The Doctor Is In

Quick First Aid for Broken ClickHouse® Clusters

Robert Hodges
Diego Nieto
Tatiana Saltykova
https://altinity.com

Important warning!

If you or a loved one is experiencing a data-threatening ClickHouse emergency, please stop listening and get help immediately.

https://www.altinity.com/slack
https://altinity.com/contact/

A brief message from our sponsor…

Robert Hodges

Database geek with 30+ years on DBMS. Kubernaut since 2018. Day job: Altinity CEO

Diego Nieto

Software engineer on Altinity with interests in databases, Python, and Rust

Altinity

ClickHouse support and services including Altinity.Cloud
Authors of Altinity Kubernetes Operator for ClickHouse, Altinity clickhouse-backup and other open source projects

Welcome to ClickHouse, a real-time analytic database

Understands SQL

Runs on bare metal to cloud

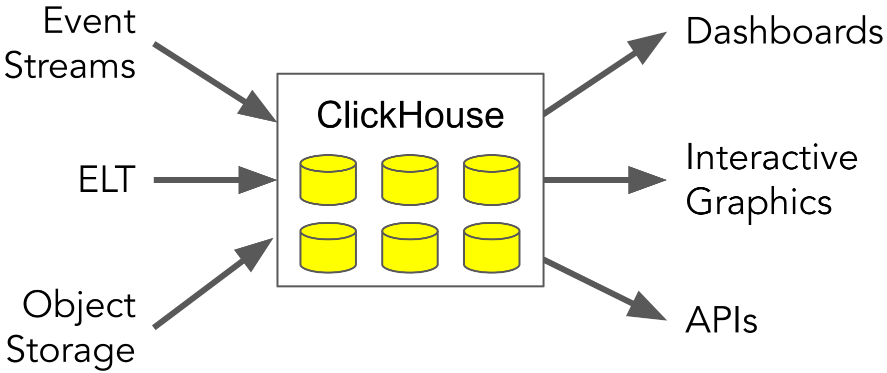
Shared nothing architecture

Stores data in columns

Parallel and vectorized execution

Scales to many petabytes

Is Open source (Apache 2.0)



It’s the core engine for low-latency analytics

Too Many Queries errors

From the annals of ClickHouse crime, case 202 :-)

```
Code: 202. DB::Exception: Received from localhost:9000.
DB::Exception: Too many simultaneous queries. Maximum: 25.
(TOO_MANY_SIMULTANEOUS_QUERIES) (version 24.3.5.47.altinitystable
(altinity build))
An error occurred while processing the query 'select avg(number)
from numbers(1000000000)': Code: 202. DB::Exception: Received from
localhost:9000. DB::Exception: Too many simultaneous queries.
Maximum: 25. Stack trace:

0. DB::Exception::Exception(DB::Exception::MessageMasked&&, int,
bool) @ 0x000000000cbd3bbb
```

Options for fixing too many query errors

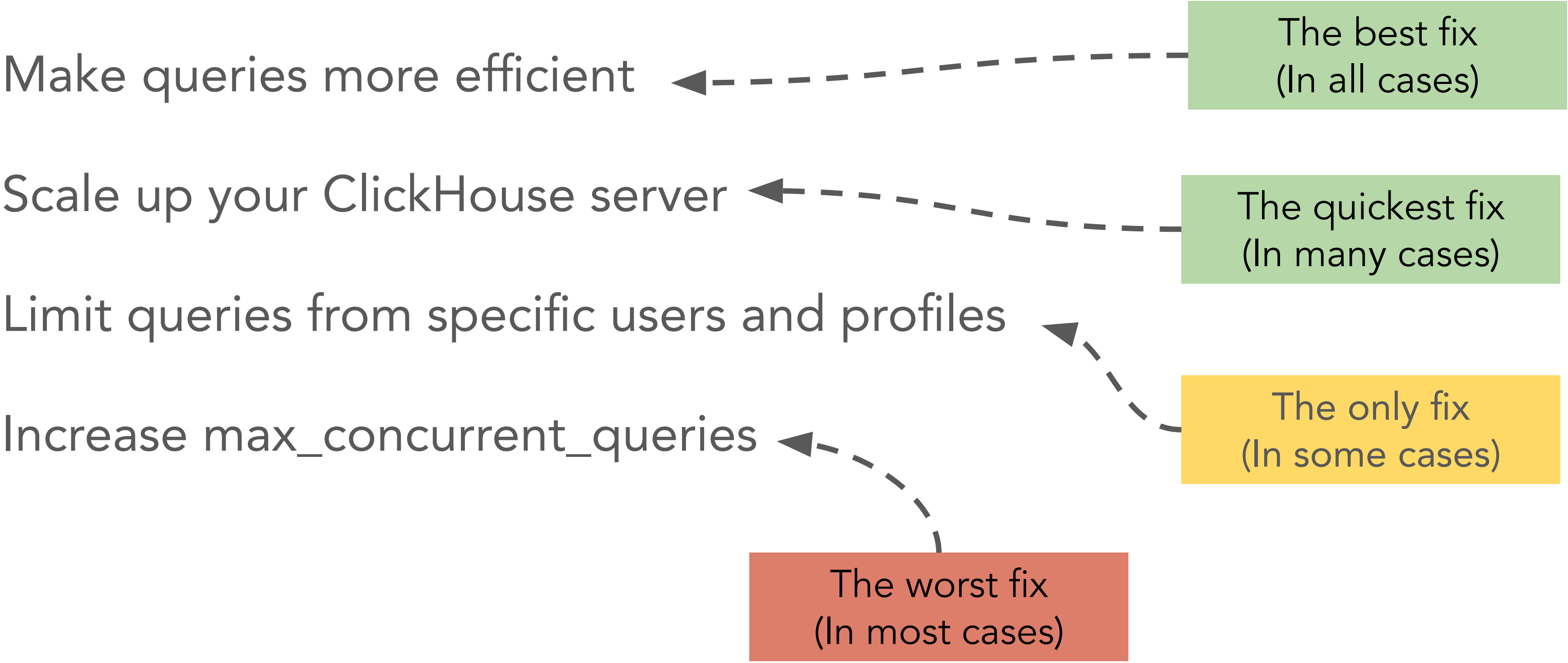
Make queries more efficient

Scale up your ClickHouse server

Limit queries from specific users and profiles

Increase max_concurrent_queries

Options for fixing too many query errors



Make queries more efficient

Scale up your ClickHouse server

Limit queries from specific users and profiles

Increase max_concurrent_queries

The best fix
(In all cases)

The quickest fix
(In many cases)

The only fix
(In some cases)

The worst fix
(In most cases)

Simple example of fixing a query

```
# This query fails.
clickhouse-benchmark -t 3 -c 26 --ignore-error \
 --query='select avg(number) from numbers(1000000000)'

# This query does not!
clickhouse-benchmark -t 3 -c 26 --ignore-error \
 --query='select (1000000000 - 1) / 2'
```

Limiting users with settings profiles

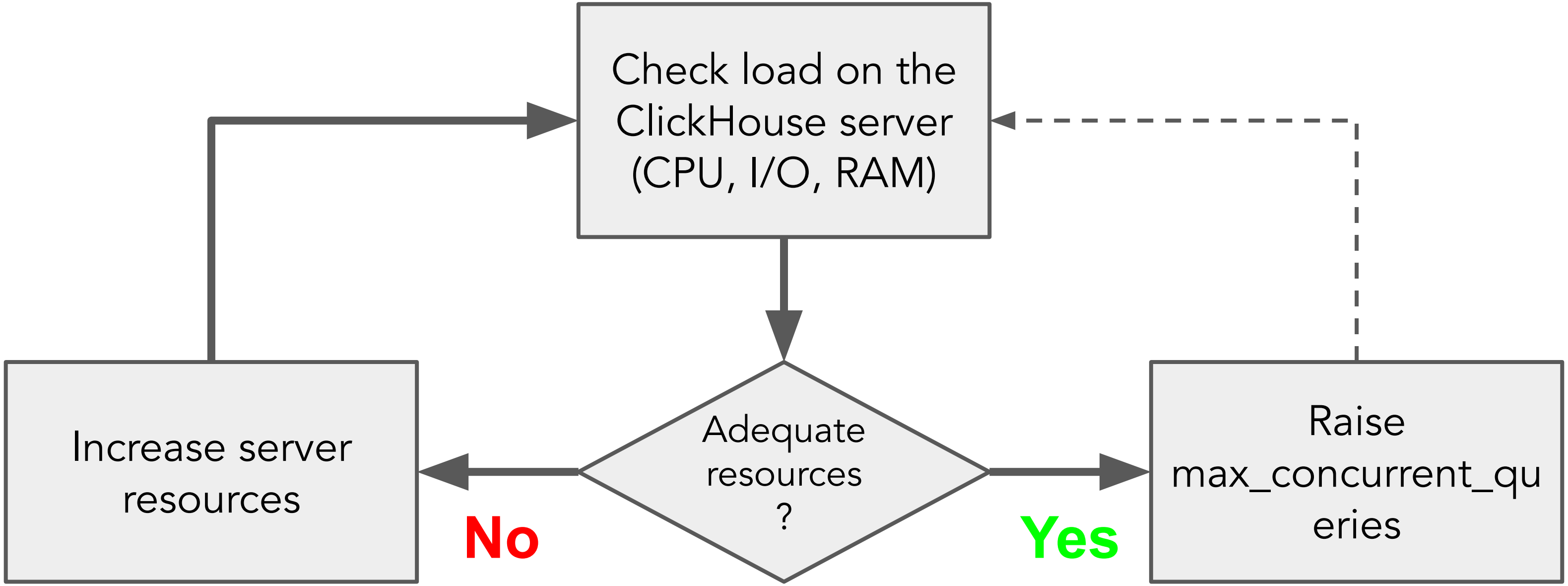
```
CREATE SETTINGS PROFILE IF NOT EXISTS `too_many` SETTINGS
  max_concurrent_queries_for_user = 15,
  max_concurrent_queries_for_all_users = 20
;

CREATE USER IF NOT EXISTS tm1
  IDENTIFIED WITH sha256_password BY 'topsecret'
  HOST LOCAL
  SETTINGS PROFILE 'too_many'
;
. . .
```

Limit for individual user

Limit for all users

How to raise max_concurrent_queries <u>safely</u>



Check load on the ClickHouse server (CPU, I/O, RAM)

Adequate resources ?

No

Increase server resources

Yes

Raise max_concurrent_queries

Altinity

© 2024 Altinity, Inc.

Reminder on updating config.xml values

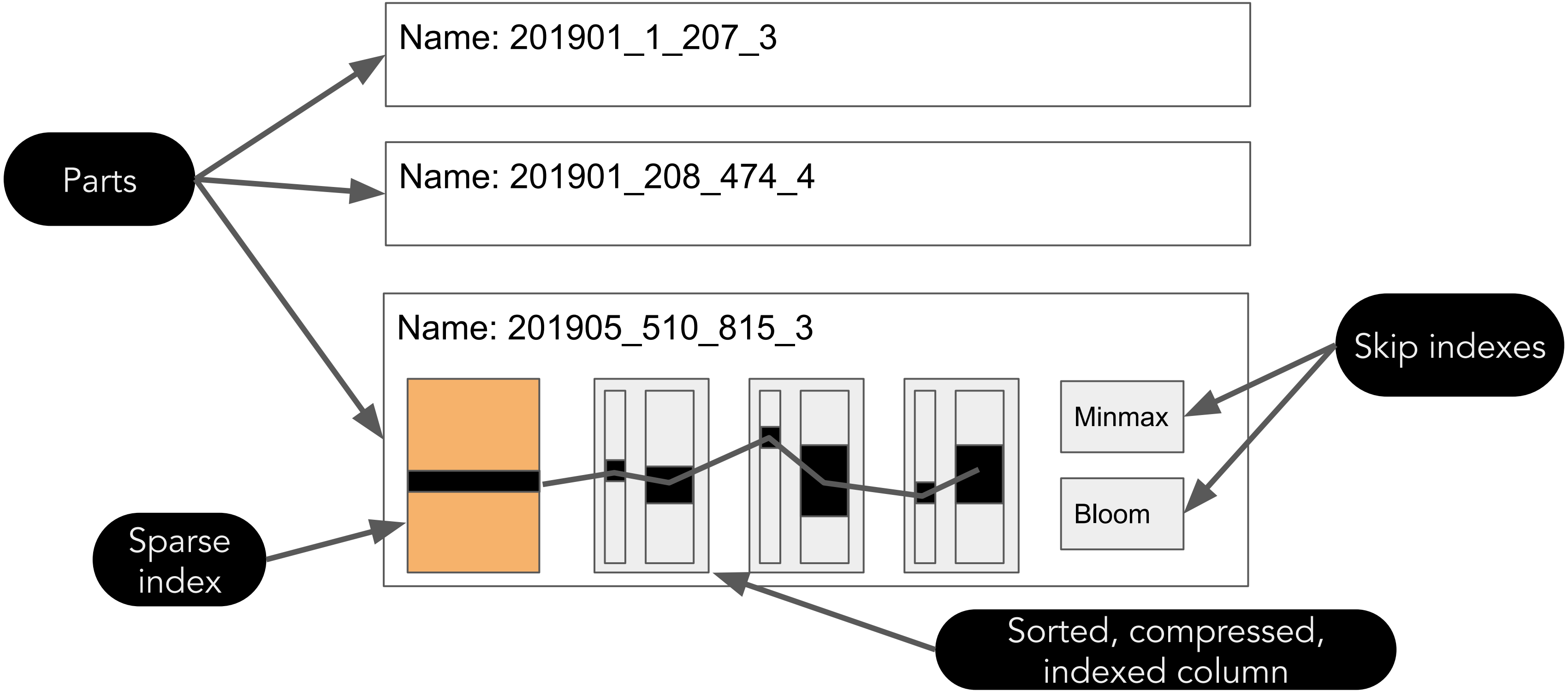
```
/etc/clickhouse-server/config.d/max_connections.xml:
<clickhouse>
   <max_concurrent_queries>100</max_concurrent_queries>
</clickhouse>
```

Can be changed without restart

Don’t update config.xml directly!

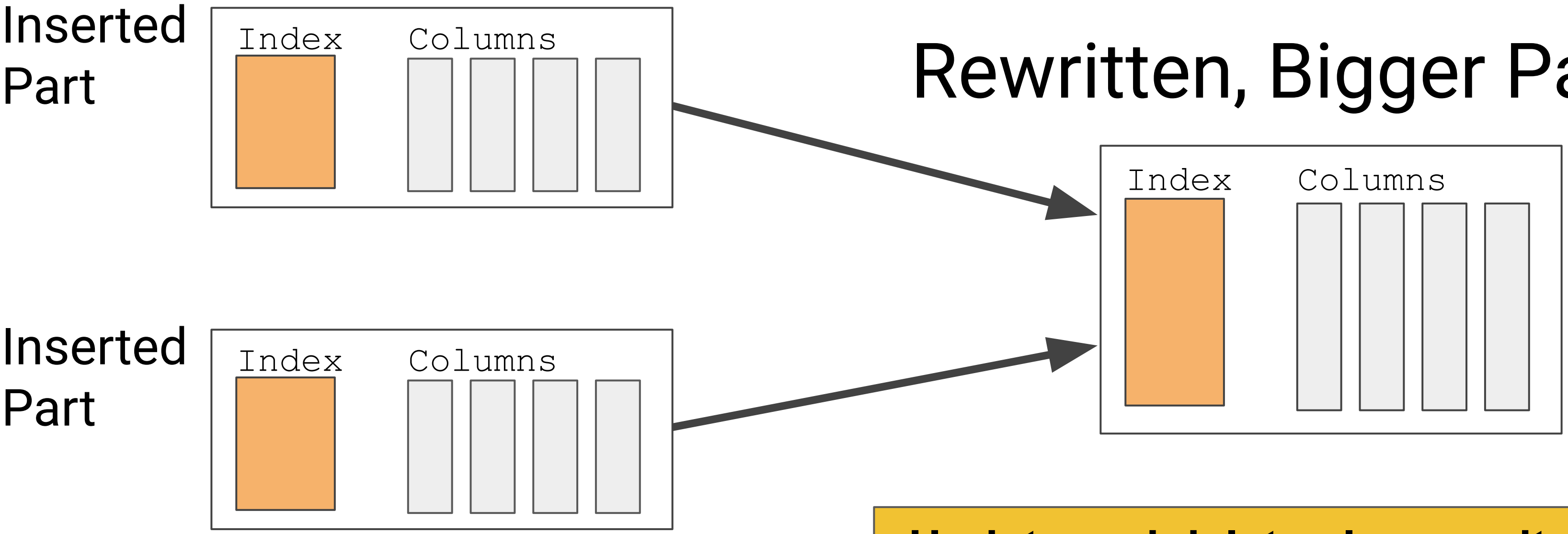
Too Many Parts

Parts are a fundamental feature of MergeTree tables



© 2024 Altinity, Inc.

Why MergeTree? Because it merges!



Update and delete also rewrite parts

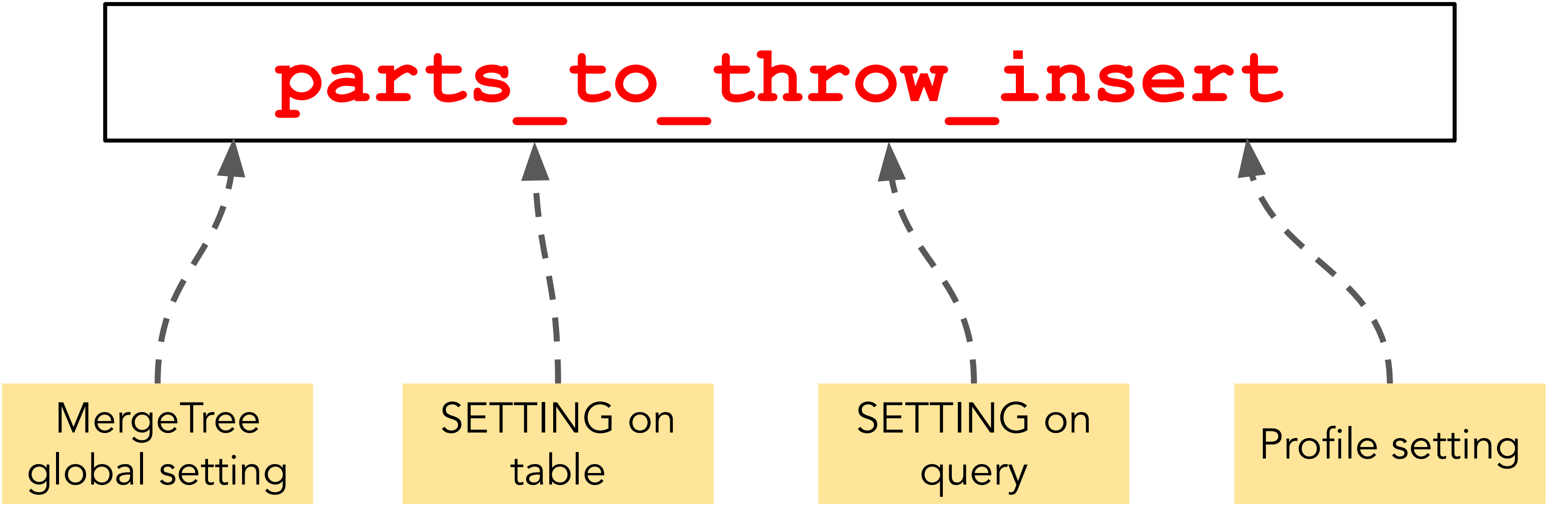
What could possibly go wrong?

```
Code: 252. DB::Exception: Received from localhost:9000.
DB::Exception: Too many parts (4 with average size of 34.19 KiB) in
table 'default.too_many_parts
(d828039d-e2fc-45fd-9962-2f863df2d829)'. Merges are processing
significantly slower than inserts. (TOO_MANY_PARTS)
```

This ain't necessarily so!!

What’s really going on?

Your INSERT hit the limit for parts per partition:



parts_to_throw_insert

MergeTree global setting

SETTING on table

SETTING on query

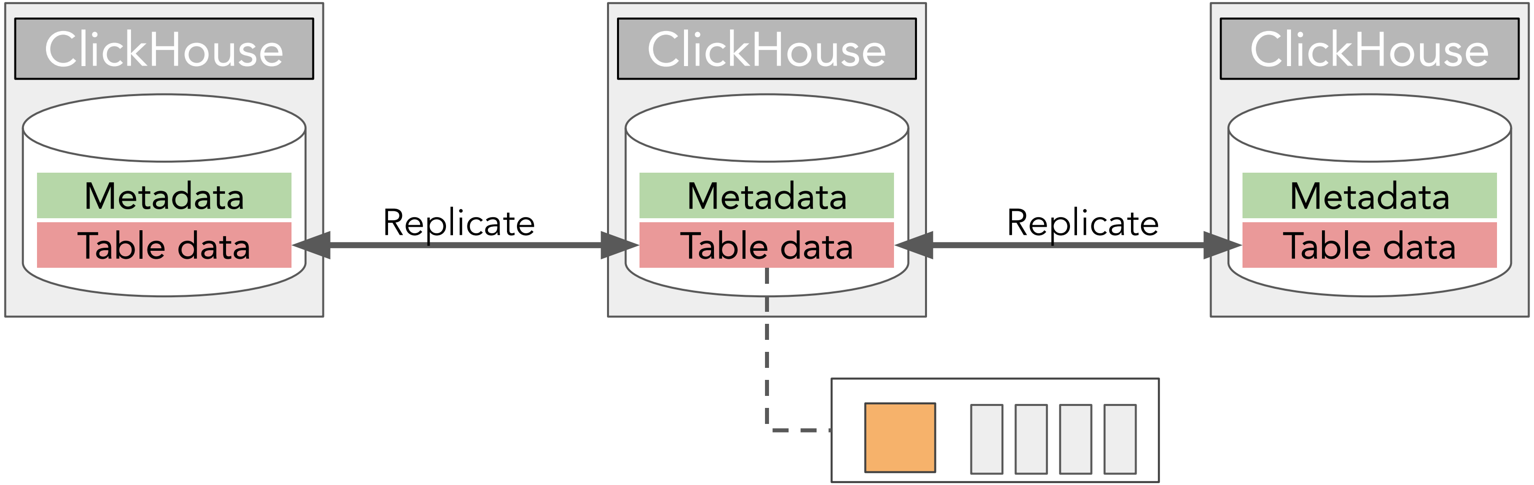
Profile setting

How to fix a TOO_MANY_PARTS error

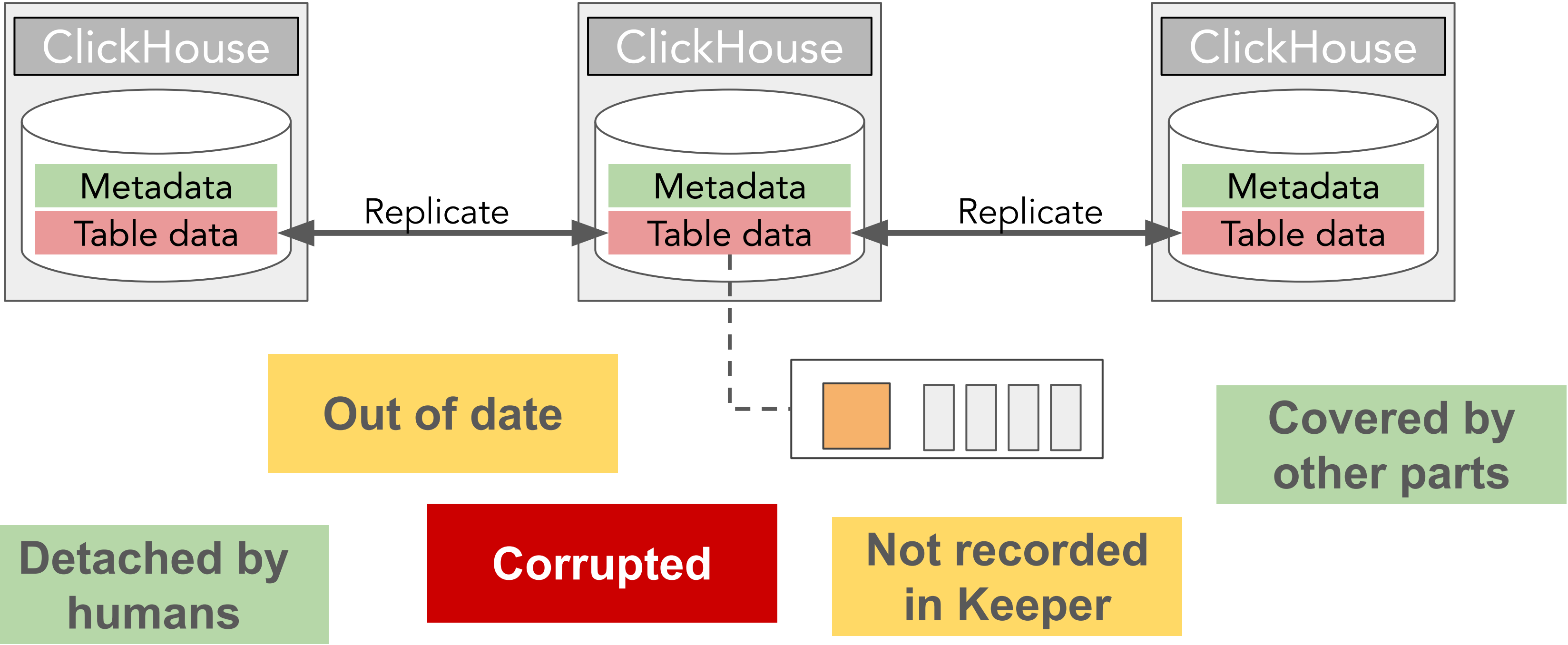
Root cause	What to do
Inserts are too small!	● Insert bigger batches ● Enable async inserts (async_insert = 1) ● Route inserts through Kafka to buffer them
Insert blocks are going to multiple partitions	● Try to match insert blocks to single partitions, modify partition key if necessary
Materialized view has different partitioning from source	● Look for materialized views that spray data across a bunch of partitions
ClickHouse is not merging small blocks fast enough	● Start reading the code for advanced settings like max_bytes_to_merge_at_min_space_in_pool

Detached and Broken Parts

What sort of bad things can happen to table parts?



What sort of bad things can happen to table parts?



Find out about detached parts in system.detached_parts

```
SELECT database, `table`, reason, count()
FROM system.detached_parts
GROUP BY database, `table`, reason
ORDER BY database ASC, `table` ASC, reason ASC

   ┌─database─┬─table─┬─reason───────────┬─count()─┐
1. │ kirpi    │ test  │                  │       1 │
2. │ kirpi    │ test  │ broken-on-start  │       1 │
   └──────────┴───────┴──────────────────┴─────────┘
```

A part that was detached deliberately

A real broken part

Investigating detached parts

```
SELECT `table`, reason, partition_id, name
FROM system.detached_parts
ORDER BY database ASC, `table` ASC, reason ASC
```

```
┌─table─┬─reason───────────┬─partition_id─┬─name─────────────────────────┐
│ test  │ broken-on-start  │ 202401       │ broken-on-start_202401_0_22_2 │
└───────┴──────────────────┴──────────────┴──────────────────────────────┘
```

A “covered” part

```
SELECT `table`, partition_id, name
FROM system.parts WHERE table = 'test'
ORDER BY partition_id, name
```

```
┌─table─┬─partition_id─┬─name─────────┐
│ test  │ 202401       │ 202401_0_22_2 │
└───────┴──────────────┴──────────────┘
```

Fixing problems with detached parts (non-exhaustive list)

Reason	Meaning	What to do
None	Detached by human	● Find out if they were detached for a reason! ● Run ALTER TABLE ATTACH PART
Ignored or covered-by-broken	A bigger part covers the same block(s)	● OK to delete - Run ALTER TABLE DROP DETACHED PART
Cloned	Left over from repairing lost replica	● OK to delete - Run ALTER TABLE DROP DETACHED PART
Broken-on-start or broken	Part has corrupt data	● Safe to delete if it's already covered by another part. (Here is the procedure.)
Unexpected	Not in Zookeeper	● ClickHouse should record it [Zoo]Keeper. If not use SYSTEM RESTORE REPLICA to fix. Don't delete!

What are suspicious parts and do they break ClickHouse?

```
Code: 231. DB::Exception: Received from localhost:9000.
DB::Exception: Suspiciously many (12) broken parts to
remove..
```

ClickHouse sees an unexpected number of broken parts on startup

How to fix it.

1. Raise max_suspicious_broken_parts in config.xml (MergeTree table setting)
2. Add the same setting to the table definition, if you can get on the host.
3. Run `sudo -u clickhouse touch /var/lib/clickhouse/flags/force_restore_data`

Warning! This could be a symptom of a misconfigured system. Check first to ensure you don’t accidentally lose data.

Long-term solutions for broken parts

Broken parts are rare on well-tuned ClickHouse clusters. We see them most commonly when overloaded systems crash. Here are some fixes:

1. Scale up hosts to add resources.
2. Tune queries to make them more efficient.
3. Reduce the number of replicated tables.
4. Avoid heavy mutations on replicated tables.

Stuck Mutations

What's a mutation?? It's how ClickHouse changes tables

Drop or create files

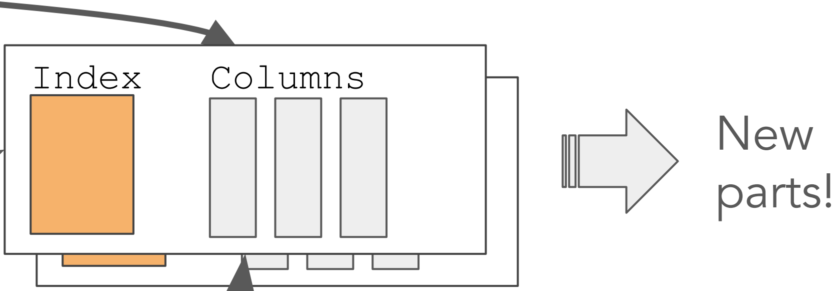
```
ALTER TABLE … MATERIALIZE/DROP INDEX/PROJECTION
ALTER TABLE … DROP/RENAME/CLEAR COLUMN
```

Mutate all columns

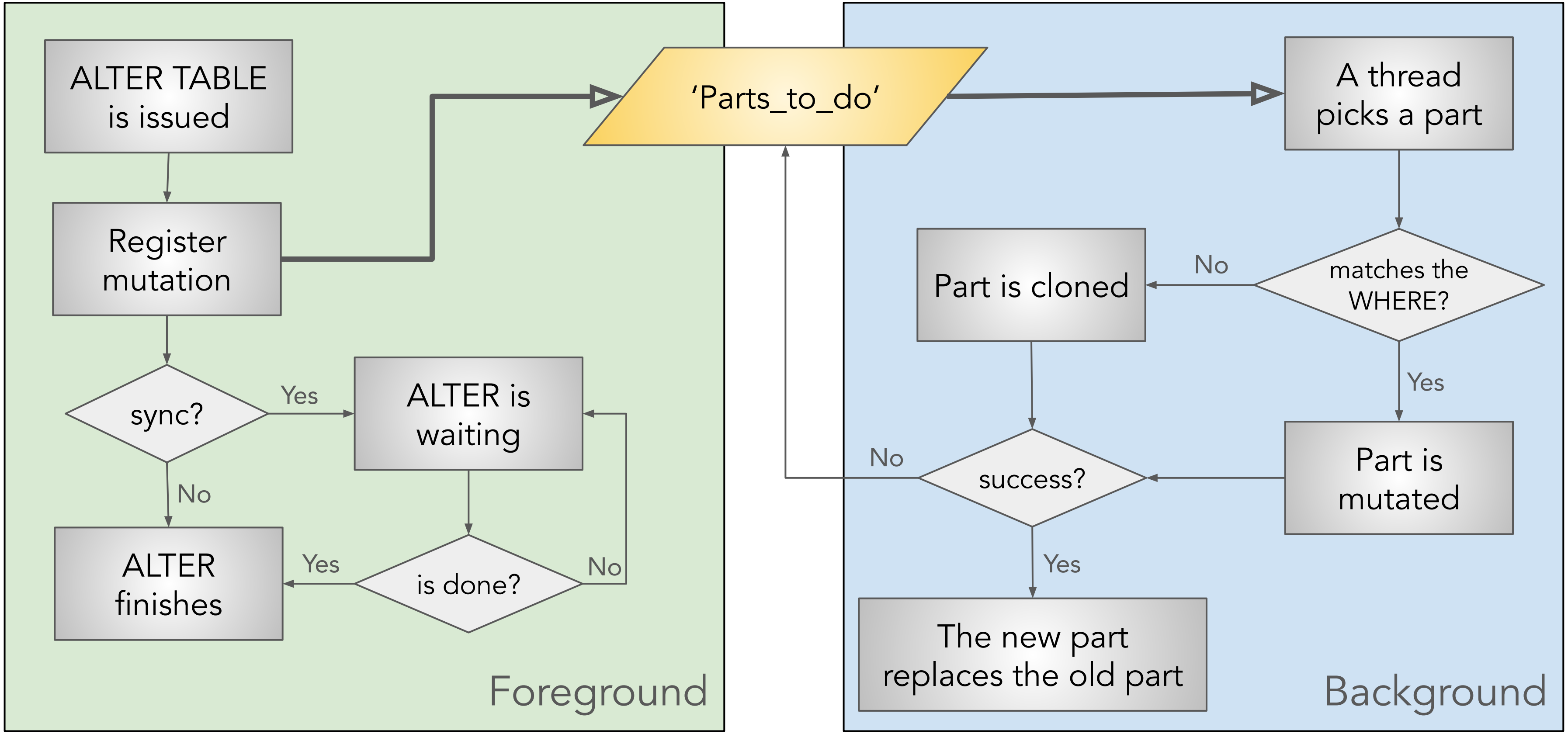
```
ALTER TABLE … DELETE
ALTER TABLE … MATERIALIZE TTL
```

Mutate some columns

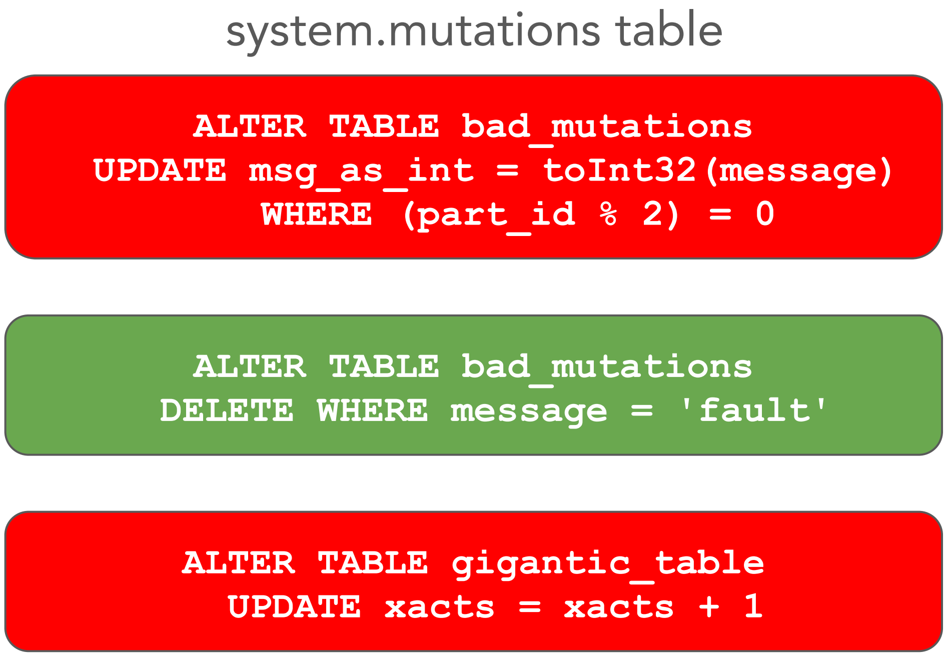
```
ALTER TABLE … UPDATE
ALTER TABLE … MATERIALIZE COLUMN
ALTER TABLE … MODIFY COLUMN (data type)
DELETE FROM … (lightweight delete)
```



How mutations are applied



Three ways mutations can get "stuck"



system.mutations table

```
ALTER TABLE bad_mutations
UPDATE msg_as_int = toInt32(message)
WHERE (part_id % 2) = 0
```

Fails on some parts

```
Code: 6. DB::Exception: Cannot
parse string 'fault' as Int32:
```

```
ALTER TABLE bad_mutations
DELETE WHERE message = 'fault'
```

OK, but blocked by 1st mutation

```
ALTER TABLE gigantic_table
UPDATE xacts = xacts + 1
```

Touches many rows, hence slow

Altinity

What is happening with my mutation?

-- Use system.merges to see if your mutation is running

```
SELECT * FROM system.merges
WHERE is_mutation
```

-- Use system.mutations to check the status

```
SELECT * FROM system.mutations
WHERE NOT is_done
```

-- Use system.mutations to find out if mutations are failing

```
SELECT * FROM system.mutations
WHERE latest_fail_time > toDateTime(0)
```

Time-honored ways to address stuck mutations

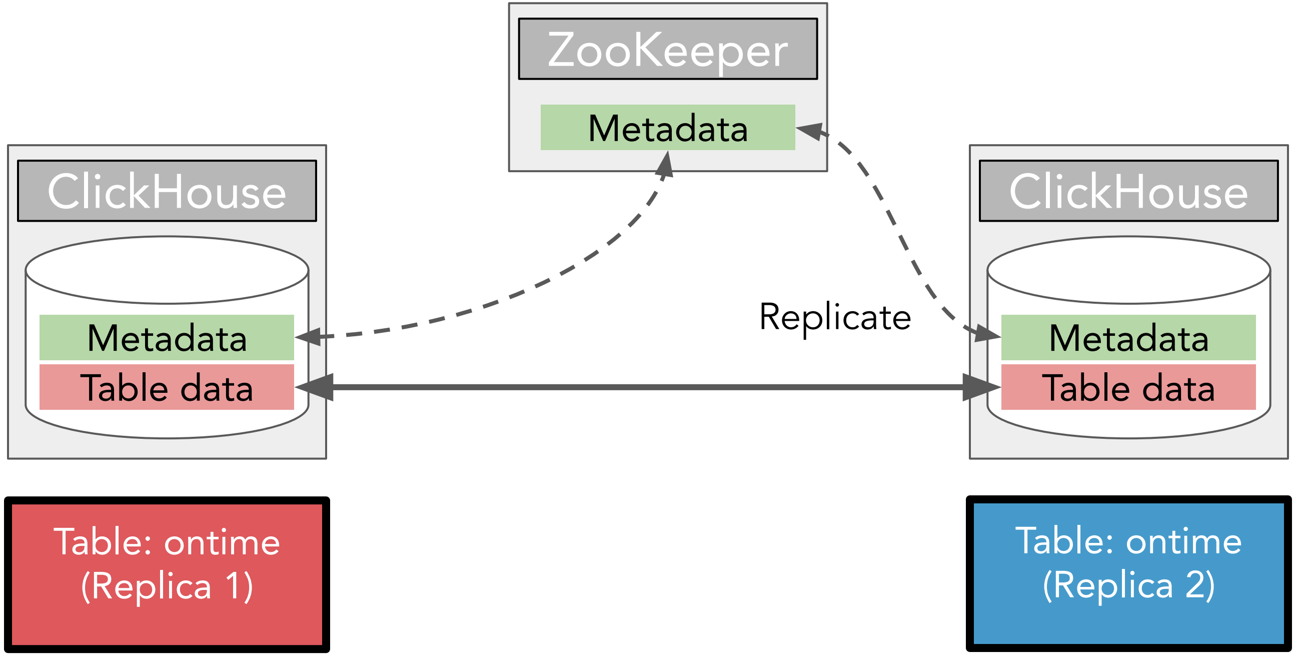
```
KILL MUTATION WHERE database = 'default' AND
  `table` = 'bad_mutations' AND
  mutation_id = 'mutation_141.txt'
```

OR

(Wait.)

Stuck Replication

Review of how replication works



Replication is asynchronous but sequential

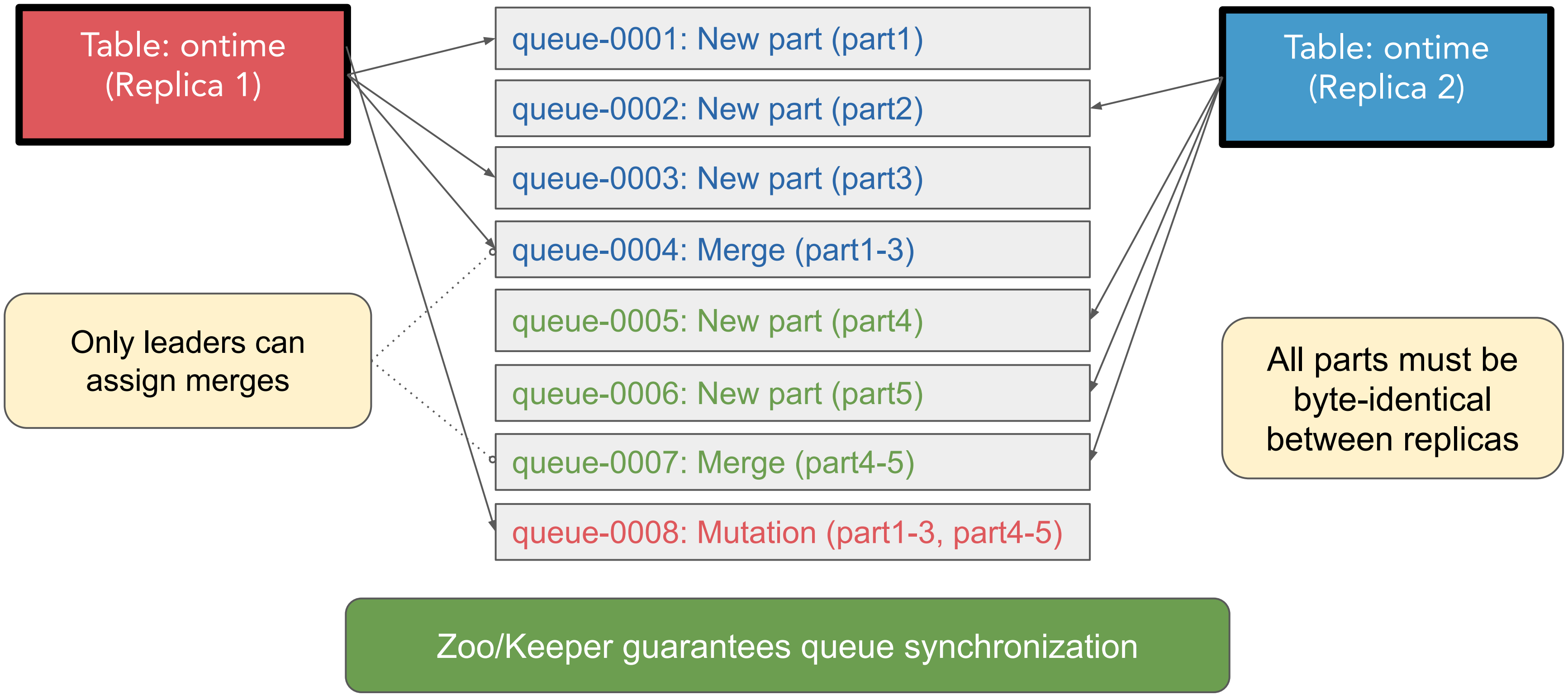


Table: ontime (Replica 1)

queue-0001: New part (part1)

queue-0002: New part (part2)

queue-0003: New part (part3)

queue-0004: Merge (part1-3)

queue-0005: New part (part4)

queue-0006: New part (part5)

queue-0007: Merge (part4-5)

queue-0008: Mutation (part1-3, part4-5)

Table: ontime (Replica 2)

Only leaders can assign merges

All parts must be byte-identical between replicas

Zoo/Keeper guarantees queue synchronization

Checking the health of replicas

```
SELECT replica_name, database, table, is_leader,
is_readonly, total_replicas,
  absolute_delay,replica_path
FROM system.replicas
WHERE `table` = 'bad_replications'\G
Row 1:
──────
replica_name:   chi-mych-clickhouse-mych-0-0
database:       default
table:          bad_replications
is_leader:      1
is_readonly:    0 …
```

Altinity

Checking if ZooKeeper is working

```
SELECT * FROM system.zookeeper_connection
```

```
Row 1:
──────
name:                           default
host:                           172.20.27.167
port:                           2181
index:                          0
connected_time:                 2024-10-15 02:28:53
session_uptime_elapsed_seconds: 129973
is_expired:                     0
keeper_api_version:             0
client_id:                      12
enabled_feature_flags:          ['FILTERED_LIST','MULTI_READ']
```

Digging deeper into ZooKeeper health

```
SELECT count()
FROM system.zookeeper
WHERE path =
'/clickhouse/tables/b11580d5-988a-4958-9472-cd88fd758547/0/r
eplicas/chi-mych-clickhouse-mych-0-0/'
```

```
   ┌─count()─┐
1. │      13 │
   └─────────┘
```

Check state of the replication queue

```
SELECT
    count() AS entries,
    countIf(last_exception != '') AS entries_with_errors
FROM system.replication_queue
```

```
   ┌─entries─┬─entries_with_errors─┐
1. │       6 │                   6 │
   └─────────┴─────────────────────┘
```

See also: https://kb.altinity.com/altinity-kb-setup-and-maintenance/altinity-kb-replication-queue/

Replication problems and solutions

Problem	How to fix
Replicas are read-only	● Make sure ClickHouse can see [Zoo]Keeper ● Restore the replica to replace ZooKeeper metadata
Replication queue blocked by mutations	● Check for large mutations in system.mutations ● Check for stuck/broken mutations (and delete them)
Old / stuck tasks in queue	● Check for old tasks with created_time > 24 hours old ● Look for high values for num_tries or num_postponed ● Delete stuck MUTATE_PARTS and MERGE_PARTS tasks
Replication overloaded	● Add resources to your ClickHouse clusters (bigger VMs!) ● Wait. It cures a lot of problems.

See also: https://kb.altinity.com/altinity-kb-setup-and-maintenance/altinity-kb-check-replication-ddl-queue/

Lost Replicas

What is a lost replica?

The Keeper log stores the last 1000 operations per table. If replica is offline longer than the lifetime of this queue it may lose its position.

ClickHouse marks a replica as lost when the replication queue gets out of date.

The replica has to poll an active replica table to find out which parts it is missing. This normally happens automatically.

But sometimes you may need to help ClickHouse by restoring the replica. This replaces [Zoo]Keeper metadata with information from the disk.

How to recreate ZooKeeper metadata from disk contents

```
-- Prepare to restore.
DETACH TABLE table_name;
-- Remove metadata
SYSTEM DROP REPLICA 'replica_name' FROM ZKPATH '/path_in_zk';
-- Bring table back in read-only mode.
ATTACH TABLE table_name;
-- Detach partitions, recreate ZK metadata, attach them again.
SYSTEM RESTORE REPLICA table_name;
-- Wait for replicas to synchronize parts.
SYSTEM SYNC REPLICA table_name;
```

See also: https://kb.altinity.com/altinity-kb-setup-and-maintenance/altinity-kb-check-replication-ddl-queue/

Wrap-up

Fixing ClickHouse broken clusters, wallet sized edition

- ClickHouse on a laptop is hard to break
- Most “interesting” problems happen on heavily loaded systems
- Prevention is the best cure:
  - Tune schema and queries for maximum efficiency
  - Provision ClickHouse clusters with adequate hardware
  - Don’t abuse mutations and avoid having too many replicated tables
- Next step is to wait – many problems fix themselves
- Your last resort: wade in and fix it

Call Altinity to get help!

You too could be an expert at fixing ClickHouse!

https://altinity.com/clickhouse-training/

More reading for an idle hour (or a moment of panic)

- Altinity Knowledge Base (https://kb.altinity.com/altinity-kb-setup-and-maintenance/rbac)
- Altinity blog (https://altinity.com/blog)
- ClickHouse code (https://github.com/ClickHouse/ClickHouse)
- ClickHouse docs (https://clickhouse.com/docs)

Thank you!

Website: https://altinity.com
Slack: https://www.altinity.com/slack

Altinity.Cloud
Altinity Stable Builds
Altinity Kubernetes Operator for ClickHouse
Enterprise Support for ClickHouse