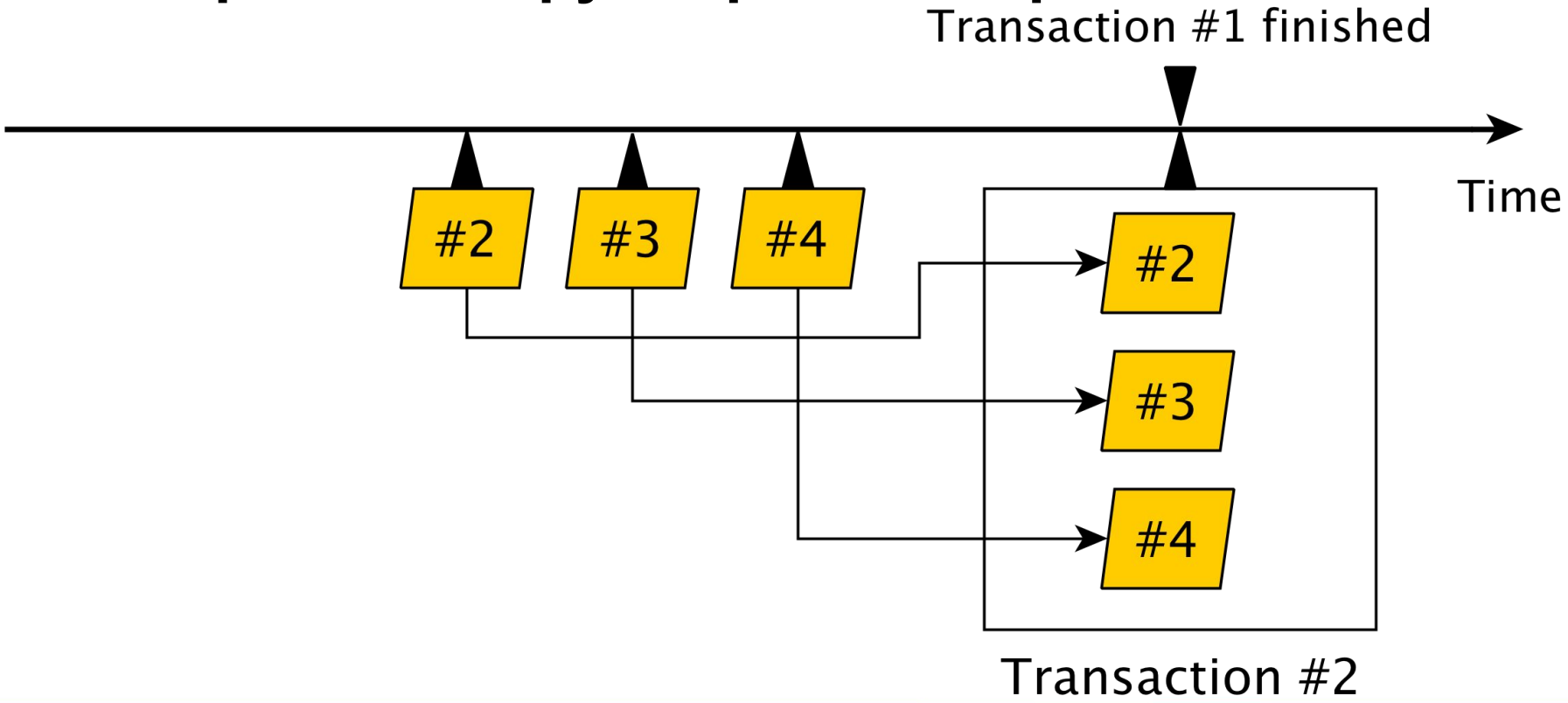
Как работает группировка запросов



Как работает группировка запросов



Как работает группировка запросов



Уменьшение количества транзакций

- Для очередей с низким рейтом — никакого оверхеда
- Для очередей с высоким рейтом — существенное улучшение
- Для очередей со средним рейтом — дополнительная задержка не более времени одной транзакции
- Пики нагрузки на очереди не ведут к пикам нагрузки в базе
- Ограничение количества одновременных транзакций в базу

Оптимизация батчинга

Эмпирически находим оптимальный для YMQ размер батча и количество параллельных транзакций

Результаты оптимизации

Для каждого шарда очереди:

Максимальный RPS:

- вырос в **60** раз!

Время пользовательских операций:

- уменьшилось в **1.5** раза!

Количество параллельных транзакций:

- упало в **2.5** раза!

Ограничение числа запросов

- Защищает от резкого роста нагрузки
- Задаётся для очереди
- Отдельное для записи/чтения/удаления

Часть 3, разработческая

Нагрузочный тест 24/7

- Простые тестовые сценарии
- Проверка системы стандартными мониторингами

Моделирование стандартных сценариев

- Проверка стандартных сценариев
- Удобно для измерений

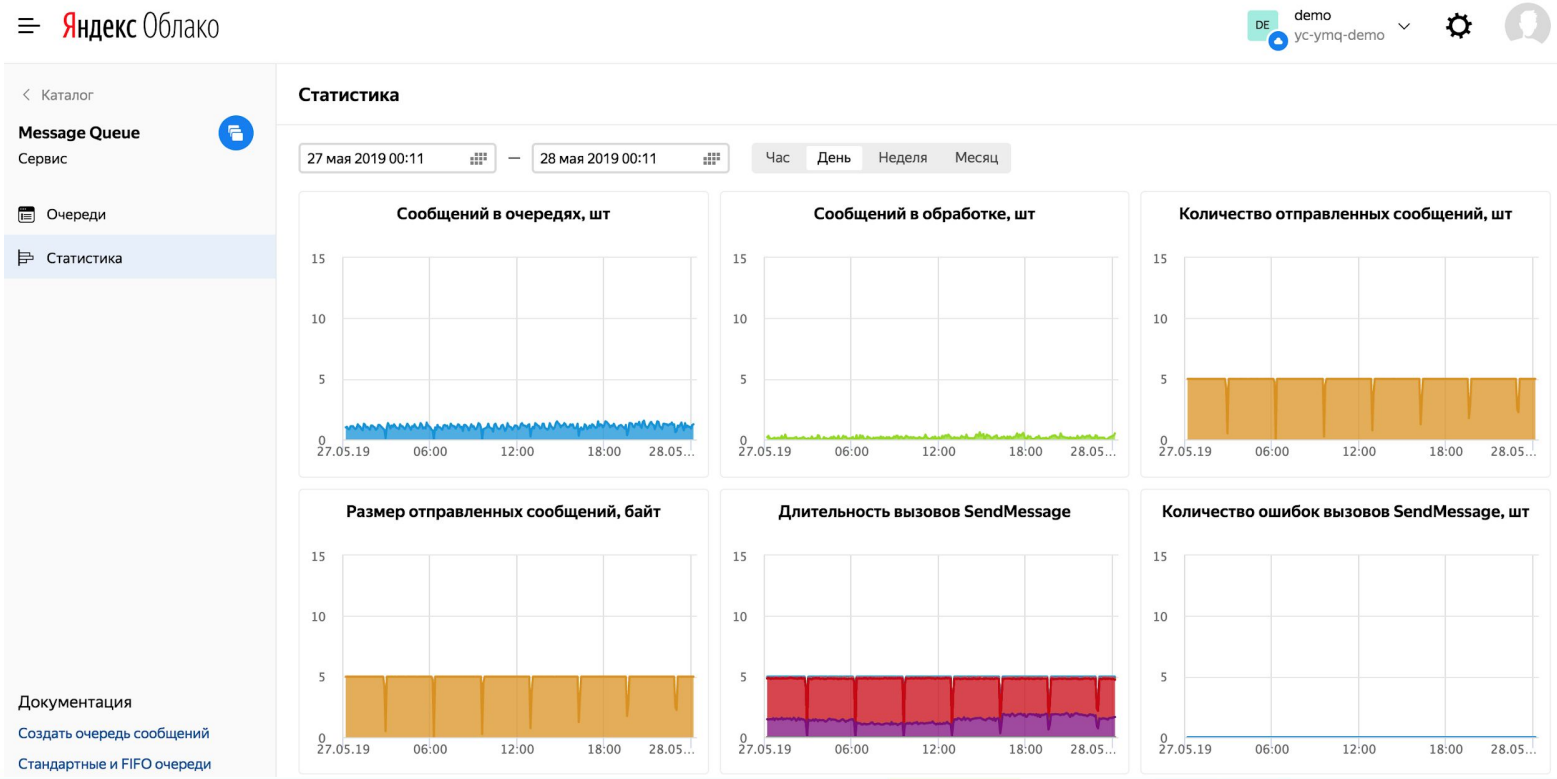
Важные эксплуатационные метрики

- Ошибки запросов (5xx)
- Время между записью и чтением
- Неуспех транзакций
- Зависание запросов

Важные пользовательские метрики

- Размер очереди
- Число попыток прочитать сообщение
- Число “невидимых” сообщений в очереди
- Возраст самого старого сообщения в очереди

Важные пользовательские метрики



Эффективная работа с логами

- Request id: привязка событий к запросу
- Сэмплирование: баланс подробности и объёма логов

Яндекс

Спасибо! Готов ответить на вопросы :-)

Василий Богонатов

разработчик сервиса

 radix@yandex-team.ru

 @drweenen