

Yandex for `developers` *//>

YATALKS

ClickHouse: настоящее и будущее

Бэкенд

Алексей Миловидов

Я расскажу

01 Почему ClickHouse — хорошая система

02 Почему ClickHouse — плохая система

03 И что с этим делать





ClickHouse — хорошая система



Это более оптимально

ClickHouse не тормозит

- Система создана из практических задач, для работы в бою
- Разработана «снизу-вверх» исходя из конкретных сценариев
- Внимание к деталям и специализация под сценарии нагрузки

Доклад «секреты оптимизации производительности ClickHouse»
youtube.com/watch?v=ltg8vstuHUU



ClickHouse — надёжная система

- Кросс-ДЦ master-master репликация
- Надёжная запись и хранение данных
- Защита от сбоев железа
- Защита от ошибок пользователя
- Защита от ошибок конфигурации
- Шифрование трафика и хранимых данных, аутентификация
- Все доступные средства тестирования в CI

ClickHouse — удобная система

Язык SQL, адаптированный
для удобства аналитики:

- Алиасы в любом месте запроса
- Массивы, кортежи, лямбда функции
- Комбинаторы агрегатных функций
- LIMIT BY, ASOF JOIN, ANY/SEMI JOIN, argMin/argMax

Функции для предметной
области из коробки:

- Click-stream: функции обработки URL и IP-адресов
- Performance monitoring: квантили
- Geospatial: geoDistance, pointInPolygon, H3, S2

ClickHouse — гибкая система



ClickHouse — доступная система

ClickHouse можно развернуть:

- На своих серверах
- В облаках; с Kubernetes
- На инфраструктуре заказчика
- На личном ноутбуке

ClickHouse доступен под разные платформы:

- x86_64, aarch64 (ARM), PowerPC 64, RISC-V
- Linux, FreeBSD, macOS

ClickHouse — настоящий open-source

- Исходники доступны публично
- Патчи от сообщества принимаются
- Открытые процессы разработки
- Низкий порог входа для контрибьюторов
- Максимальное поощрение и вовлечение сообщества

Доклад «как организовать живое сообщество вокруг open-source продукта»
youtube.com/watch?v=xddKLojmkus&t=4165s



ClickHouse — плохая* система



Это не оптимально

* — не идеальная.

Репликация требует ZooKeeper

ZooKeeper — отдельный от ClickHouse компонент, написанный на Java, требующий тщательной настройки и отдельных серверов.

ZooKeeper уходит!

1. Clickhouse-keeper — 100% совместимый с ZooKeeper по протоколу и модели данных
 - Compressed logs and snapshots
 - No issues with zxid overflow
 - No issues with large packets
 - Better memory usage
 - No issues with GC and Java heap
2. Может запускаться встроенным в clickhouse-server
 - Нет необходимости в отдельном сервисе

Данные необходимо вставлять пачками

- Можно вставлять **миллионы строк в секунду** на каждый сервер
- Но эти строки должны быть всего лишь **в нескольких пачках в секунду**
- Можно использовать Kafka или RabbitMQ таблицы

Асинхронные INSERT запросы

- Возможность делать много частых INSERT
- Из множества параллельных соединений
- Без Kafka и без Buffer таблиц!
- Множество мелких INSERTs комбинируются вместе в одну пачку в оперативной памяти
- Вставки надёжные по-умолчанию: клиент получает ответ, когда данные записаны в таблицу

Уже в продакшене!

Разработчики: Anton Popov, Ivan Lezhankin.

Отсутствие поддержки транзакций

Зачем в ClickHouse транзакции?

- Для атомарной вставки в несколько таблиц и представлений
- Для атомарной вставки на кластер
- Для выполнения множества SELECT из одного снимка

Недостаточная совместимость SQL

- Язык SQL в ClickHouse изначально сделан нестандартным для удобства
- Из-за сложного механизма разрешения имён и типов, запросы сложно анализировать

Есть способ поддержать все возможности стандарта и сохранить все расширения ClickHouse!

2021: Window Functions, ANY/ALL, EXISTS, GROUPING SETS...

2022: Correlated Subqueries

Отсутствие оптимизаций JOIN

- Не учитывается сортировка таблицы для JOIN
- Нет cost based optimizer для переупорядочивания JOIN
- Нет grace hash алгоритма для JOIN
- Нет shuffle для распределённых JOIN
- И вообще распределённые JOIN плохо работают

Отсутствие UPSERT

- Отсутствие точечных UPDATE и DELETE, а также UNIQUE KEY CONSTRAINT
- Реализовать unique key в распределённой системе — нетривиальная задача

Сложность масштабирования

- ClickHouse прекрасно масштабируется до тысяч серверов и 100 ПБ данных
- Но изменение числа серверов в кластере — **боль**
- Нужно заботиться о конфигурации шардов и реплик.
- Перешардирование данных **осуществляется вручную**
- Решение — cloud-native ClickHouse.

Кстати, а что это значит?

Сложность разделения ресурсов

- Разделение CPU и IO между запросами
- Приоритеты запросов
- Memory overcommit

Недостаточные возможности по интеграции

- Нет родного UI для ClickHouse
- Не хватает официальных интеграций с BI и ETL

Недостаточная известность в США и Европе

- Недостаточно развитая документация
- Отсутствие обучающих материалов, курсов и поддержки

И что с этим делать?



Компания ClickHouse, Inc

- Создана вместе с Яндексом
- Уже привлекли 300 млн \$
при оценке 2 млрд

Направления работы ClickHouse Inc

1. Создать облачный сервис для ClickHouse в serverless формате с динамическим масштабированием
2. Развитие и поддержка ClickHouse в open-source с целью увеличения размера рынка ClickHouse
3. Исследования и эксперименты для поиска новых ниш и возможностей ClickHouse

Новые горизонты для ClickHouse

- Поддержка полуструктурированных данных
- Функции обработки текста на естественном языке
- Поточковые запросы и complex event processing
- Key-value витрины данных, инкрементальная агрегация в оперативке
- Выполнение запросов с использованием GPU
- Интеграция с ML & AI. Обработка графов
- Batch jobs
- Data Hub

Support For Semistructured Data

JSO data type:

```
CREATE TABLE games (data JSON) ENGINE = MergeTree;
```

- You can insert arbitrary nested JSONs
- Types are automatically inferred on INSERT and merge
- Data is stored in columnar format: columns and subcolumns
- Query nested data naturally

Support For Semistructured Data

Example: NBA games dataset

```
CREATE TABLE games (data String)
ENGINE = MergeTree ORDER BY tuple();

SELECT JSONExtractString(data, 'teams', 1, 'name')
FROM games;
```

— 0.520 sec.

```
CREATE TABLE games (data JSON)
ENGINE = MergeTree;
SELECT data.teams.name[1] FROM games;
```

— 0.015 sec.

Support For Semistructured Data

```
DESCRIBE TABLE games
SETTINGS describe_extend_object_types = 1

name: data
type: Tuple(                                     <-- inferred type
  `_id.$oid` String,
  `_date.$date` String,
  `teams.abbreviation` Array(String),
  `teams.city` Array(String),
  `teams.home` Array(UInt8),
  `teams.name` Array(String),
  `teams.players.ast` Array(Array(Int8)),
  `teams.players.blk` Array(Array(Int8)),
  `teams.players.drb` Array(Array(Int8)),
  `teams.players.fg` Array(Array(Int8)),
  `teams.players.fg3` Array(Array(Int8)),
  `teams.players.fg3_pct` Array(Array(String)),
  `teams.players.fg3a` Array(Array(Int8)),
  `teams.players.fg_pct` Array(Array(String)),
  `teams.players.fga` Array(Array(Int8)),
  `teams.players.ft` Array(Array(Int8)),
  `teams.players.ft_pct` Array(Array(String)),
  `teams.players.fta` Array(Array(Int8)),
  `teams.players.mp` Array(Array(String)),
  `teams.players.orb` Array(Array(Int8)),
  `teams.players.pf` Array(Array(Int8)),
  `teams.players.player` Array(Array(String)),
```


Support For Semistructured Data

- Flexible schema
- You can have columns with strict and flexible schema in one table
- Queries work **as fast as** with predefined types!

Planned for Q1 2022.

Developer: Anton Popov.

Выводы

Вместе с новой компанией
и open-source сообществом
мы сделаем ClickHouse
лучшей аналитической
СУБД в мире!

Yandex for `developers` *//>

YATALKS

Спасибо

Бэкенд

Алексей Миловидов

СТО

milovidov@clickhouse.com