

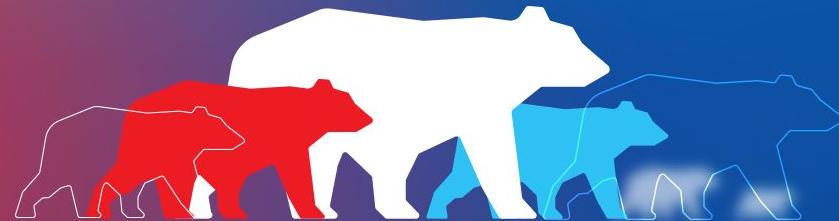
ClickHouse и тысяча графиков

Антон Алексеев, 2ГИС



HighLoad⁺⁺
Siberia 2019

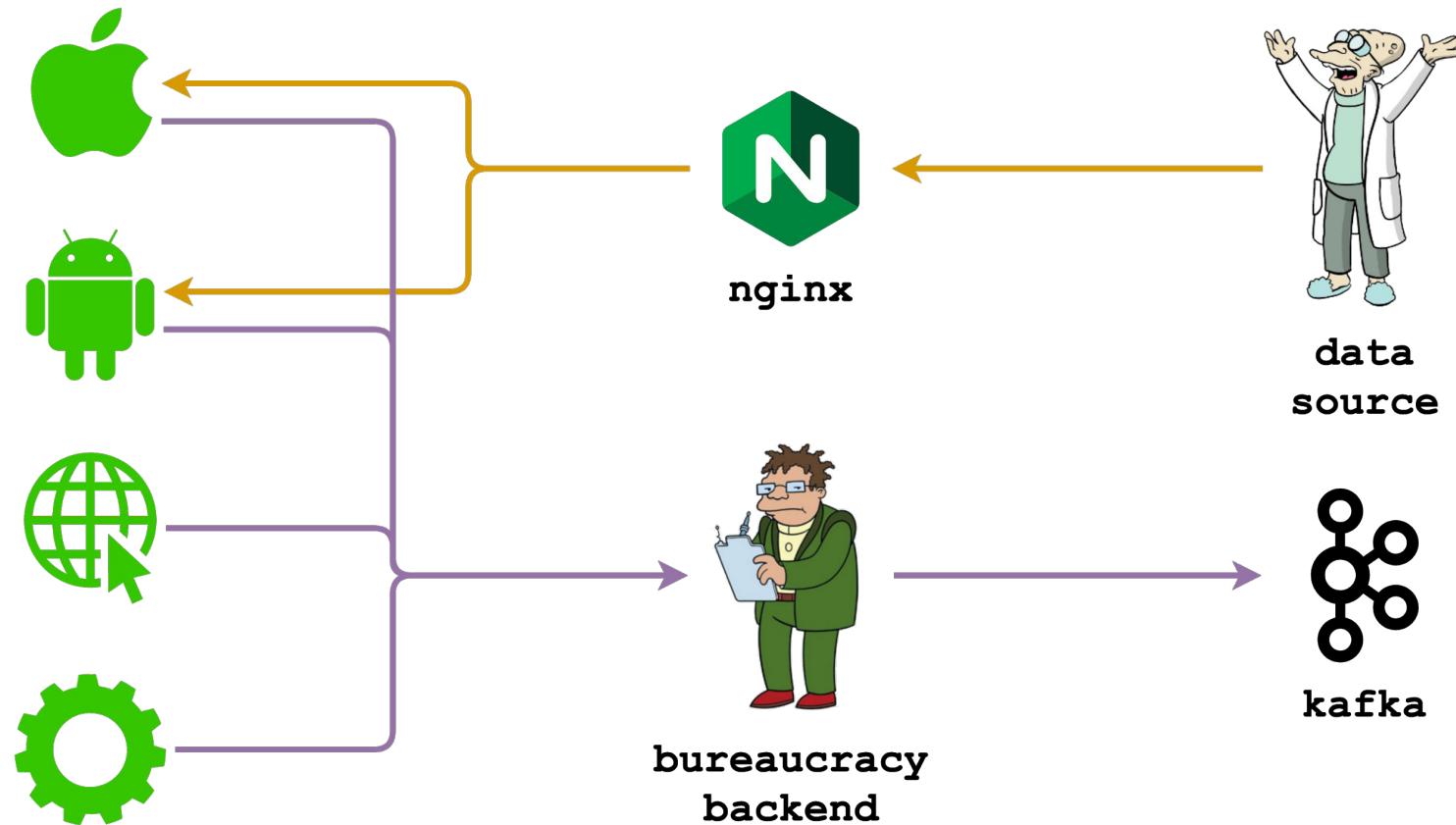
Профессиональная конференция
для разработчиков высоконагруженных
систем



Мы и ClickHouse

- 2 года
- 2 сервера
- 128 ГБ RAM
- 16 CPU
- 4 ТБ SSD

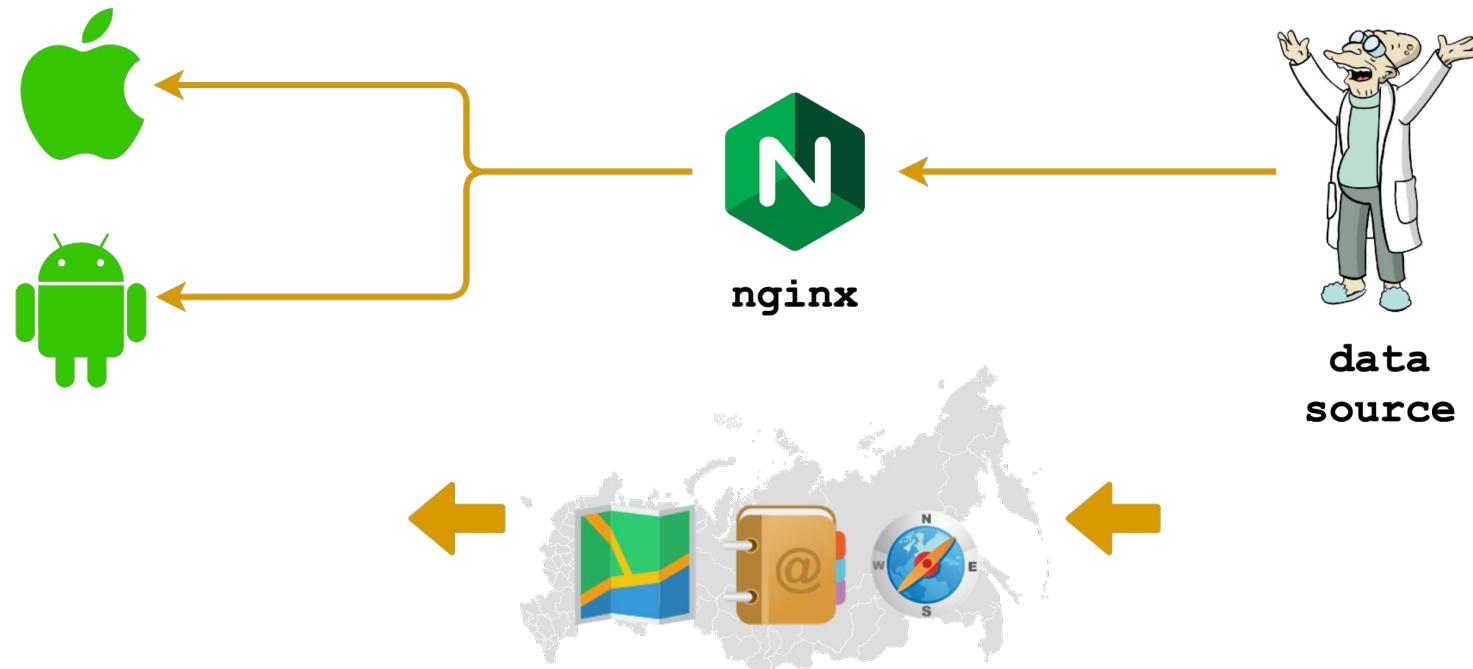
Наши сервисы





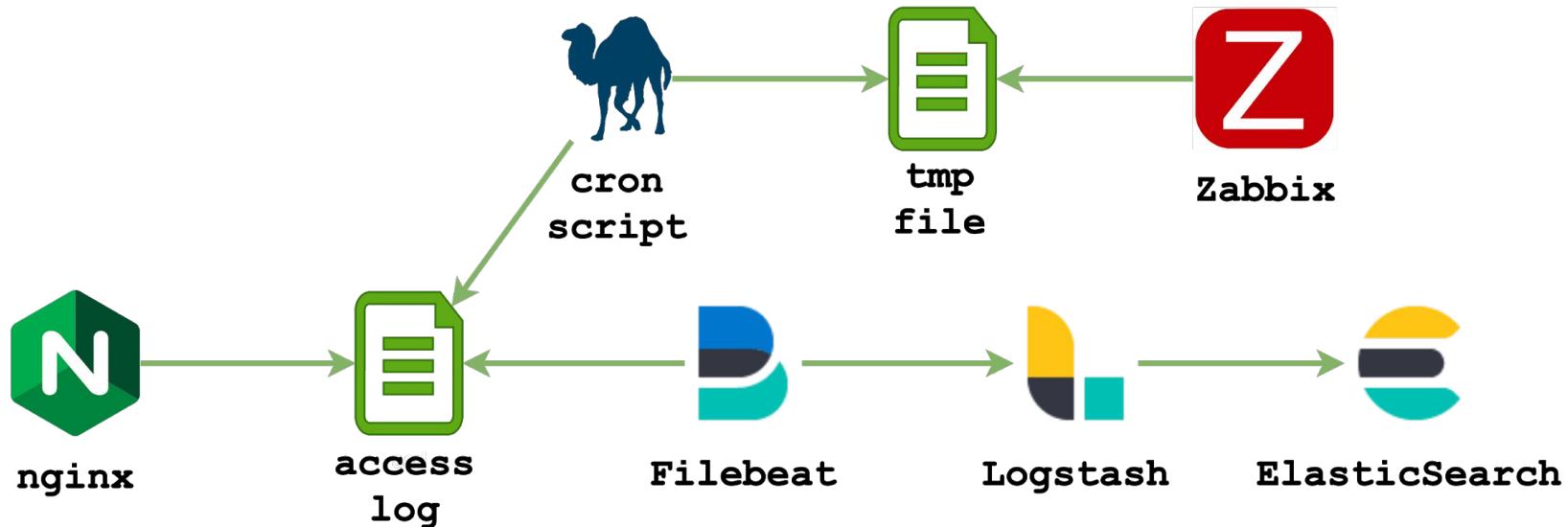
CDN

2ГИС CDN



Как было раньше

Как было раньше



Хотелки

- Разные срезы запросов и трафика:
 - по версиям приложений
 - по регионам

Хотелки

- Разные срезы запросов и трафика:
 - по версиям приложений
 - по регионам
- Процентили:
 - время отдачи
 - скорость скачивания

Хотелки

- Разные срезы запросов и трафика:
 - по версиям приложений
 - по регионам
- Процентили:
 - время отдачи
 - скорость скачивания
- Достаточная глубина хранения сырых данных

Способ решения №1: Закидать деньгами

- Куча скриптов для подсчёта агрегатов

Способ решения №1: Закидать деньгами

- Куча скриптов для подсчёта агрегатов
- ElasticSearch на несколько терабайт

Способ решения №1: Закидать деньгами

- Куча скриптов для подсчёта агрегатов
- ElasticSearch на несколько терабайт

Недостатки:

- Деньги надо где-то взять

Способ решения №1: Закидать деньгами

- Куча скриптов для подсчёта агрегатов
- ElasticSearch на несколько терабайт

Недостатки:

- Деньги надо где-то взять
- Минимум 2 различные системы

Способ решения №1: Закидать деньгами

- Куча скриптов для подсчёта агрегатов
- ElasticSearch на несколько терабайт

Недостатки:

- Деньги надо где-то взять
- Минимум 2 различные системы
- Новый агрегат требует времени для набора истории

Способ решения №2: Ищем альтернативу

- Набор записей с одинаковой плоской структурой

Способ решения №2: Ищем альтернативу

- Набор записей с одинаковой плоской структурой
- Числа или повторяющиеся строки

Способ решения №2: Ищем альтернативу

- Набор записей с одинаковой плоской структурой
- Числа или повторяющиеся строки
- Для запроса нужна пара полей

Способ решения №2: Ищем альтернативу

- Набор записей с одинаковой плоской структурой
- Числа или повторяющиеся строки
- Для запроса нужна пара полей

Всем пунктам удовлетворяет поколоночная БД

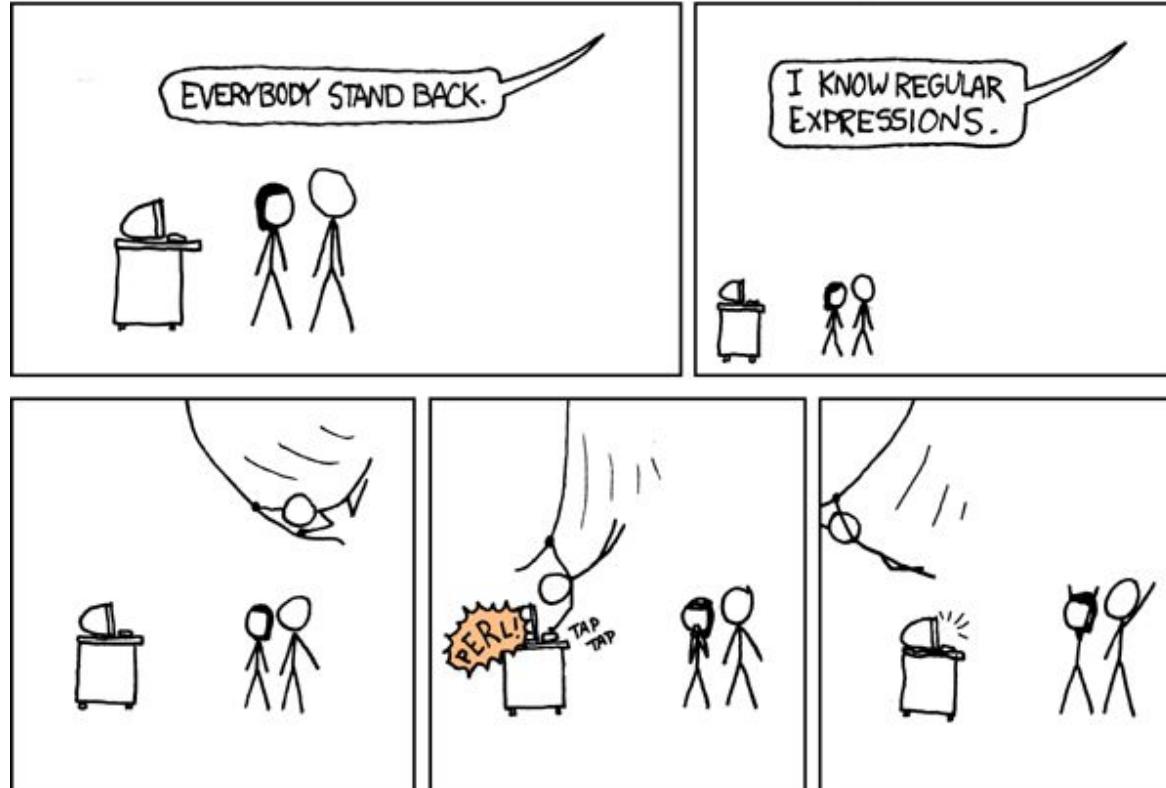
Пробуем ClickHouse: создаём таблицу

```
CREATE TABLE cdn_logs (
    date Date DEFAULT toDate(now()) ,
    datetime DateTime DEFAULT now() ,
    request_type Enum8('META' = 1, 'DATA' = 2) ,
    http_status UInt16,
    cache_status Enum8('EXPIRED' = 1, 'HIT' = 2, ...),
    bytes_sent UInt32,
    ...
)
ENGINE =
    MergeTree(date, (date, request_type, http_status), ...)
```

Пробуем ClickHouse: первичный ключ

```
CREATE TABLE cdn_logs (
    date Date DEFAULT toDate(now()),
    datetime DateTime DEFAULT now(),
    request_type Enum8('META' = 1, 'DATA' = 2),
    http_status UInt16,
    cache_status Enum8('EXPIRED' = 1, 'HIT' = 2, ...),
    bytes_sent UInt32,
    ...
)
ENGINE =
MergeTree(date, (date, request_type, http_status), ...)
```

Пробуем ClickHouse: наливаем данные



Супер-простая заливка данных

```
curl -X POST -H 'Content-Type: application/json' \
'http://localhost:8123?input_format_skip_unknown_fields=1' \
-d @- <<EOF

INSERT INTO cdn_logs FORMAT JSONEachRow
{
  "request_type": "DATA", "source": "/var/log/nginx/access.log"
}
{
  "request_type": "META", "source": "/var/log/nginx/access.log"
}

EOF
```

ClickHouse и Grafana: плагин от Vertamedia

- Работает из коробки с самой первой версии (0.0.1)
- Поддерживает переменные, ad-hoc-фильтры и многое другое
- Приносит счастье нашим логам

Не является рекламой!

ClickHouse и Grafana: какие данные нужны

t	requests	users
1 500 000 000 000	532	23
1 500 000 060 000	897	62
1 500 000 120 000	704	86

ClickHouse и Grafana: какие данные нужны

t	values
1 500 000 000 000	[('r', 532), ('u', 23)]
1 500 000 060 000	[('r', 897), ('u', 62)]
1 500 000 120 000	[('r', 704), ('u', 86)]

ClickHouse и Grafana: пишем SELECT

```
SELECT
    intDiv(toUInt32(datetime), $interval) * $interval * 1000 AS t,
    count() as requests,
    uniq(user_id) as users
...
GROUP BY t
ORDER BY t
```

ClickHouse и Grafana: пишем SELECT

```
SELECT
    intDiv(toUInt32(datetime), $interval) * $interval) * 1000 AS t,
    count() as requests,
    uniq(user_id) as users
...
GROUP BY t
ORDER BY t
```

ClickHouse и Grafana: пишем SELECT

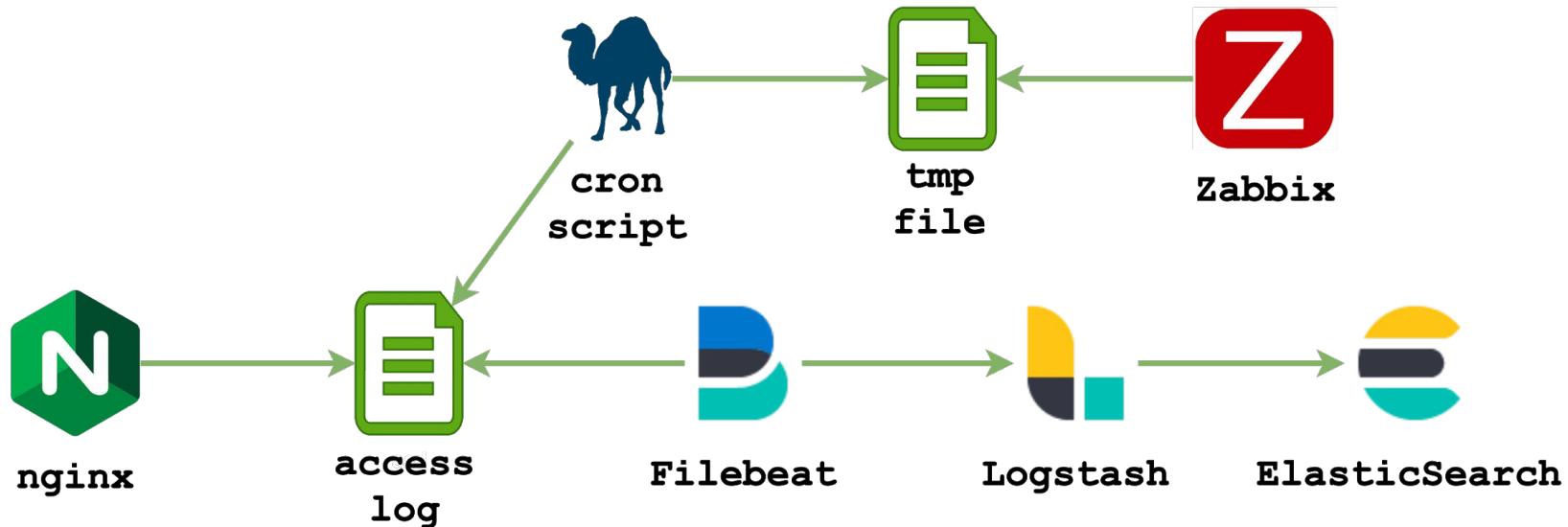
```
SELECT
    intDiv(toUInt32(datetime), $interval) * $interval) * 1000 AS t,
    [
        ('r', count()),
        ('u', uniq(user_id))
    ] as values
...
GROUP BY t
ORDER BY t
```

ClickHouse и Grafana: полезные массивы

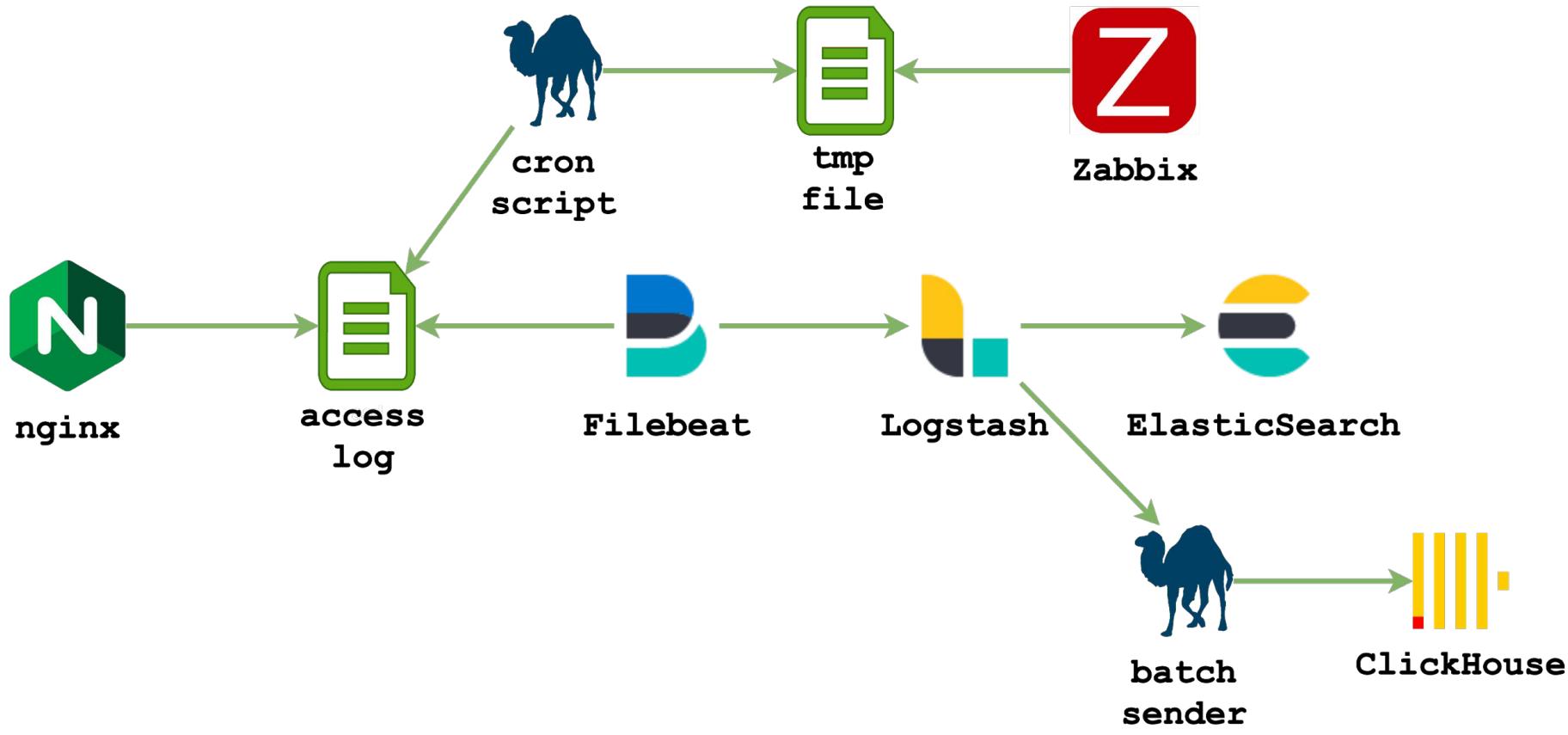
```
SELECT
    intDiv(toUInt32(datetime), $interval) * $interval) * 1000 AS t,
    arrayMap(
        x, y -> (x, y),
        ['75th', '50th', '25th'],
        quantiles(0.75, 0.5, 0.25)
    ) as res
...
GROUP BY t
ORDER BY t
```

Оперативные графики

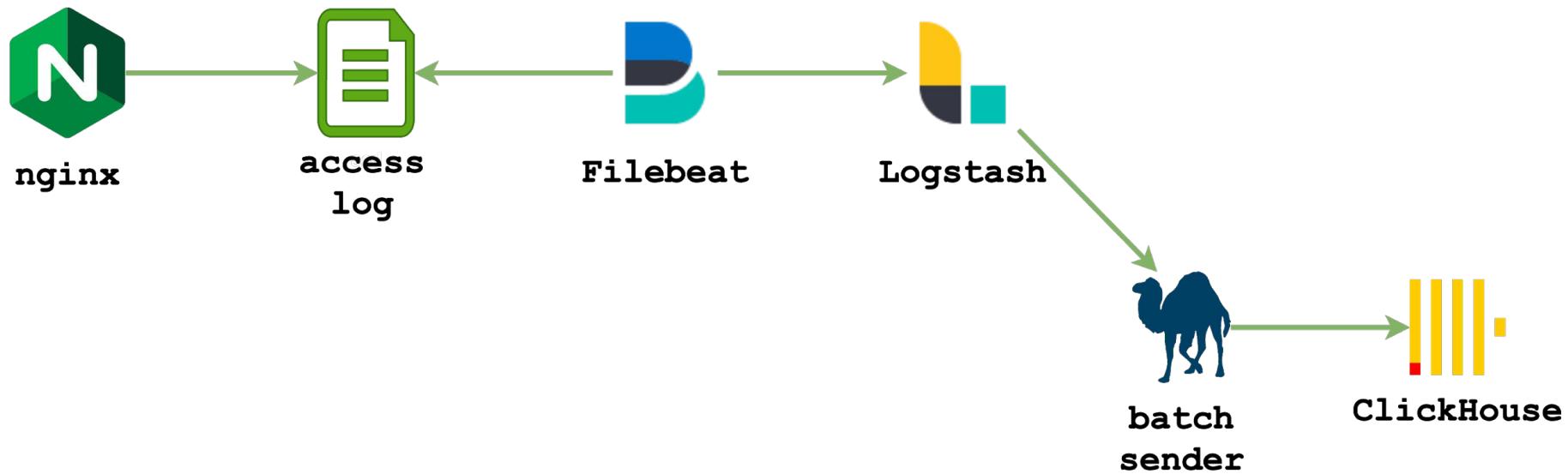
Как было раньше



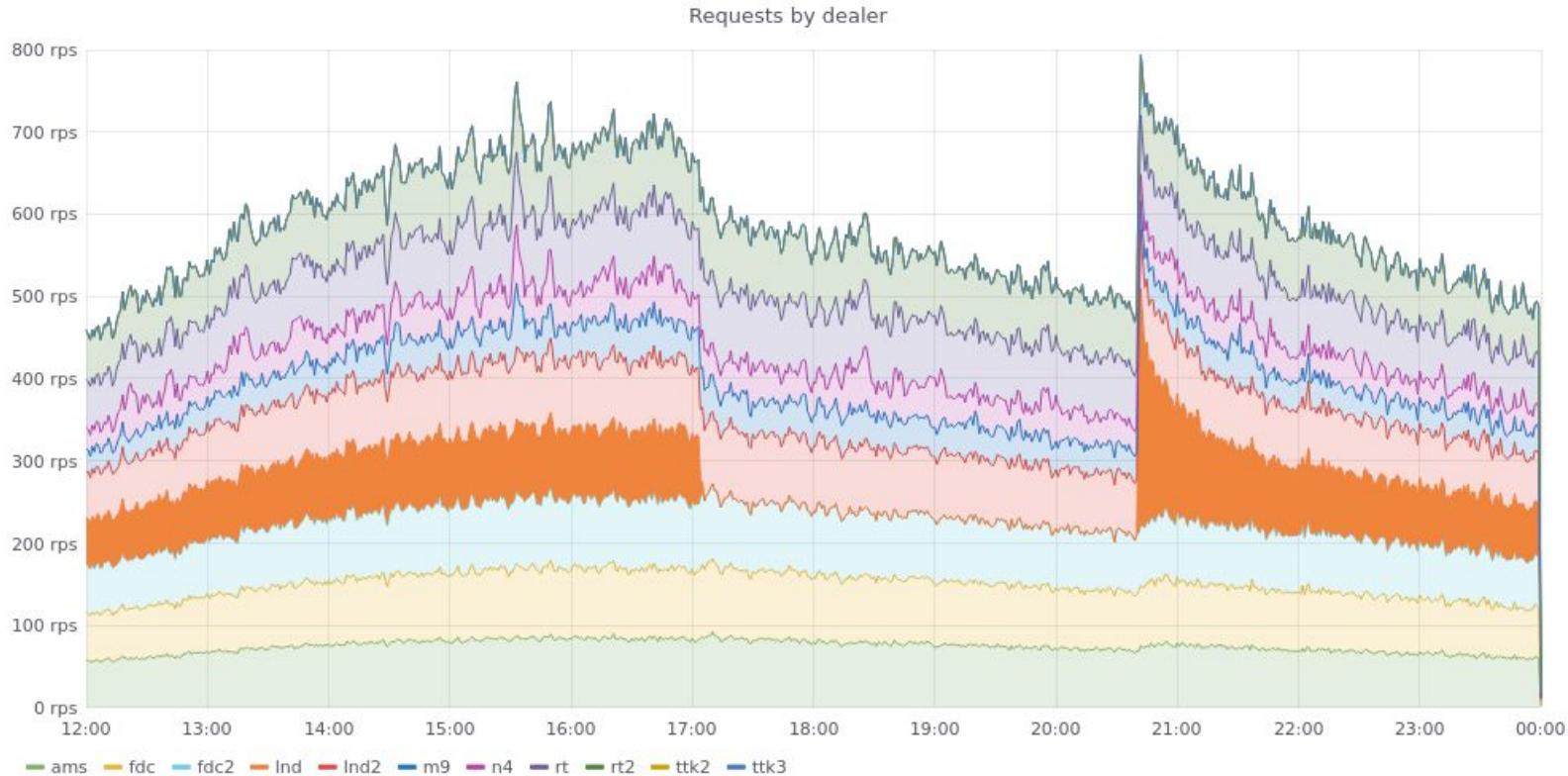
Как стало



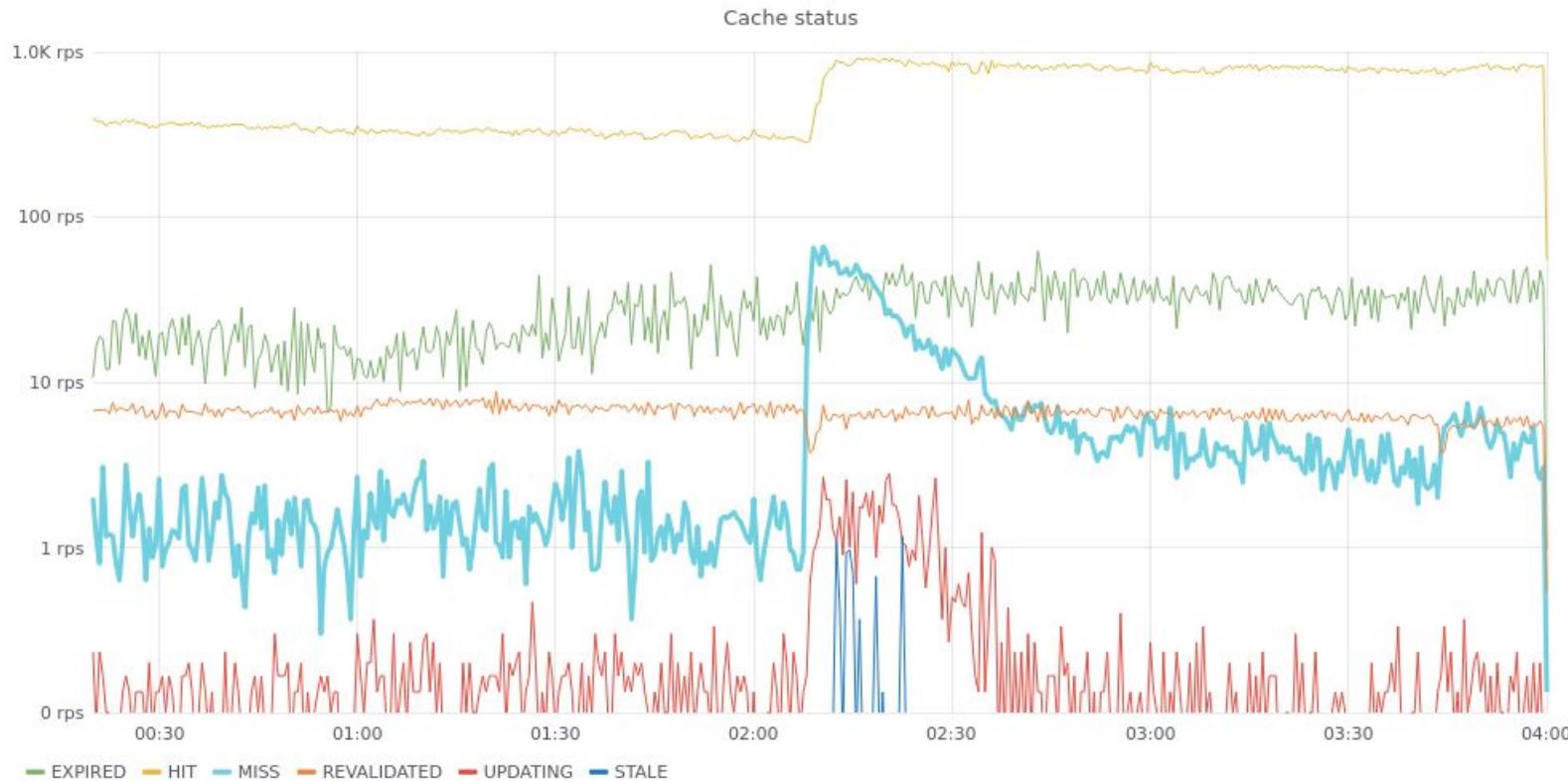
Как стало (спойлер)



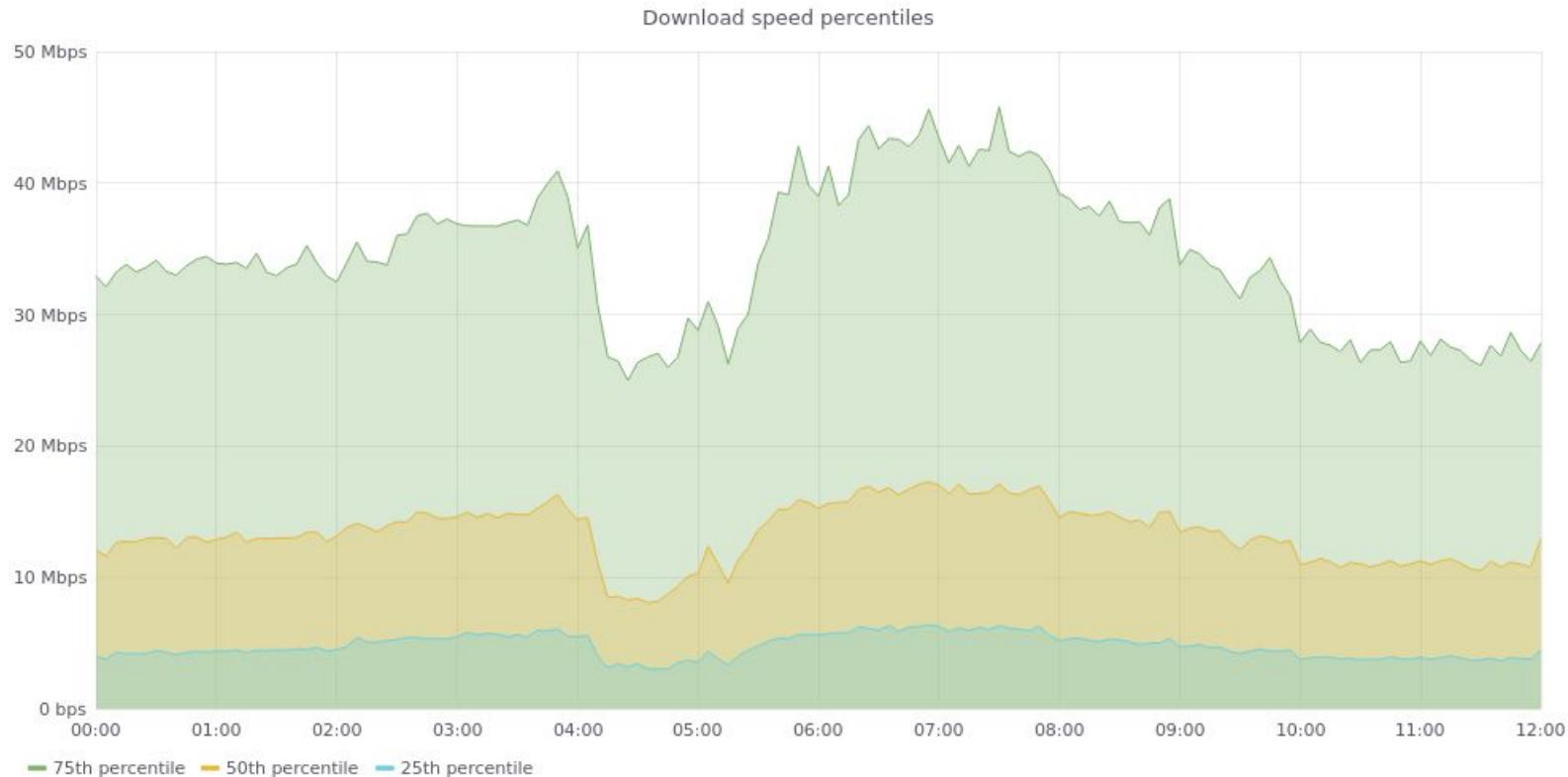
ClickHouse + Grafana: запросы по серверу



ClickHouse + Grafana: статусы кэша



ClickHouse + Grafana: скорость скачивания



Новые поля — легко!

Nginx:

```
log_format cdn_host_json escape=json '{
    ...
    "custom_header": "$http_custom_header",
    ...
}
```

ClickHouse:

```
ALTER TABLE cdn_logs ADD COLUMN custom_header String;
```

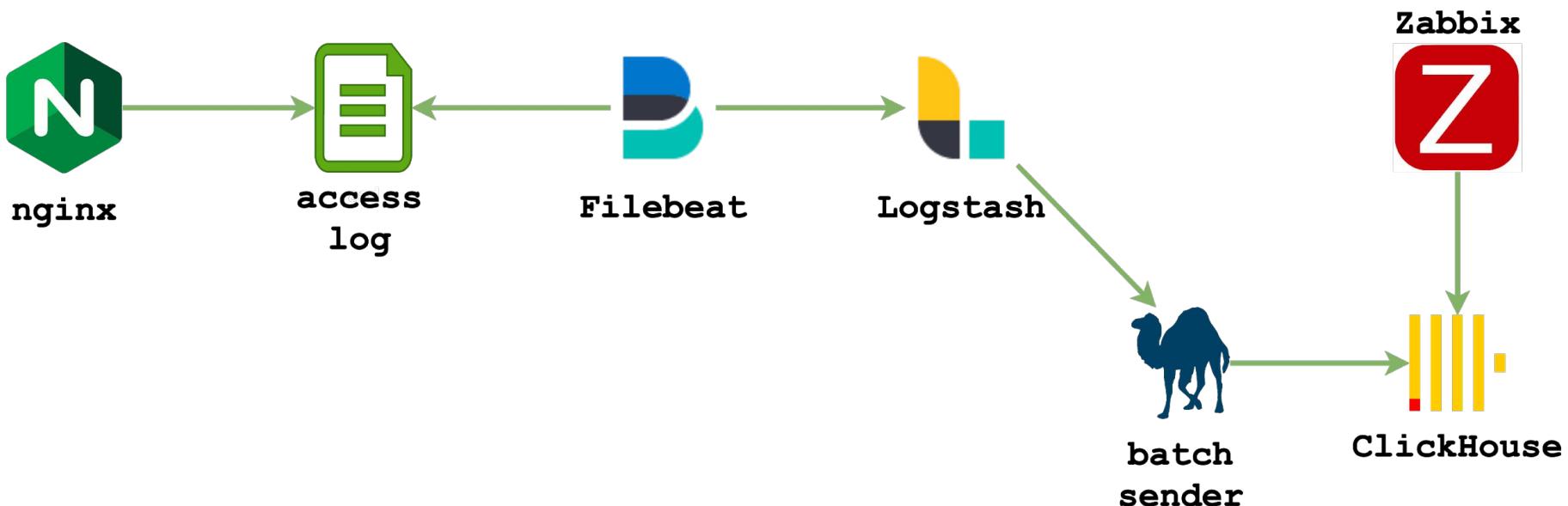
А есть есть?

Алфавитный указатель переменных

[\\$ancient_browser](#)
[\\$arg_](#)
[\\$args](#)
[\\$binary_remote_addr \(ngx_http_core_module\)](#)
[\\$binary_remote_addr \(ngx_stream_core_module\)](#)
[\\$body_bytes_sent](#)
[\\$bytes_received](#)
[\\$bytes_sent \(ngx_http_core_module\)](#)
[\\$bytes_sent \(ngx_http_log_module\)](#)
[\\$bytes_sent \(ngx_stream_core_module\)](#)
[\\$connection \(ngx_http_core_module\)](#)
[\\$connection \(ngx_http_log_module\)](#)
[\\$connection \(ngx_stream_core_module\)](#)
[\\$connection_requests \(ngx_http_core_module\)](#)
[\\$connection_requests \(ngx_http_log_module\)](#)
[\\$connections_active](#)
[\\$connections_reading](#)
[\\$connections_waiting](#)
[\\$connections_writing](#)
[\\$content_length](#)

© 2018 Yury Slobodchikov

Алертинг через Zabbix



Те же графики, но за год?

Больше интервал — дольше строится график

Неделя:

1009 rows in set. Elapsed: **7.218 sec.** Processed 373.72 million rows, 11.53 GB (51.77 million rows/s., 1.60 GB/s.)

Месяц:

721 rows in set. Elapsed: **24.863 sec.** Processed 1.37 billion rows, 42.31 GB (55.17 million rows/s., 1.70 GB/s.)

Полгода:

725 rows in set. Elapsed: **161.333 sec.** Processed 8.35 billion rows, 257.27 GB (51.74 million rows/s., 1.59 GB/s.)

Сэмплирование в ClickHouse

date	http_status
2019-05-28	200
2019-05-28	200
2019-05-28	200
2019-05-28	200
2019-05-28	304
2019-05-28	304
2019-05-29	200
2019-05-29	200
2019-05-29	304
2019-05-29	304

Сэмплирование в ClickHouse

date	http_status
2019-05-28	200
2019-05-28	200
2019-05-28	200
2019-05-28	200
2019-05-28	304
2019-05-28	304
2019-05-29	200
2019-05-29	200
2019-05-29	304
2019-05-29	304

SAMPLE 0.5

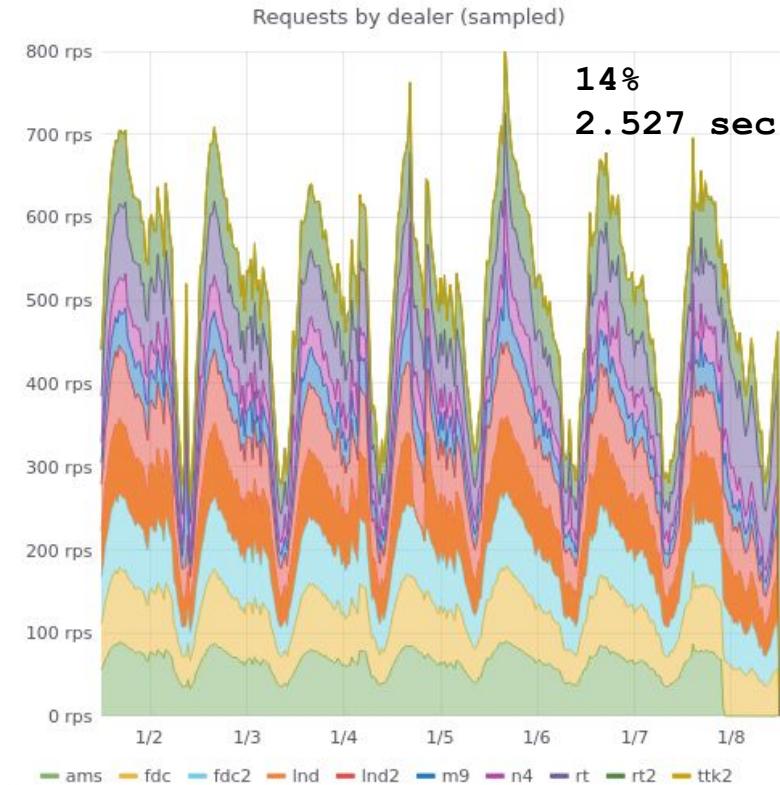
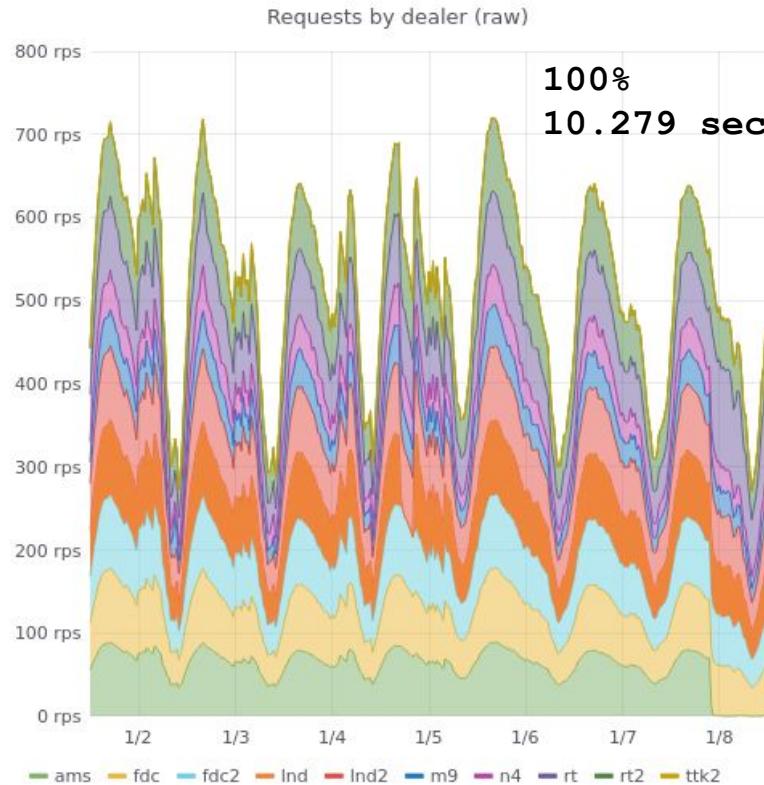


date	http_status
2019-05-28	200
2019-05-28	200
2019-05-28	200
2019-05-28	200
2019-05-28	304
2019-05-28	304
2019-05-29	200
2019-05-29	200
2019-05-29	304
2019-05-29	304

Сэмплирование в ClickHouse: _sample_factor

```
SELECT  
    count() * _sample_factor  
FROM cdn_logs  
SAMPLE $sample_lines
```

Сэмплированные и точные графики



Больше интервал — а нам всё равно!

Неделя:

1011 rows in set. Elapsed: **3.711 sec.** Processed 101.17 million rows, 3.93 GB (27.26 million rows/s., 1.06 GB/s.)

Месяц:

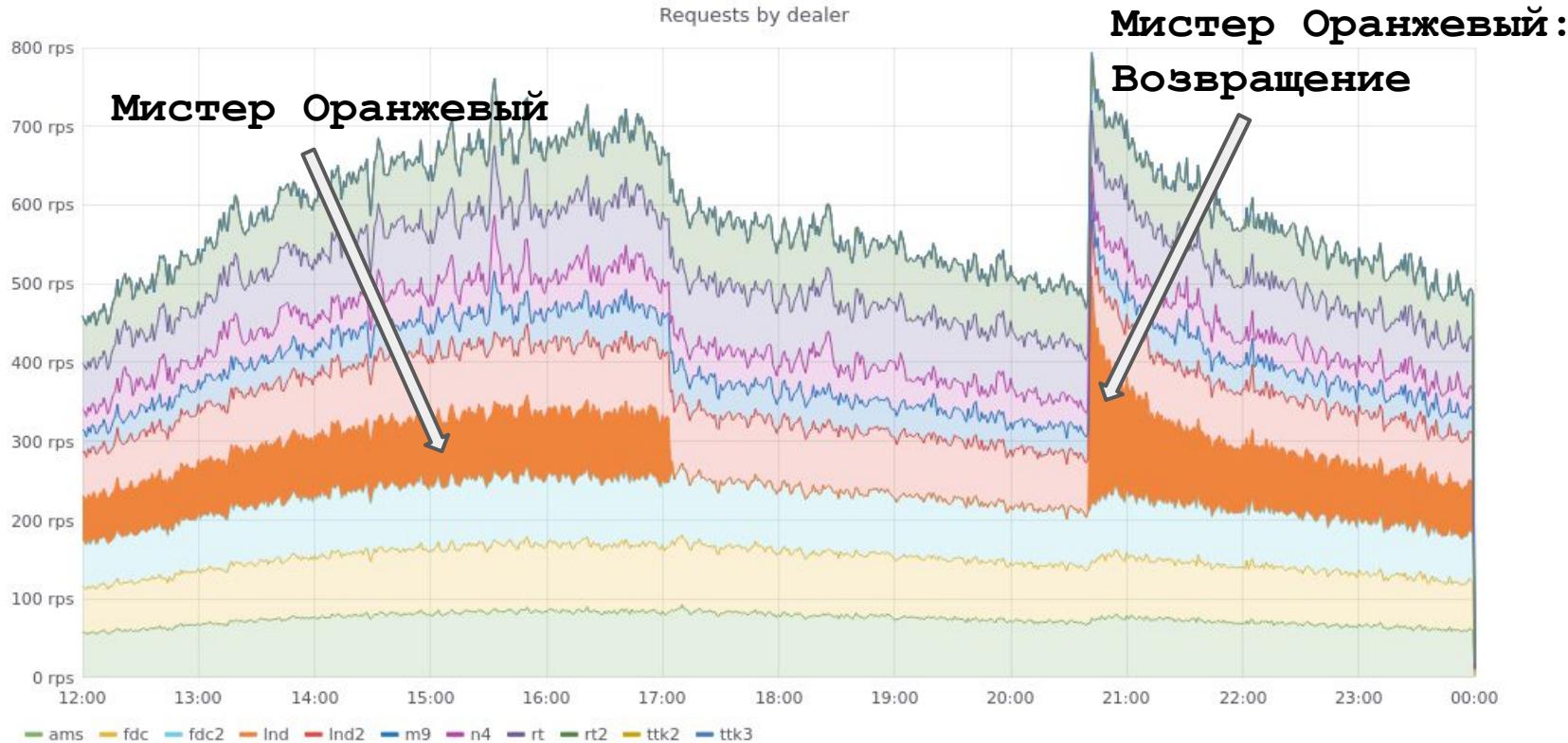
722 rows in set. Elapsed: **2.035 sec.** Processed 103.08 million rows, 4.00 GB (50.64 million rows/s., 1.97 GB/s.)

Полгода:

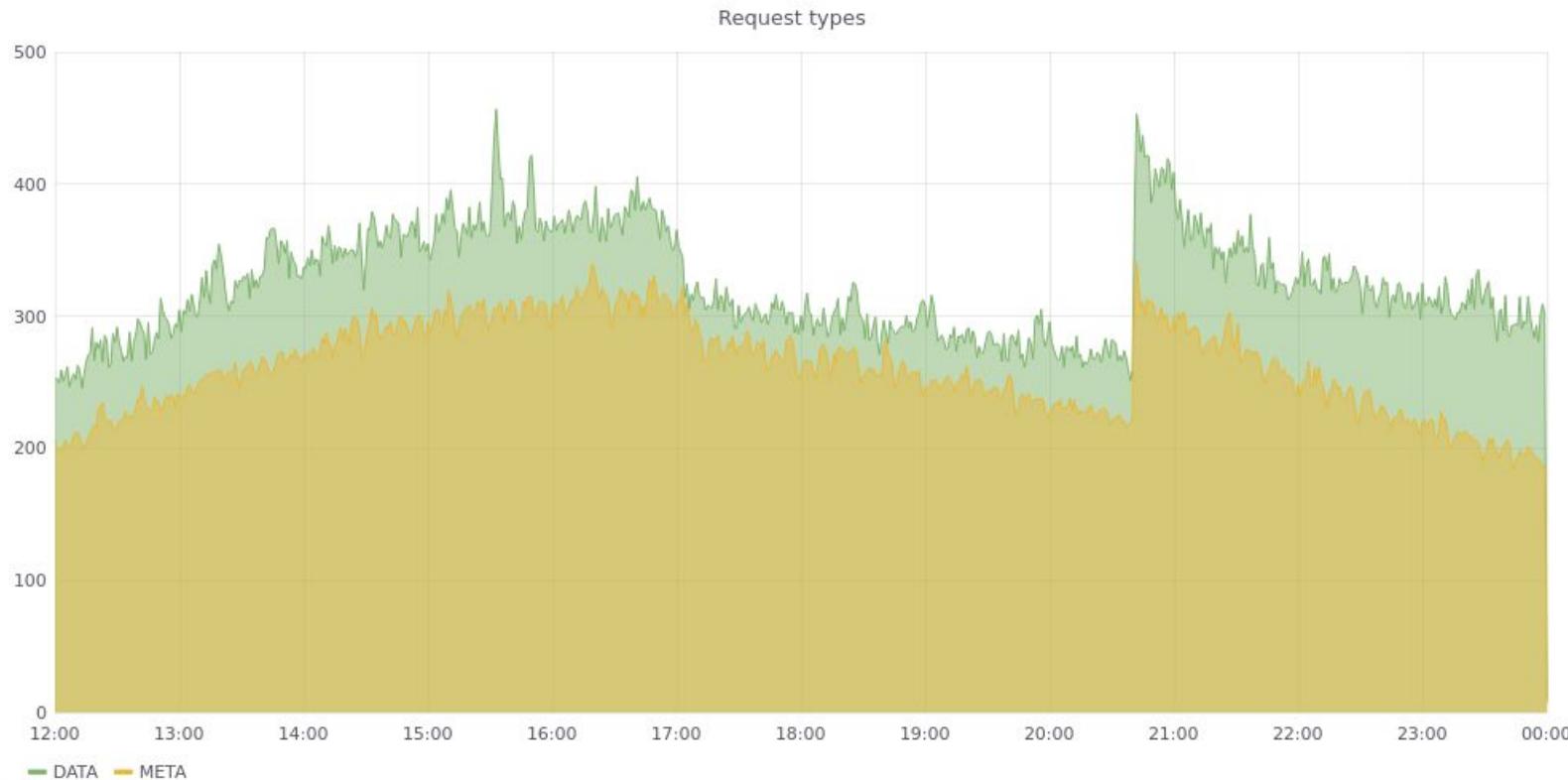
725 rows in set. Elapsed: **2.880 sec.** Processed 117.56 million rows, 4.56 GB (40.82 million rows/s., 1.58 GB/s.)

Пример расследования

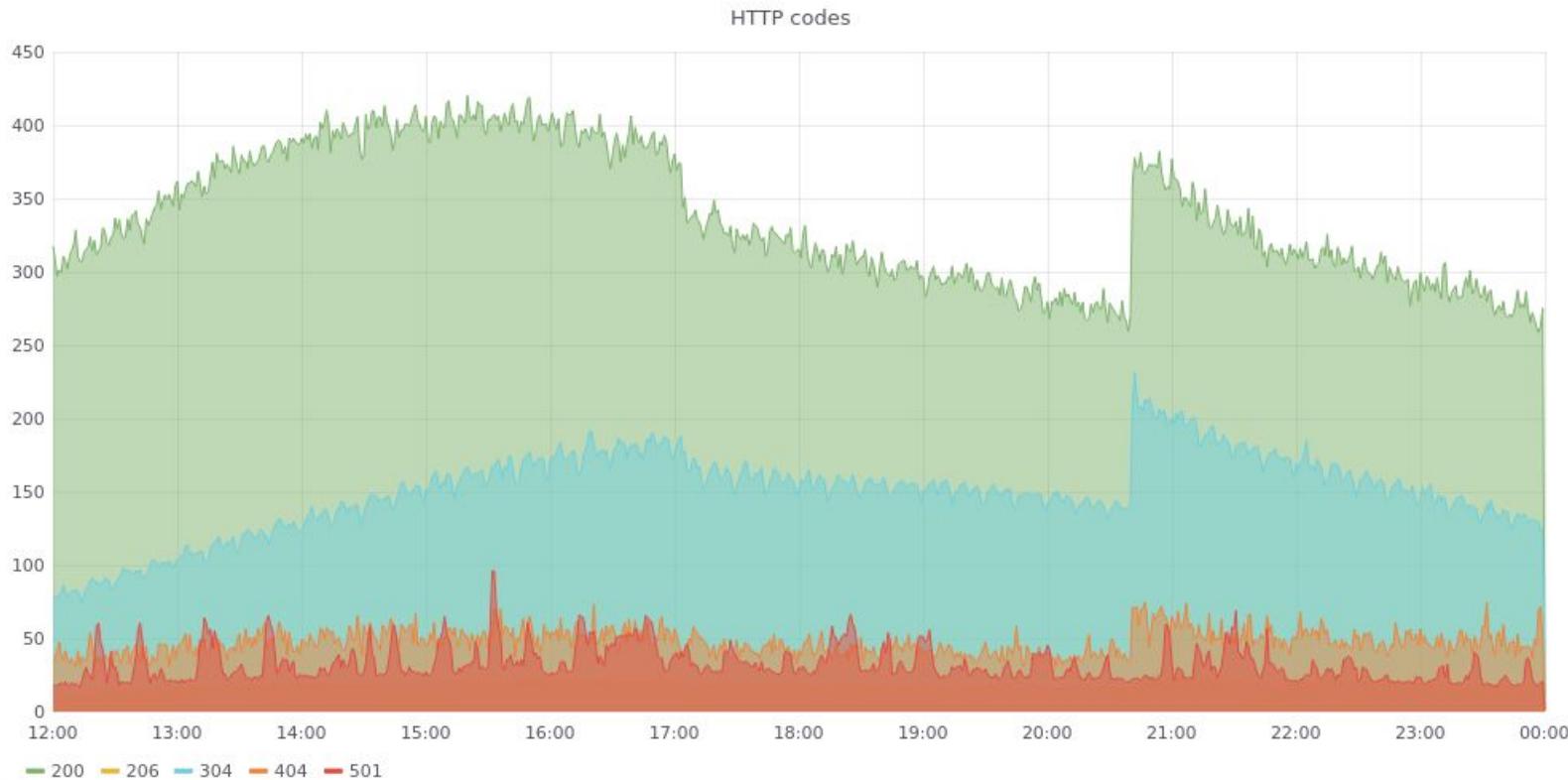
Недобалансировка при blackhole



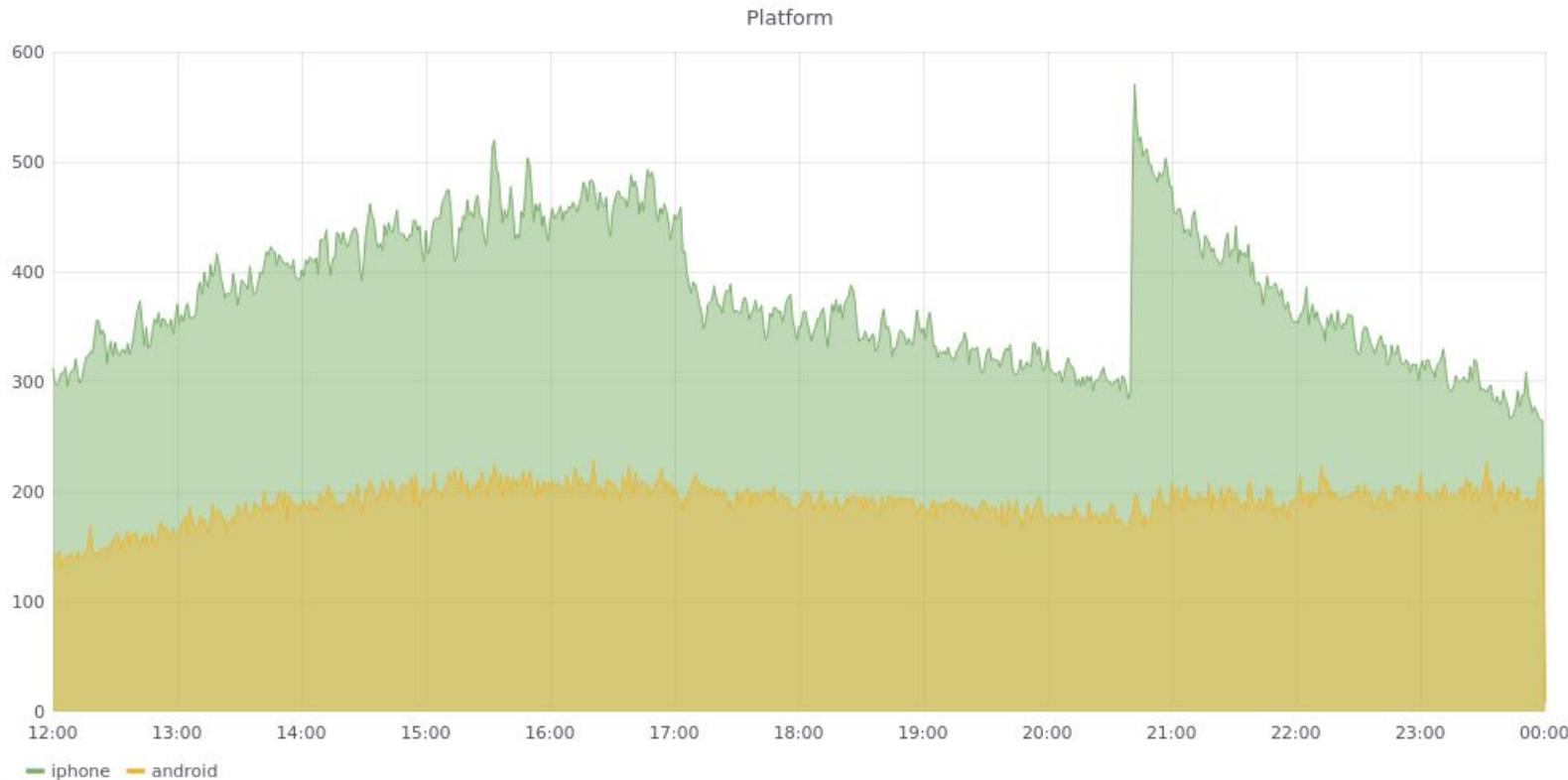
Корреляции по типам запроса нет



Корреляции по кодам ответа нет



Корреляция по платформам есть!



Если линий слишком много

```
PREWHERE app_version in (
    SELECT
        app_version
    WHERE $timeFilter
    GROUP BY app_version
    ORDER BY count() DESC
    LIMIT $limit
)
```

Если линий слишком много

```
PREWHERE app_version in (
    SELECT
        app_version
    WHERE $timeFilter
    GROUP BY app_version
    ORDER BY count() DESC
    LIMIT $limit
)
```

Если линий слишком много

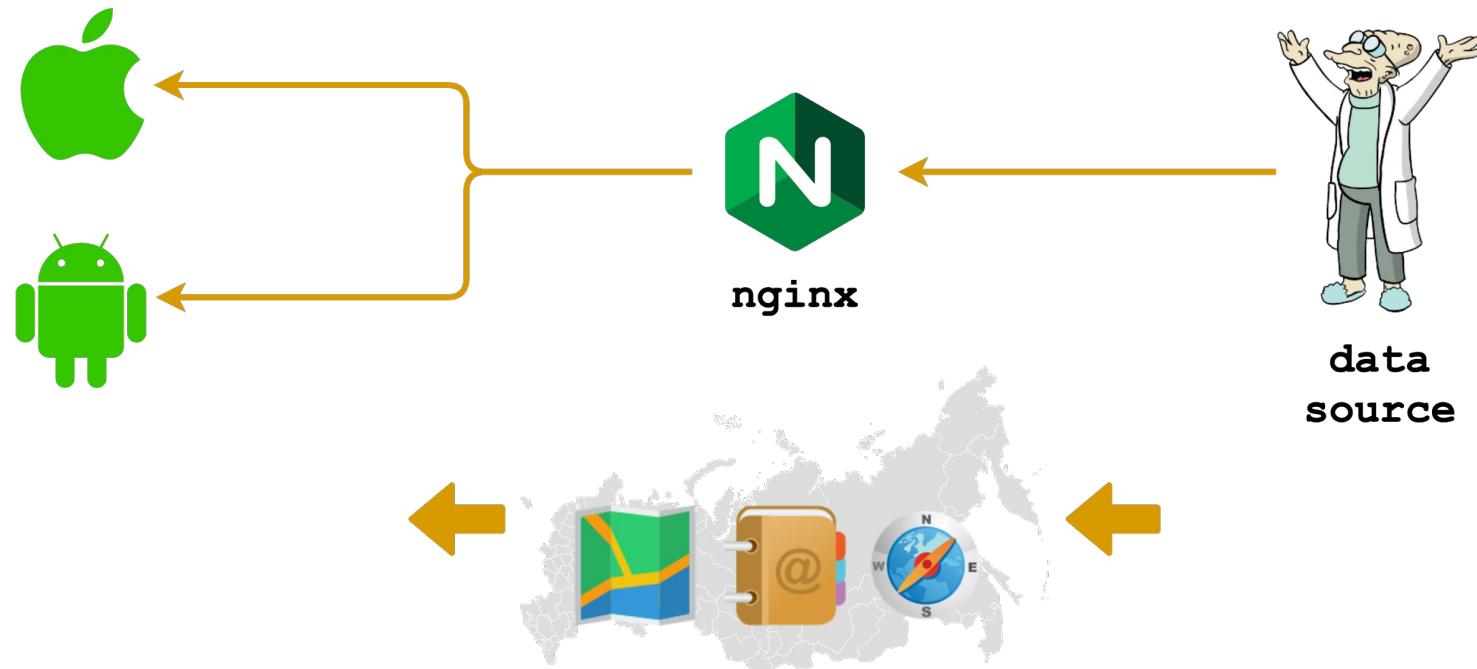
```
PREWHERE app_version in (
    SELECT
        app_version
    WHERE $timeFilter
    GROUP BY app_version
    ORDER BY count() DESC
    LIMIT $limit
)
```

Если линий слишком много

```
PREWHERE app_version in (
    SELECT
        app_version
    WHERE $timeFilter
    GROUP BY app_version
    ORDER BY count() DESC
    LIMIT $limit
)
```

А что чувствует пользователь?

Пользователь качает регион (несколько файлов)



Считаем время скачивания региона

SELECT

```
min(datetime - toUInt32(request_time)) as dl_start,  
max(datetime) as dl_end,  
dl_end - dl_start as dl_duration
```

...

GROUP BY

```
user_id,  
pkg_name_region,  
pkg_release_date
```

Считаем время скачивания региона

```
SELECT
```

```
    min(datetime - toUInt32(request_time)) as dl_start,  
max(datetime) as dl_end,  
    dl_end - dl_start as dl_duration
```

```
...
```

```
GROUP BY
```

```
    user_id,  
    pkg_name_region,  
    pkg_release_date
```

Считаем время скачивания региона

```
SELECT
```

```
    min(datetime - toUInt32(request_time)) as dl_start,  
    max(datetime) as dl_end,  
    dl_end - dl_start as dl_duration
```

```
...
```

```
GROUP BY
```

```
    user_id,  
    pkg_name_region,  
    pkg_release_date
```

Считаем время скачивания региона

SELECT

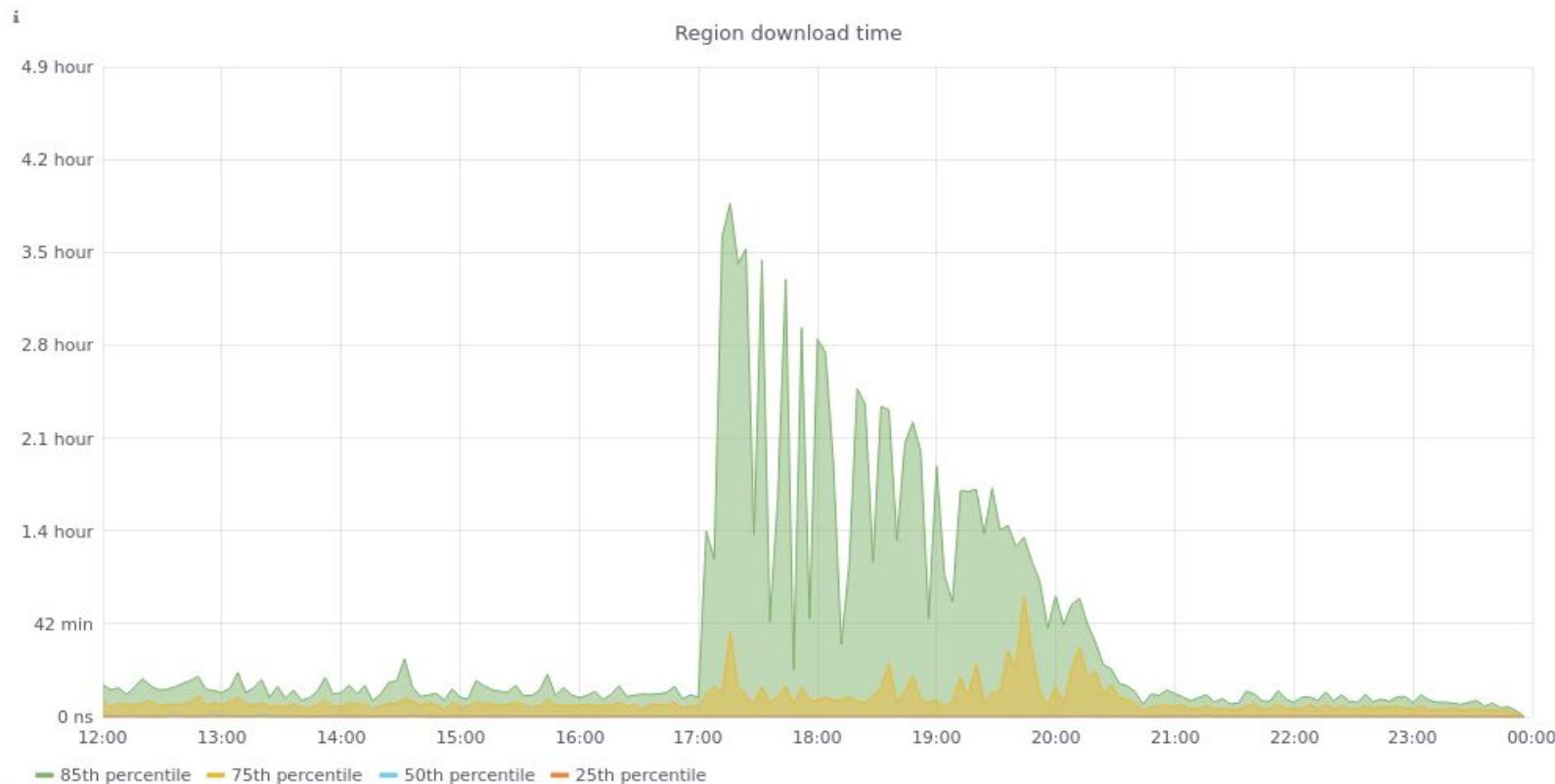
```
min(datetime - toUInt32(request_time)) as dl_start,  
max(datetime) as dl_end,  
dl_end - dl_start as dl_duration
```

...

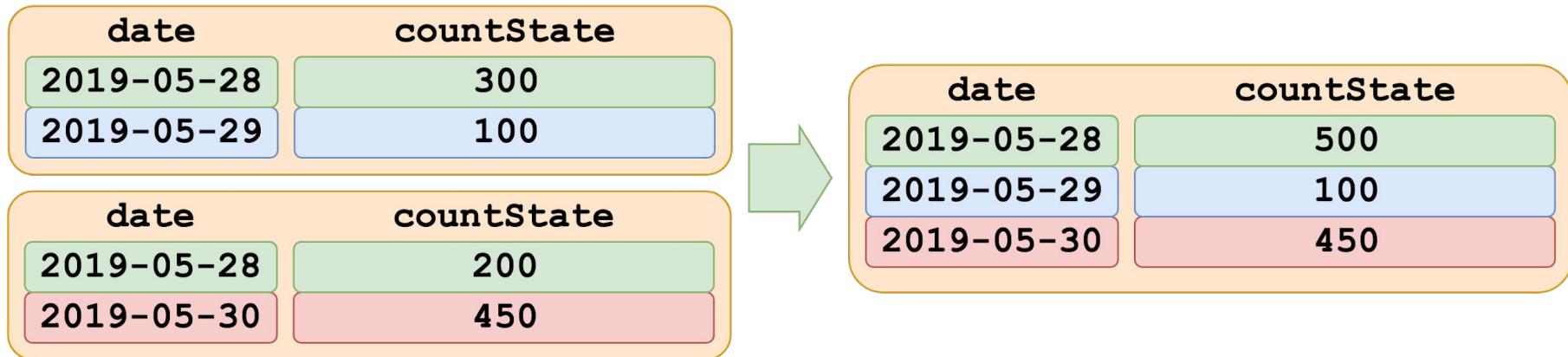
GROUP BY

```
user_id,  
pkg_name_region,  
pkg_release_date
```

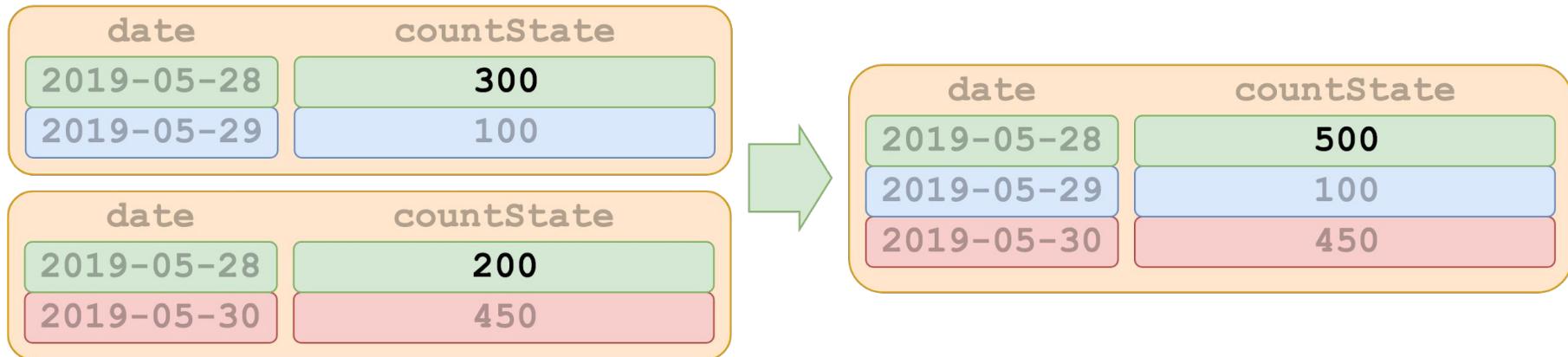
График времени скачивания Москвы



AggregatingMergeTree: count



AggregatingMergeTree: count



AggregatingMergeTree: uniq

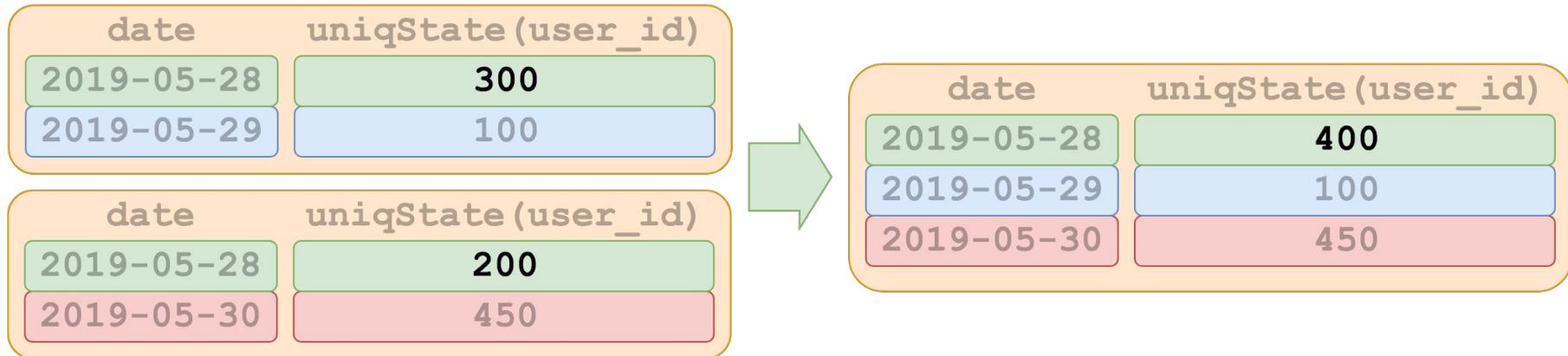
date	uniqState (user_id)
2019-05-28	300
2019-05-29	100

date	uniqState (user_id)
2019-05-28	200
2019-05-30	450



date	uniqState (user_id)
2019-05-28	400
2019-05-29	100
2019-05-30	450

AggregatingMergeTree: uniq



Выигрыш по времени на полугодовом графике

Из исходной таблицы:

364 rows in set. Elapsed: **175.341 sec.** Processed 8.32 billion rows, 132.52 GB (47.47 million rows/s., 755.79 MB/s.)

Из Materialized View:

362 rows in set. Elapsed: **15.690 sec.** Processed 48.73 million rows, 6.51 GB (3.11 million rows/s., 415.16 MB/s.)

Из Materialized View с сэмплированием:

362 rows in set. Elapsed: **1.569 sec.** Processed 48.73 million rows, 6.90 GB (31.06 million rows/s., 4.40 GB/s.)

Недостатки

- Отсутствие генерализации

Недостатки

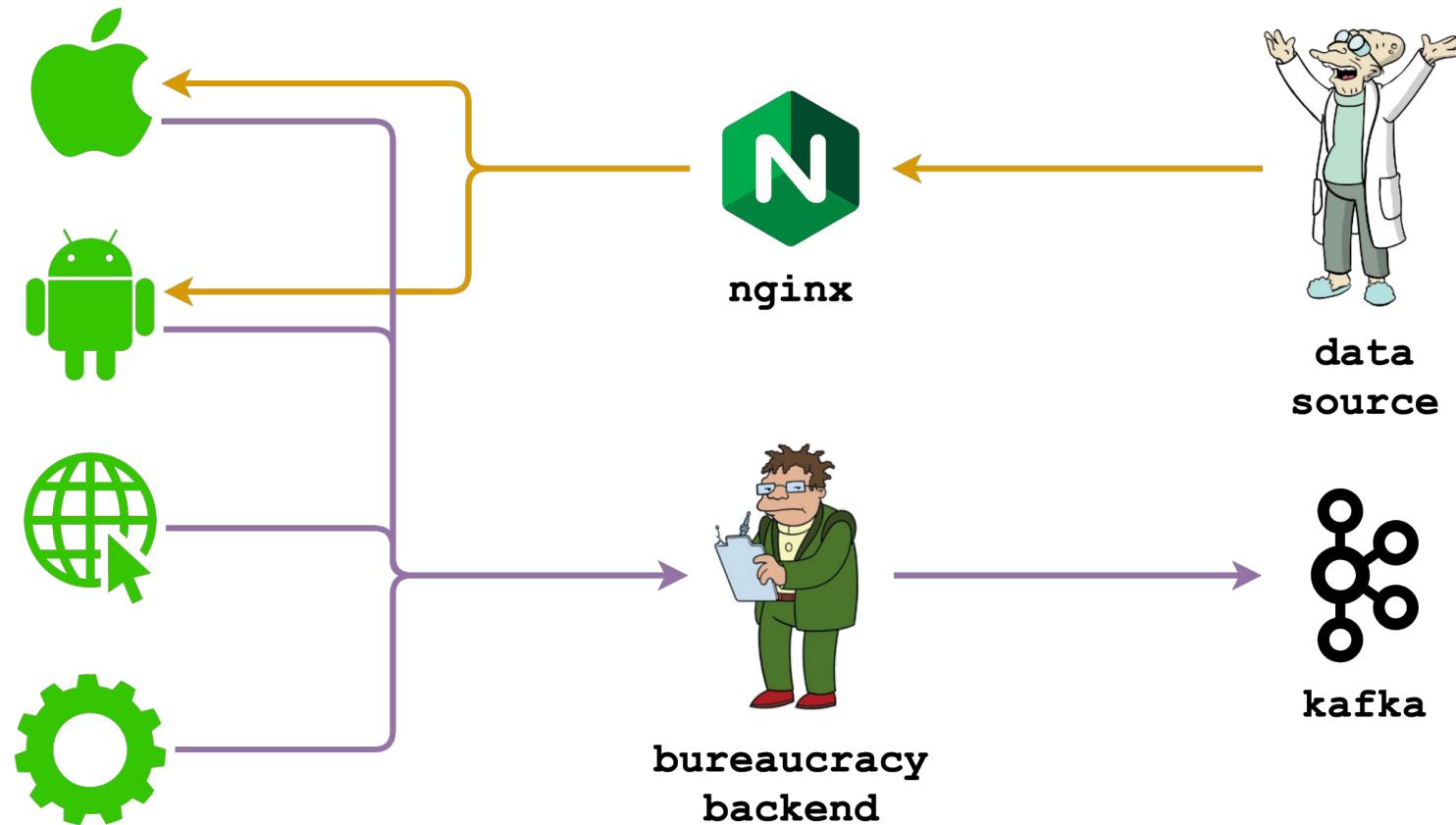
- Отсутствие генерализации
- Самописная прослойка между Logstash и ClickHouse

Недостатки

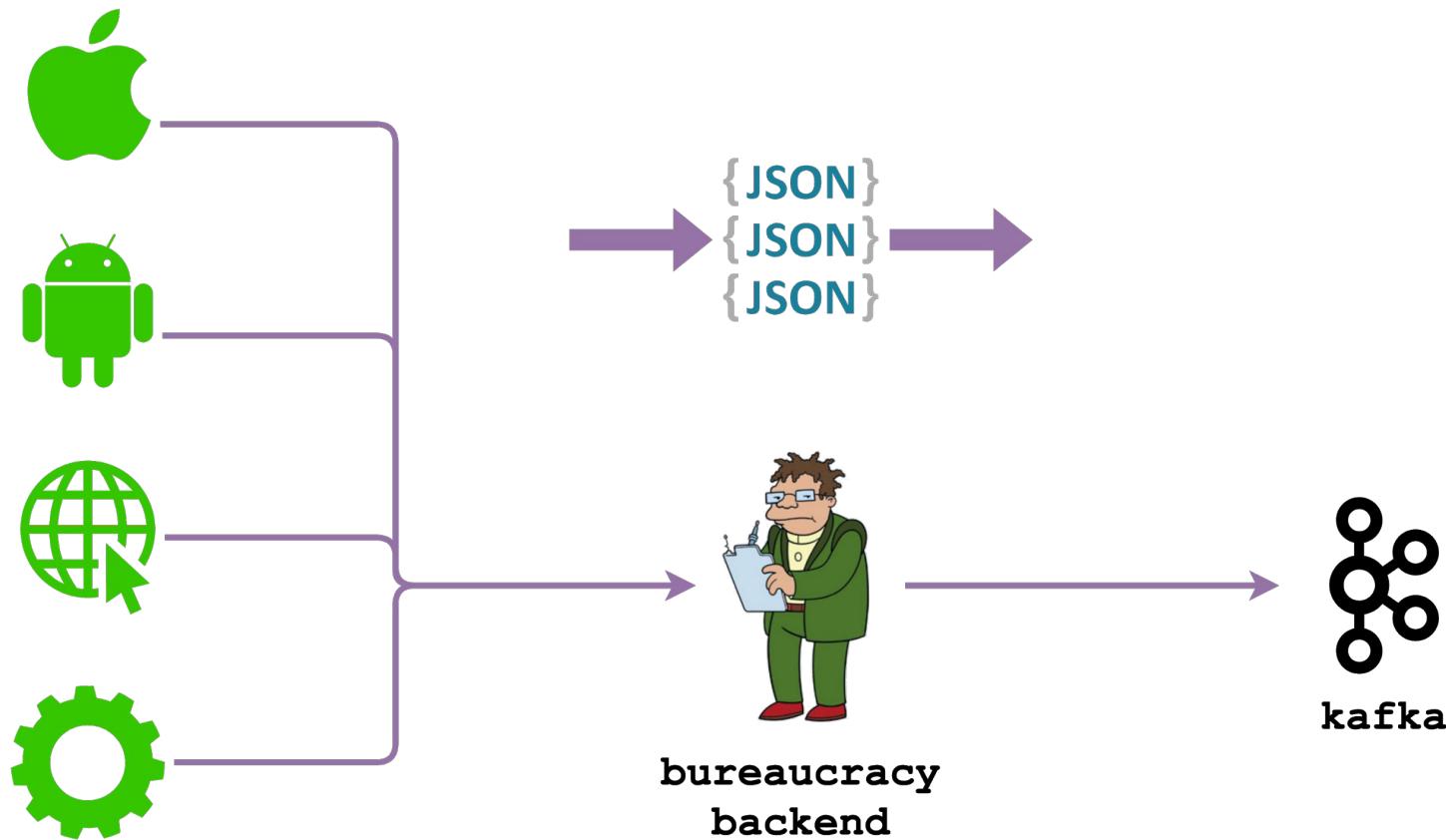
- Отсутствие генерализации
- Самописная прослойка между Logstash и ClickHouse
- Нет аналогов Kibana

Сервис транспорта статистики

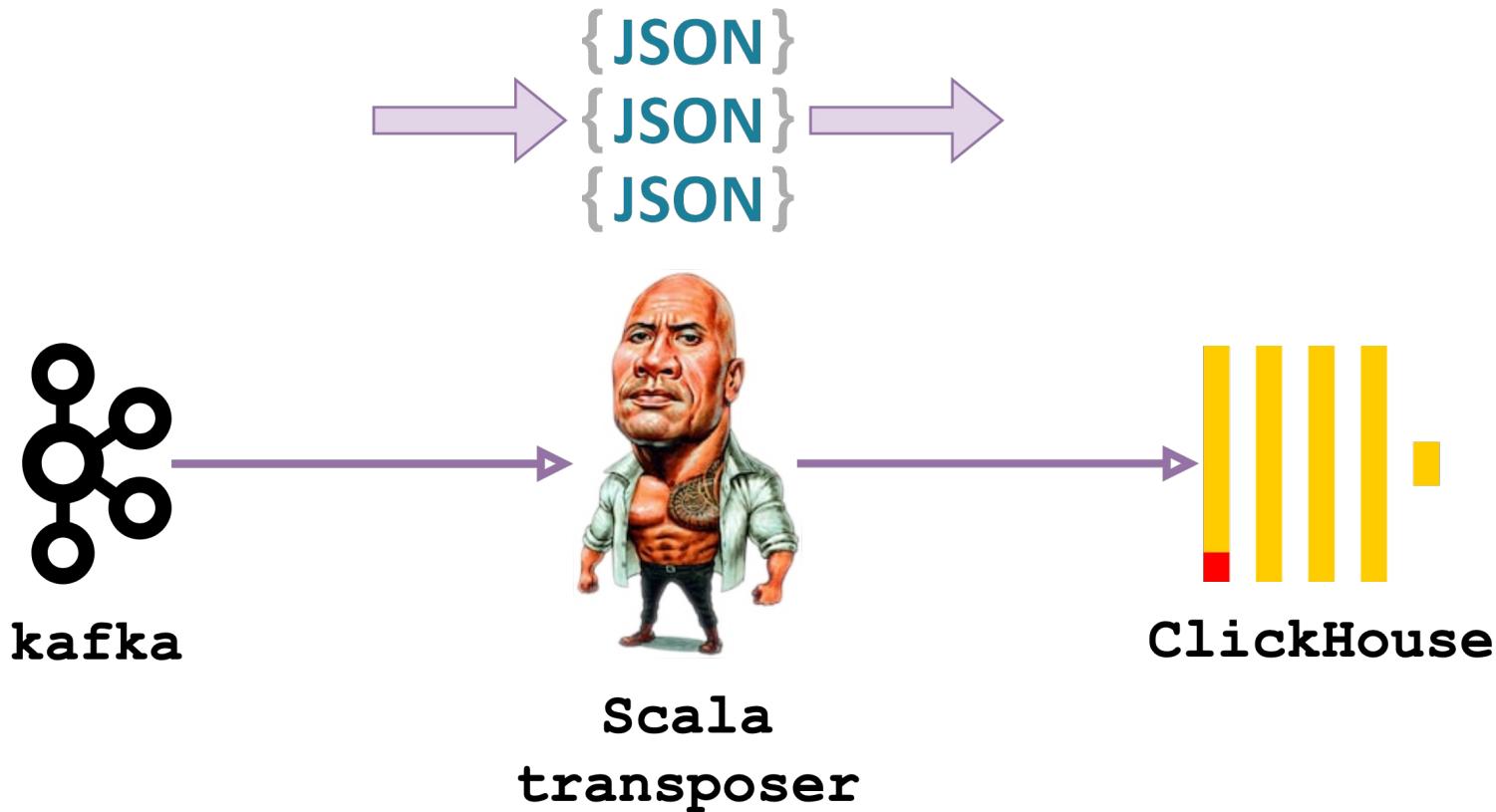
Наши сервисы



Транспорт статистики



Eat your own dog food



Создаём таблицу

```
CREATE TABLE messages (
    date Date,
    user_ts DateTime,
    recv_ts DateTime,
    kafka_ts DateTime,
    type UInt16,
    product UInt16,
    app_version String,
    ...
)
ENGINE = ReplicatedMergeTree(....)
PARTITION BY date
...
```

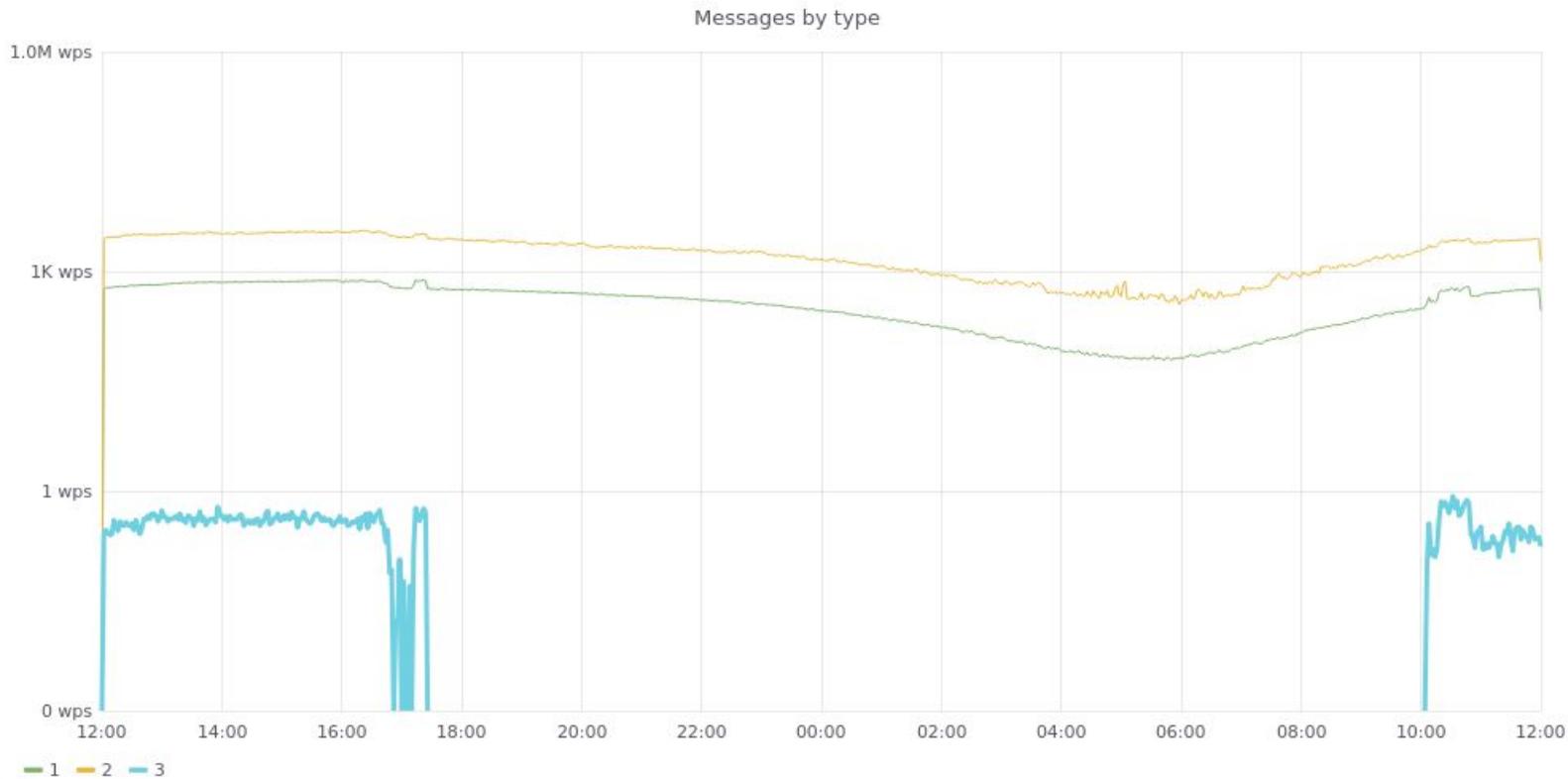
Время прохождения через транспорт



Разбивка по продуктам

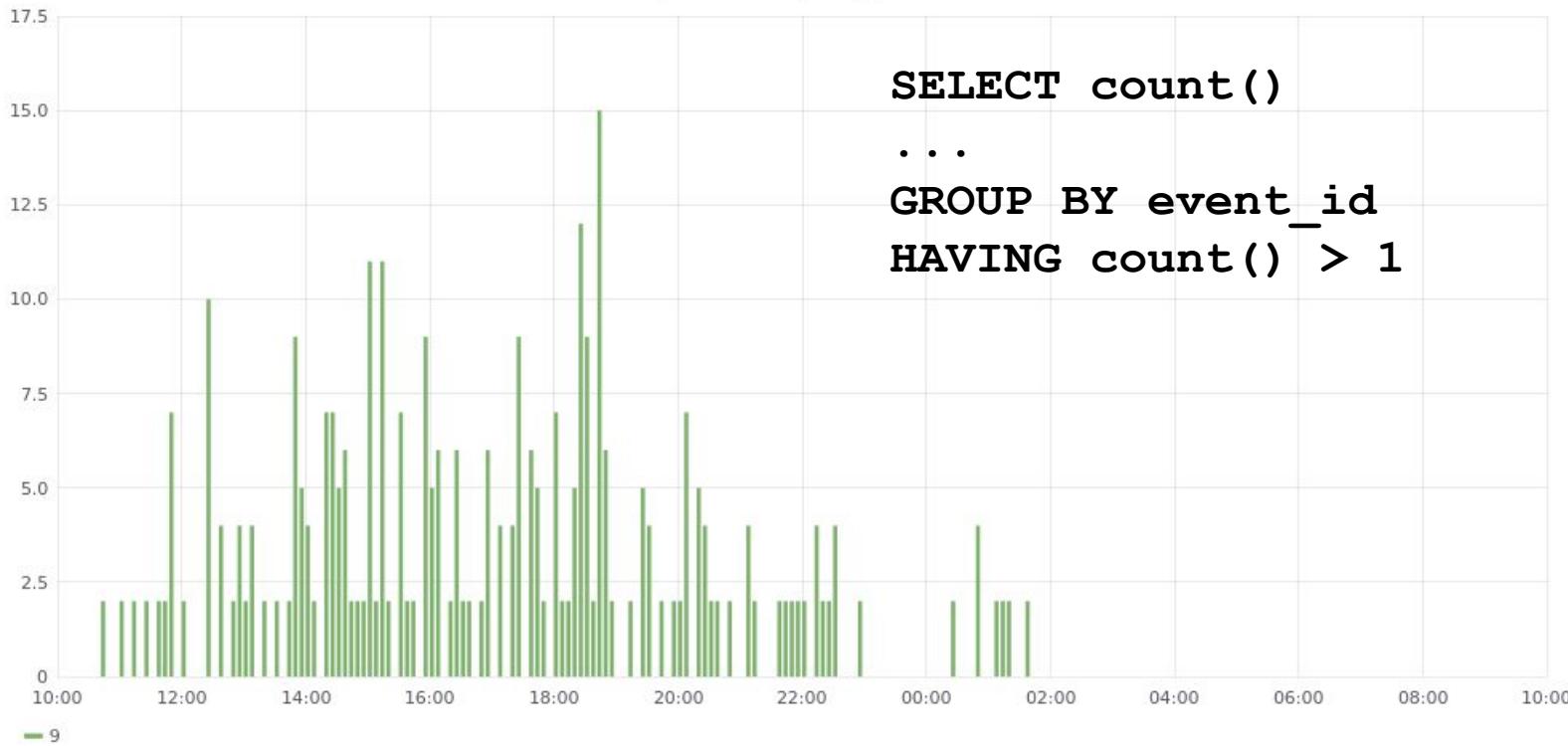


Разбивка по типам сообщений

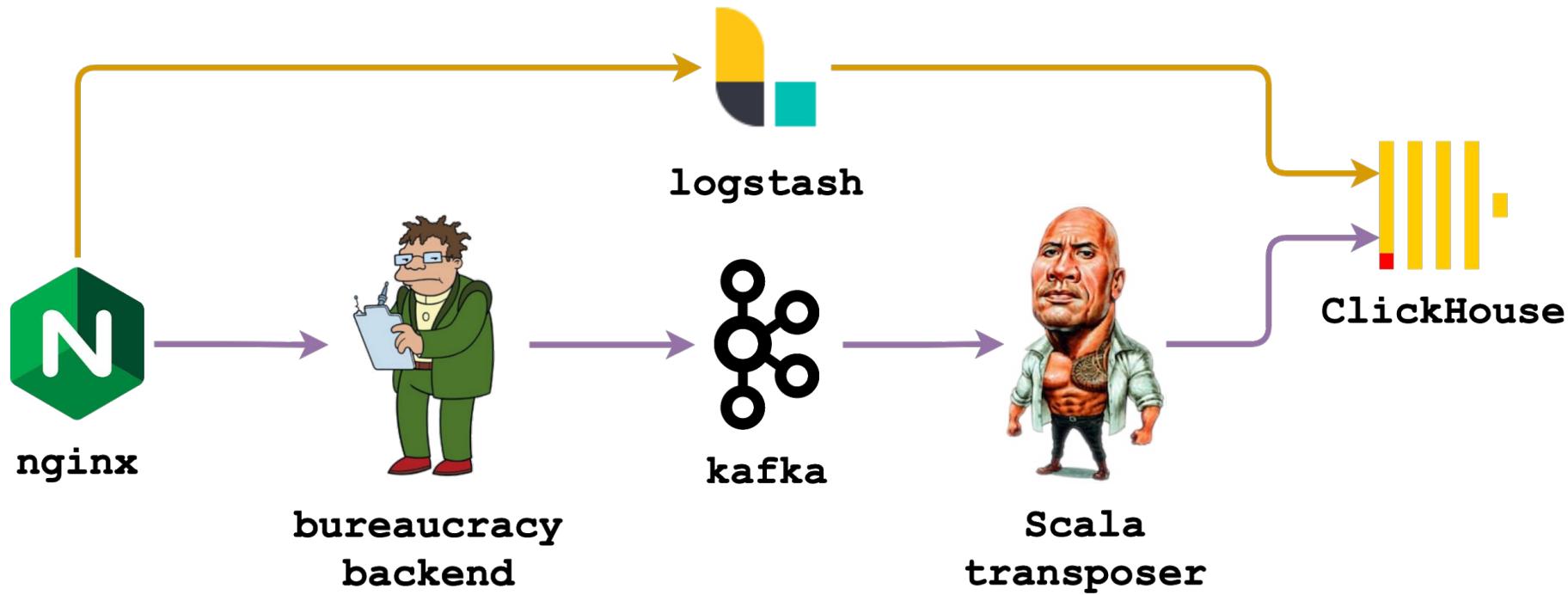


Поиск дубликатов

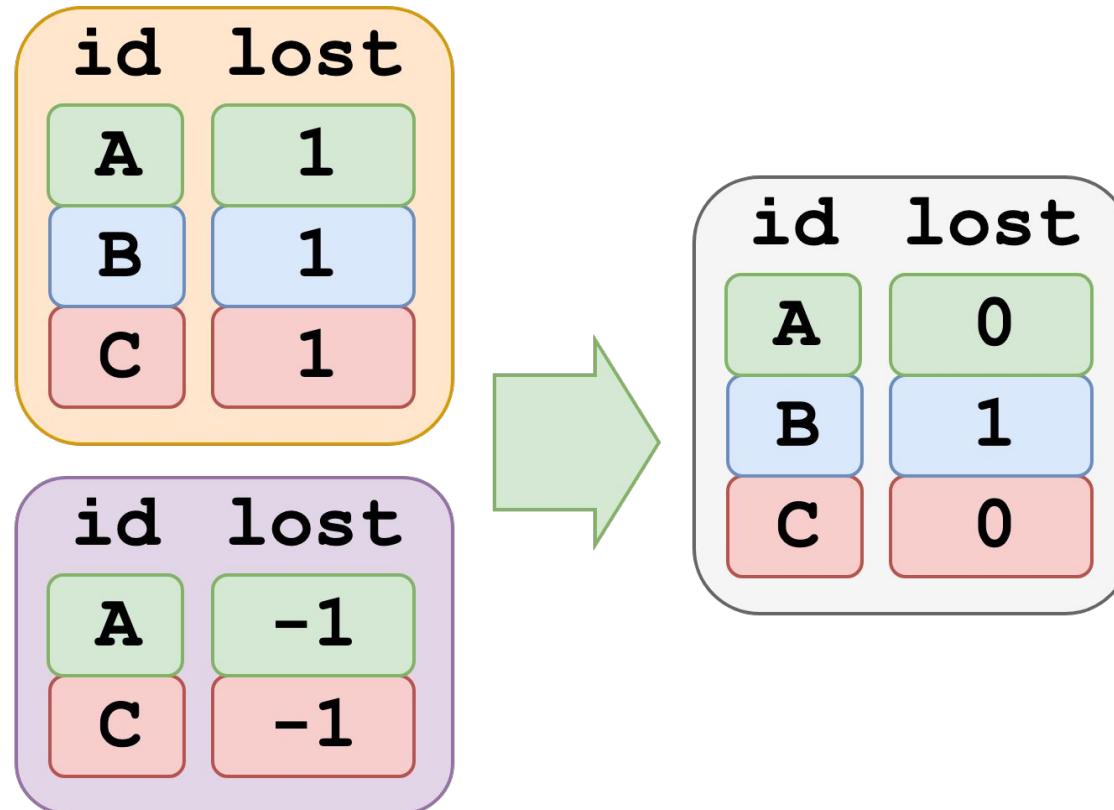
Duplicate messages by product



Поиск потерь: в одной таблице и логи, и запросы

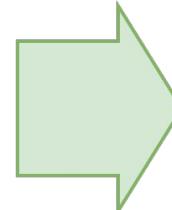


Поиск потерь: SummingMergeTree



Поиск потерь: SummingMergeTree

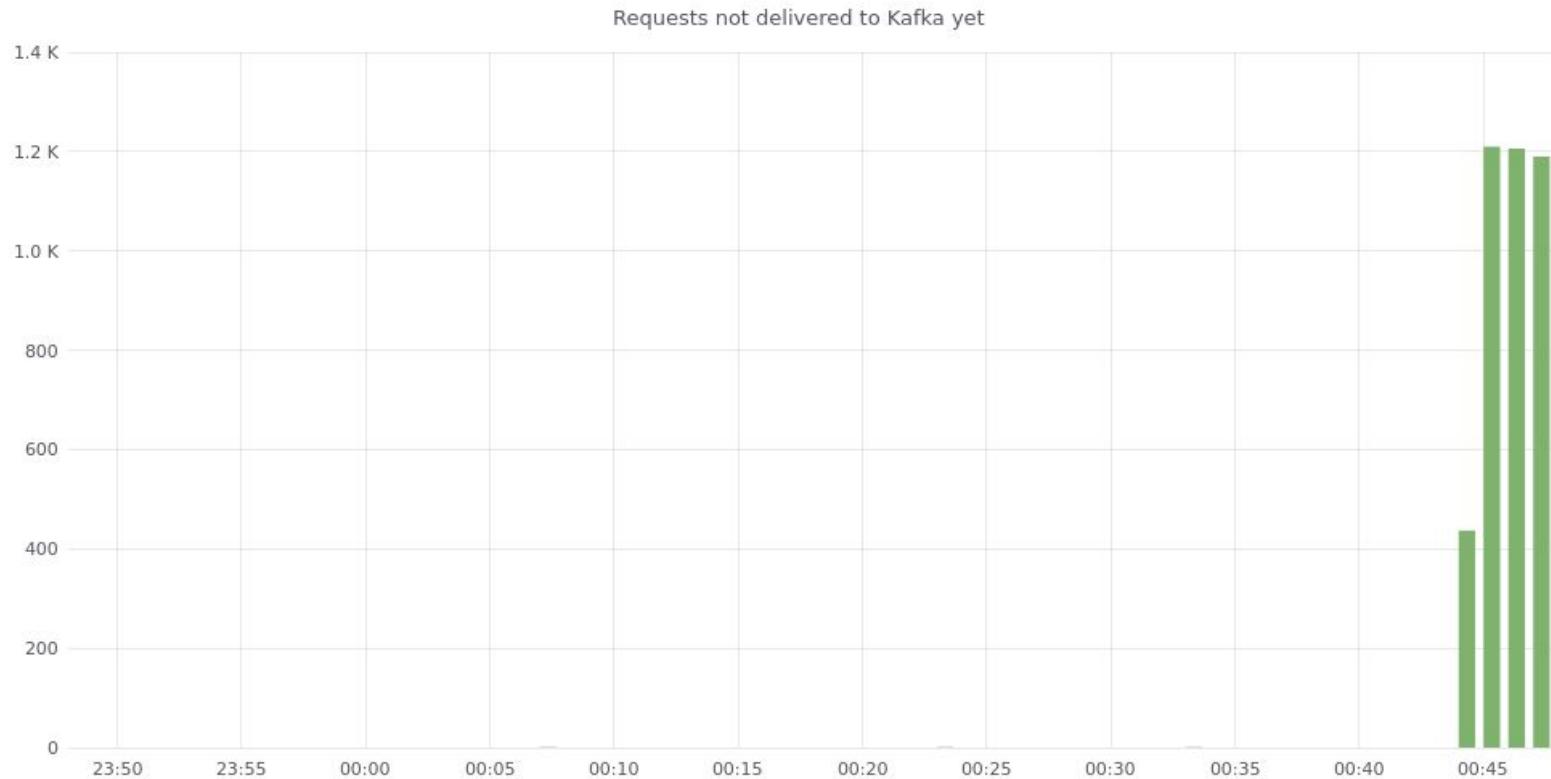
id lost	
A	1
B	1
C	1



id lost	
A	-1
C	-1

id lost	
A	0
B	1
C	0

Поиск потерь



Заключение

Если через тебя проходит поток структурированных данных:

- Направляешь этот поток в ClickHouse
- Рисуешь крутые графики
- Если тормозят — используешь SAMPLE и Materialized View
- Выводишь понимание работы сервиса на новый уровень
- PROFIT

Полезные ссылки

- <https://github.com/Vertamedia/clickhouse-grafana>
- <http://nginx.org/ru/docs/varindex.html>
- <https://clickhouse.yandex/docs/ru/>

Вопросы?

antony.alekseyev@gmail.com