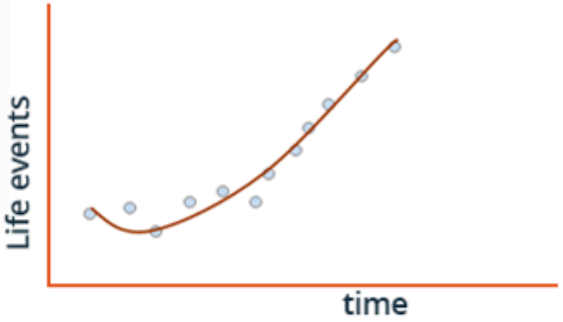

ClickHouse for Time-Series
Alexander Zaitsev
Altinity

Altinity

- Мы делаем ClickHouse еще быстрее и еще удобнее
- Как мы это делаем:
  - Дорабатываем ClickHouse вместе с Яндексом
  - Разрабатываем дополнительный софт (Kubernetes, cluster manager, tools & utilities)
  - Обучаем и помогаем построить решение на ClickHouse
  - Обеспечиваем 24x7 поддержку ClickHouse-инсталляций

Что такое time-series?



Набор упорядоченных во времени событий, представляющих изменения какого-то процесса во времени

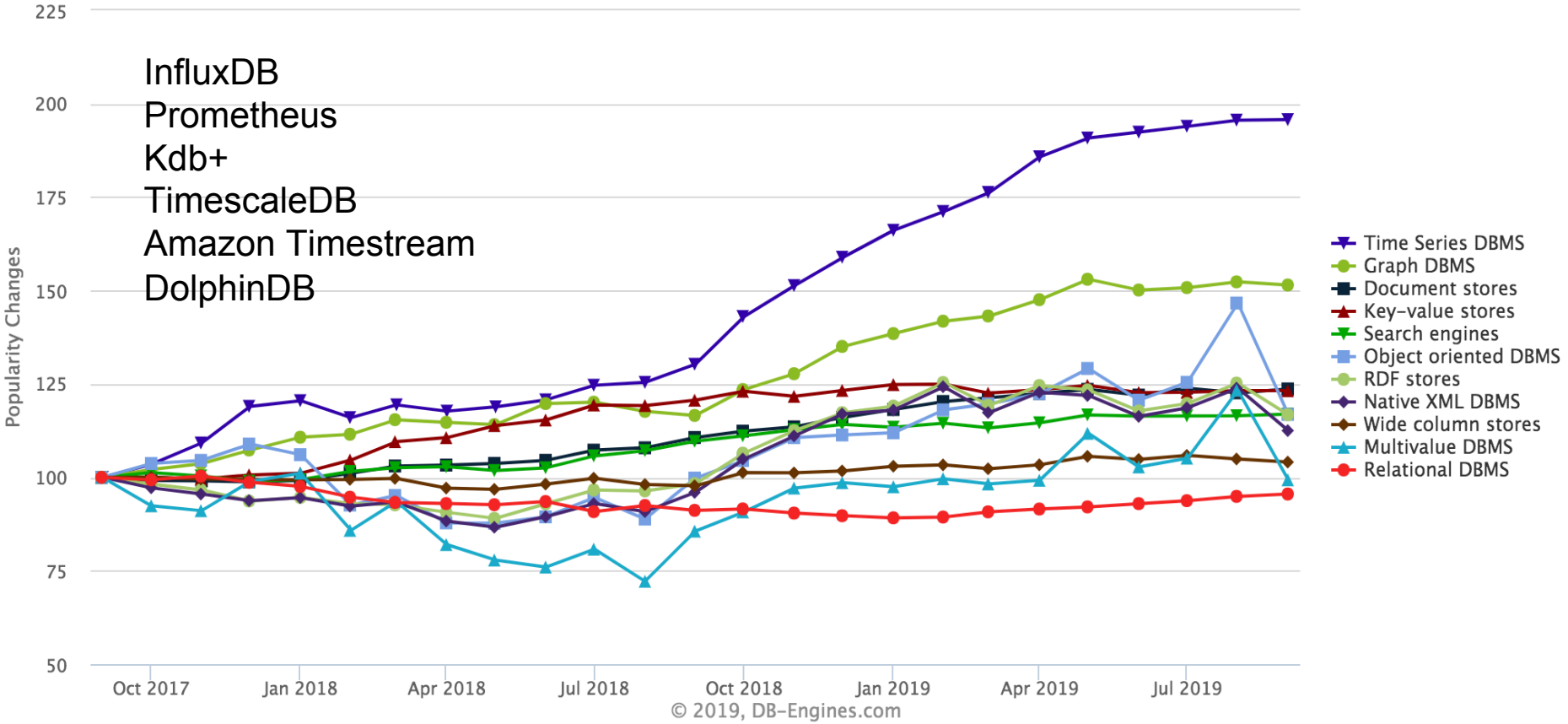
Мониторинг
Финансы
Интернет вещей

Зачем?

Измеряем ***изменения***:

- Как что-то изменилось по сравнению с прошлым
- Как что-то меняется прямо сейчас
- Предсказать изменения в будущем

Специальные time-series DBMS растут!



Что особенного в time-series DBMS?

- Оптимизированы на быструю вставку INSERT
- Эффективно и гибко хранят и “забывают” данные
- Автоматически считают агрегаты (downsampling)
- Эффективно запрашивают агрегированные данные

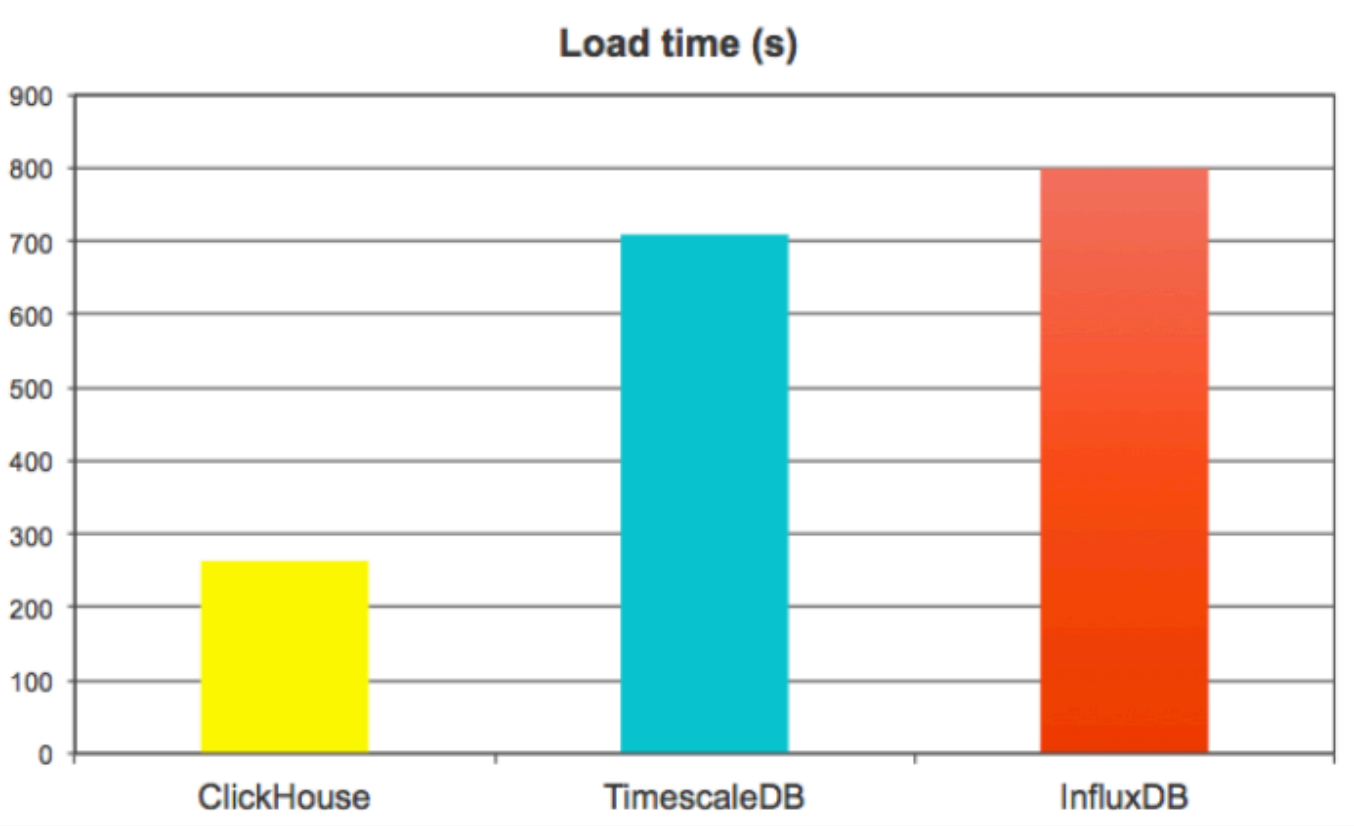
Так это же ClickHouse!

Бенчмарк ноября 2018. TSBS

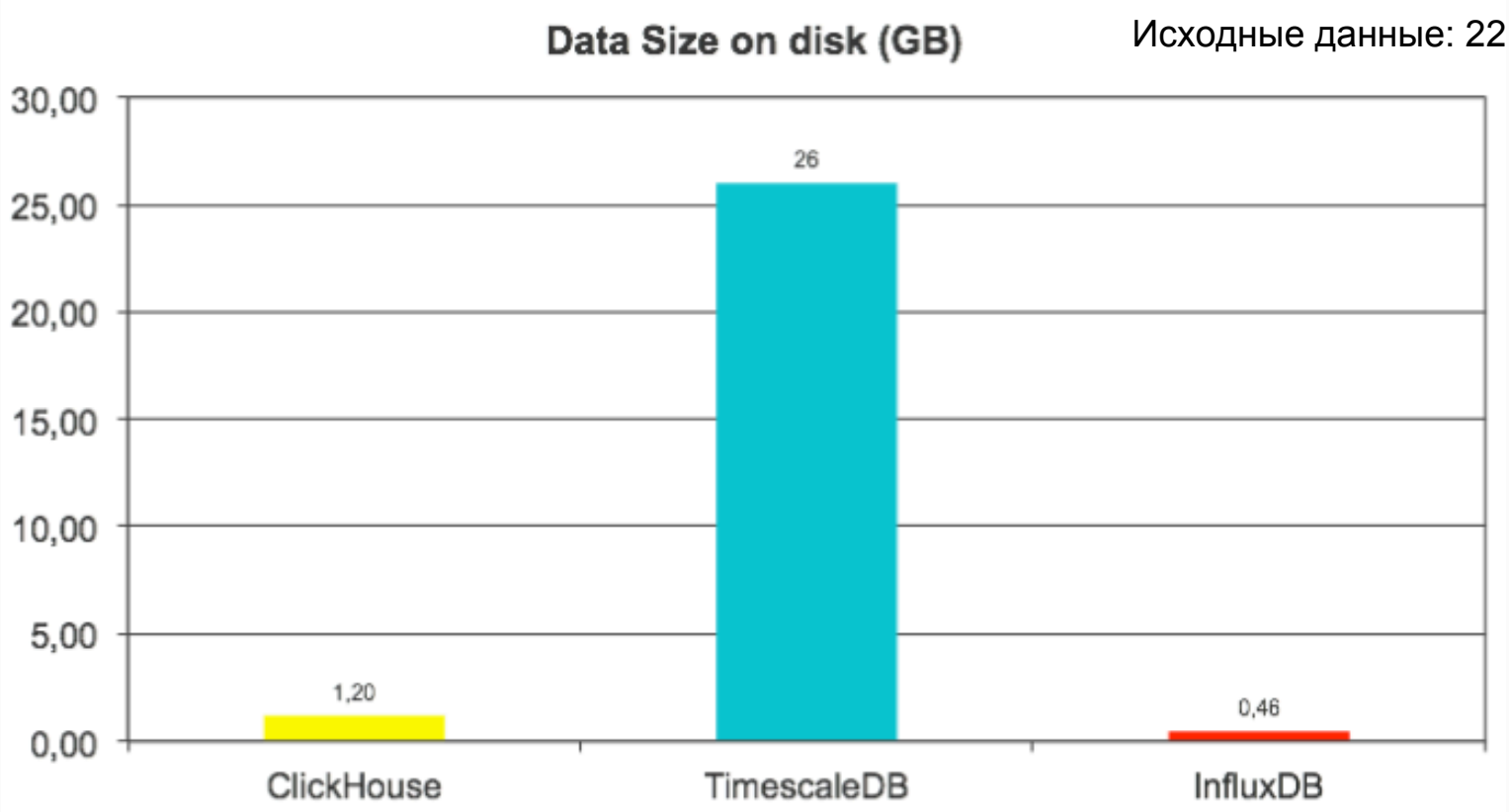
- https://github.com/timescale/tsbs
- ClickHouse vs TimescaleDB vs InfluxDB (vs Cassandra)
- Amazon r5.2xlarge instance, 8 vCPUs, 64GB RAM, EBS storage
- 100M строк, 10 метрик (колонок) + метаданные
- 15 тестовых запросов, типичных для time series

https://www.altinity.com/blog/clickhouse-for-time-series

Бенчмарк ноября 2018. TSBS

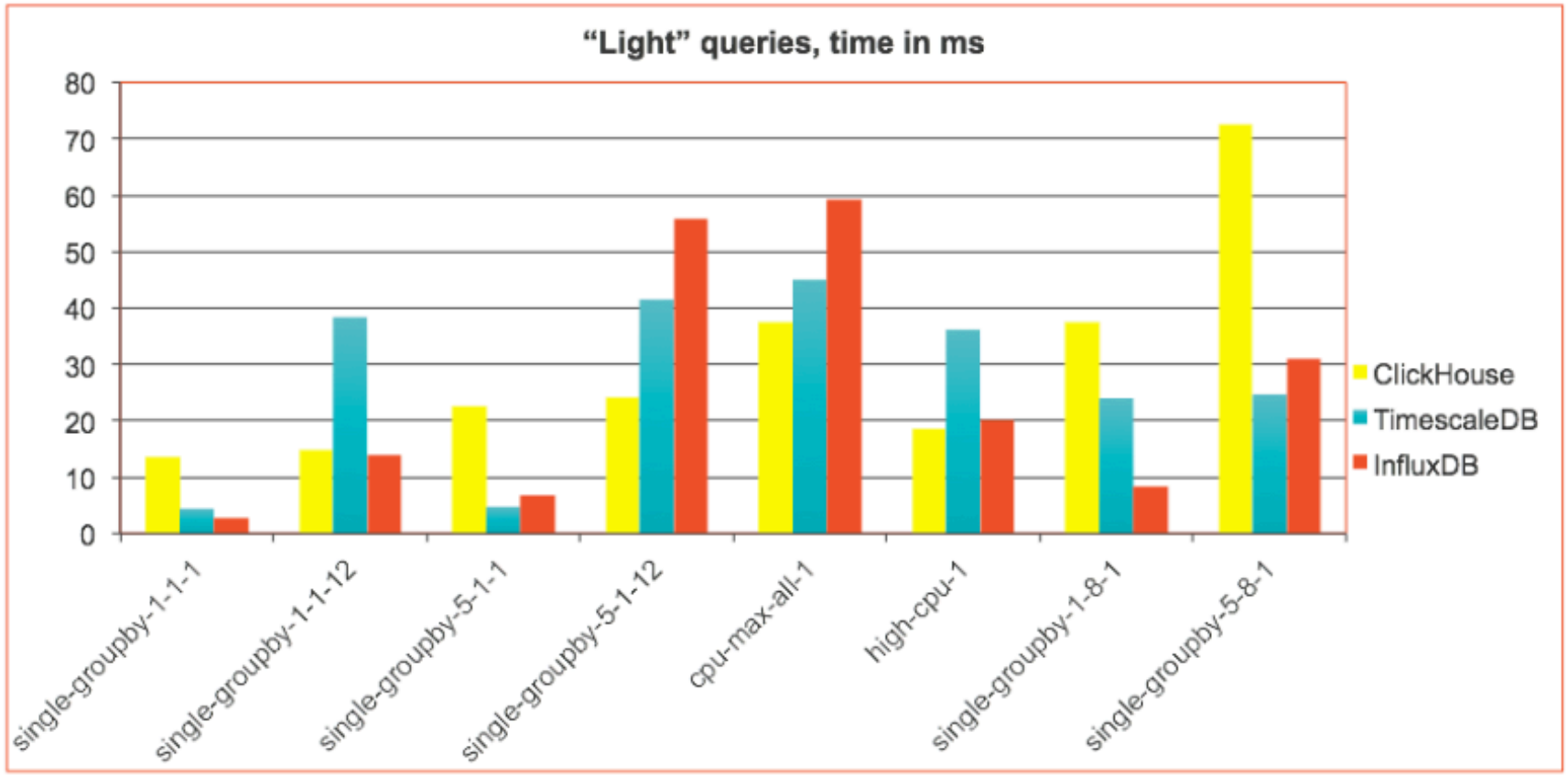


Бенчмарк ноября 2018. TSBS

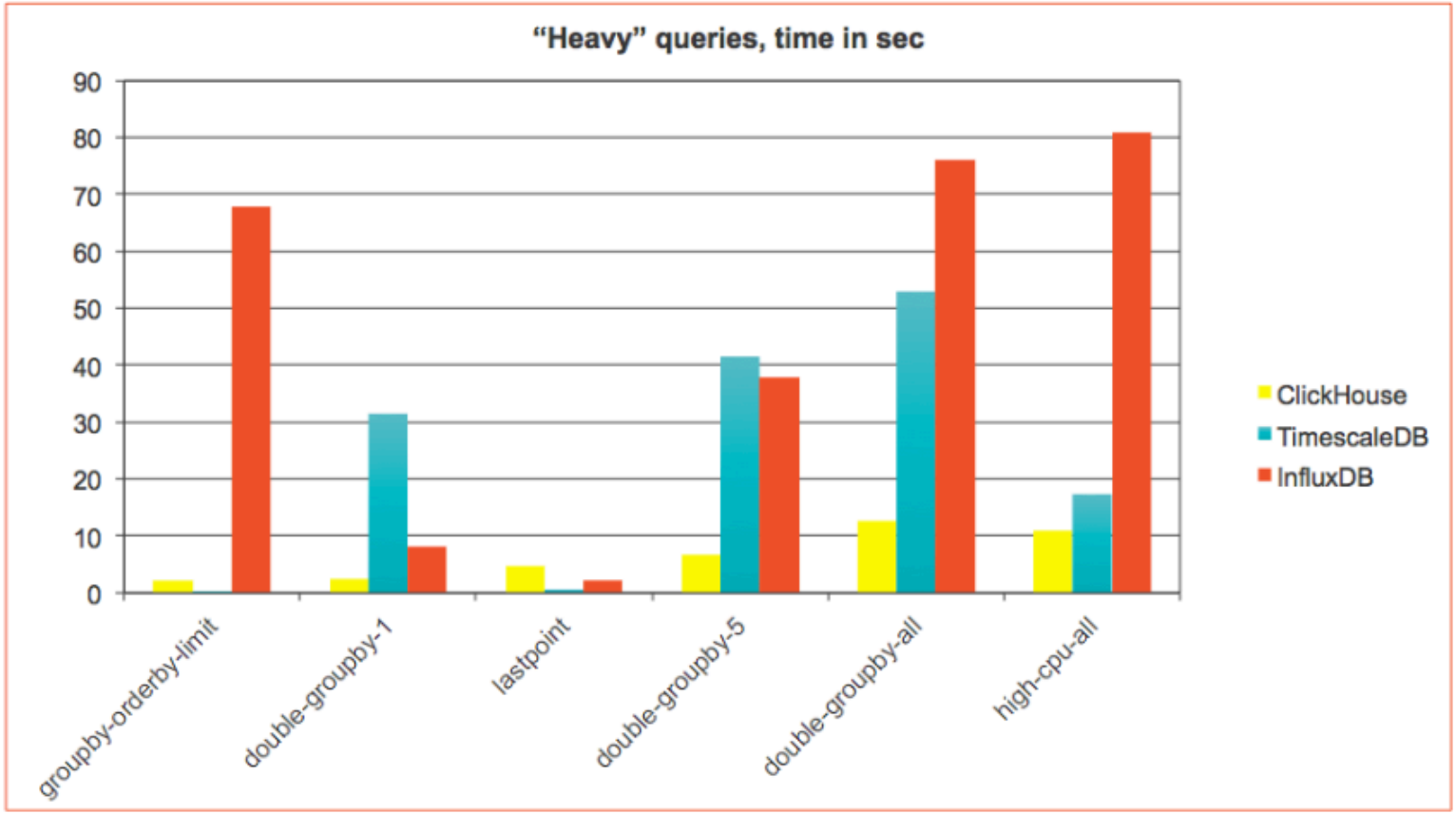


Исходные данные: 22.5GB

Бенчмарк ноября 2018. TSBS



Бенчмарк ноября 2018. TSBS



Промежуточные итоги

- Грузим очень быстро
- Сжимаем довольно неплохо, но можно лучше
- Запросы выполняем довольно неплохо, но можно лучше

Как надо строить time-series на ClickHouse

Схема

- Регулярные данные (знаем метрики заранее)
- Нерегулярные данные (не знаем метрики заранее)
- Произвольные аннотации и теги

Схема. Регулярные данные. Колонки.

```
CREATE TABLE cpu (
  created_date Date DEFAULT today(),
  created_at DateTime DEFAULT now(),
  time String,
  tags_id UInt32,  /* join to dim_tag */
  usage_user Float64,
  usage_system Float64,
  usage_idle Float64,
  usage_nice Float64,
  usage_iowait Float64,
  usage_irq Float64,
  usage_softirq Float64,
  usage_steal Float64,
  usage_guest Float64,
  usage_guest_nice Float64
) ENGINE = MergeTree(created_date, (tags_id, created_at), 8192);
```

Схема. Нерегулярные данные. Массивы.

```
CREATE TABLE cpu_alc (
  created_date Date,
  created_at DateTime,
  time String,
  tags_id UInt32,
  metrics Nested(
    name LowCardinality(String),
    value Float64
  )
) ENGINE = MergeTree(created_date, (tags_id, created_at), 8192);
```

```
SELECT max(metrics.value[indexOf(metrics.name,'usage_user')]) FROM ...
```

indexOf не оптимизирован: https://github.com/yandex/ClickHouse/issues/6005

Схема. Нерегулярные данные. Строки.

```
CREATE TABLE cpu_rlc (
  created_date Date,
  created_at DateTime,
  time String,
  tags_id UInt32,
  metric_name LowCardinality(String),
  metric_value Float64
) ENGINE = MergeTree(created_date, (metric_name, tags_id, created_at),
8192);


SELECT
    maxIf(metric_value, metric_name = 'usage_user'),
    ...
FROM cpu_r
WHERE metric_name IN ('usage_user', ...)
```

Схема. Сравнение.

Тип схемы	Размер на диске	Плюсы	Минусы
Колонки	1.23 GB	- Максимальное сжатие - Максимальная скорость	- Фиксированная схема
Массивы	1.48 GB	- Отлично сжимается - Подходит для нерегулярных данных	- Скорость деградирует с размером массива
Строки	4.7 GB	- Максимально простая схема - Отличная скорость для одной метрики	- Плохо сжимается, много строк - Запросы по нескольким метрикам сразу не оптимальны

Детали: https://www.altinity.com/blog/2019/5/23/handling-variable-time-series-efficiently-in-clickhouse

Компрессия и кодирование

- Кодек vs компрессор
- Типы кодирования:
  - RLE
  - Dictionary encoding
  - Entropy coding

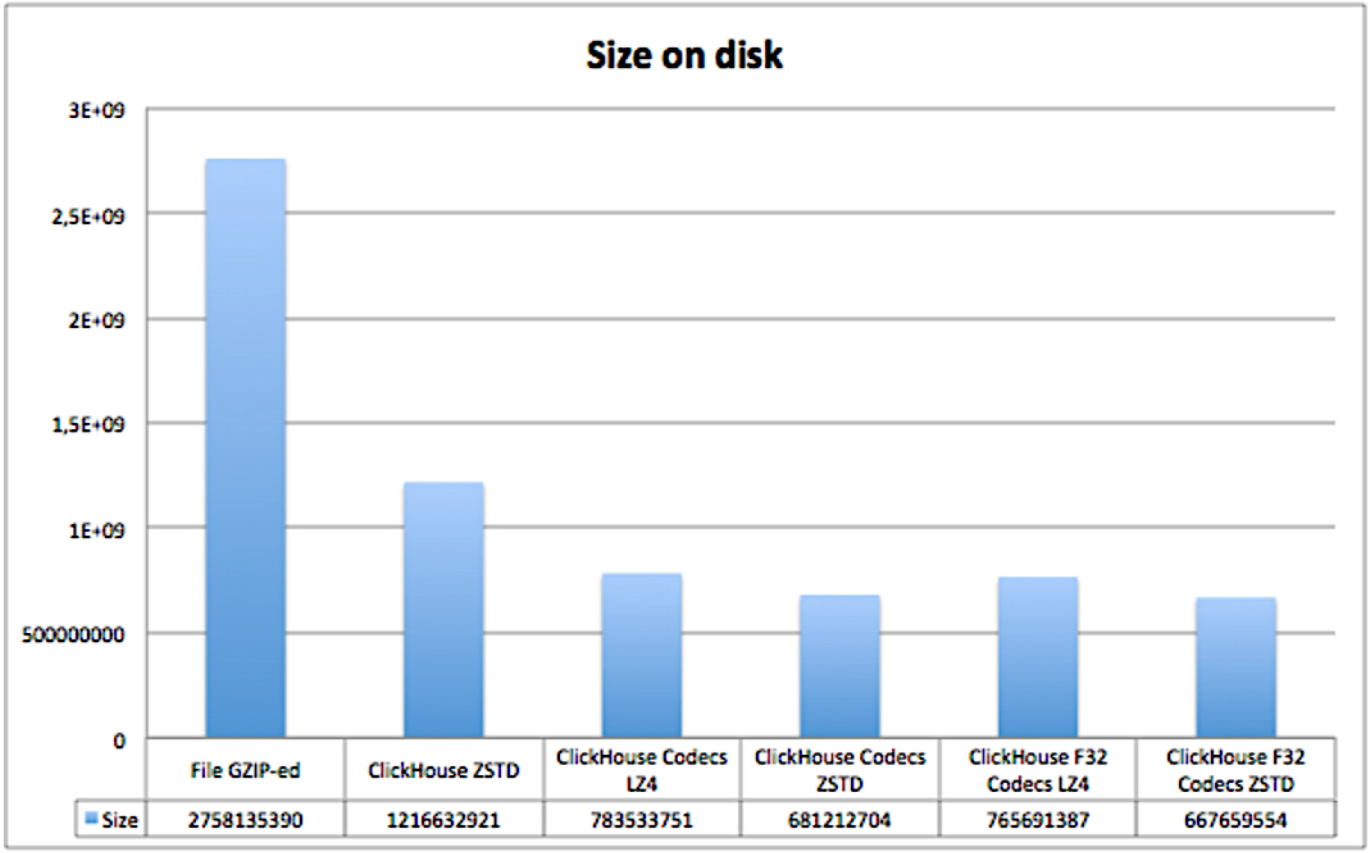
Кодеки в ClickHouse

- Delta
- DoubleDelta
- Gorilla
- T64
- Компрессоры LZ4 и ZSTD
- Композиция кодеков

Кодеки в ClickHouse

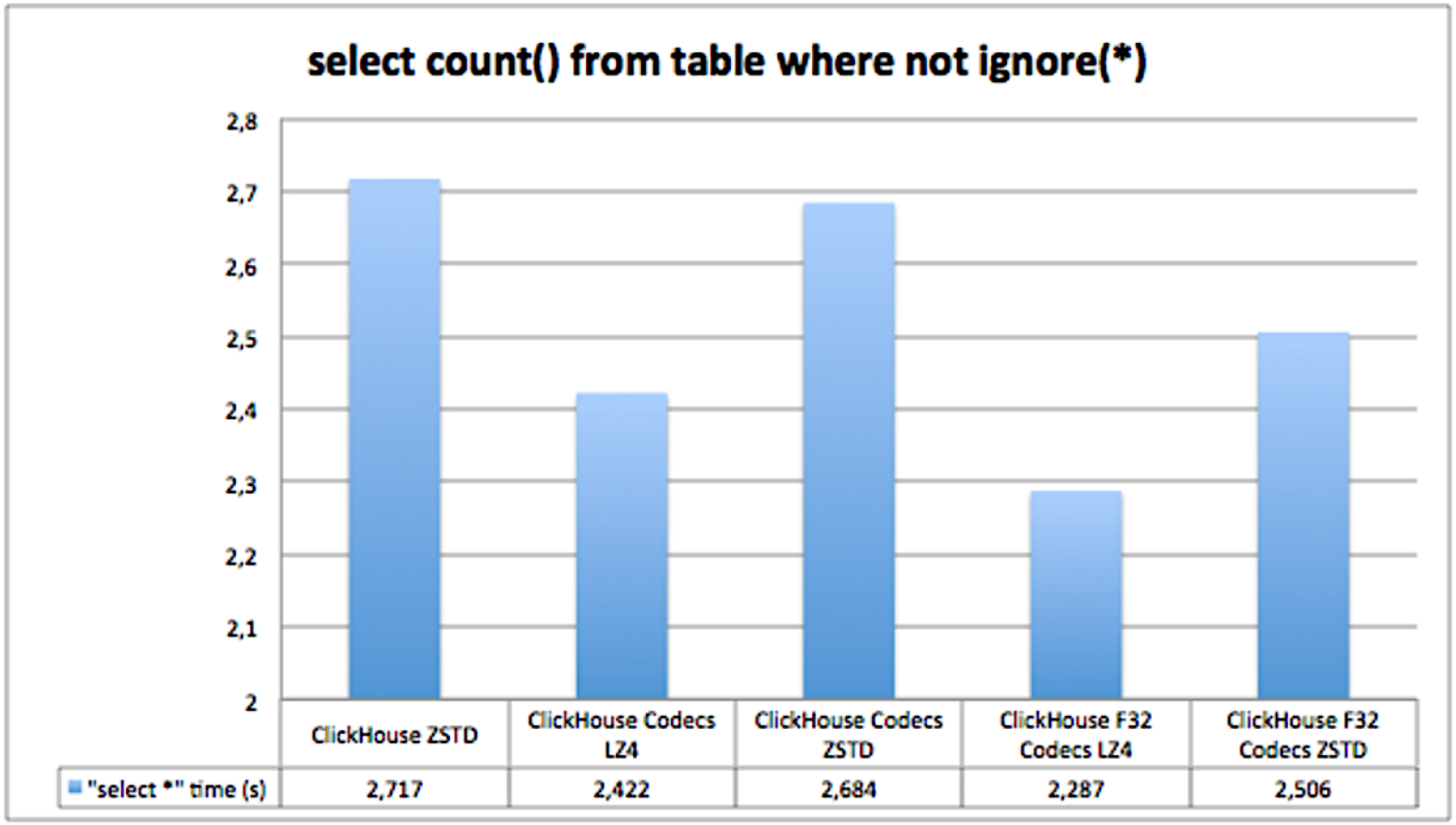
```
CREATE TABLE benchmark.cpu_codecs_lz4 (
    created_date Date DEFAULT today(),
    created_at DateTime DEFAULT now() Codec(DoubleDelta, LZ4),
    tags_id UInt32,
    usage_user Float64 Codec(Gorilla, LZ4),
    usage_system Float64 Codec(Gorilla, LZ4),
    usage_idle Float64 Codec(Gorilla, LZ4),
    usage_nice Float64 Codec(Gorilla, LZ4),
    usage_iowait Float64 Codec(Gorilla, LZ4),
    usage_irq Float64 Codec(Gorilla, LZ4),
    usage_softirq Float64 Codec(Gorilla, LZ4),
    usage_steal Float64 Codec(Gorilla, LZ4),
    usage_guest Float64 Codec(Gorilla, LZ4),
    usage_guest_nice Float64 Codec(Gorilla, LZ4),
    additional_tags String DEFAULT ''
)
ENGINE = MergeTree(created_date, (tags_id, created_at), 8192);
```

Кодеки в ClickHouse



InfluxDB:
456MB :-/

Кодеки в ClickHouse

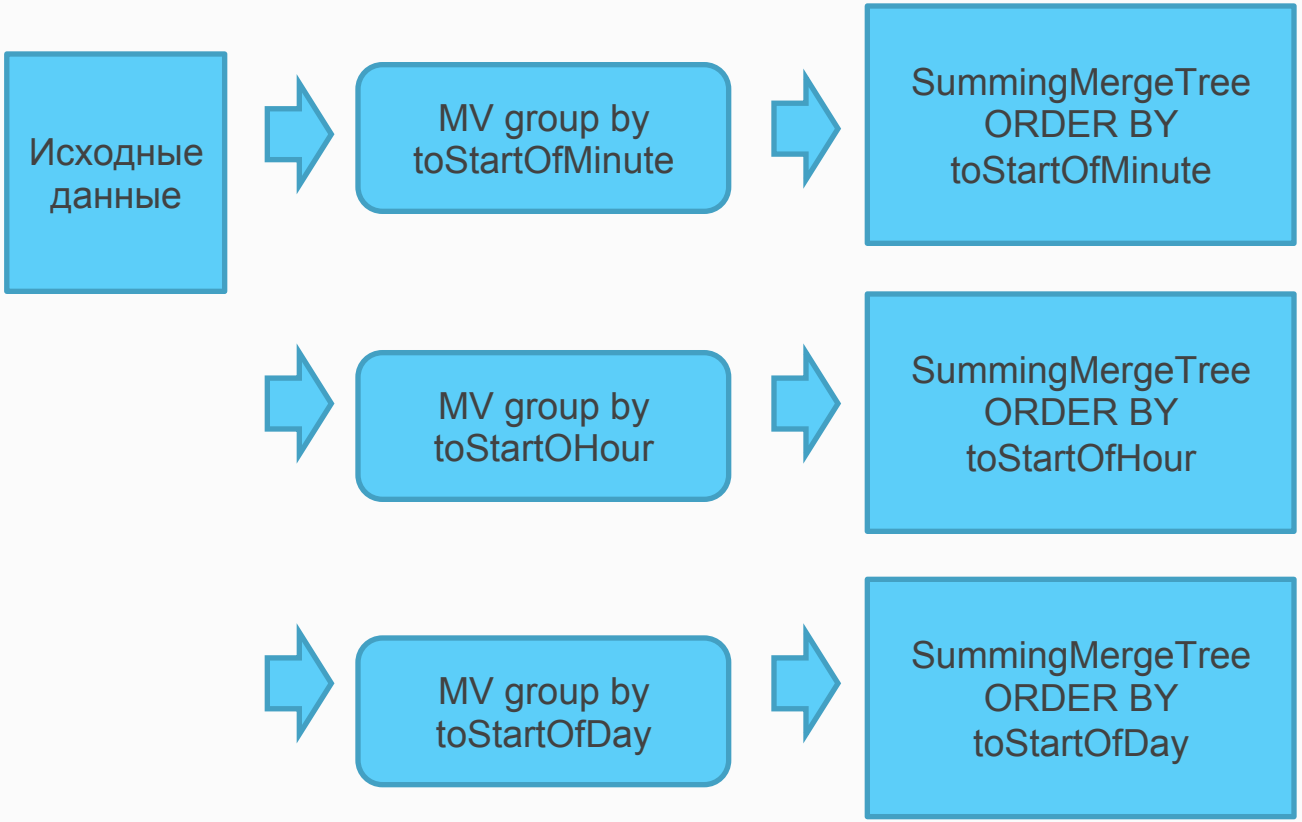


Кодеки в ClickHouse. Итого

- Кодеки – это хорошо (используйте 19.11.7 или выше)
- Есть проблемы со скоростью битового кодирования (DoubleDelta особенно)
- Все еще не так хорошо, как InfluxDB
- Можно сделать лучше, некоторые идеи:
  - Кодировать фреймами (как в middle-out)
  - Приводить к целочисленным типам (как VictoriaMetrics)
  - Не делать в битовое кодирование, оставить это ZSTD

Детали: https://github.com/yandex/clickhouse-presentations/blob/master/meetup26/time_series.pdf

Агрегация



- Агрегация сумм и юников!
- Каскады с 19.14
- Нет возможности выбирать автоматически источник запроса

TTL – забываем старые данные

```
CREATE TABLE aggr_by_minute
…
TTL time + interval 1 day

CREATE TABLE aggr_by_day
…
TTL time + interval 30 day

CREATE TABLE aggr_by_week
…
/* no TTL */
```

Специфичные для time-series запросы

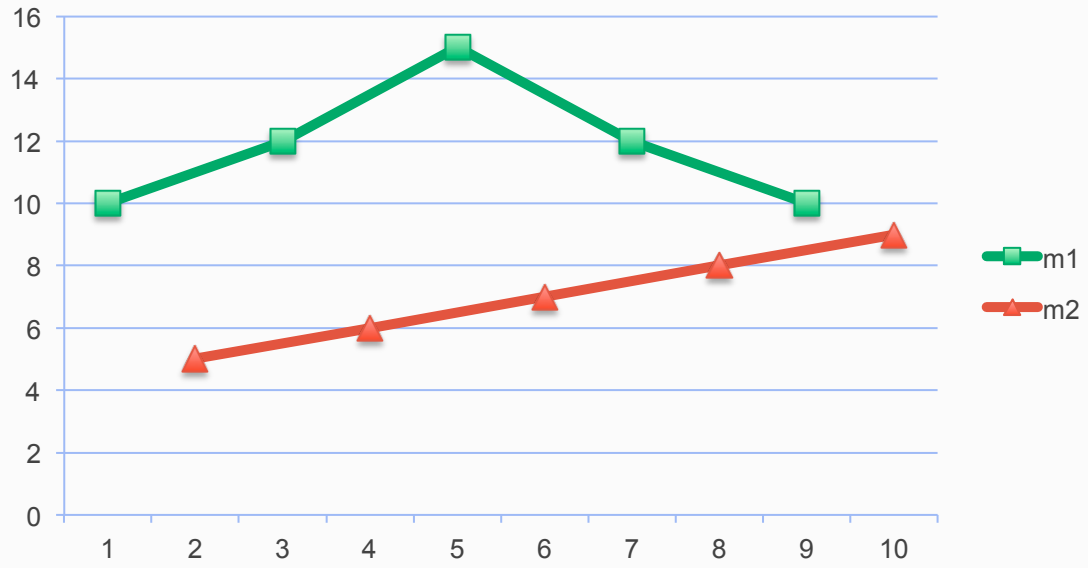
Вернуть последние значения каждого cpu

```
SELECT *
  FROM cpu
 WHERE (tags_id, created_at) IN
    (SELECT tags_id, max(created_at)
        FROM cpu
        GROUP BY tags_id)
```

```
SELECT
    argMax(usage_user), created_at),
    argMax(usage_system), created_at),
...
 FROM cpu
```

```
SELECT now() as created_at,
       cpu.*
  FROM (SELECT DISTINCT tags_id from cpu) base
  ASOF LEFT JOIN cpu USING (tags_id, created_at)
```

ASOF JOIN – «склеивание» рядов с разным временем




```
SELECT m1.*, m2.*
  FROM m1
  LEFT ASOF JOIN m2 USING (timestamp)
```

Аналитические фунции

```
SELECT origin,
       timestamp,
       timestamp -LAG(timestamp, 1) OVER (PARTITION BY origin ORDER BY
timestamp) AS duration,
       timestamp -MIN(timestamp) OVER (PARTITION BY origin ORDER BY
timestamp) AS startseq_duration,
       ROW_NUMBER() OVER (PARTITION BY origin ORDER BY timestamp) AS
sequence,
       COUNT() OVER (PARTITION BY origin ORDER BY timestamp) AS nb
  FROM mytable
ORDER BY origin, timestamp;
```

Это не ClickHouse!

Аналитические фунции. ClickHouse way.

```
SELECT
    origin,
    timestamp,
    duration,
    timestamp - ts_min AS startseq_duration,
    sequence,
    ts_cnt AS nb
FROM (
    SELECT
        origin,
        groupArray(timestamp) AS ts_a,
        arrayMap((x, y) -> (x - y), ts_a, arrayPushFront(arrayPopBack(ts_a), ts_a[1])) AS ts_diff,
        min(timestamp) as ts_min,
        arrayEnumerate(ts_a) AS ts_row,
        count() AS ts_cnt
    FROM mytable
    GROUP BY origin
)
ARRAY JOIN ts_a AS timestamp, ts_diff AS duration, ts_row AS sequence
ORDER BY origin, timestamp
```

НЕ очень просто и НЕ очень удобно, но работает

Специальные функции

- runningDifference, runningAccumulate, neighbor – в пределах блоках
- sumMap(key, value)
- timeSeriesGroupSum(uid, timestamp, value)
- timeSeriesGroupRateSum(uid, timestamp, value)
- skewPop, skewSamp, kurtPop, kurtSamp
- WITH FILL
- simpleLinearRegression, stochasticLinearRegression

Применения в других системах

- GraphHouse – замена бекенда в Graphite
- PromHouse – замена бекенда в Prometheus
- PMM – мониторинг производительности MySQL
- Apache Traffic Control – мониторинг CDN
- Сам ClickHouse – system.metric_log в 19.14!
- … и внутри многих неназываемых коммерческих систем

Заключение

- ClickHouse Rocks!
- Time-series работают не хуже специализированных DBMS
- Всегда есть, что улучшать:
  - Еще лучше кодеки
  - Более эффективная работа с массивами
  - Оконные функции или их эффективная замена
  - Автоматический выбор таблиц в случае даунсемплинга

Questions?

Thank you!

P.S. We are hiring!

Contacts:
info@altinity.com

Visit us at:
https://www.altinity.com

Read Our Blog:
https://www.altinity.com/blog