History of ClickHouse at Ahrefs

Yasunari Watanabe

ahrefs

Agenda

- **About us**
- Road to ClickHouse adoption
- Upstream patch: In-memory marks compression

ahrefs

About me

- Joined Ahrefs ~2 years ago
- Backend developer
- Working on our web crawler

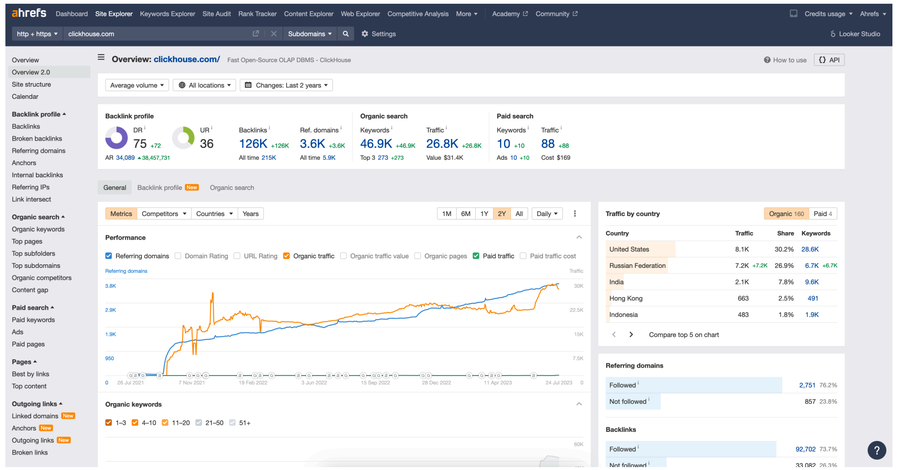
ahrefs

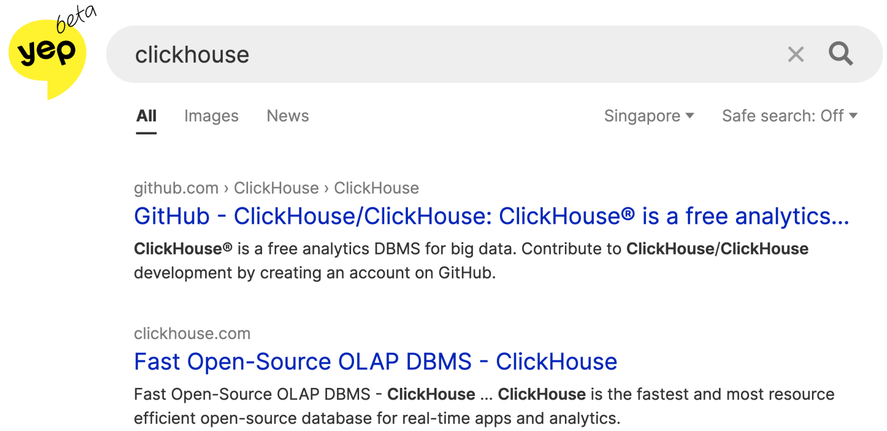
About Ahrefs: Who we are

- Founded in 2010, HQ in Singapore
- ~120 people, two thirds remote
- 50% engineers/data scientists
  - 19 backend developers, including 1 ClickHouse developer
  - 6 DevOps engineers

About Ahrefs: What we do

- Process big data about the Web and transform into relevant indicators
  - SEO metrics
  - Our new search engine, **Yep**





About Ahrefs: Data flow

Quick facts

- Most active web crawler in SEO industry
- World’s largest index of live backlinks
- Running on our own hardware

ahrefs.com/big-data

Every minute, we crawl
5M
Pages

There are
170T
Rows in our key-value database

External backlinks history
35.0T
Records

3300
Servers

596K
CPU cores

4PB
RAM

33PB
HDD

346PB
SSD

Pages in index
357.4B

Domains (post-vetting)
162.7M

Internal backlinks
24.7T

Agenda

- About us
- **Road to ClickHouse adoption**
- Upstream patch: In-memory marks compression

Early days

- Tried all available solutions (Cassandra, Hypertable); none fast enough
- Developed custom solution optimized for crawling the web with limited resources (tens of servers, circa 2010)
- QFS to handle unevenness among shards
- Elasticsearch for various non-crawler use cases

ahrefs

Worked well, but…

- Limitations of custom storage
  - Lack of versatile querying API
  - Lack of bells and whistles
- Evolving feature requirements, size of the Web, our infra…

ahrefs

Enter ClickHouse

- Started experimenting with ClickHouse in 2019
- Similarities in design, but ClickHouse has dedicated time + resources to features and optimizations
  - SQL interface
  - Many input/output formats
- Notable difference: column-based

ahrefs

Migration to ClickHouse

- Custom storage
  - Migrated part of the web crawler a few years ago
  - Rest of migration ongoing
  - Challenge: need to keep everything running during migration
- Elasticsearch
  - Most indices rebuilt in ClickHouse

ahrefs

ClickHouse deployment

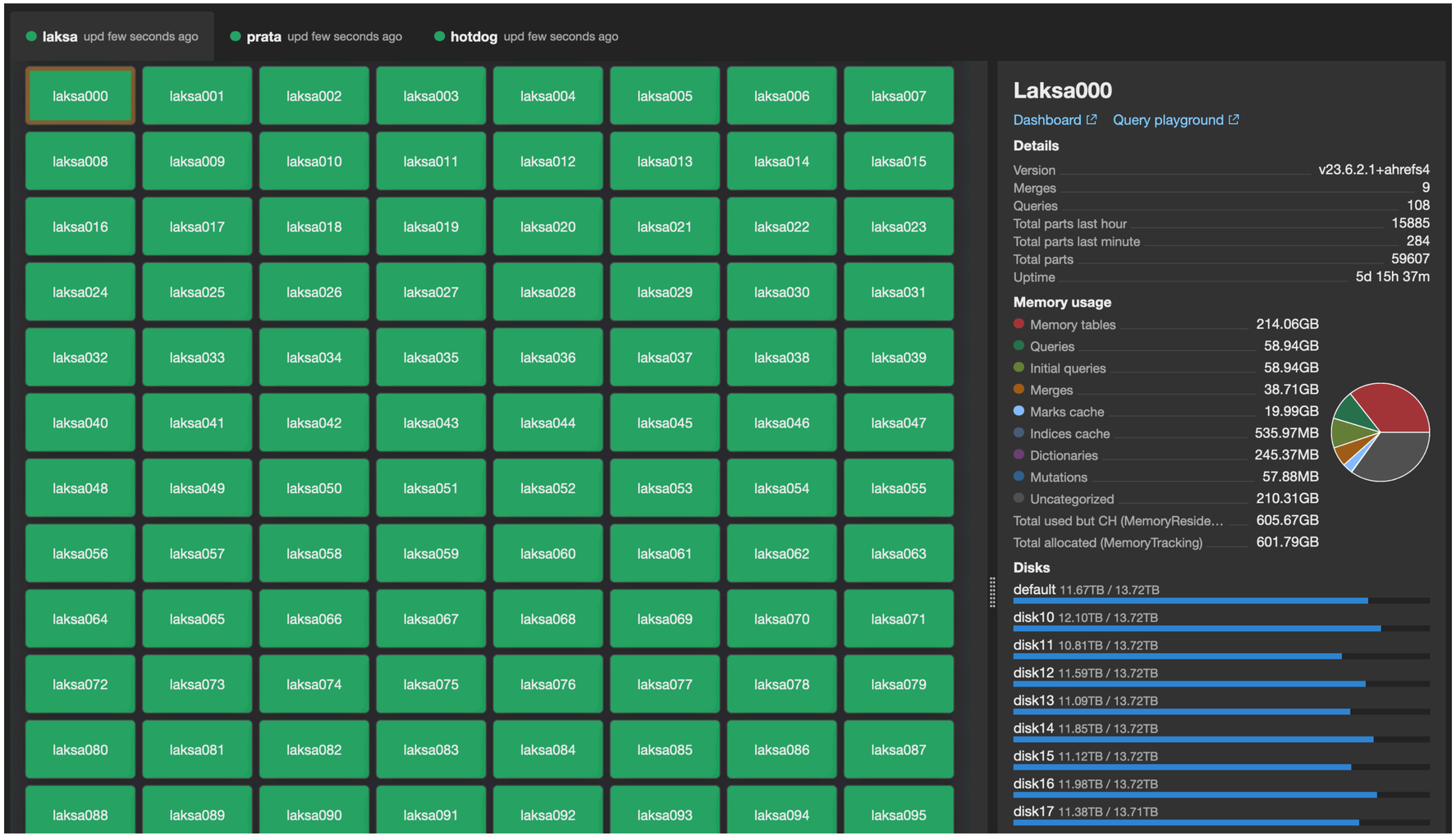
- Multiple clusters each with hundreds of hosts
- Geo-replicated main cluster, designated read/write replicas
- Large tables with trillions of rows, tens of columns
- More capacity to come

ahrefs

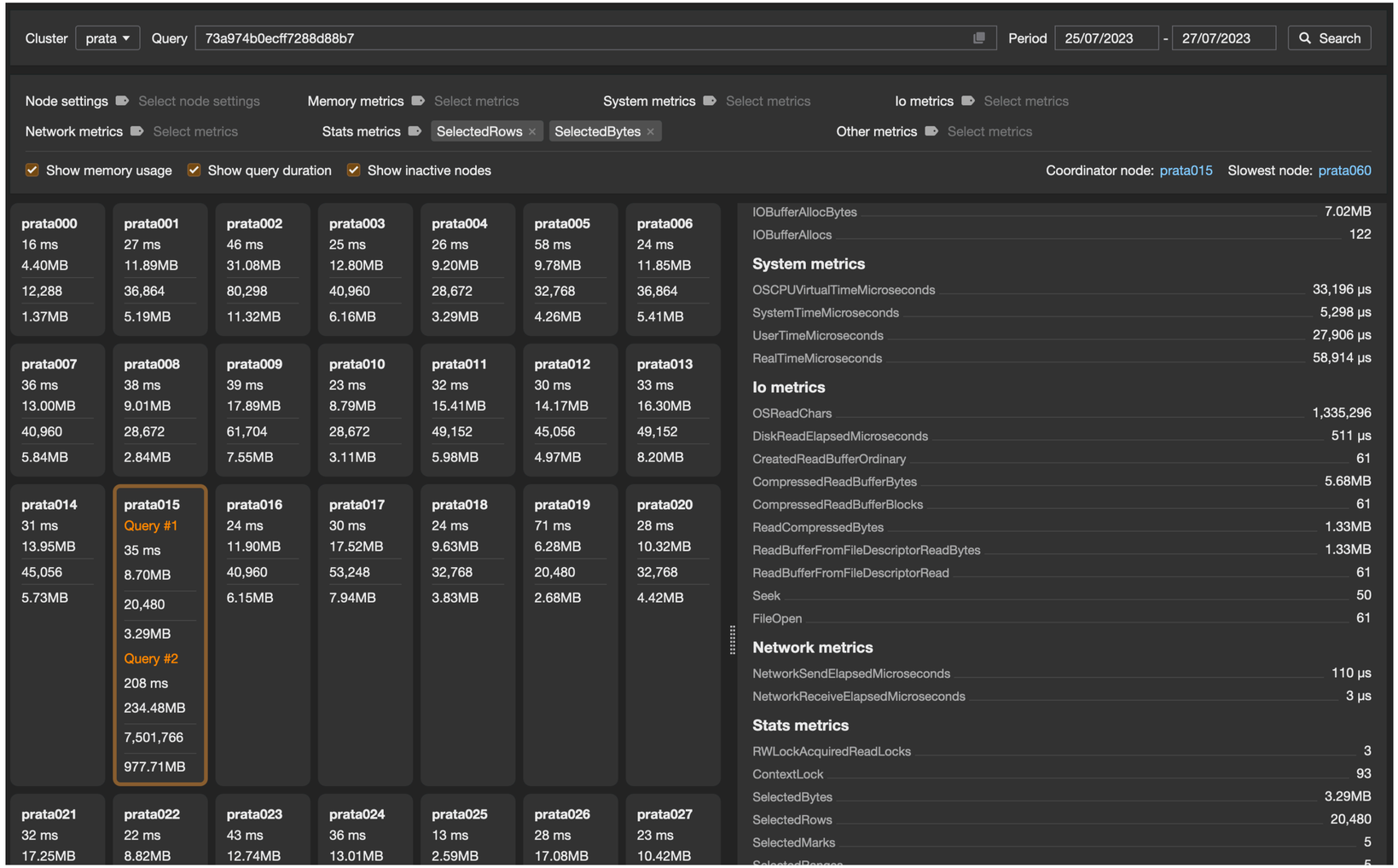
ClickHouse in our workflow

- Techniques
  - Optimizing insert: custom code for partitioned buffers
  - Fetch/attach to move parts between servers efficiently
- Heavy tooling on top of SQL interface
  - Metaprogramming for tight integration with application code (in OCaml)

Tools: Cluster monitoring



Tools: Query analyzer



ahrefs

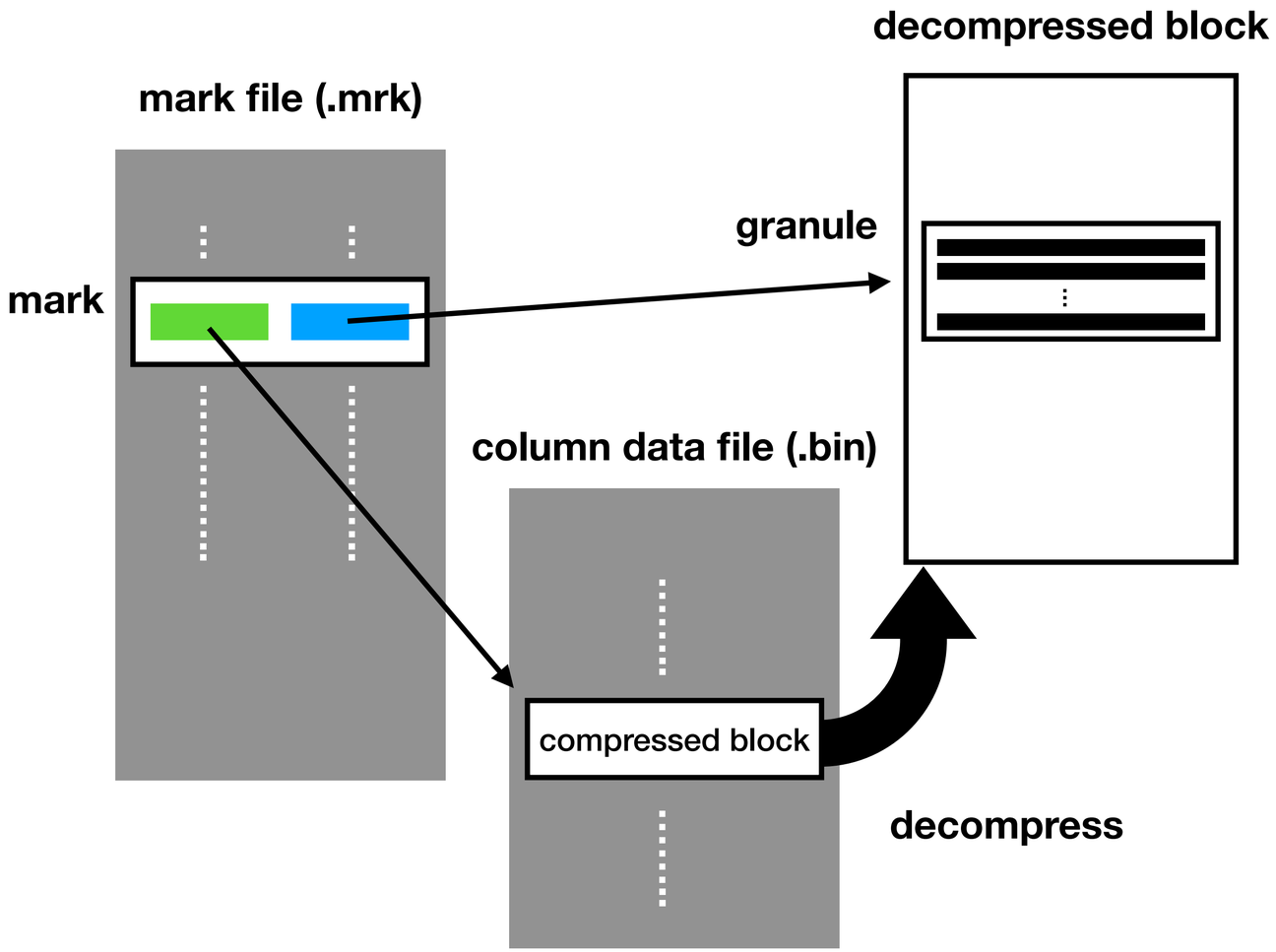
Agenda

- About us
- Road to ClickHouse adoption
- **Upstream patch: In-memory marks compression**

ahrefs

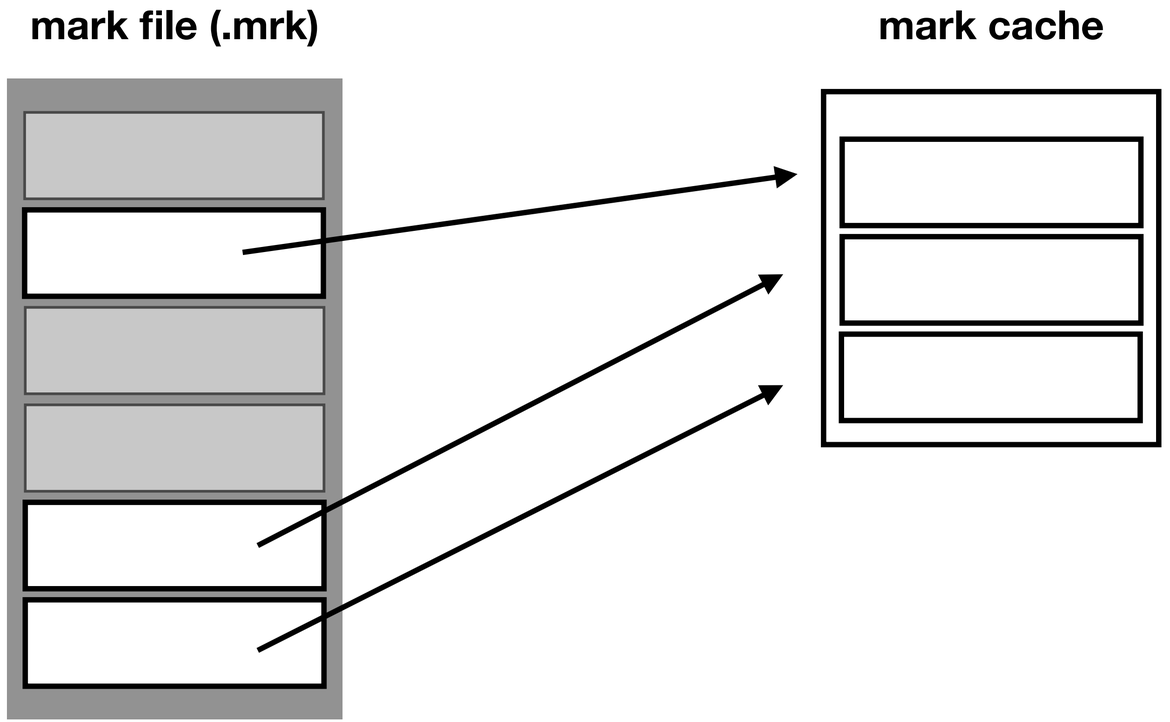
Background

- ClickHouse uses offset pairs (“marks”) to locate groups of rows in compressed data files
- Marks are stored in “mark files” and cached in memory



Problem

- Large queries read rows from many marks
  → poor cache performance
- At Ahrefs, want all marks to fit in memory for max performance
- Need more memory-efficient marks representation



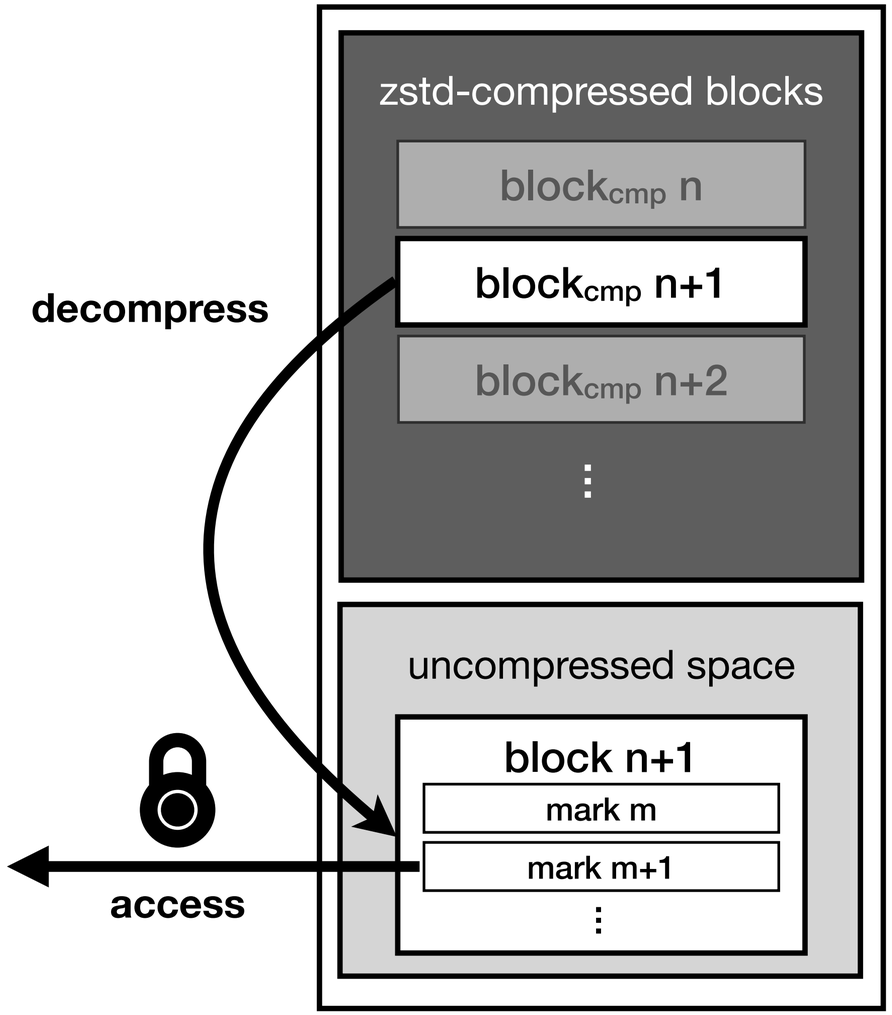
Insight

- The marks are suitable for compression

  - First offset is monotonically increasing
  - Second offset is almost always (but sometimes not) zero
  - But both stored as 64-bit integers (wasteful)

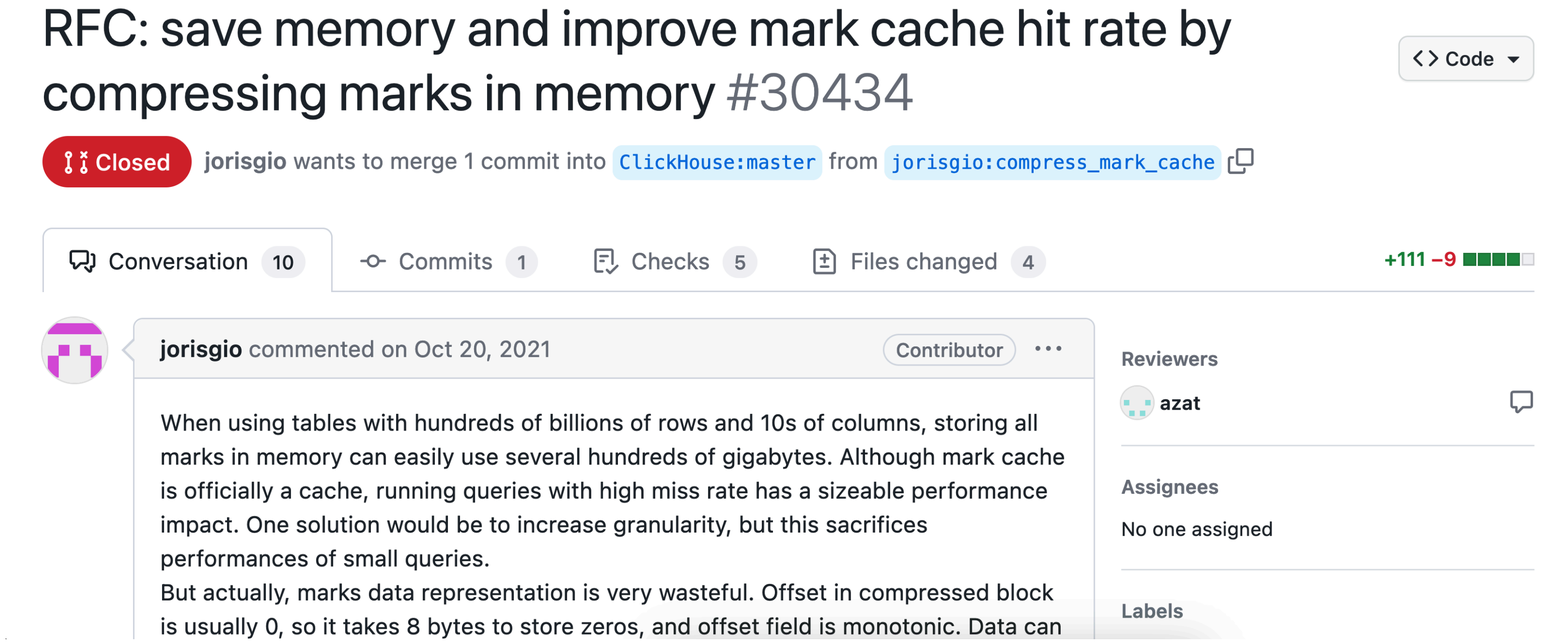
Early implementation at Ahrefs

- Split marks into blocks, store blocks compressed with zstd in memory
- Allocate space for one uncompressed block
- Guard access with mutex
  - Downside: contention if many threads
  - In practice, okay



Early implementation at Ahrefs

Oct 2021



RFC: save memory and improve mark cache hit rate by compressing marks in memory #30434

Code

Closed **jorisgio** wants to merge 1 commit into `ClickHouse:master` from `jorisgio:compress_mark_cache`

Conversation 10 · Commits 1 · Checks 5 · Files changed 4 · +111 −9

jorisgio commented on Oct 20, 2021 — Contributor

When using tables with hundreds of billions of rows and 10s of columns, storing all marks in memory can easily use several hundreds of gigabytes. Although mark cache is officially a cache, running queries with high miss rate has a sizeable performance impact. One solution would be to increase granularity, but this sacrifices performances of small queries.
But actually, marks data representation is very wasteful. Offset in compressed block is usually 0, so it takes 8 bytes to store zeros, and offset field is monotonic. Data can

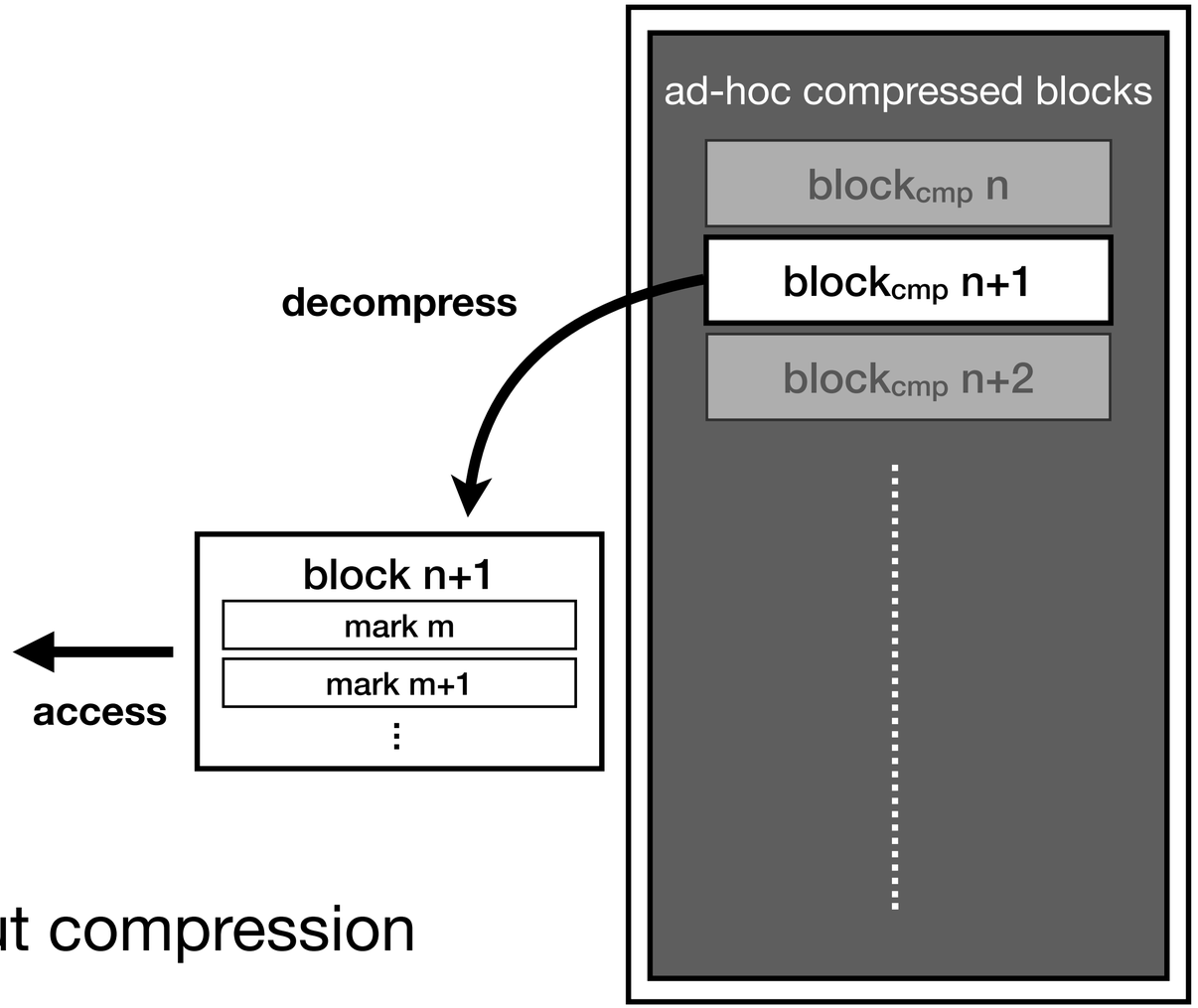
Reviewers
azat

Assignees
No one assigned

Labels

Eventual upstream solution

- Also split marks into blocks,
  but use ad-hoc compression
  - Store various deltas, bitpacked
- All in-memory marks compressed
  → no mutex
  - Downside: decompression
    for each mark access
- Result: 3-6x less memory than without compression



Eventual upstream solution

March 2023

New: Key-value interface

Feature discussion: fast key-value query over http #52194

Open **canhld94** opened this issue last week · 3 comments

canhld94 commented last week — Contributor

This is more like a continuation of #33581. Some have been done since #33581:

- An abstract class for key-value entity `IKeyValueEntity`
- `IKeyValueEntity` methods has been implemented for `EmbeddedRocksDB` and `Dictionary`

Assignees

No one assigned

Labels

feature

ClickHouse at Ahrefs

- Great success overall
- Performance that meets our usage demands
- Active feature development and bug fixes, regular monthly releases

We’re hiring!

ahrefs.com/jobs

- D/C++ developers
- OCaml developers
- Data scientists