Распределённая трассировка с помощью Jaeger и Yandex Database. Опыт Auto.ru и Yandex.Cloud

Александр Салтыков
Разработчик Auto.ru

Александр Щербаков
Разработчик Yandex.Cloud

DevOps Conf 2021

auto.ru

Яндекс Недвижимость

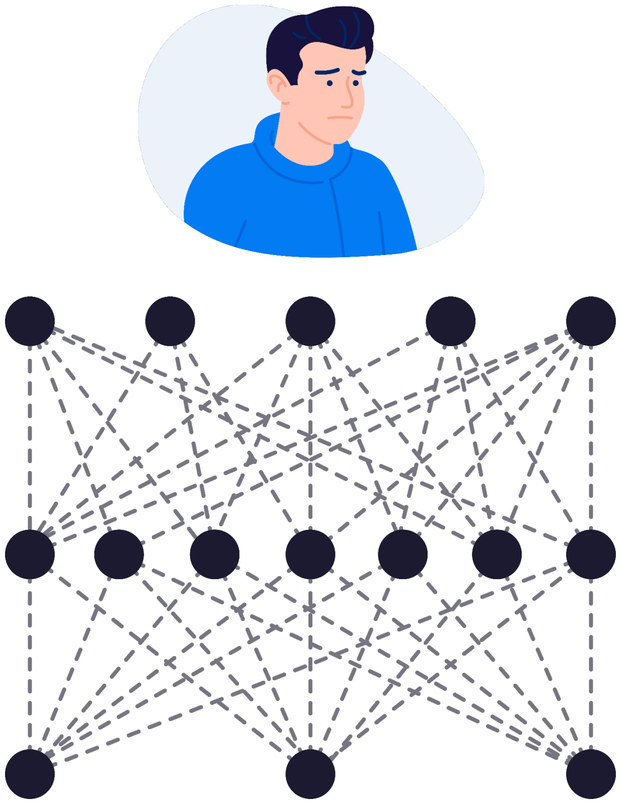
Yandex Cloud

Что такое Distributed / Open Tracing

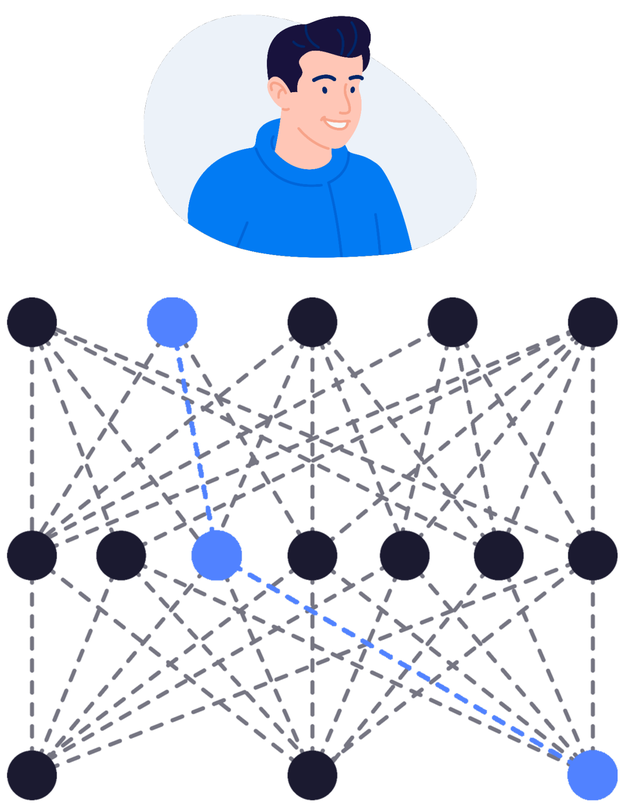
OPENTRACING

Что случилось с моим запросом?

Без распределённой трассировки



С распределённой трассировкой



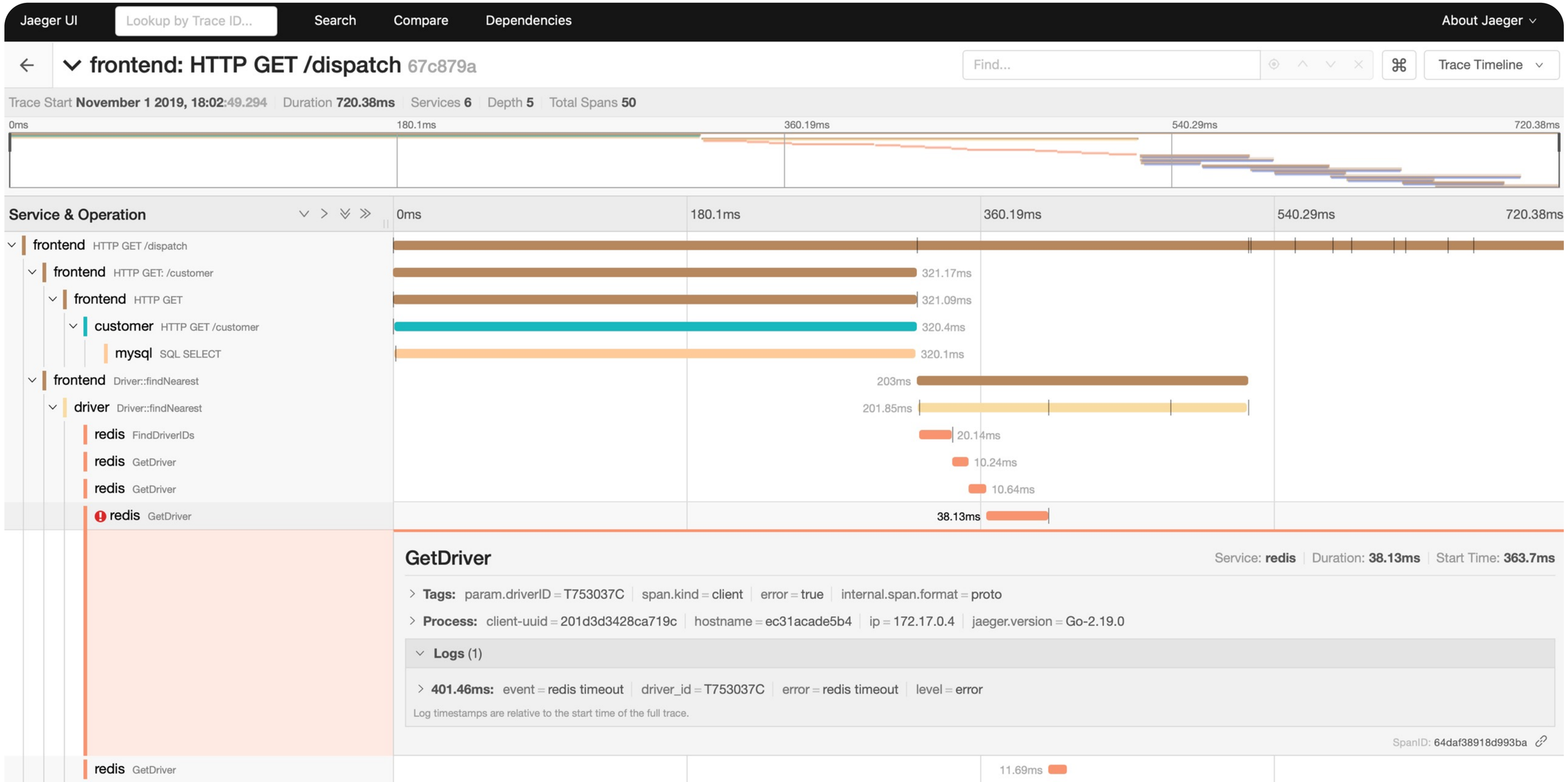
Писать всё

- По любой trace_id получить трассу
  (ни одна ошибка не уйдёт обиженной)
- Для этого — писать трассы на все запросы
  (убрать sampling — пролив части трасс)
- ~300K spans/s, исходя из prod-нагрузки

Jaeger tracing

- Golang
- Есть система плагинов
- На Zipkin тоже смотрели, не зашло

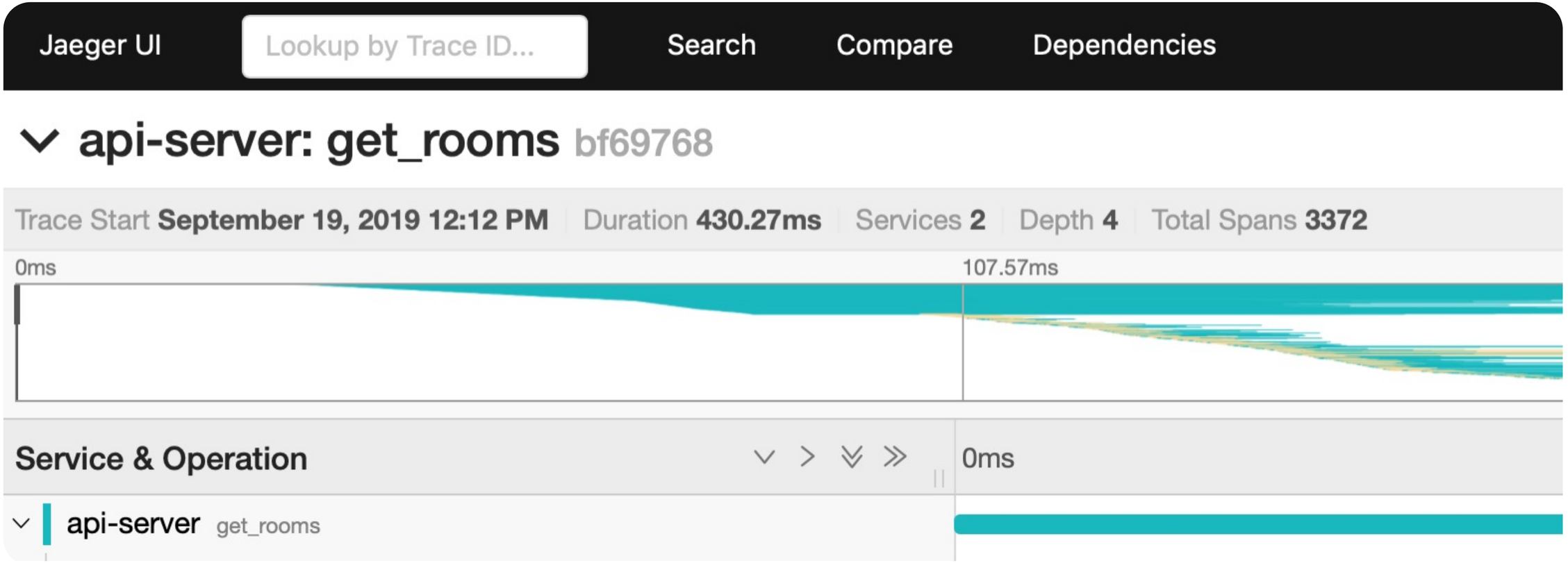
Jaeger tracing



Запрос в API

JAEGER

Запрос в API мультиплицируется в 3300+ запросов внутри



Jaeger tracing — компоненты

- Agent (sidecar)
- Collector
- Query (UI)
- DB

Требования к базе

- Георасределённая
- Отказоустойчивая
- Горизонтально масштабируемая
- С высокой пропускной способностью
- С малым временем отклика

Multiple Storage Backends

Yandex Database

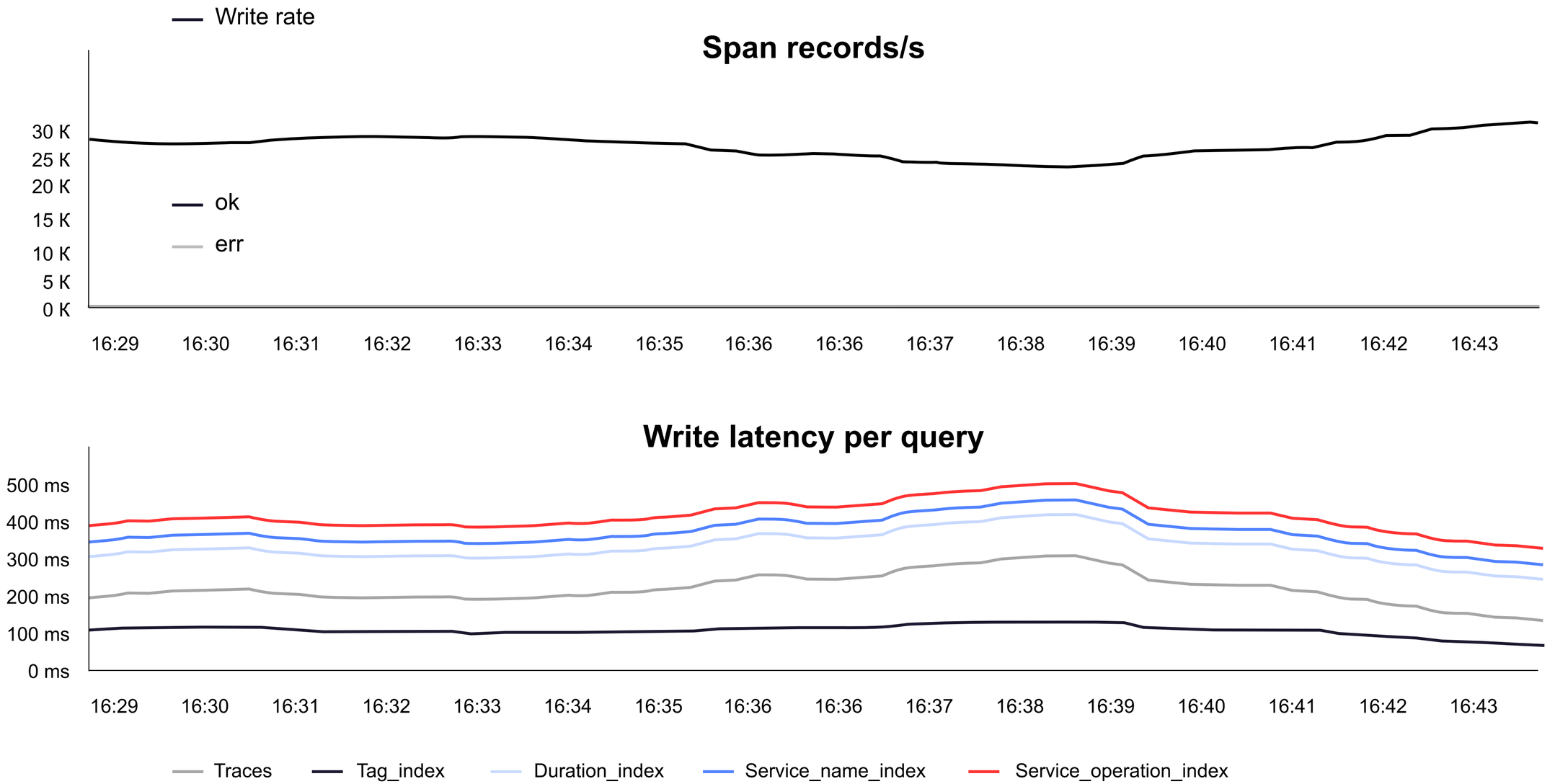
Multiple Storage Backends

- Нагрузка искусственная
- Мы написали генератор нагрузки, который собирает и пишет рандомные трейсы
- Ищем хранилище с лучшим показателем spans/core/s

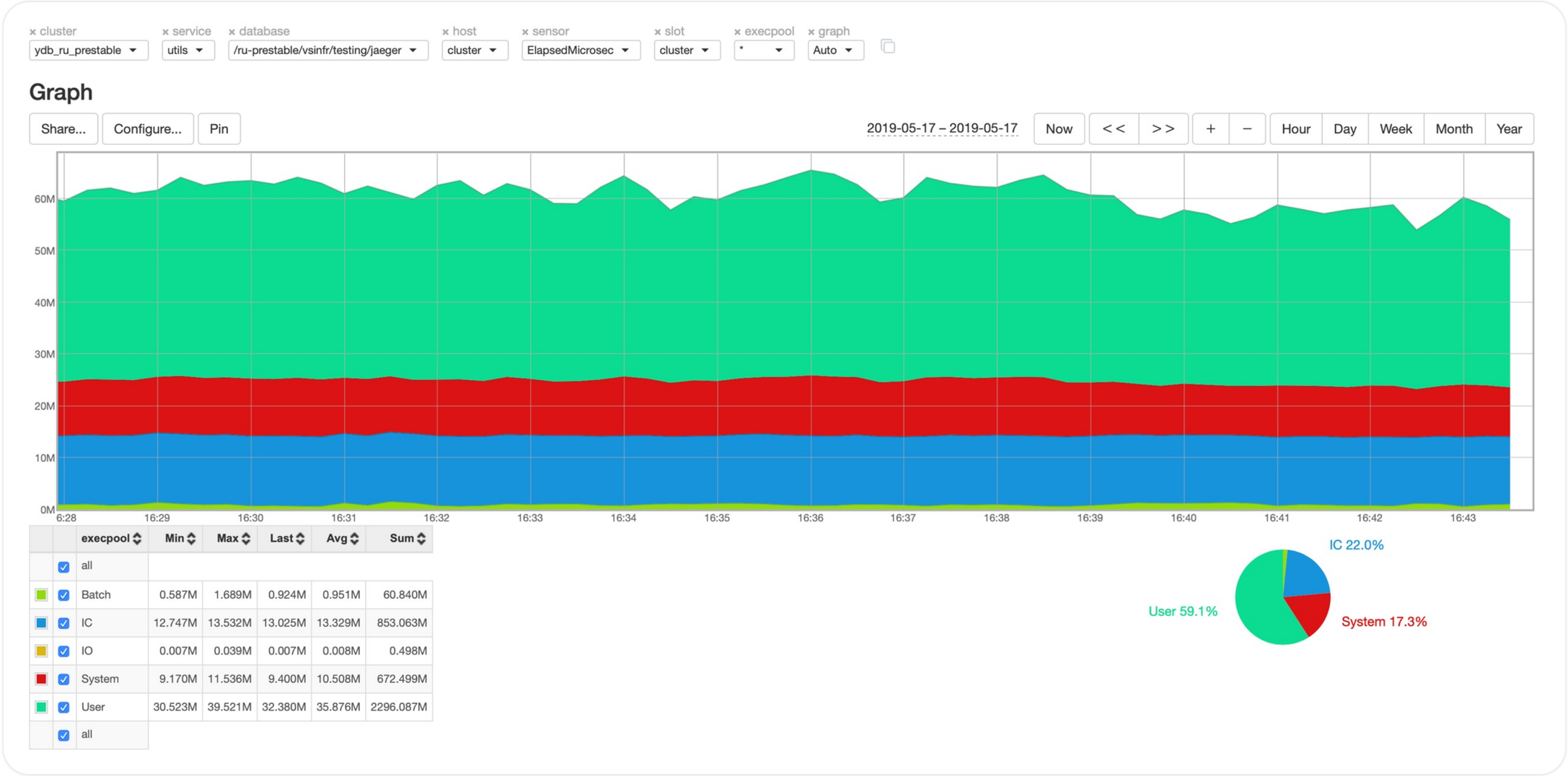
Плагин для Yandex Database

- Spans
- Index table by service
- Index table by key-value tags
- Index table by request duration

Yandex Database — первый блин комом



Yandex Database — первый блин комом



Yandex Database — первый блин комом

- 3 DC × 5 nodes × (4 cores, 16 GiB RAM)
- YQL API
- 350 span/core/s

Получается очень дорого

Ушли думать

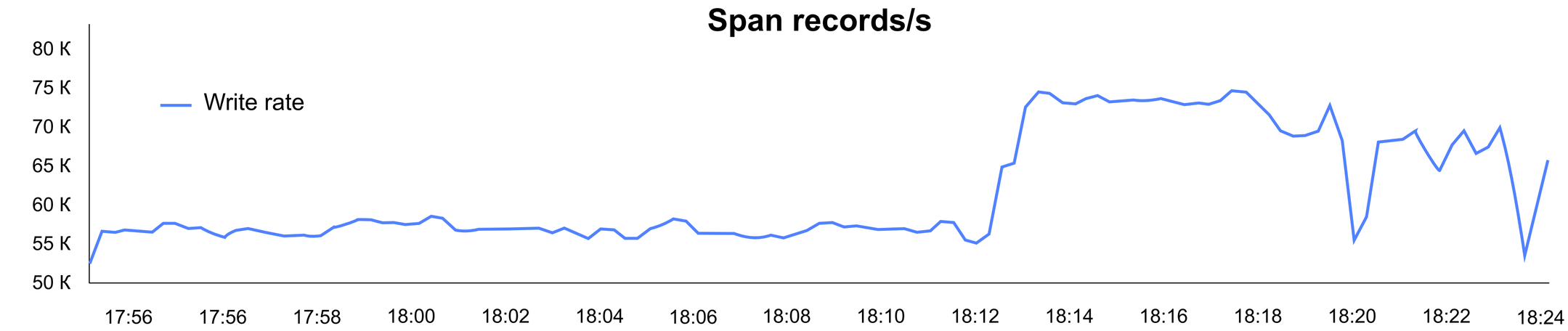
Cassandra™

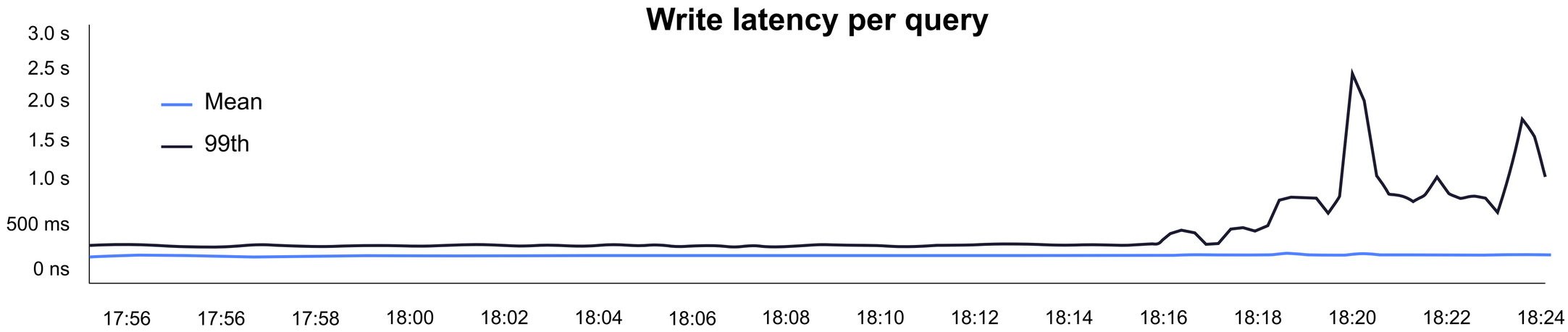
- Кластер из 3 VM × (2 × Xeon E5-2660, 256GB RAM, SSD RAID1)
- Max 312 span/core/s
- При 10−15K spans/s выедаем CPU
- Экономия на репликации и сжатии помогает несильно

MongoDB®

- Replica-set из 3 VM × (48 vCPU, 192GB RAM)
- 1000−1500 span/core/s
- Запись в каждую реплику MongoDB® потребляет 2/3 CPU от потребления мастера
- При плотной записи из мастера почти невозможно читать, а на репликах остаётся не очень много ресурсов

MongoDB®


Span records/s
80 K
75 K
70 K
65 K
60 K
55 K
50 K
Write rate
17:56
17:56
17:58
18:00
18:02
18:04
18:06
18:08
18:10
18:12
18:14
18:16
18:18
18:20
18:22
18:24


Write latency per query
3.0 s
2.5 s
2.0 s
1.5 s
1.0 s
500 ms
0 ns
Mean
99th
17:56
17:56
17:58
18:00
18:02
18:04
18:06
18:08
18:10
18:12
18:14
18:16
18:18
18:20
18:22
18:24

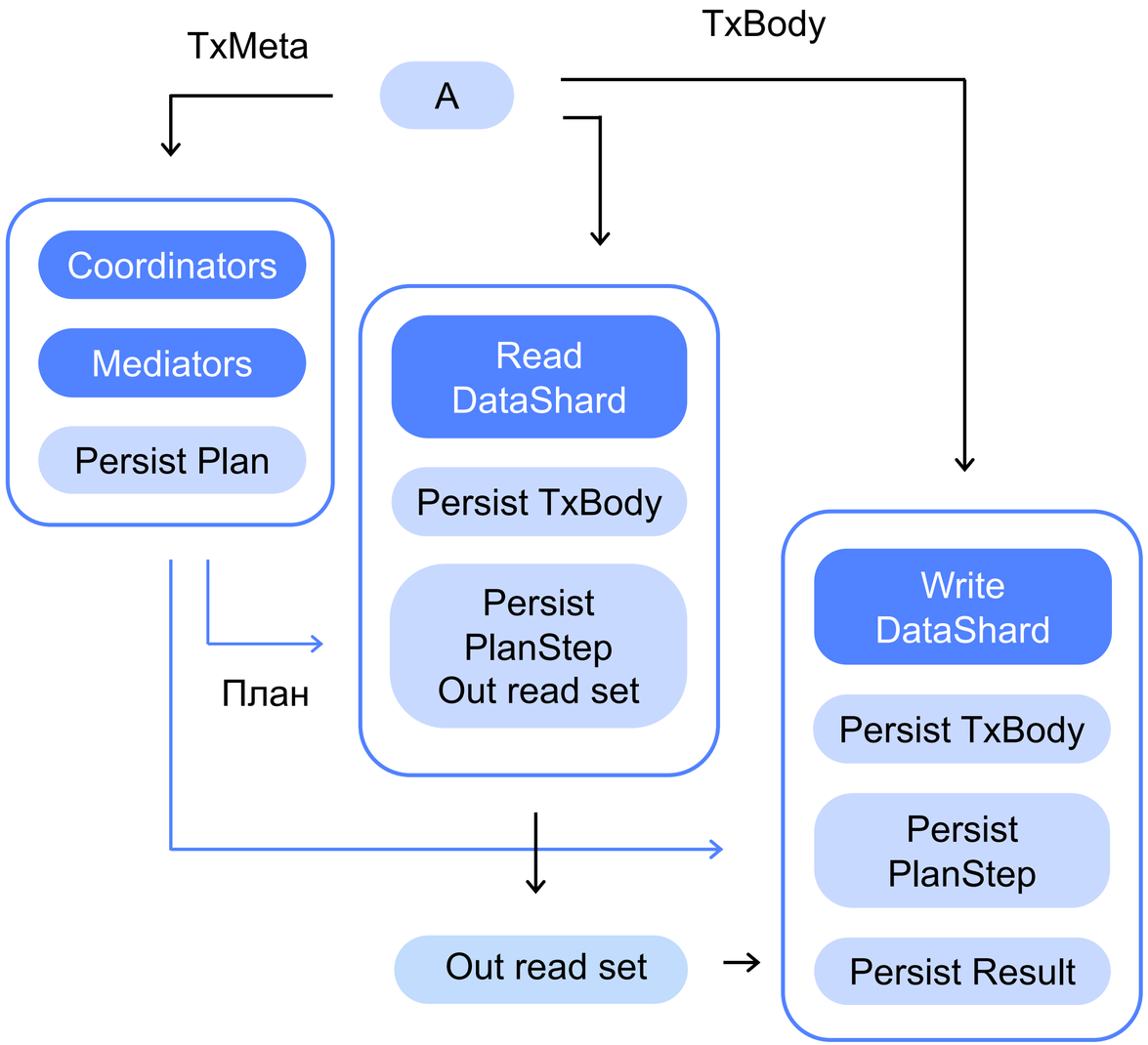
Hold my beer!

…и другие плохие идеи

Yandex Database — Query Processing

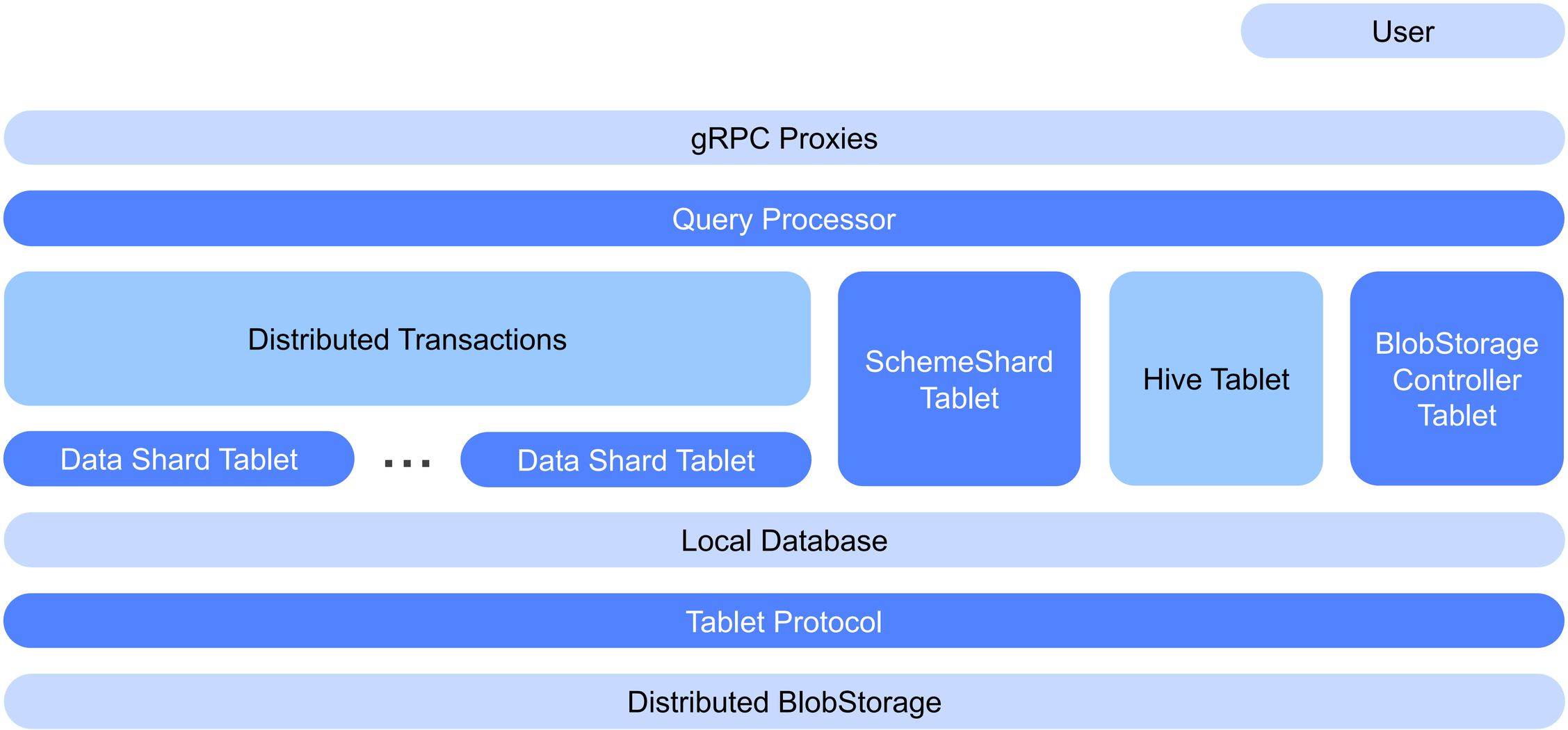
- YQL — диалект SQL
- OLTP-нагрузка
- Запрос выполняется как набор deterministic transactions
- Между распределёнными транзакциями берутся optimistics locks
- Уровни изоляции
  - Serializable
  - Read committed
  - …

Yandex Database — Distributed Transactions

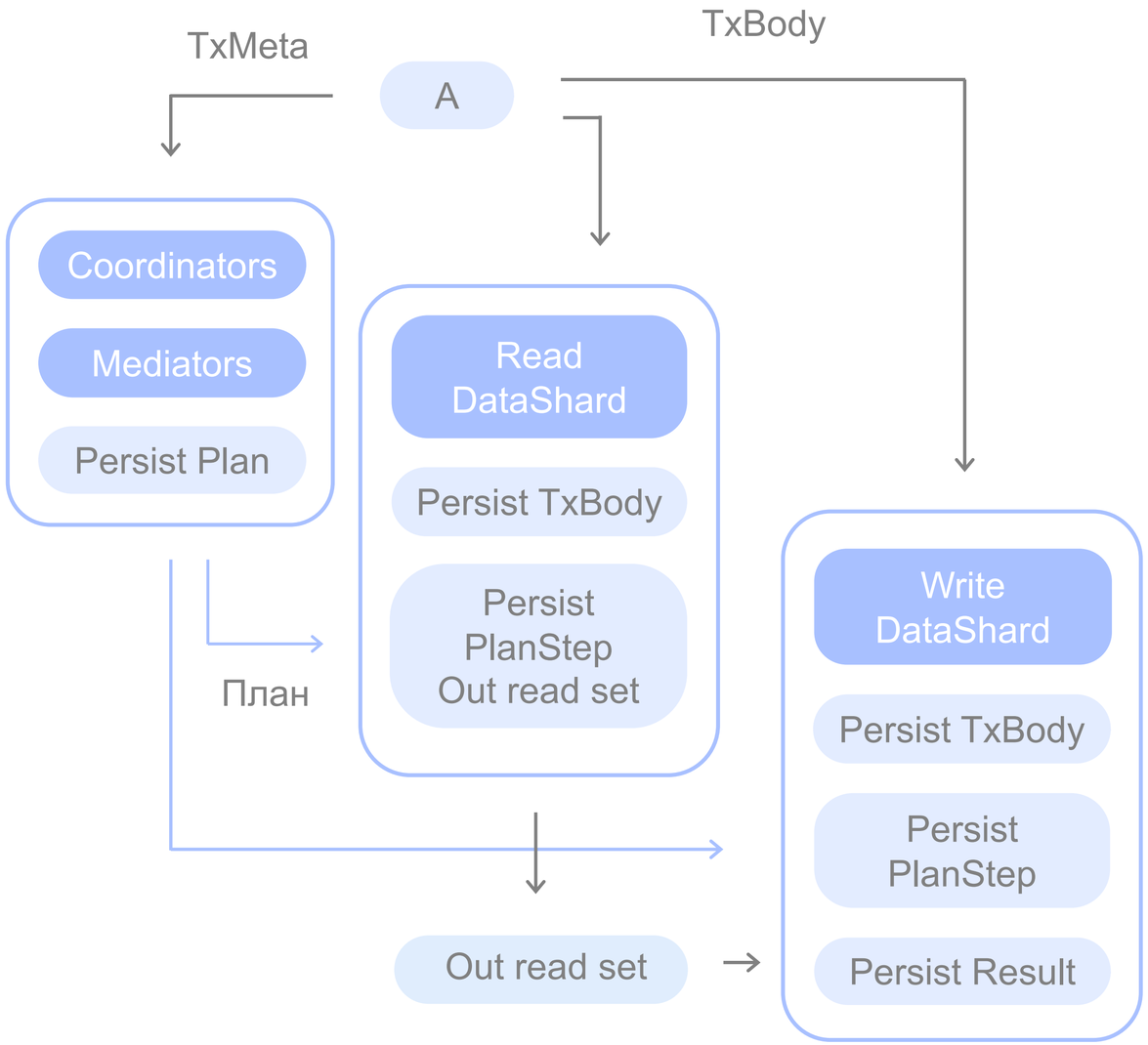


- ACID, serializable, но можно ослаблять
- Распределённые транзакции дороги, но иногда необходимы
- Традиционно распределённые транзакции реализуются через 2PC, в Yandex Database — deterministic transactions

Yandex Database — технологический стек



Distributed Transactions не нужны*

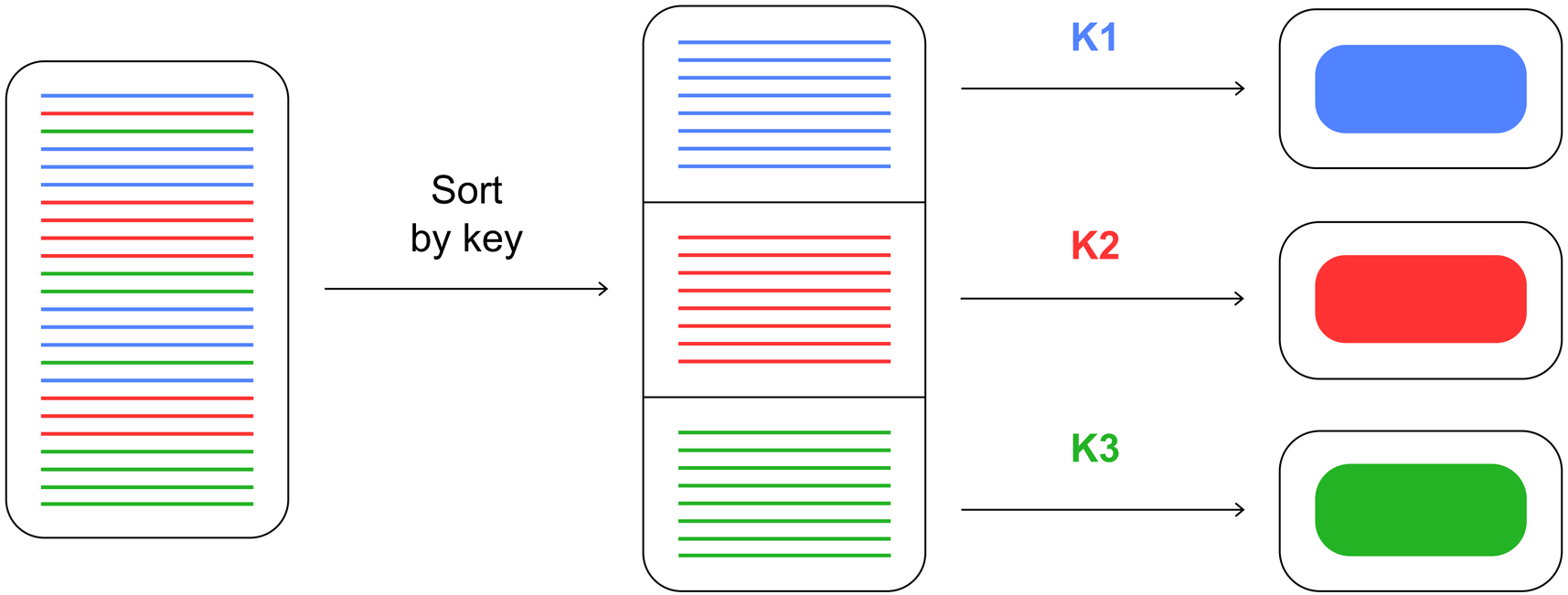


- Пишем много
- Читаем мало
- Читаем не в real-time

Да будет свет

Неизвестный монтёр

Как работает BulkUpsert



Как работает BulkUpsert

- Убран overhead на обработку YQL (parse, prepare, plan, ...)
- Убрали transaction contention (вставка данных происходит по каждой партиции отдельно, портим transaction locks всем остальным)
- Не пишем во вторичные индексы ради ускорения записи

На словах ты — Лев Толстой…

Недоверчивый слушатель

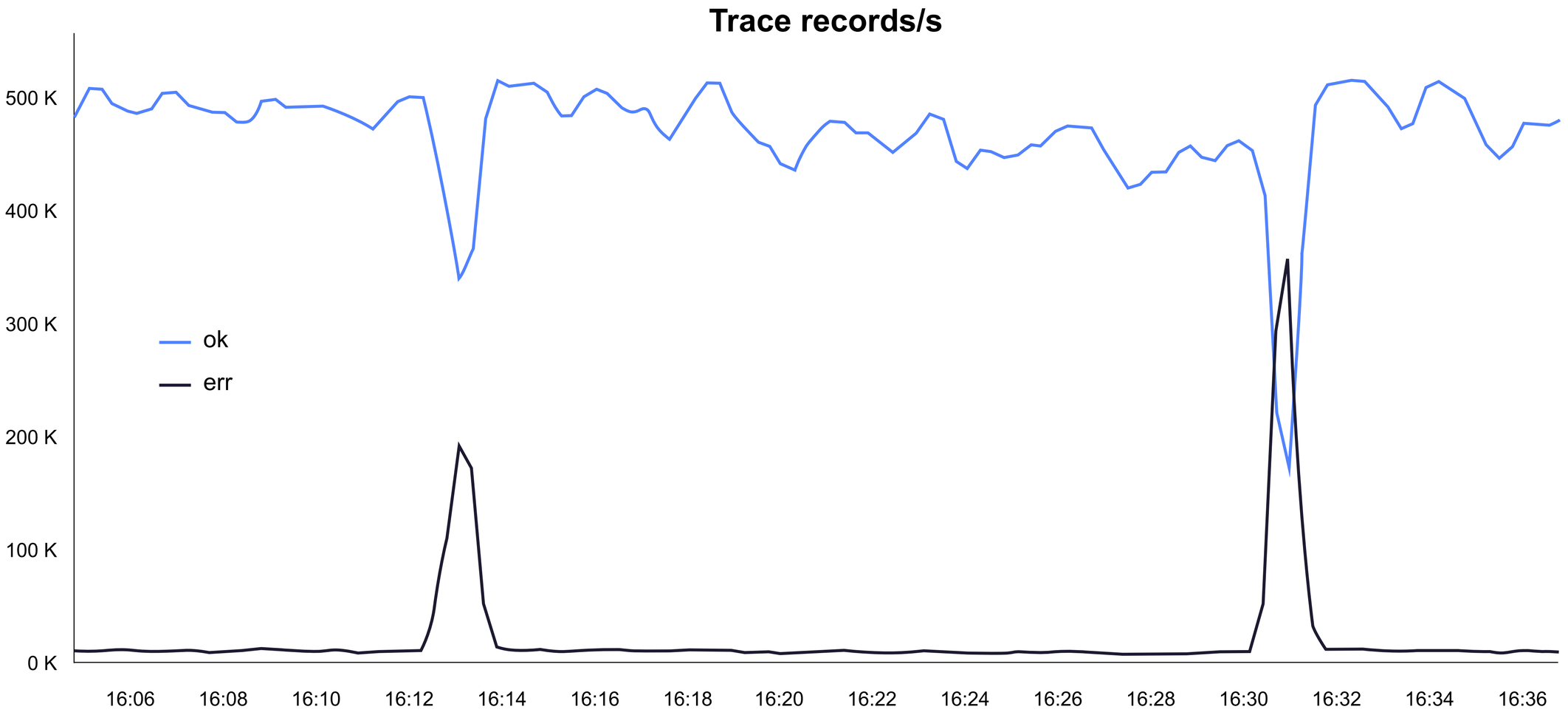
Yandex Database — BulkUpsert alpha

- 1 DC × 8 nodes × (32 cores, 32 GiB RAM)

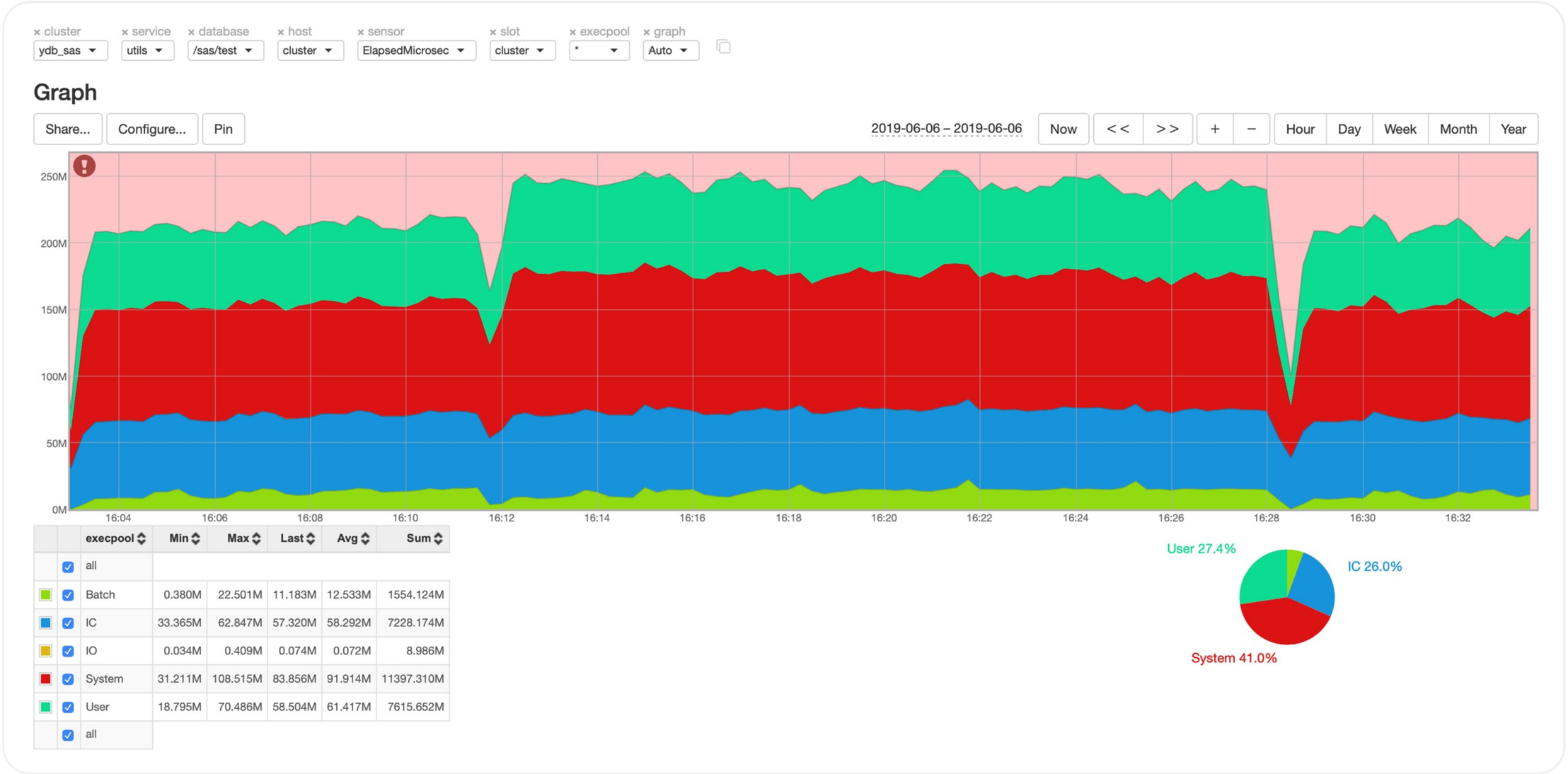
С новым API получилось 400−500K span/s на 210 ядер

2150 span / core — уже намного интересней

Yandex Database — BulkUpsert



Yandex Database — BulkUpsert



		execpool	Min	Max	Last	Avg	Sum
	☑	all					
■	☑	Batch	0.380M	22.501M	11.183M	12.533M	1554.124M
■	☑	IC	33.365M	62.847M	57.320M	58.292M	7228.174M
■	☑	IO	0.034M	0.409M	0.074M	0.072M	8.986M
■	☑	System	31.211M	108.515M	83.856M	91.914M	11397.310M
■	☑	User	18.795M	70.486M	58.504M	61.417M	7615.652M
	☑	all					

И на деле — Лев Толстой!

Гордый разработчик

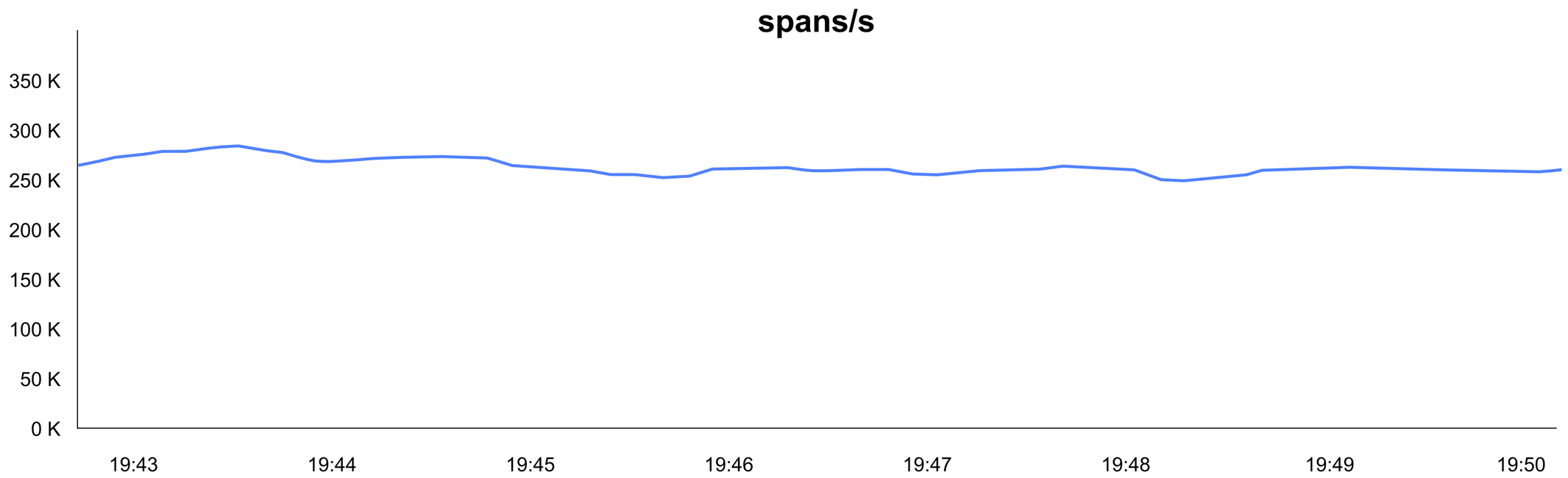
Yandex Database — BulkUpsert v1.0

- 3 DC, 19 VM × (4 cores, 50 GiB RAM)
- 3600 spans/core/s

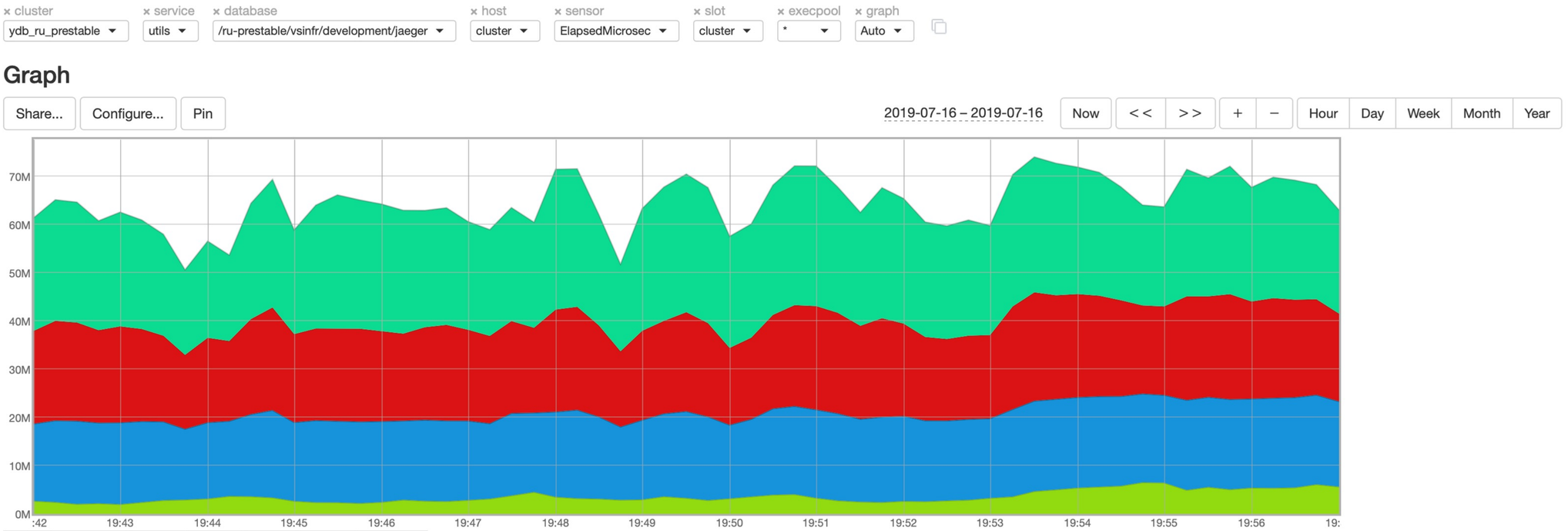
Получили 275K спанов в секунду на 76 ядрах

Совсем хорошо

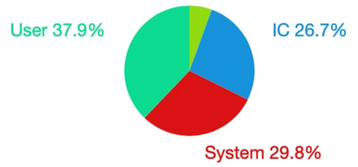
Yandex Database — BulkUpsert на версии 19-4



Yandex Database — BulkUpsert на версии 19-4



		execpool	Min	Max	Last	Avg	Sum
	☑	all					
	☑	Batch	2.000M	6.537M	5.643M	3.612M	220.305M
	☑	IC	14.691M	18.796M	17.700M	17.295M	1055.002M
	☑	IO	0.009M	0.024M	0.009M	0.010M	0.599M
	☑	System	15.342M	22.491M	18.182M	19.295M	1176.994M
	☑	User	17.659M	29.117M	21.517M	24.512M	1495.236M
	☑	all					



Результаты

- Cassandra™ — 312 span/core
- MongoDB® — 1500 span/core
- Yandex Database — 3600 span/core

Профит

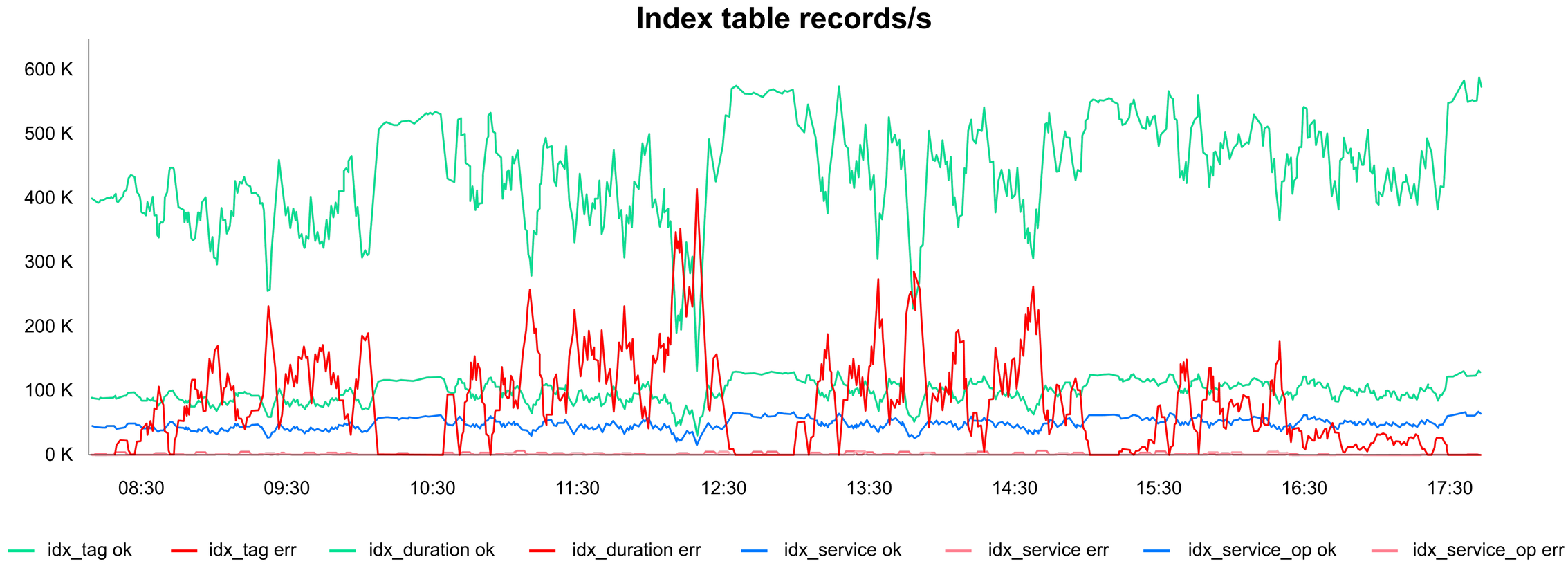
- Managed
- Cross-dc
- Проверили, работает
- В сравнении с Cassandra™ и MongoDB® получается дешевле запись/ядро CPU

А теперь начинается реальная жизнь

- Разбили таблицы на партиции по времени (10 партиций в день)
- Пустили прод-нагрузку

Проблемы на production

- Некоторые шарды перегружены
- Некоторые недонагружены
- Проливаем данные



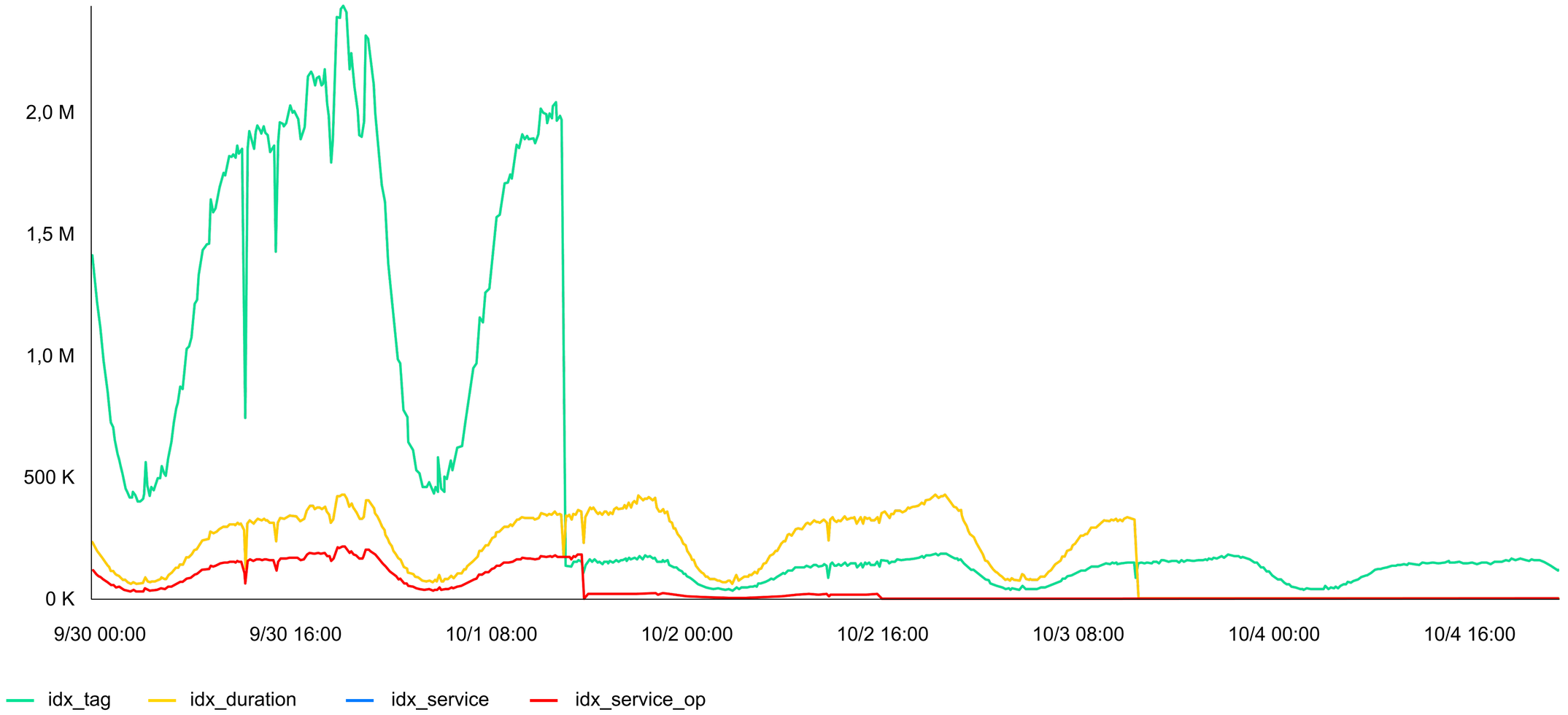
Эксперименты

- В пике пишем 4M записей/с в базу
- Очень круто
- Но жалко базу

Оптимизация

- Снижаем поток записи
- Оптимизируем индексные таблицы
- Пишем в индексную таблицу не на каждый спан, а на группу спанов за 5 секунд или 100 записей внутри коллектора
- В индекс по продолжительности пишем приблизительное время (40−50 ms, 100−200 ms, etc.)

Index table records/s



Оптимизация

Рефакторим, чтобы прекратить насилие над базой

- Получаем 30% экономии места
- Примерно на 30% сократили CPU Utilization
- И в 10 раз сокращаем запись
- 400 000 вместо 4 000 000 rows/s

Open source

- Jaeger storage плагин для YDB в open source
- Постоянные улучшения

clck.ru/Usgku

Используем два года — полёт нормальный

- По косвенным признакам одна из самых нагруженных (~0,5M span/s) инсталляций Jaeger в мире
- В Yandex.Cloud есть своя инсталляция с текущей нагрузкой 60−70K span/s

Yandex Database

Нужна БД, но не простая, а:

- геораспределённая
- отказоустойчивая
- со строгой консистентностью транзакций
- горизонтально масштабируемая
- с высокой пропускной способностью
- с малым временем отклика

Yandex Database в Яндексе

Яндекс Турбо-страницы

Яндекс ID

auto.ru

Яндекс Репетитор

Яндекс Алиса

Яндекс Новости

Яндекс Директ

Яндекс Дзен

Яндекс Погода

Яндекс Толока

Яндекс Go Такси

Яндекс Услуги

КиноПоиск

Yandex Database в Yandex.Cloud

Сервисы Yandex.Cloud используют Yandex Database для хранения данных

- Compute Cloud
- Load Balancer
- Billing
- Managed Service for Kubernetes®

….

- YDBaaS (dedicated- и serverless-режимы)

Yandex Database Serverless

- Pay as you go
- Платим за storage и за запросы (request units)
- Не нужно заниматься масштабированием самостоятельно
- Document API — слой совместимости с AWS DynamoDB

Yandex Database Serverless + Jaeger

- Тестировали с помощью **tracegen**
- **~0,6 request units/span** при **~8K span/s**
- Эффективно использовать при нагрузке не более **668 span/s**
- Дальше dedicated-кластер из 1 машины получается дешевле при пересчёте на **₽/span**

Рецепты в репозитории плагина

Jaeger + Yandex Database = ♡

- Yandex Database отлично подходит для задач с горизонтальным масштабированием по данным
- Нашли точки роста и сделали новый API для задач с преобладающей записью
- Сделали plugin для Jaeger и выложили в open source
- Можно попробовать в Yandex.Cloud

Yandex Cloud

Спасибо!

Александр Салтыков

Разработчик Auto.ru

alexander-s@yandex-team.ru

Александр Щербаков

Разработчик Yandex.Cloud

a-shch@yandex-team.ru

Ссылки

Jaeger plugin

clck.ru/Usgku

Yandex Database

clck.ru/V8r67