Yerevan, Armenia
Aug 28, 2025

PG ARMENIA X PERCONA UNIVERSITY

How to Find Bottlenecks in ClickHouse

Eugene Klimov

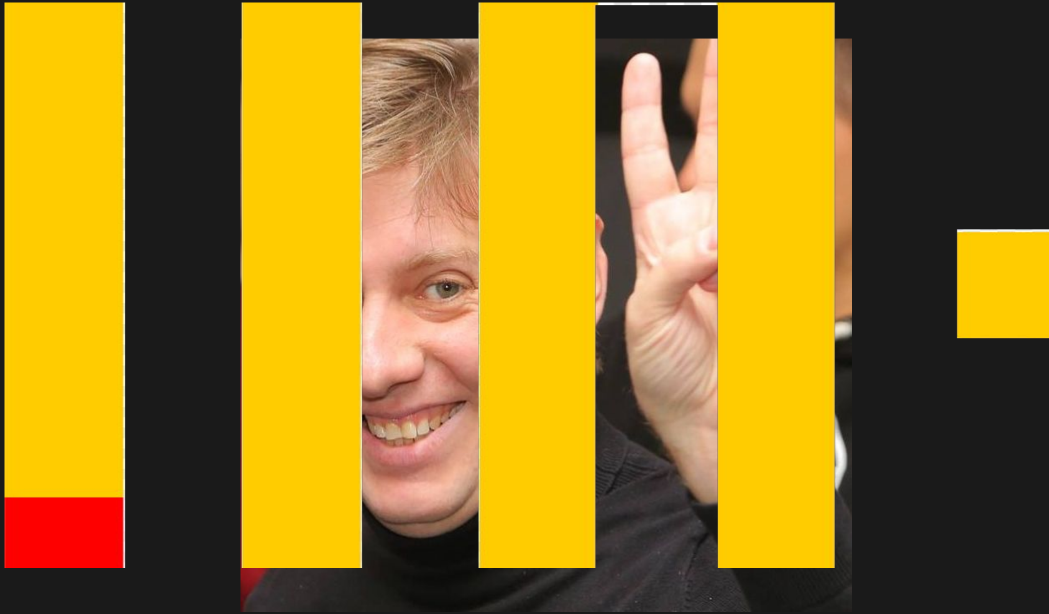
Clickhouse Software Engineer at Altinity

Who am I, and why I talking about this?

- I've been cleaning up some IT shit since the previous millennium. ;)
- Last 5 years in 24x7x365 mostly ClickHouse

Maintainer/Contributor

- clickhouse-backup
- clickhouse-grafana
- clickhouse-operator
- altinity-mcp



https://github.com/Slach

ClickHouse Quiz: Raise your hand!

1. Who knows what ClickHouse is?
2. Who works with ClickHouse right now?
3. Who has compressed data more than 10TB right now?

Common usage of resources: IO, CPU, RAM, NET.

ClickHouse is a data excavator: for each query, it tries to allocate the maximum allowable resources to process your query for maximum efficiency.

Spike workloads. Many concurrent queries will struggle for resources.

Background merges, message broker streaming, and dictionaries loading also allocate resources.

Don't mix clickhouse with other software components in production (including keeper) on the same hardware.

Recommendations of proportions: 10 TB of compressed data per server with 128 GB RAM and 32 CPU cores.

10 Gbit network is always better than a 1 Gbit network. ;)

ClickHouse store a lot of internal telemetry

```
# metrics related
metrics
events
metric_log
asynchronous_metrics
asynchronous_metric_log

dimensional_metrics
histogram_metrics

latency_buckets
latency_log

# stack related
trace_log
stack_trace

# logs
text_log

errors
warnings
```

```
# runtime
processes
merges
moves
mutations

user_processes
view_refreshes
dictionaries

asynchronous_inserts
asynchronous_loader

query_cache
query_condition_cache


# data streaming
azure_queue
s3queue
kafka_consumers
iceberg_history
```

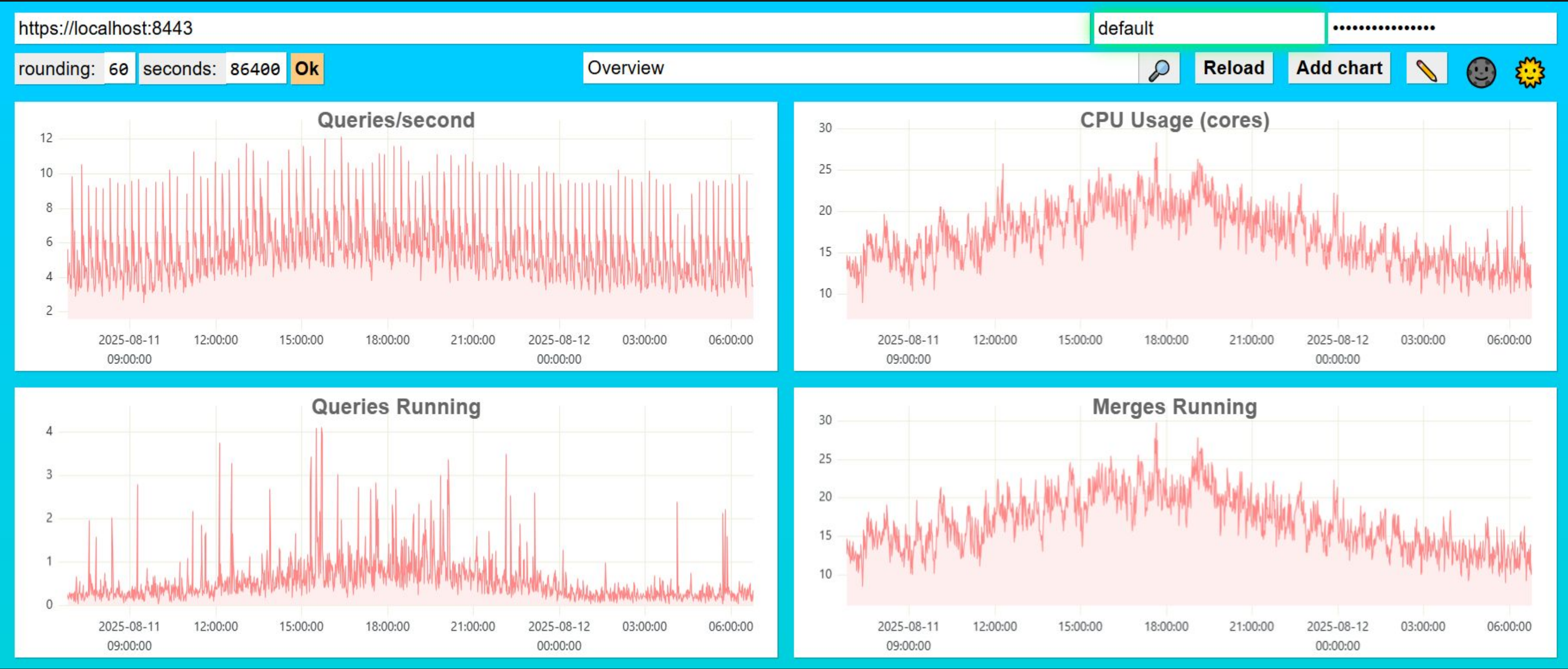
```
# replication and clusters
distributed_ddl_queue
distribution_queue
part_moves_between_shards
replicas
replicated_fetches
replication_queue

# disk space usage
disks
parts
tables
parts_columns
projection_parts
projection_parts_columns
projections
data_skipping_indices

backups
backup_log
```

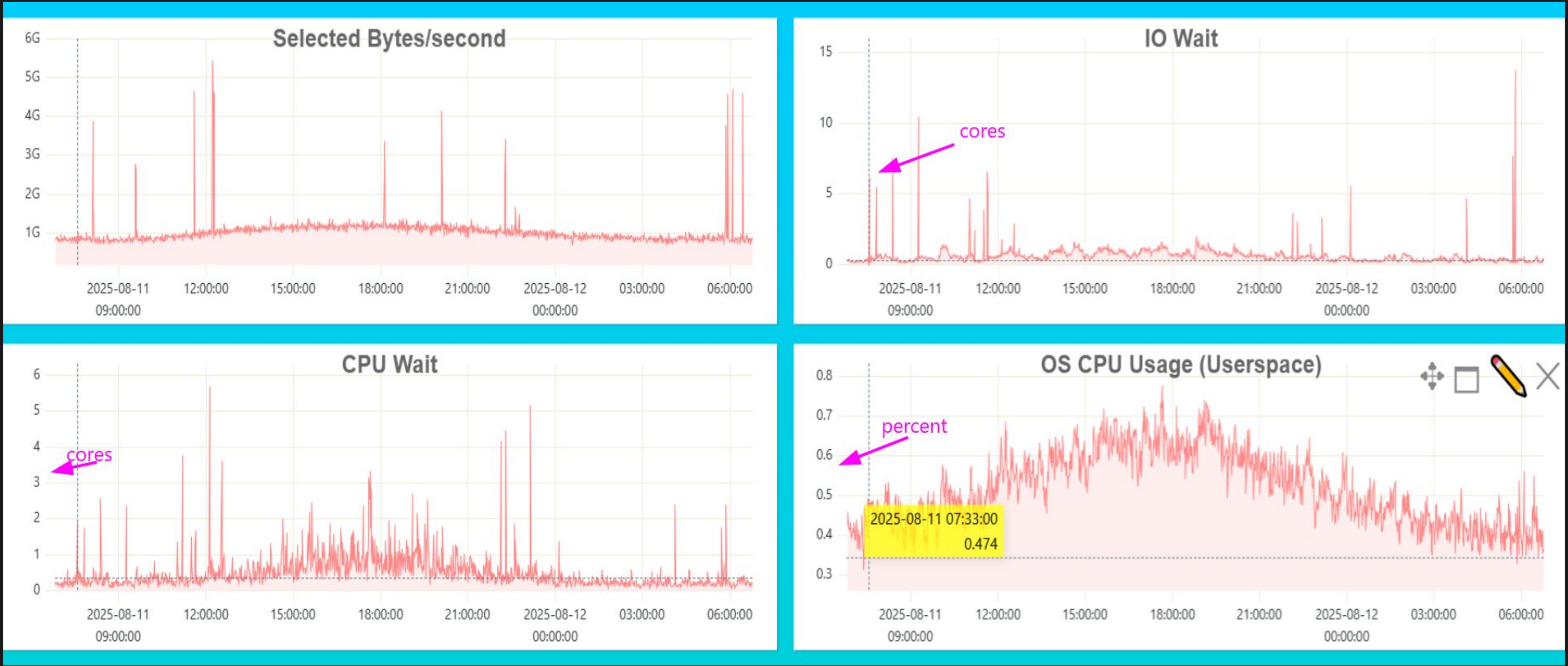
Internal basic monitoring - part 1/4

http://localhost:8123/dashboards



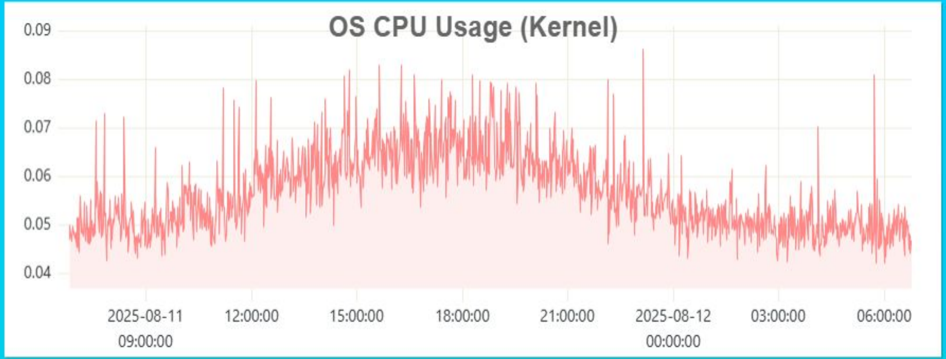
Internal basic monitoring - part 2/4

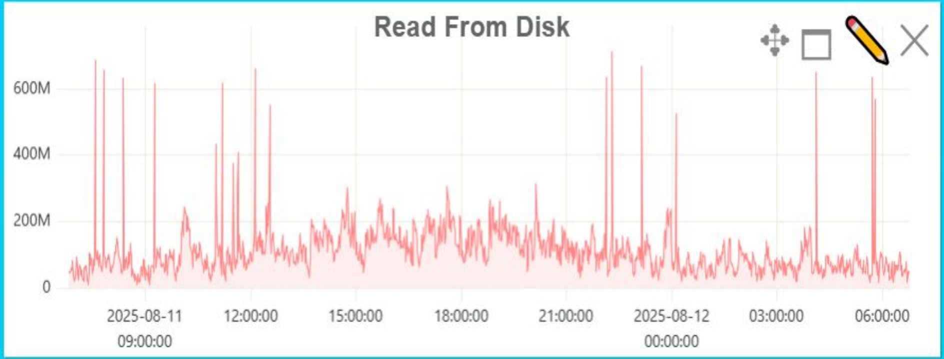
http://localhost:8123/dashboards

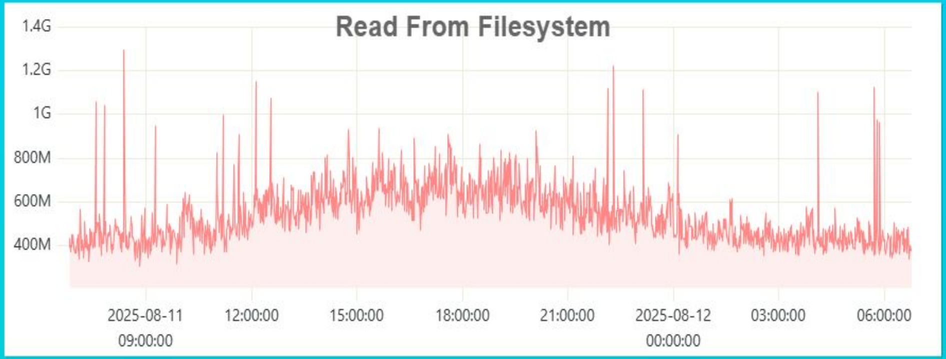


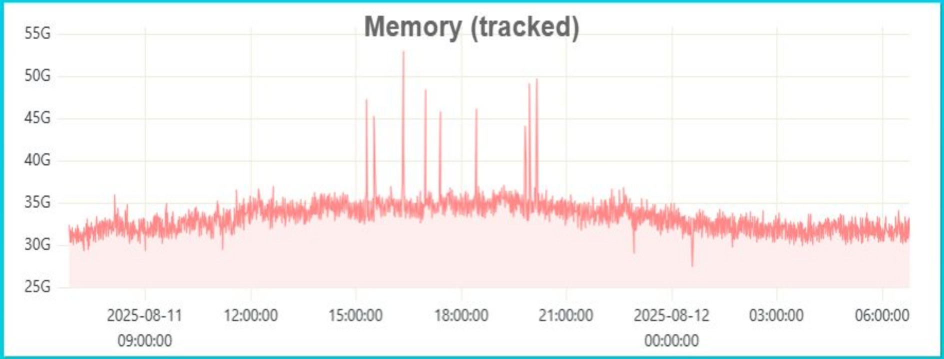
Internal basic monitoring - part 3/4

http://localhost:8123/dashboards



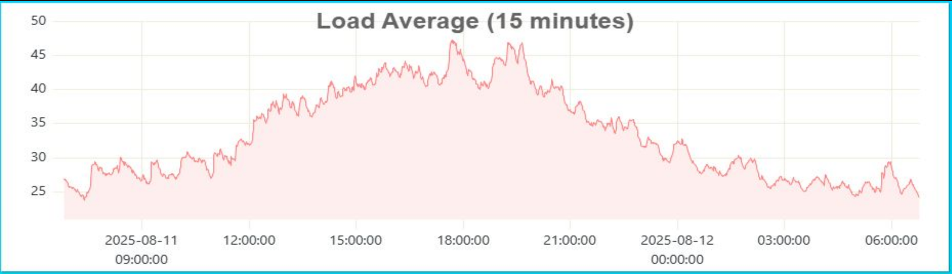


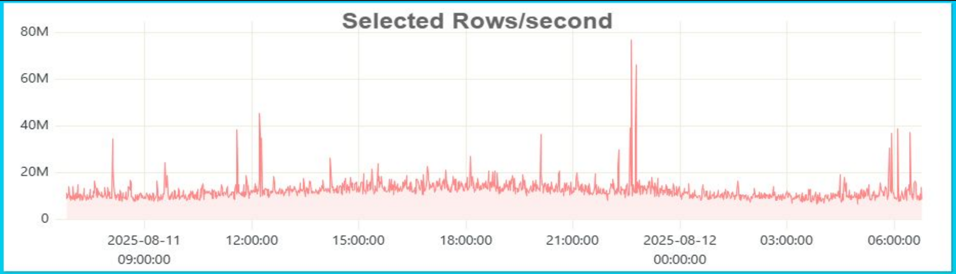


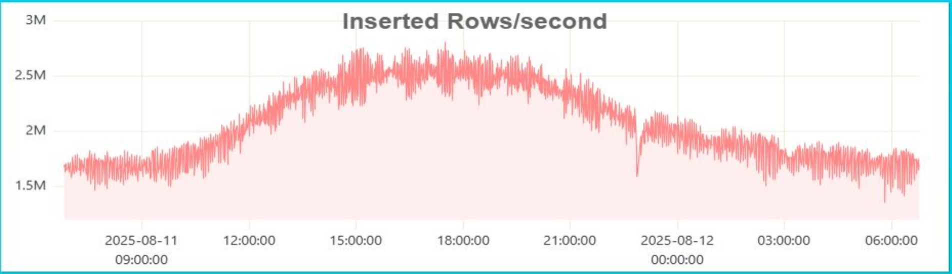


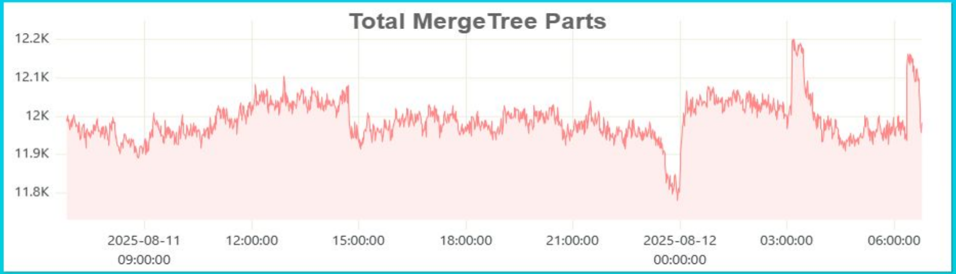
Internal basic monitoring - part 4/4

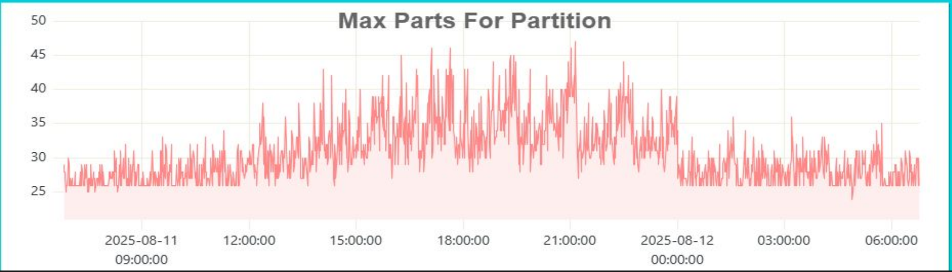
http://localhost:8123/dashboards

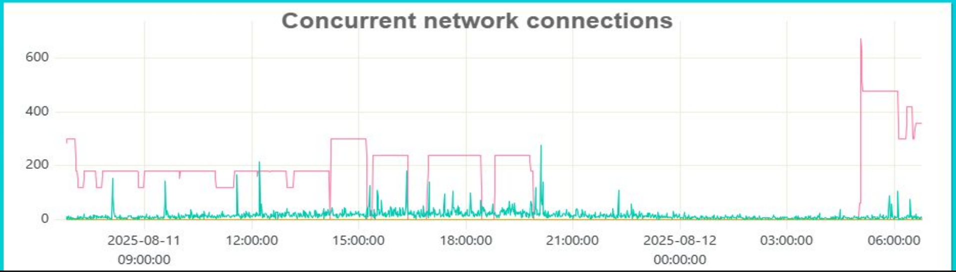






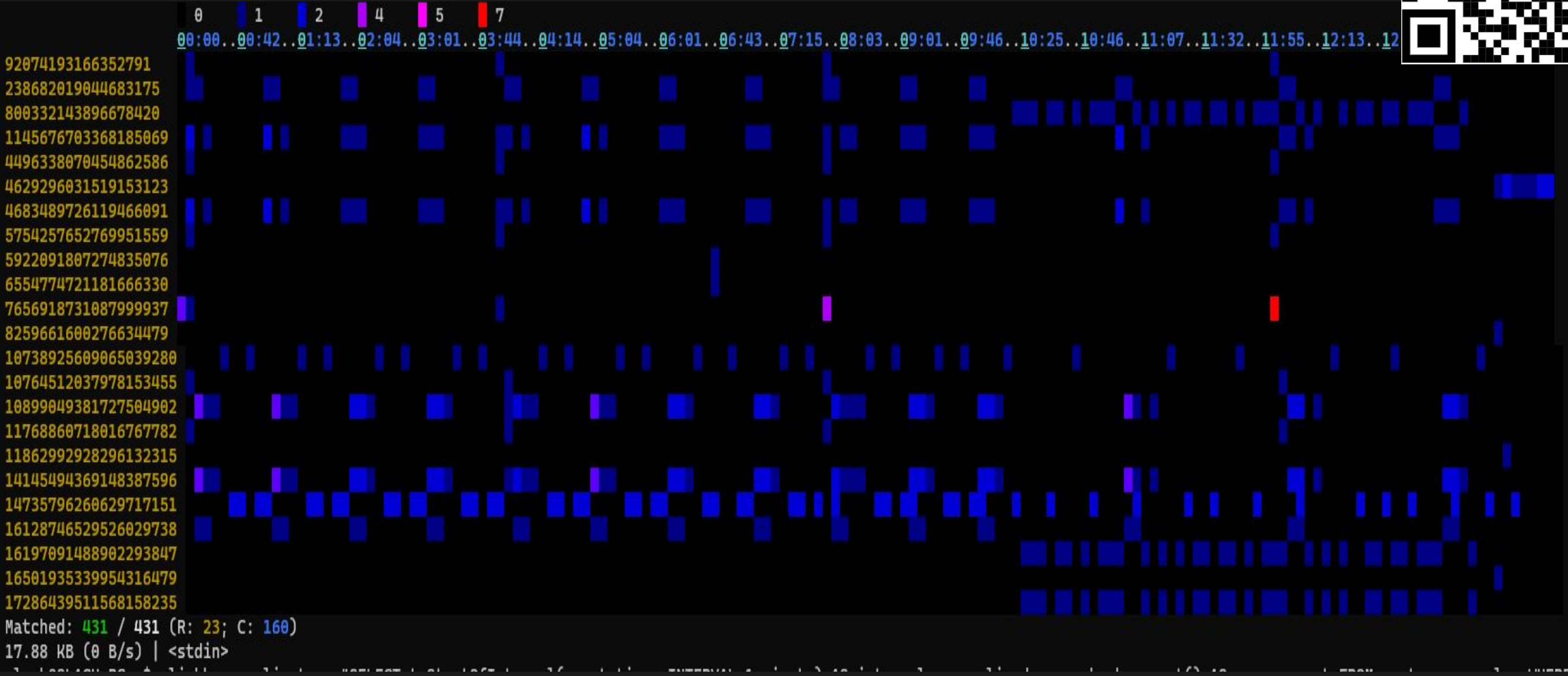




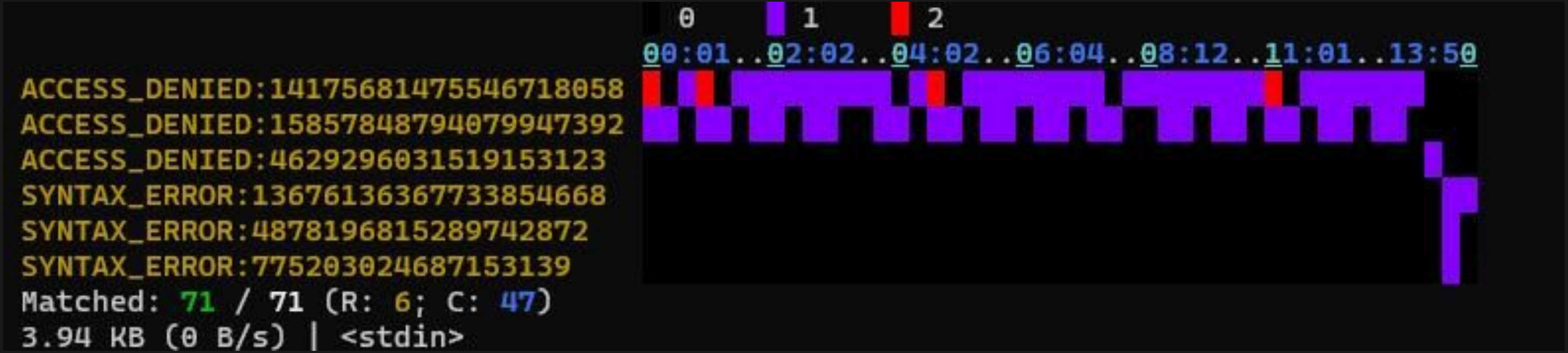


Heatmap queries by frequency

https://github.com/zix99/rare + https://github.com/Slach/clickhouse-timeline

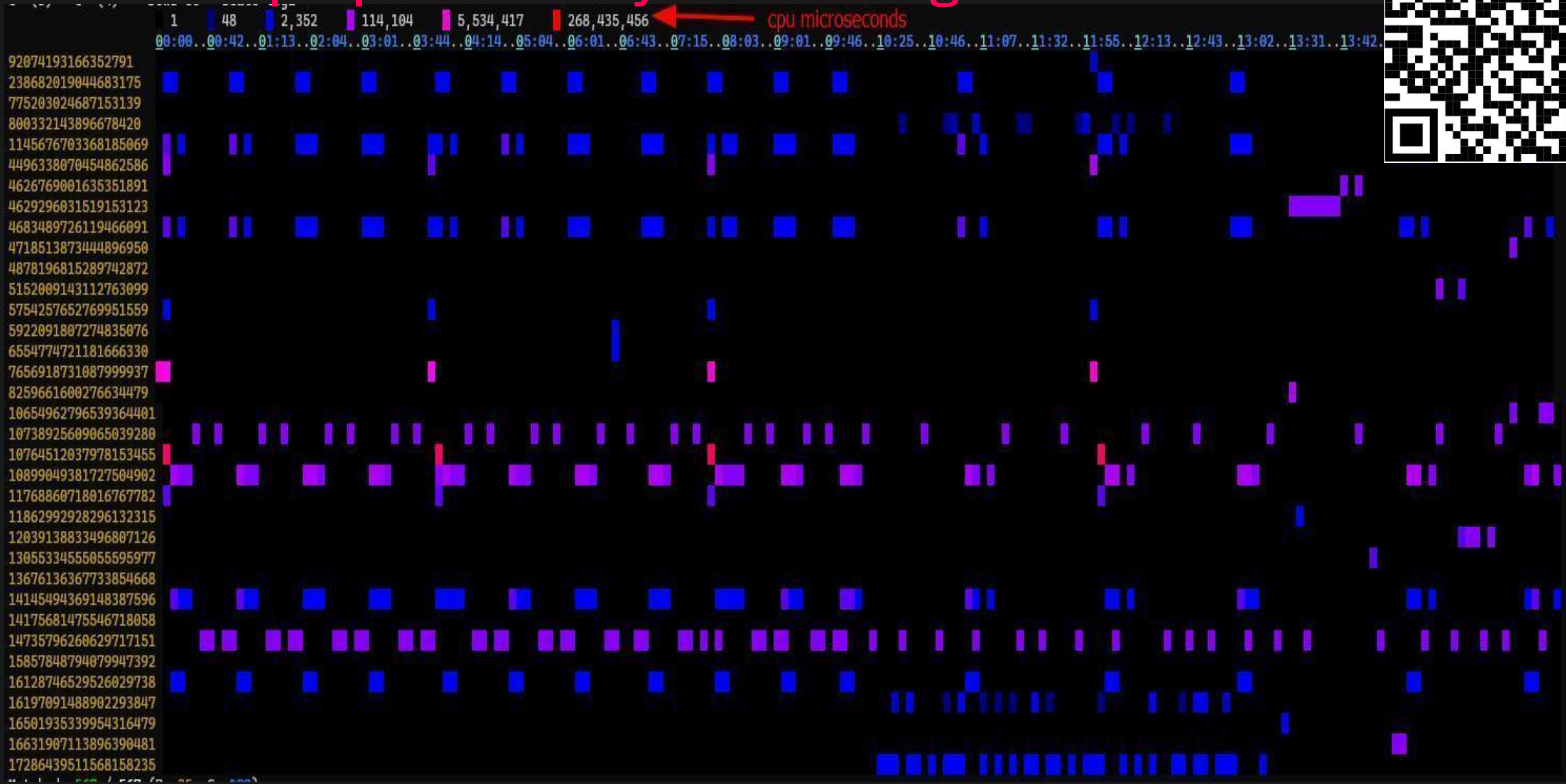

```
                      0    1    2    4    5    7
                    00:00..00:42..01:13..02:04..03:01..03:44..04:14..05:04..06:01..06:43..07:15..08:03..09:01..09:46..10:25..10:46..11:07..11:32..11:55..12:13..12
92074193166352791
238682019044683175
800332143896678420
1145676703368185069
4496338070454862586
4629296031519153123
4683489726119466091
5754257652769951559
5922091807274835076
6554774721181666330
7656918731087999937
8259661600276634479
10738925609065039280
10764512037978153455
10899049381727504902
11768860718016767782
11862992928296132315
14145494369148387596
14735796260629717151
16128746529526029738
16197091488902293847
16501935339954316479
17286439511568158235
Matched: 431 / 431 (R: 23; C: 160)
17.88 KB (0 B/s) | <stdin>
```

Heatmap error:query_hash by frequency



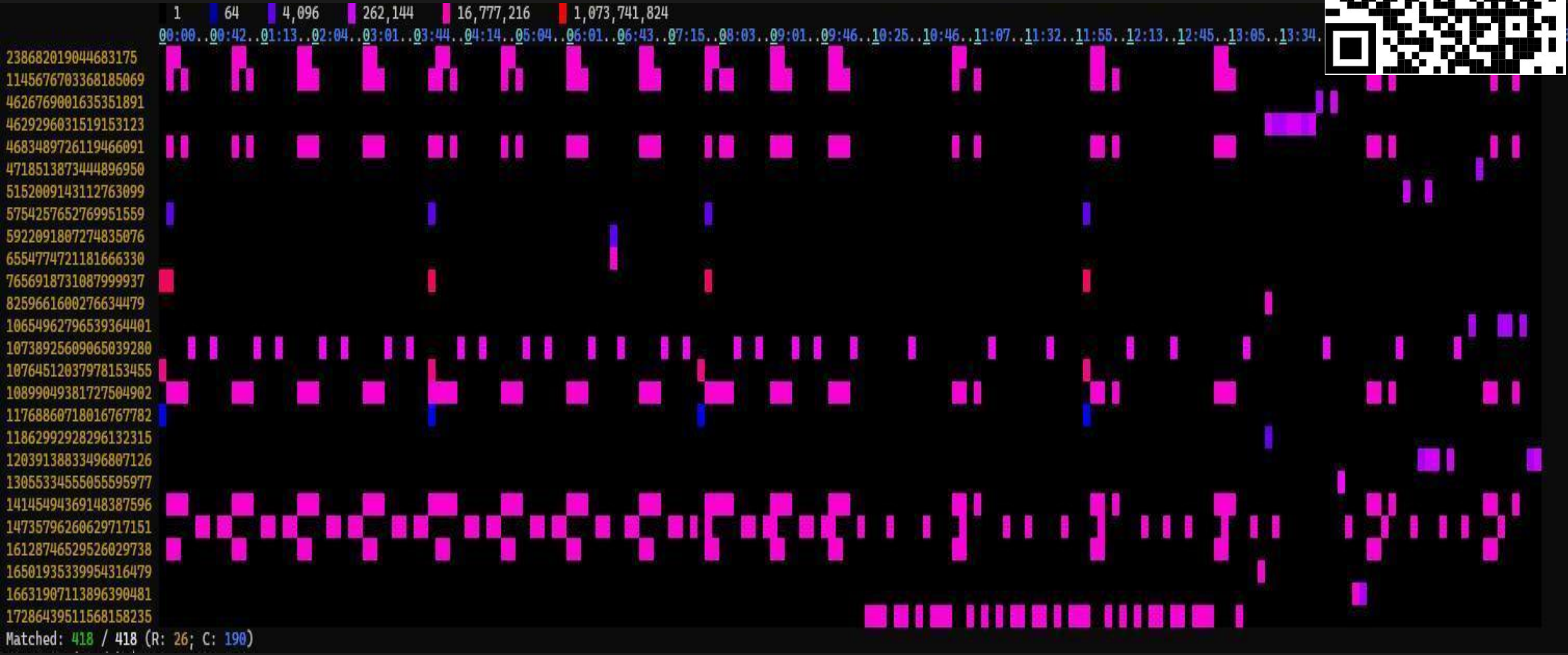
```
                                             0     1     2
                                           00:01..02:02..04:02..06:04..08:12..11:01..13:50
ACCESS_DENIED:14175681475546718058
ACCESS_DENIED:15857848794079947392
ACCESS_DENIED:4629296031519153123
SYNTAX_ERROR:13676136367733854668
SYNTAX_ERROR:4878196815289742872
SYNTAX_ERROR:7752030246871531139
Matched: 71 / 71 (R: 6; C: 47)
3.94 KB (0 B/s) | <stdin>
```

Heatmap queries by CPU Usage



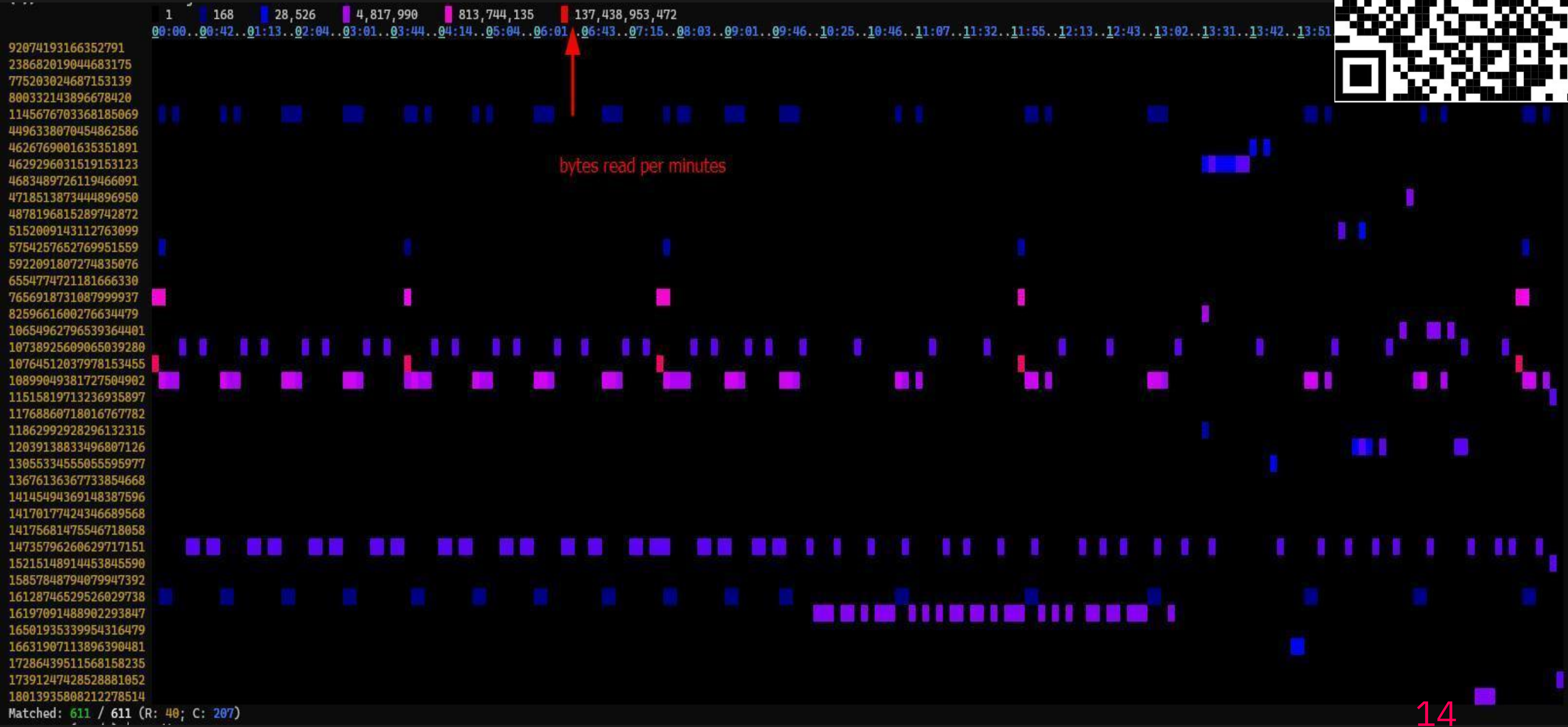
```
 1   48   2,352   114,104   5,534,417   268,435,456
00:00..00:42..01:13..02:04..03:01..03:44..04:14..05:04..06:01..06:43..07:15..08:03..09:01..09:46..10:25..10:46..11:07..11:32..11:55..12:13..12:43..13:02..13:31..13:42.
92074193166352791
238682019044683175
775203024687153139
800332143896678420
1145676703368185069
4496338070454862586
4626769001635351891
4629296031519153123
4683489726119466091
4718513873444896950
4878196815289742872
5152009143112763099
5754257652769951559
5922091807274835076
6554774721181666330
7656918731087999937
8259661600276634479
10654962796539364401
10738925609065039280
10764512037978153455
10899049381727504902
11768860718016767782
11862992928296132315
12039138833496807126
13055334555055595977
13676136367733854668
14145494369148387596
14175681475546718058
14735796260629717151
15857848794079947392
16128746529526029738
16197091488902293847
16501935339954316479
16631907113896390481
17286439511568158235
```

Heatmap queries by Memory Usage



```
   ⬛ 1   🟦 64   🟪 4,096   🟪 262,144   🟪 16,777,216   🟥 1,073,741,824
   00:00..00:42..01:13..02:04..03:01..03:44..04:14..05:04..06:01..06:43..07:15..08:03..09:01..09:46..10:25..10:46..11:07..11:32..11:55..12:13..12:45..13:05..13:34.
2386820190446831 75
11456767033681850 69
4626769001635351891
4629296031519153123
4683489726119466091
4718513873444896950
5152009143112763099
5754257652769951559
5922091807274835076
6554774721181666330
7656918731087999937
8259661600276634479
10654962796539364401
10738925609065039280
10764512037978153455
10899049381727504902
11768860718016767782
11862992928296132315
12039138833496807126
13055334555055595977
14145494369148387596
14735796260629717151
16128746529526029738
16501935339954316479
16631907113896390481
17286439511568158235
Matched: 418 / 418 (R: 26; C: 190)
```

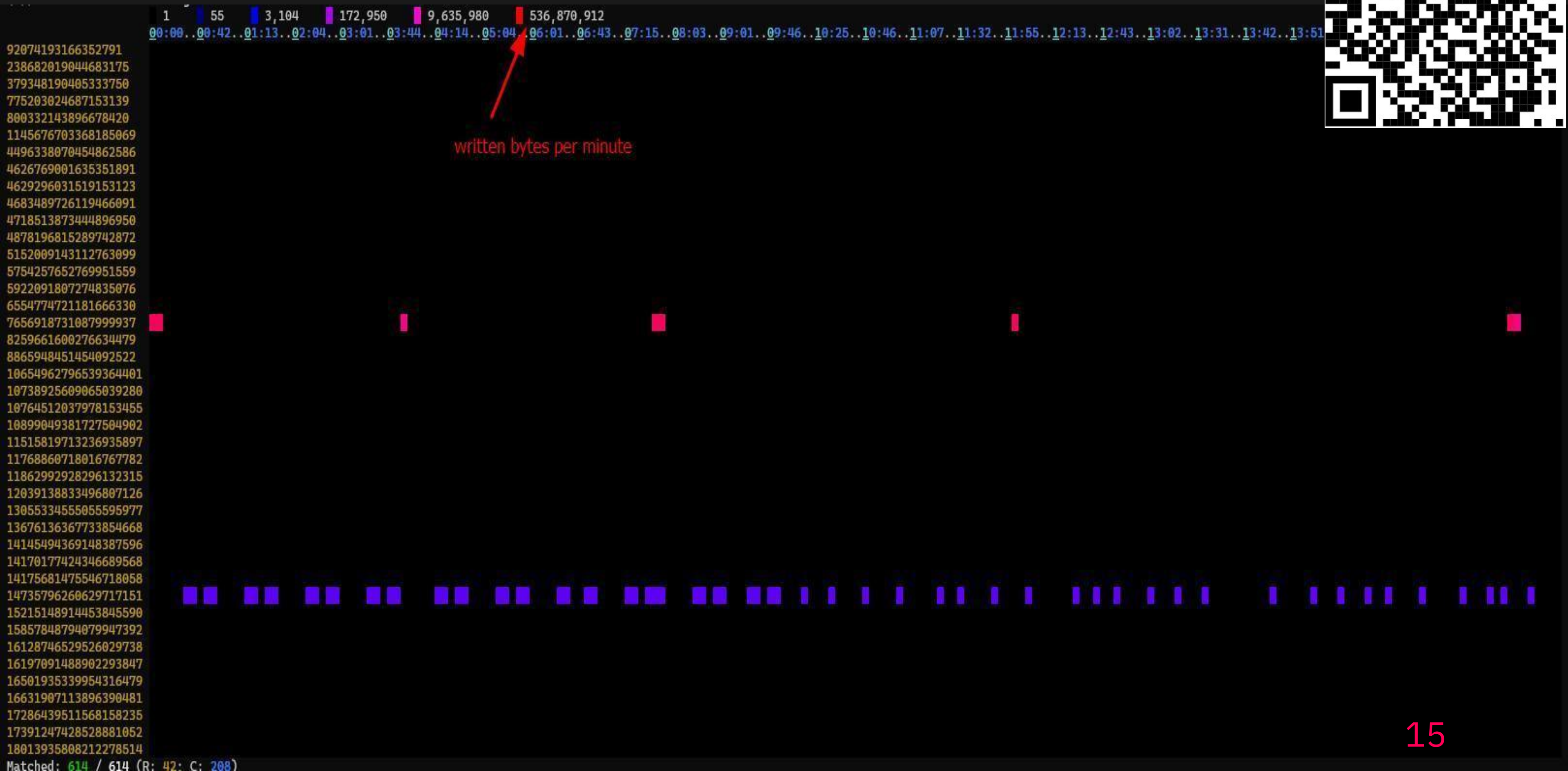
Heatmap queries by read_bytes



```
   ■ 1   ■ 168   ■ 28,526   ■ 4,817,990   ■ 813,744,135   ■ 137,438,953,472
   00:00..00:42..01:13..02:04..03:01..03:44..04:14..05:04..06:01..06:43..07:15..08:03..09:01..09:46..10:25..10:46..11:07..11:32..11:55..12:13..12:43..13:02..13:31..13:42..13:51

920741931663527910
2386820190446831750
7752030246871531390
8003321438966784200
11456767033681850690
4496338070454862586
4626769001635351891
4629296031519153123
4683489726119466091
4718513873444896950
4878196815289742872
5152009143112763099
5754257652769951559
5922091807274835076
6554774721181666330
7656918731087999937
8259661600276634479
10654962796539364401
10738925609065039280
10764512037978153455
10899049381727504902
11515819713236935897
11768860718016767782
11862992928296132315
12039138833496807126
13055334555055595977
13676136367733854668
14145494369148387596
14170177424346689568
14175681475546718058
14735796260629717151
15215148914453845590
15857848794079947392
16128746529526029738
16197091488902293847
16501935339954316479
16631907113896390481
17286439511568158235
17391247428528881052
18013935808212278514
Matched: 611 / 611 (R: 40; C: 207)
```

Heatmap queries by written_bytes



```
     1   55   3,104   172,950   9,635,980   536,870,912
     00:00..00:42..01:13..02:04..03:01..03:44..04:14..05:04..06:01..06:43..07:15..08:03..09:01..09:46..10:25..10:46..11:07..11:32..11:55..12:13..12:43..13:02..13:31..13:42..13:51
920741931663527 91
2386820190446831 75
3793481904053337 50
7752030246871531 39
8003321438966784 20
11456767033681850 69
44963380704548625 86
46267690016353518 91
46292960315191531 23
46834897261194660 91
47185138734448969 50
48781968152897428 72
51520091431127630 99
57542576527699515 59
59220918072748350 76
65547747211816663 30
76569187310879999 37
82596616002766344 79
88659484514540925 22
106549627965393644 01
107389256090650392 80
107645120379781534 55
108990493817275049 02
115158197132369358 97
117688607180167677 82
118629929282961323 15
120391388334968071 26
130553345550555959 77
136761363677338546 68
141454943691483875 96
141701774243466895 68
141756814755467180 58
147357962606297171 51
152151489144538455 90
158578487940799473 92
161287465295260297 38
161970914889022938 47
165019353399543164 79
166319071138963904 81
172864395115681582 35
173912474285288810 52
180139358082122785 14
Matched: 614 / 614 (R: 42; C: 208)
```

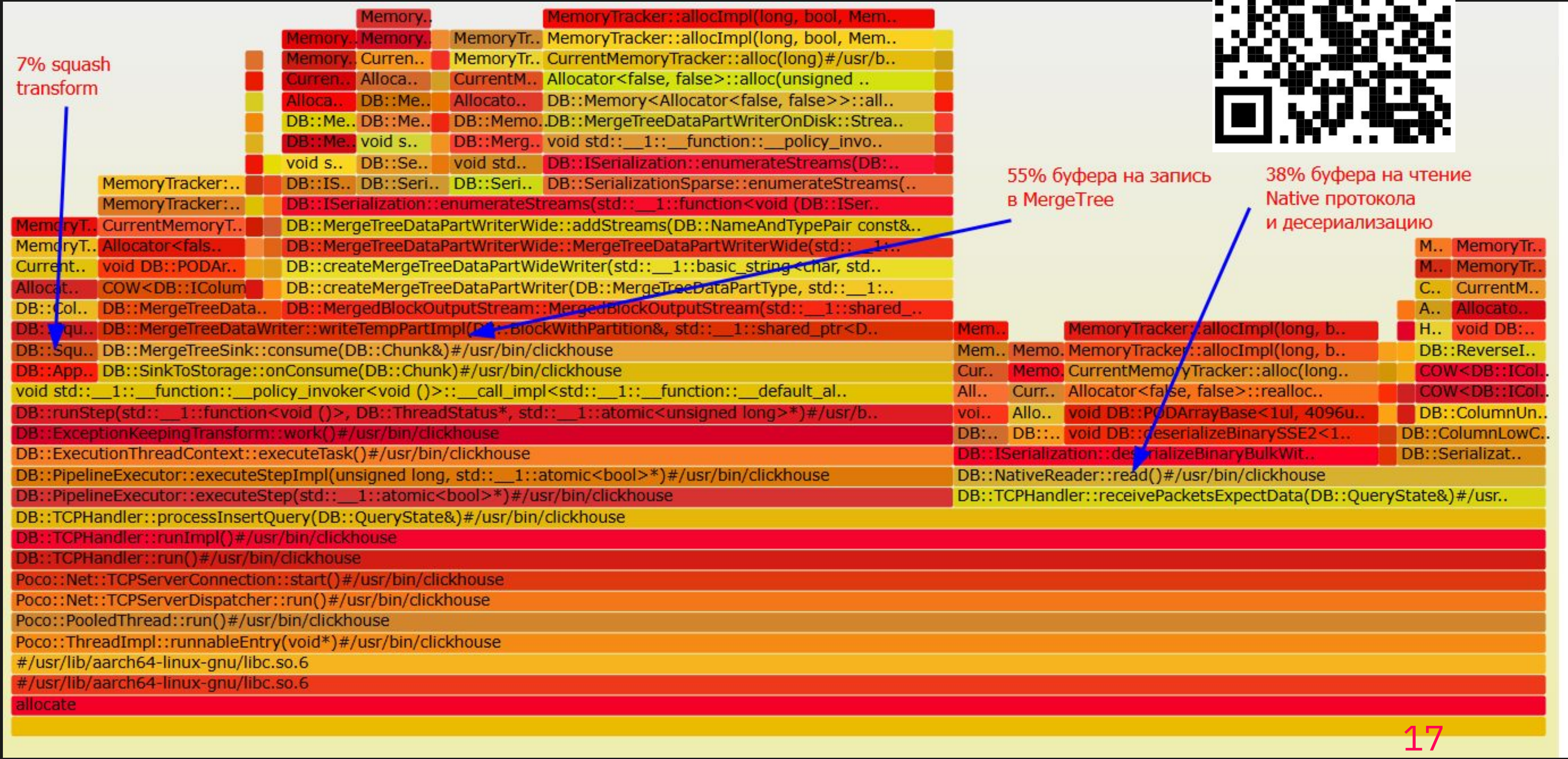
Found bad normalized_query_hash and what?

Make flamegraph https://github.com/Slach/clickhouse-flamegraph

```
clickhouse-client -q "SELECT 'clickhouse-flamegraph
--query-ids=' ||
arrayStringConcat(groupArray(10)(query_id),',') ||
'\n' FROM system.query_log WHERE
normalized_query_hash=? AND event_date=? AND
event_time BETWEEN ? AND ? ORDER BY
query_duration_ms DESC LIMIT 10
FORMAT TSVRaw" | bash
```

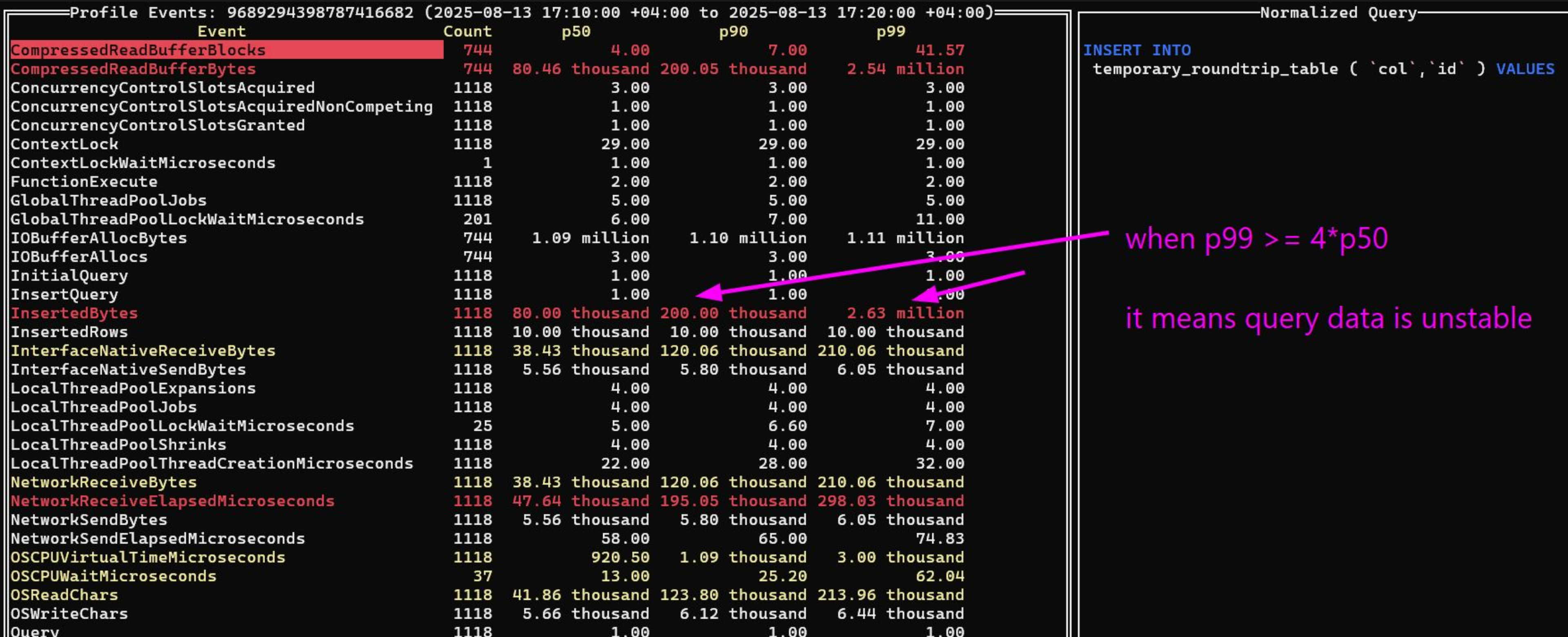
Flamegraph: what is it?

https://github.com/YS-L/flamelens



Found bad normalized_query_hash - what next?

Look percentiles for ProfileEvents



```
Profile Events: 9689294398787416682 (2025-08-13 17:10:00 +04:00 to 2025-08-13 17:20:00 +04:00)
                Event                          Count        p50             p90             p99
CompressedReadBufferBlocks                      744           4.00            7.00           41.57
CompressedReadBufferBytes                       744  80.46 thousand 200.05 thousand    2.54 million
ConcurrencyControlSlotsAcquired                1118           3.00            3.00            3.00
ConcurrencyControlSlotsAcquiredNonCompeting    1118           1.00            1.00            1.00
ConcurrencyControlSlotsGranted                 1118           1.00            1.00            1.00
ContextLock                                    1118          29.00           29.00           29.00
ContextLockWaitMicroseconds                       1           1.00            1.00            1.00
FunctionExecute                                1118           2.00            2.00            2.00
GlobalThreadPoolJobs                           1118           5.00            5.00            5.00
GlobalThreadPoolLockWaitMicroseconds            201           6.00            7.00           11.00
IOBufferAllocBytes                              744    1.09 million    1.10 million    1.11 million
IOBufferAllocs                                  744           3.00            3.00            3.00
InitialQuery                                   1118           1.00            1.00            1.00
InsertQuery                                    1118           1.00            1.00            1.00
InsertedBytes                                  1118  80.00 thousand 200.00 thousand    2.63 million
InsertedRows                                   1118  10.00 thousand  10.00 thousand  10.00 thousand
InterfaceNativeReceiveBytes                    1118  38.43 thousand 120.06 thousand 210.06 thousand
InterfaceNativeSendBytes                       1118   5.56 thousand   5.80 thousand   6.05 thousand
LocalThreadPoolExpansions                      1118           4.00            4.00            4.00
LocalThreadPoolJobs                            1118           4.00            4.00            4.00
LocalThreadPoolLockWaitMicroseconds              25           5.00            6.60            7.00
LocalThreadPoolShrinks                         1118           4.00            4.00            4.00
LocalThreadPoolThreadCreationMicroseconds      1118          22.00           28.00           32.00
NetworkReceiveBytes                            1118  38.43 thousand 120.06 thousand 210.06 thousand
NetworkReceiveElapsedMicroseconds              1118  47.64 thousand 195.05 thousand 298.03 thousand
NetworkSendBytes                               1118   5.56 thousand   5.80 thousand   6.05 thousand
NetworkSendElapsedMicroseconds                 1118          58.00           65.00           74.83
OSCPUVirtualTimeMicroseconds                   1118         920.50   1.09 thousand   3.00 thousand
OSCPUWaitMicroseconds                            37          13.00           25.20           62.04
OSReadChars                                    1118  41.86 thousand 123.80 thousand 213.96 thousand
OSWriteChars                                   1118   5.66 thousand   6.12 thousand   6.44 thousand
Query                                          1118           1.00            1.00            1.00
```

```
Normalized Query
INSERT INTO
 temporary_roundtrip_table ( `col`,`id` ) VALUES
```

when p99 >= 4*p50

it means query data is unstable

MEMORY_LIMIT_EXCEED - How Memory Tracker works

When memory allocates, clickhouse looks to the memory tracker counter.
If value is more than the soft limit value, then wait memory_usage_overcommit_max_wait_microseconds. If memory is not freed, then any next allocation will throw an exception, and any other query could be canceled.

Memory limit exceeded (for query)

Tweak max_bytes_before_external_sort, max_bytes_before_external_group_by, max_threads, max_insert_threads, max_memory_usage

Memory limit exceeded (for user) - 90% use cases another queries.

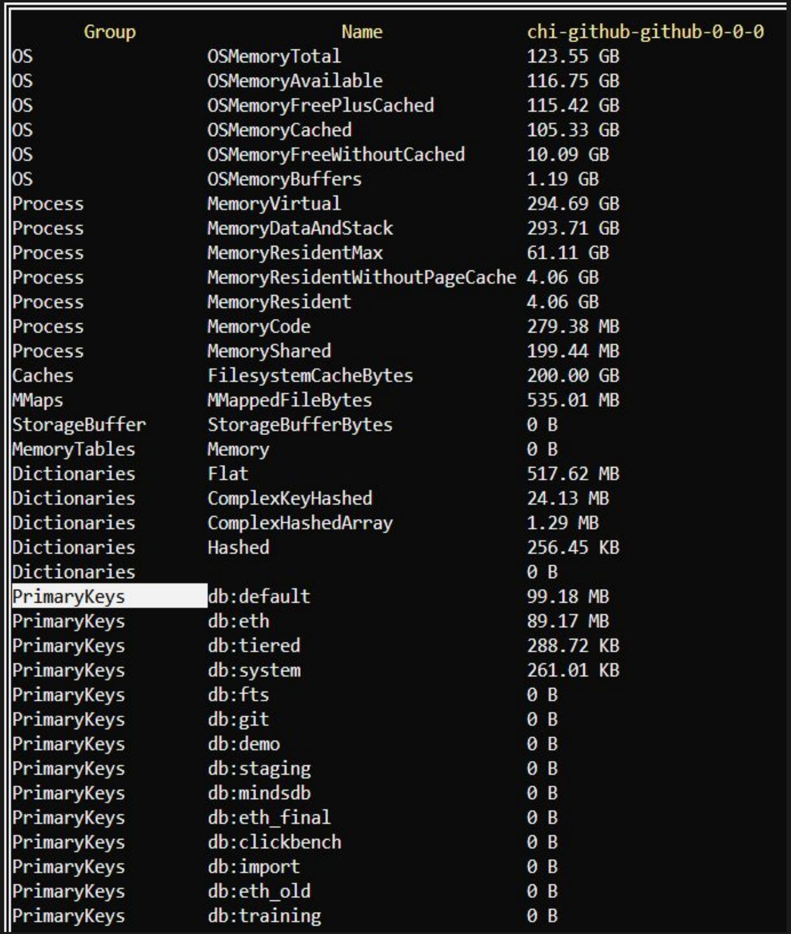
max_memory_usage_for_user

Memory limit exceeded (total)

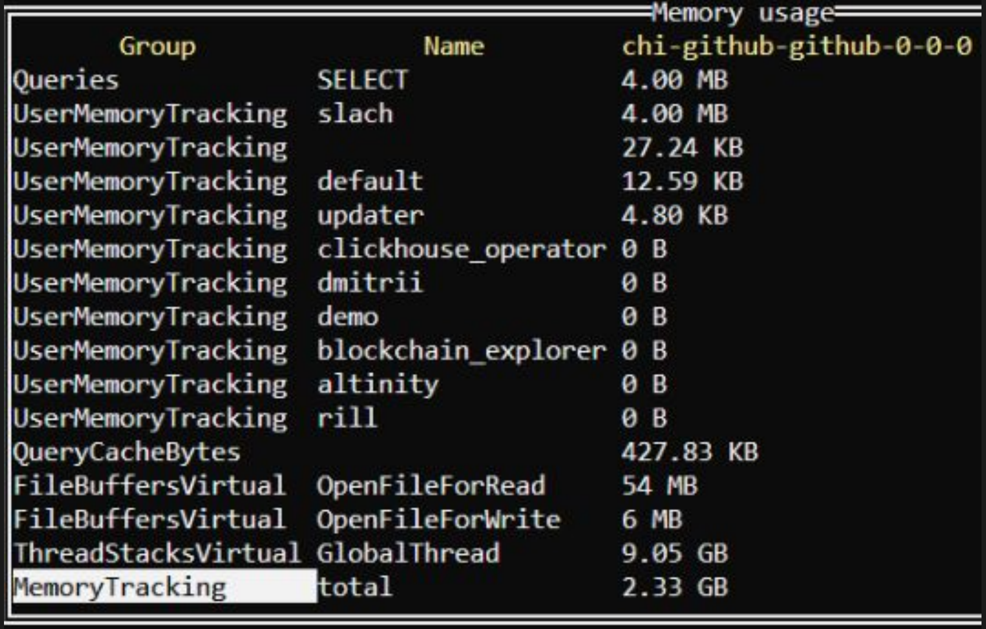
Try to decrease max_server_memory_usage_ratio (to avoid OOM)

https://kb.altinity.com/altinity-kb-setup-and-maintenance/altinity-kb-who-ate-my-memory/

Who ate my memory

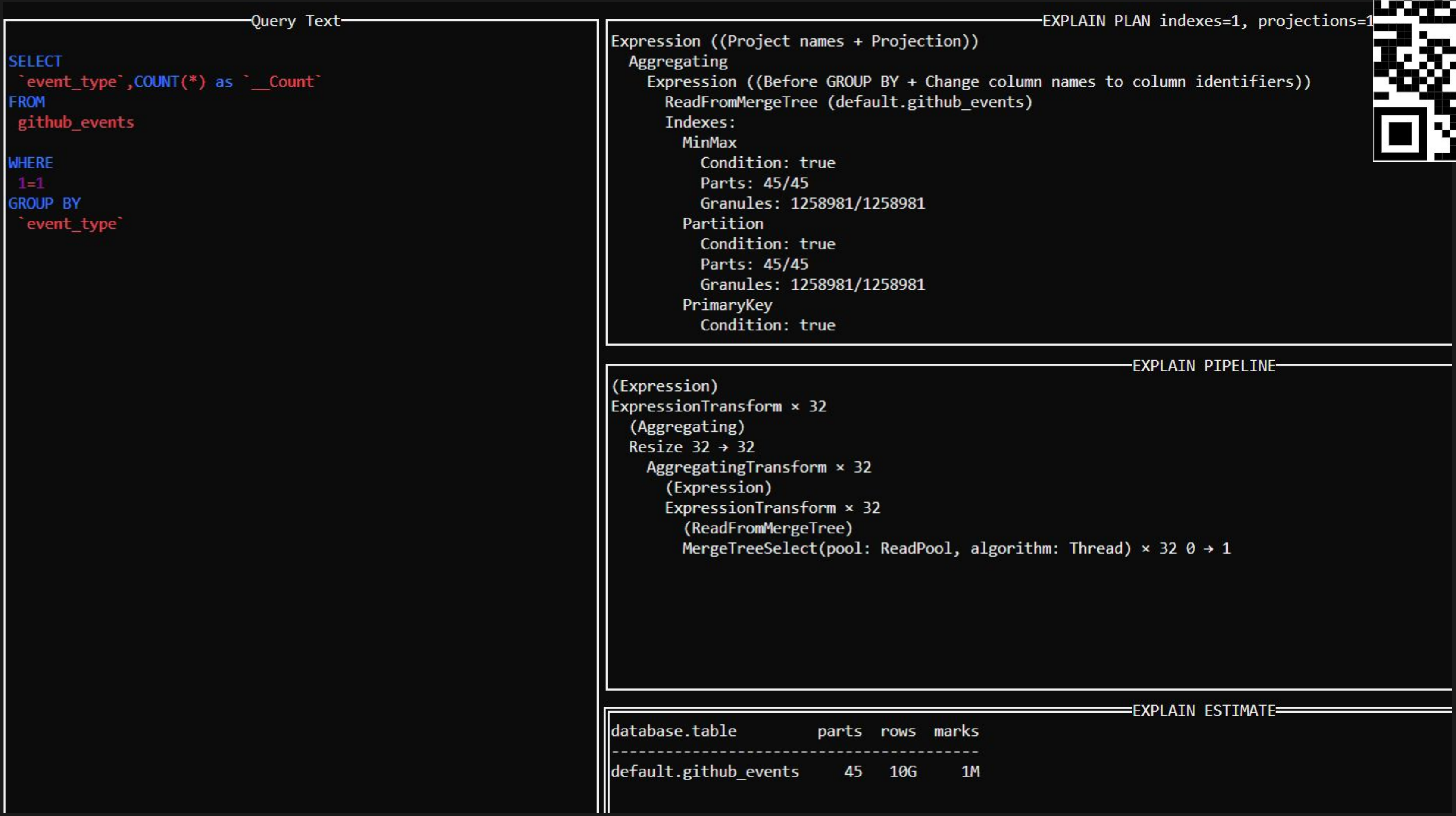


```
     Group                      Name                     chi-github-github-0-0-0
OS                  OSMemoryTotal                        123.55 GB
OS                  OSMemoryAvailable                    116.75 GB
OS                  OSMemoryFreePlusCached               115.42 GB
OS                  OSMemoryCached                       105.33 GB
OS                  OSMemoryFreeWithoutCached            10.09 GB
OS                  OSMemoryBuffers                      1.19 GB
Process             MemoryVirtual                        294.69 GB
Process             MemoryDataAndStack                   293.71 GB
Process             MemoryResidentMax                    61.11 GB
Process             MemoryResidentWithoutPageCache       4.06 GB
Process             MemoryResident                       4.06 GB
Process             MemoryCode                           279.38 MB
Process             MemoryShared                         199.44 MB
Caches              FilesystemCacheBytes                 200.00 GB
MMaps               MMappedFileBytes                     535.01 MB
StorageBuffer       StorageBufferBytes                   0 B
MemoryTables        Memory                               0 B
Dictionaries        Flat                                 517.62 MB
Dictionaries        ComplexKeyHashed                     24.13 MB
Dictionaries        ComplexHashedArray                   1.29 MB
Dictionaries        Hashed                               256.45 KB
Dictionaries                                             0 B
PrimaryKeys         db:default                           99.18 MB
PrimaryKeys         db:eth                               89.17 MB
PrimaryKeys         db:tiered                            288.72 KB
PrimaryKeys         db:system                            261.01 KB
PrimaryKeys         db:fts                               0 B
PrimaryKeys         db:git                               0 B
PrimaryKeys         db:demo                              0 B
PrimaryKeys         db:staging                           0 B
PrimaryKeys         db:mindsdb                           0 B
PrimaryKeys         db:eth_final                         0 B
PrimaryKeys         db:clickbench                        0 B
PrimaryKeys         db:import                            0 B
PrimaryKeys         db:eth_old                           0 B
PrimaryKeys         db:training                          0 B
```



```
                                    Memory usage
       Group                 Name                  chi-github-github-0-0-0
Queries               SELECT                     4.00 MB
UserMemoryTracking    slach                      4.00 MB
UserMemoryTracking                               27.24 KB
UserMemoryTracking    default                    12.59 KB
UserMemoryTracking    updater                    4.80 KB
UserMemoryTracking    clickhouse_operator        0 B
UserMemoryTracking    dmitrii                    0 B
UserMemoryTracking    demo                       0 B
UserMemoryTracking    blockchain_explorer        0 B
UserMemoryTracking    altinity                   0 B
UserMemoryTracking    rill                       0 B
QueryCacheBytes                                  427.83 KB
FileBuffersVirtual    OpenFileForRead            54 MB
FileBuffersVirtual    OpenFileForWrite           6 MB
ThreadStacksVirtual   GlobalThread               9.05 GB
MemoryTracking        total                      2.33 GB
```

Explain bad normalized_query_hash



Query Text

```
SELECT
 `event_type`,COUNT(*) as `__Count`
FROM
 github_events

WHERE
 1=1
GROUP BY
 `event_type`
```

EXPLAIN PLAN indexes=1, projections=1

```
Expression ((Project names + Projection))
  Aggregating
    Expression ((Before GROUP BY + Change column names to column identifiers))
      ReadFromMergeTree (default.github_events)
      Indexes:
        MinMax
          Condition: true
          Parts: 45/45
          Granules: 1258981/1258981
        Partition
          Condition: true
          Parts: 45/45
          Granules: 1258981/1258981
        PrimaryKey
          Condition: true
```

EXPLAIN PIPELINE

```
(Expression)
ExpressionTransform × 32
  (Aggregating)
  Resize 32 → 32
    AggregatingTransform × 32
      (Expression)
      ExpressionTransform × 32
        (ReadFromMergeTree)
        MergeTreeSelect(pool: ReadPool, algorithm: Thread) × 32 0 → 1
```

EXPLAIN ESTIMATE

```
database.table         parts  rows  marks
-----------------------------------------
default.github_events     45   10G     1M
```

Thanks for your attention :)
Please ask questions!

@BLOODJAZMAN

Altinity