A Brief Overview of Caching in ClickHouse

Kseniia Sumarokova, ClickHouse Core SWE

Caches – what and why?

- Reserved storage location that collects temporary data
- Faster storage → faster access to data

Caches – what and why?

- Reserved storage location that collects temporary data
- Faster storage → faster access to data

Examples:

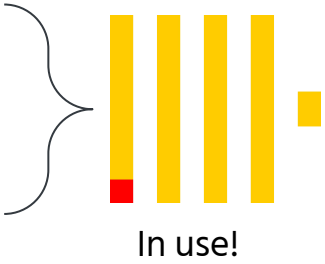
- CPU cache (L1, L2, L3)
- Page cache (memory)
- Hardware Disk cache
- DNS cache
- CDN
- Web Browser cache
- …

Caches – what and why?

- Reserved storage location that collects temporary data
- Faster storage → faster access to data

Examples:

- CPU cache (L1, L2, L3)
- Page cache (memory)
- Hardware Disk cache
- DNS cache
- CDN
- Web Browser cache
- …



In use!

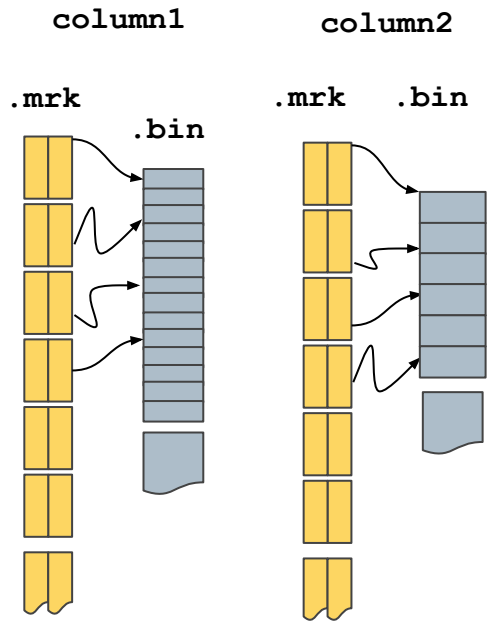
Marks Cache

– cache of marks used by table engines of Merge Tree family

- Caches a pair of offsets for each mark of each file:
  - offset in compressed data
  - offset after decompression
- Hash table + LRU eviction policy
- Hold space in cache includes:
  - Key: file path + mark number
  - Value: 2 * Int64

Settings:

- **mark_cache_size** (default: 5GiB)



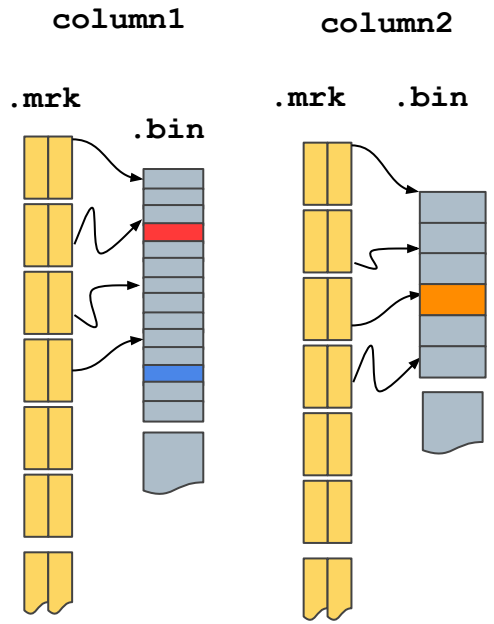
Uncompressed Cache

– cache of uncompressed blocks for table engines of Merge Tree family

- Caches uncompressed data of columns
- Can significantly reduce latency and increase throughput in case of frequent short queries
- Disables automatically for large queries
- Hash table + LRU eviction policy

Settings:

- **use_uncompressed_cache** (default: false)
- **uncompressed_cache_size** (default: 8GiB)



Metadata Cache

– cache of metadata file for table engines of Merge Tree family

- Caches data parts metadata files:
  - index, marks, checksums, rows counts, ...
- Significantly speed up server startup in case of huge amount of tables
- Persistent RocksDB storage on disk

Settings:

- **merge_tree_metadata_cache** (default: false)
- **lru_cache_size**
- **continue_if_corrupted** *(not so reliable :( )*

Remote Filesystem Cache

– cache for remote storages: S3, HDFS, Azure*

- Cache layer for the underlying data storage
- Faster access to remote storage
- File segments + Hash table + LRU eviction policy

Settings:

- **data_cache_enabled*** (default: false)
- **enable_filesystem_cache** (default: true)

Remote Filesystem Cache

– cache for remote storages: S3, HDFS, Azure*

- Cache layer for the underlying data storage
- Faster access to remote storage
- File segments + Hash table + LRU eviction policy

Settings:

- **data_cache_enabled*** (default: false)
- **enable_filesystem_cache** (default: true)

This is more optimal!

Query	With cache			Without cache		
SELECT UserID, toMinute(EventTime) AS m, :	x1.04 (20.905 s.)	(3.334 s.)	(3.184 s.)	(20.118 s.)	x5.96 (19.871 s.)	x6.11 (19.455 s.)
SELECT UserID FROM hits_1000m WHERE U	(0.806 s.)	(0.077 s.)	(0.091 s.)	x18.17 (14.645 s.)	x192.74 (14.841 s.)	x167.56 (15.248 s.)
SELECT count() FROM hits_1000m WHERE U	x1.04 (41.856 s.)	(0.987 s.)	(1.028 s.)	(40.414 s.)	x40.13 (39.613 s.)	x38.72 (39.804 s.)
SELECT SearchPhrase, any(URL), count() AS	(20.255 s.)	(0.965 s.)	(0.952 s.)	x2.48 (50.140 s.)	x50.75 (48.970 s.)	x50.99 (48.546 s.)

Compiled Expressions Cache

– JIT-compilation for simple functions and aggregate functions

- ClickHouse uses JIT for complex expressions and aggregation
- Improves performance
  - expression execution: 30%-200%
  - aggregation: 15%-200%
- Compilation can be slow – store compiled expressions in cache
- Hash table + LRU eviction policy

Settings:

- **compile_expressions** (default: true)
- **compile_aggregate_expressions** (default: true)
- **min_count_to_compile** (default: 3)

DNS Cache

– internal cache of DNS lookup

- Caches resolved IP addresses
- Reason: DNS can be slow (seconds for each request)
- Recommended for operating ClickHouse in systems with frequently changing infrastructure such as Kubernetes

Settings:

- **disable_internal_dns_cache** (default: false)
- **dns_cache_update_period** (in seconds, default: 15)

Opened Files Cache

– cache for opened file descriptors

- Cached opened file descriptors
- Allows to share file descriptors when doing reading with `pread` syscalls, opened for reading
- Open / close of files is very cheap on Linux, the purpose is to decrease the chance exhausting opened files limit

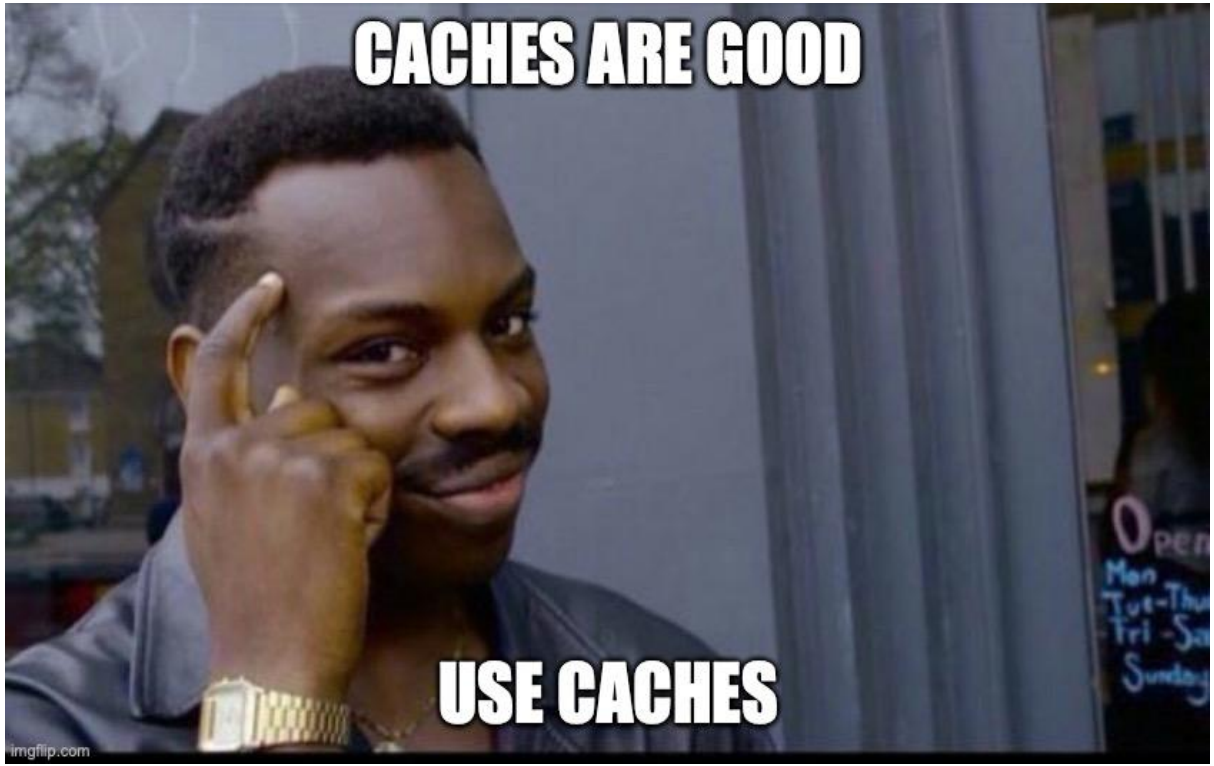
Settings:

- **local_filesystem_read_method** = `pread` or `pread_threadpool`

Future work

- **Schema Inference cache** (author: Pavel Kruglov)
  - Cache for table functions which use schema inference: S3, HDFS, File, ...
  - Cache is verified by file modification time
  - Already implemented, available in the next release
- **Query results cache**
- **External table functions, engines cache**
  - Cache for S3, HDFS, Hive table functions and table engines
  - Cache is verified by file modification time

Conclusion



Thanks for attention!

Questions?