ГРУППОВАЯ КУРСОВАЯ РАБОТА

(программный проект)

на тему

Оптимизация сортировки в ClickHouse

Выполнили студенты:
Гумеров Арслан Рустемович, группа 176, 3 курс
Кидрачев Альберт Тимурович, группа 176, 3 курс
Морозов Василий Игоревич, группа 176, 3 курс

Научный руководитель:
Руководитель группы разработки ClickHouse, Миловидов Алексей Николаевич

Основные задачи

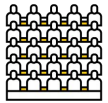
Изучение и реализация способов оптимизации сортировок в ClickHouse:

- Изучить и оптимизировать неэффективную обработку блоков данных при частичной сортировке
- Изучить реализацию сортировки колонок-кортежей и уменьшить в ней количество виртуальных вызовов
- Изучить и оптимизировать **Radix Sort**, избавившись от неэффективной работы с данными

А это нужно?

ClickHouse не тормозит

Одна из самых быстрых аналитических СУБД в мире

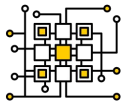


Большая фан-база

Большие требования

ORDER BY наше все

~ $4 \cdot 10^5$



Вундерваффе

Наш вклад

- Оптимизация обработки запросов с модификаторами ORDER BY & LIMIT

  Кидрачев Альберт, 176

- Оптимизация сортировки колонок-кортежей на основе идеи Radix Sort

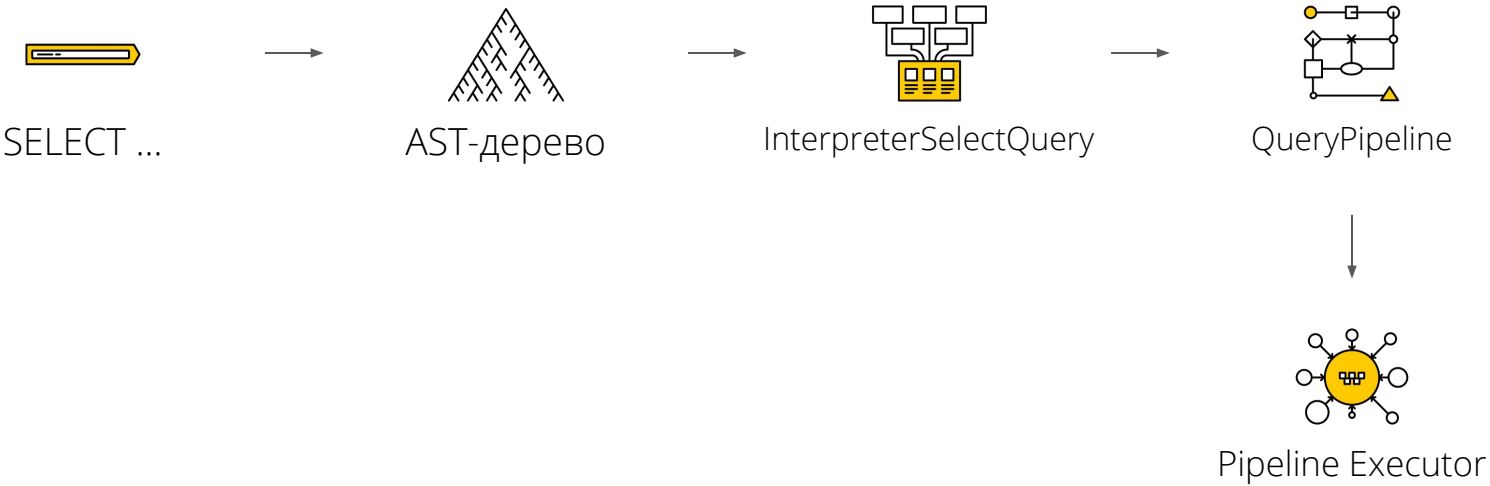
  Морозов Василий, 176

- Оптимизация реализации алгоритма сортировки LSD Radix Sort

  Гумеров Арслан, 176

Оптимизация сортировки с модификатором LIMIT

Обработка запросов в ClickHouse



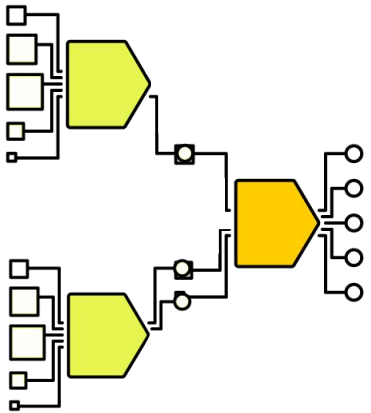
Да кто этот ваш Пайплайн?!

QueryPipeline - это ориентированный граф.

Вершины называются Процессорами.

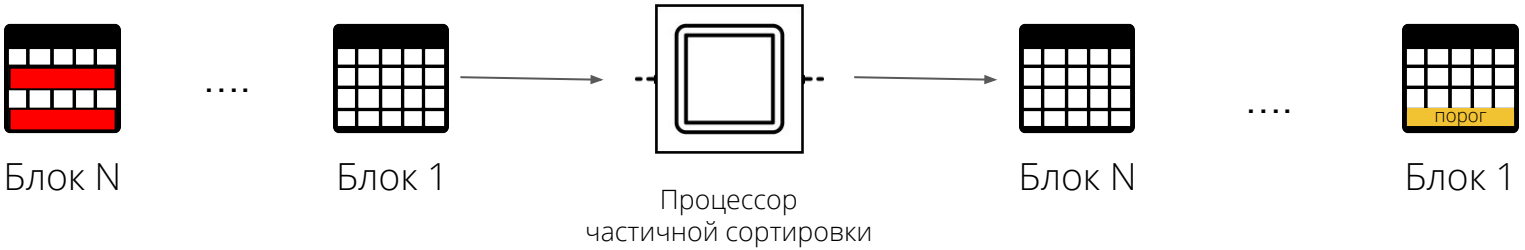
Данные между Процессорами передаются в Блоках.

Блок - множество колонок с метаинформацией о них.



Pipeline

PartialSortingTransform



Профит

На некотором классе запросов скорость обработки выросла на 10-60% согласно нагрузочным тестам ClickHouse.

Changes in performance

Old, s	New, s	Relative difference (new − old) / old	p < 0.001 threshold
0.71	0.229	-0.678	0.676
0.927	0.331	-0.643	0.635
0.7799	0.3319	-0.575	0.571
0.586	0.264	-0.55	0.547
0.425	0.1949	-0.542	0.538
0.483	0.238	-0.508	0.505
0.504	0.26	-0.485	0.48
0.062	0.091	0.467	0.451
0.362	0.194	-0.465	0.461
0.416	0.23	-0.448	0.444
0.065	0.093	0.43	0.415
0.185	0.118	-0.363	0.362
0.218	0.147	-0.326	0.316
0.709	0.577	-0.187	0.184
0.662	0.541	-0.183	0.181
0.853	0.734	-0.14	0.131
0.079	0.071	-0.102	0.101

Уменьшение количества виртуальных вызовов

Профит

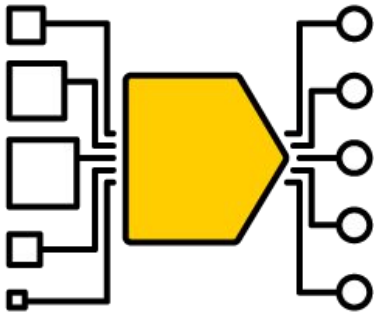
На некотором классе запросов зафиксирован рост производительности до 30%

Changes in performance

Old, s.	New, s.	Relative difference (new – old) / old	p < 0.001 threshold	Test
0.2513	0.1742	-0.307	0.305	logical_functions_small
0.4298	0.3242	-0.246	0.243	mingroupby-orderbylimit1
0.1238	0.0951	-0.232	0.228	joins_in_memory_pmj
0.1224	0.0955	-0.22	0.218	joins_in_memory_pmj
0.0653	0.0519	-0.206	0.195	parse_engine_file
0.344	0.286	-0.169	0.165	logical_functions_medium
0.3137	0.2653	-0.155	0.151	mingroupby-orderbylimit1
0.0849	0.0729	-0.142	0.14	string_sort
0.117	0.1034	-0.117	0.111	string_sort

Оптимизация сортировки LSD Radix Sort

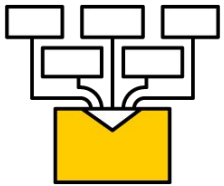
Устройство сортировок в ClickHouse



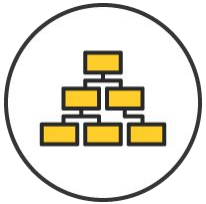
Сортировки вычисляют порядок, в котором данные будут отсортированы

Реализация в ClickHouse

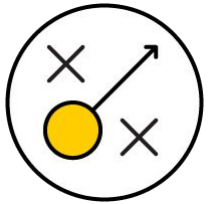
Нужны полные данные для сортировки



Данные склеиваются с исходным индексом



Сортируются в виде сложных элементов



После сортировки извлекается порядок

Выводы и результаты

- Избавились от лишнего копирования данных
- К сожалению, нагрузочные тесты не показали значительного прироста в производительности

Заключение

В ходе выполнение работы была частично изучена архитектура СУБД ClickHouse, а также реализовано несколько подходов в оптимизации сортировок

- Была оптимизирована сортировка кортежей
- При частичной сортировки данных были оптимизированы лишние сравнения для элементов, заведомо не попавших в ответ.
- Было убрано лишние копирование данных для оптимизации реализованной в ClickHouse LSD Radix Sort

В совокупности данные улучшения позволили ускорить выполнение некоторых классов запросов более чем на 60%

Спасибо за внимание