

ClickHouse

Александр Смирнов

17.09.2020

Сценарии работы с данными

- ▶ Какие производятся запросы, как часто и в каком соотношении
- ▶ Сколько читается данных на запросы каждого вида - строк, столбцов, байт
- ▶ Как соотносятся чтения и обновления данных
- ▶ Какой рабочий размер данных и насколько локально он используется
- ▶ Какие требования к дублированию данных и логической целостности
- ▶ Требования к задержкам на выполнение и пропускной способности запросов каждого вида

Сценарии работы с данными (2)

- ▶ Не существует системы, одинаково хорошо подходящей под существенно различные сценарии работы
- ▶ Если система подходит под широкое множество сценариев работы, то при достаточно большой нагрузке, система будет справляться со всеми сценариями работы плохо, или справляться хорошо только с одним из сценариев работы

OLAP – интерактивная аналитическая обработка

- ▶ Подавляющее большинство запросов – на чтение
- ▶ Данные обновляются достаточно большими пачками (> 1000 строк), а не по одной строке, или не обновляются вообще
- ▶ Данные добавляются в БД, но не изменяются
- ▶ При чтении, вынимается достаточно большое количество строк из БД, но только небольшое подмножество столбцов
- ▶ Таблицы являются широкими, то есть, содержат большое количество столбцов
- ▶ При выполнении простых запросов, допустимы задержки в районе 50 мс
- ▶ Результат выполнения запроса существенно меньше исходных данных – то есть, данные фильтруются или агрегируются, результат выполнения помещается в оперативку на одном сервере

ClickHouse

- ▶ Распределенная аналитическая столбцовая (column-oriented) СУБД

Почему column-oriented?

- ▶ В обычной row-oriented СУБД данные хранятся по строкам
- ▶ Хорошо подходит для обычных сценариев
 - ▶ изменить поле у пользователя
- ▶ Плохо подходит для задачи аналитики
 - ▶ пусть много столбцов, e.g. время, браузер, модель телефона
 - ▶ строки — события, e.g. просмотр страницы
 - ▶ нужен отчет по моделям телефона

Почему column-oriented? (2)

- ▶ Подходит система где данные хранятся по столбцам
- ▶ Данные, лежащие по столбцам, лучше сжимаются (однородные)

Преимущества

- ▶ Скорость
- ▶ Гибкость
 - ▶ доступны все данные, что хотим то и делаем с ними
- ▶ Масштабируемость
 - ▶ распределенное хранение данных
- ▶ Цена
 - ▶ цена разработчиков, не дорого

Недостатки

- ▶ Ограниченная поддержка JOIN
 - ▶ не больше одного подзапроса
- ▶ Нет UPDATE и DELETE
- ▶ Слабая совместимость со стандартом SQL

Как хранятся данные в таблицах

- ▶ ClickHouse – модульная система
- ▶ Разрабатывается снизу вверх
 - ▶ сначала работает на одной локальной машине
 - ▶ распределённая система собирается из множества локальных систем
- ▶ Поддерживает работу различных Storage Engines

Индекс БД

- ▶ Объект базы данных, создаваемый с целью повышения производительности поиска данных
- ▶ Часто реализуется В-деревом
 - ▶ хотим быстро доставать данные по идентификатору пользователя
 - ▶ отображение из значений ключа (e.g. идентификатор пользователя) в смещения на диске
 - ▶ В-деревья – дерево поиска с широкими node-ми

MySQL

- ▶ **Запись данных**
 - ▶ данные поступают по времени
 - ▶ записываются в конец
 - ▶ поступают перемешанно со всех сайтов
 - ▶ записи хранятся друг за другом последовательно
 - ▶ есть B-дерево, которое эти записи адресует по ключу
- ▶ **Чтение данных**
 - ▶ выбрать данные по одному сайту за период времени
 - ▶ ключ будет составным: идентификатор сайта, дата
 - ▶ по дереву вынимаем пачку смещений
 - ▶ выборка будет размазана по всем данным
- ▶ **Достоинство: просто**
- ▶ **Недостаток: очень долго, данные расположены не оптимально для чтения**

Log-structured merge-tree

- ▶ Заведем clustered index по сайтам, чтобы данные были физически упорядочены
- ▶ Вместо того, чтобы сразу записывать в файл, будем хранить новые записи в оперативной памяти
- ▶ Когда оперативная память заканчивается, сливаем записи с записями на диске
- ▶ Предоставляет быстрый доступ по индексу в условиях частых запросов на вставку

Векторная обработка запросов

- ▶ Проблема: как обрабатывать сложные запросы
 - ▶ сделать интерпретатор выражений
 - ▶ в аналитической СУБД нужно быстро фильтровать и агрегировать миллиарды строк
 - ▶ будет тормозить по сри
- ▶ Решения
 - ▶ Кодогенерация для компиляции запроса
 - ▶ компилируем машинный код и применяем для каждой строки
 - ▶ Векторная обработка запроса
 - ▶ все операции пишутся не для отдельных значений, а для векторов
 - ▶ не только храним данные по столбцам, но и обрабатываем по столбцам

История

- ▶ Yandex.Metrika
- ▶ Данные по действиям пользователей на сайтах
 - ▶ как сохранить данные так, чтобы быстро строить отчеты
 - ▶ изначально предагрегировали данные, но пользователи хотели больше вариативности
- ▶ Пробовали разные столбцовые субд, не подошло

Распространение

- ▶ Open-source
- ▶ Есть своя небольшая ниша
- ▶ В России
 - ▶ Яндекс
 - ▶ Mail
 - ▶ СКБ Контур
 - ▶ Rambler
- ▶ За рубежом
 - ▶ Cloudfare
 - ▶ Wikimedia
 - ▶ Nvidia