Mnesia ACL schema migration

Ilya Averyanov

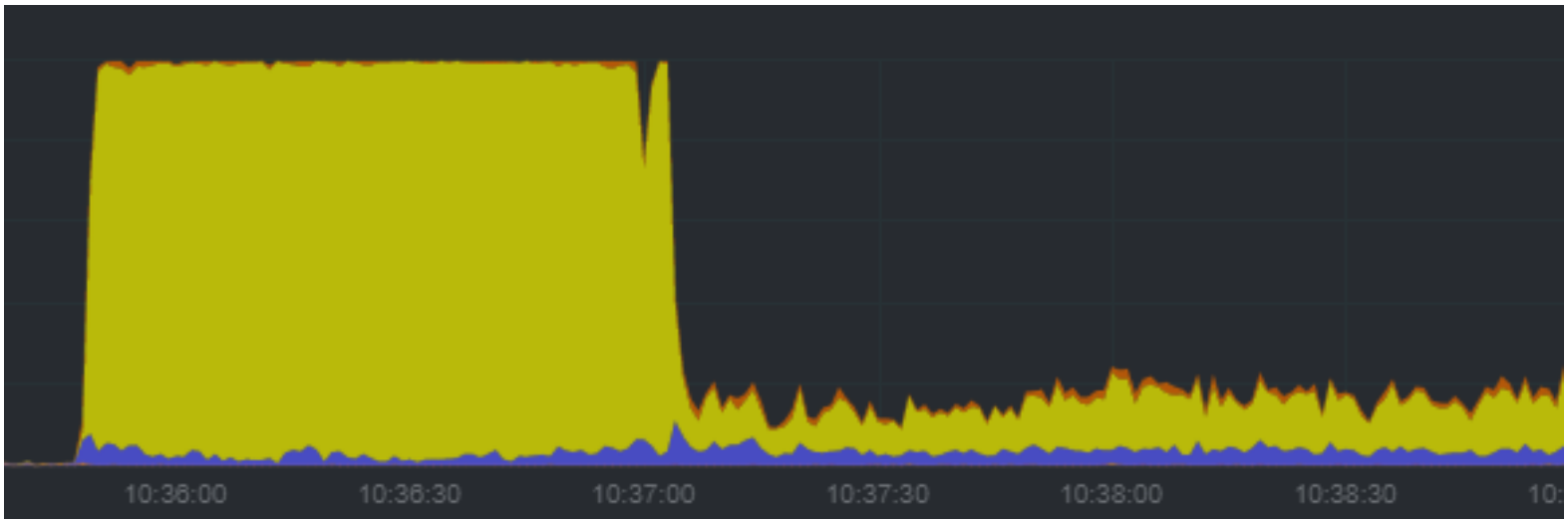
2021

Initial problem
100% High CPU on v4.3.6 vs v4.2.6

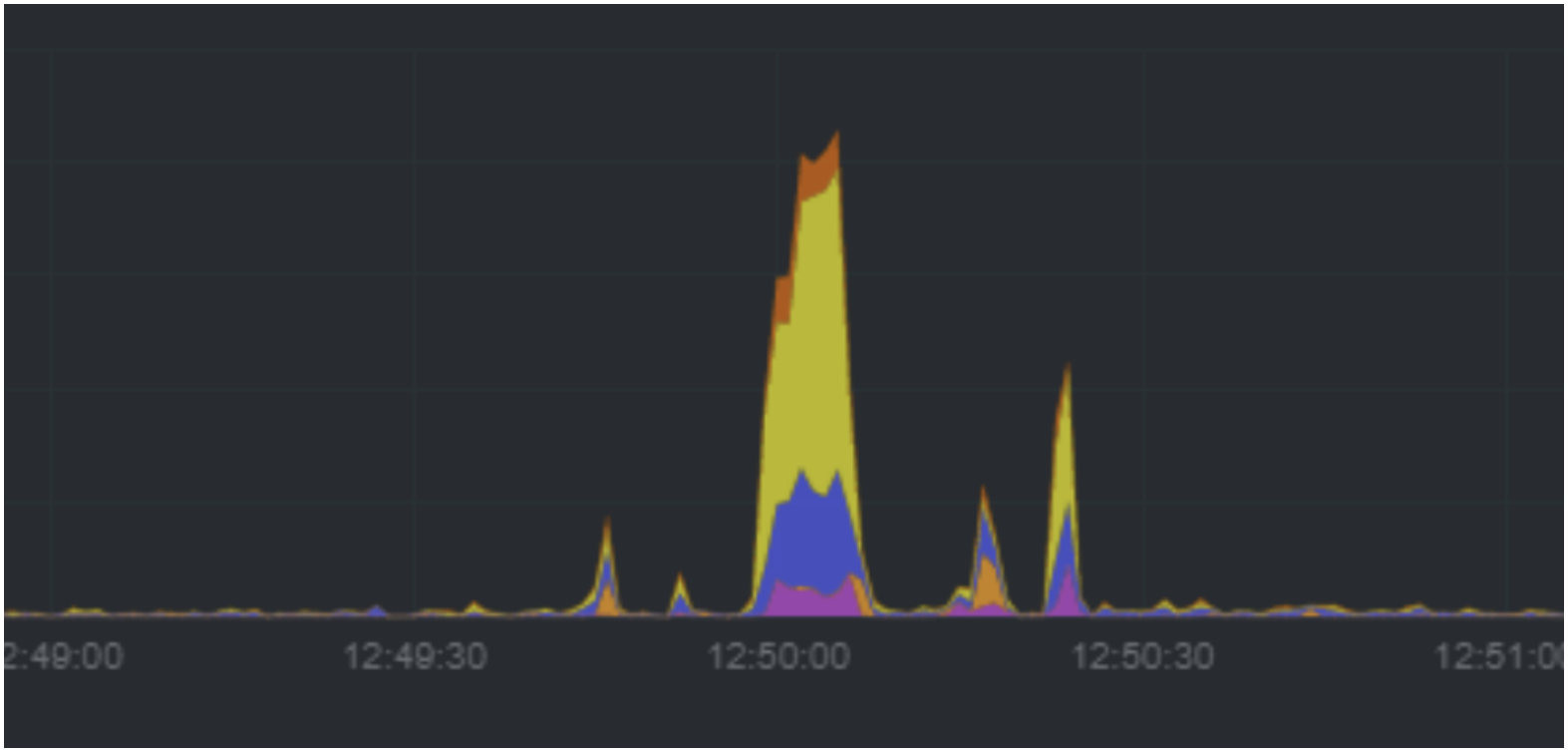
- https://github.com/emqx/emqx/issues/5506
- User has 5000 clients
- Clients connect and subscribe simultaneously
- Small CPU usage peak on 4.2
- Long lasting 100% CPU flatline on 4.3

Initial problem

v4.3.6, reproduced by the user



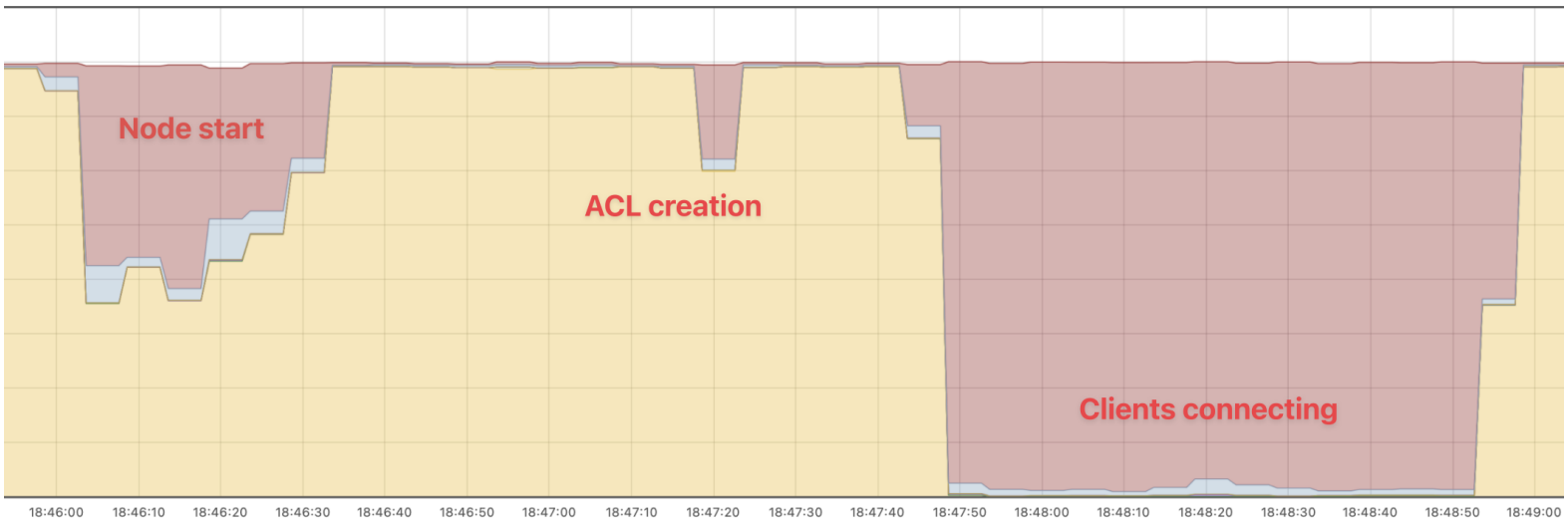
Initial problem
v4.2.6 comparison



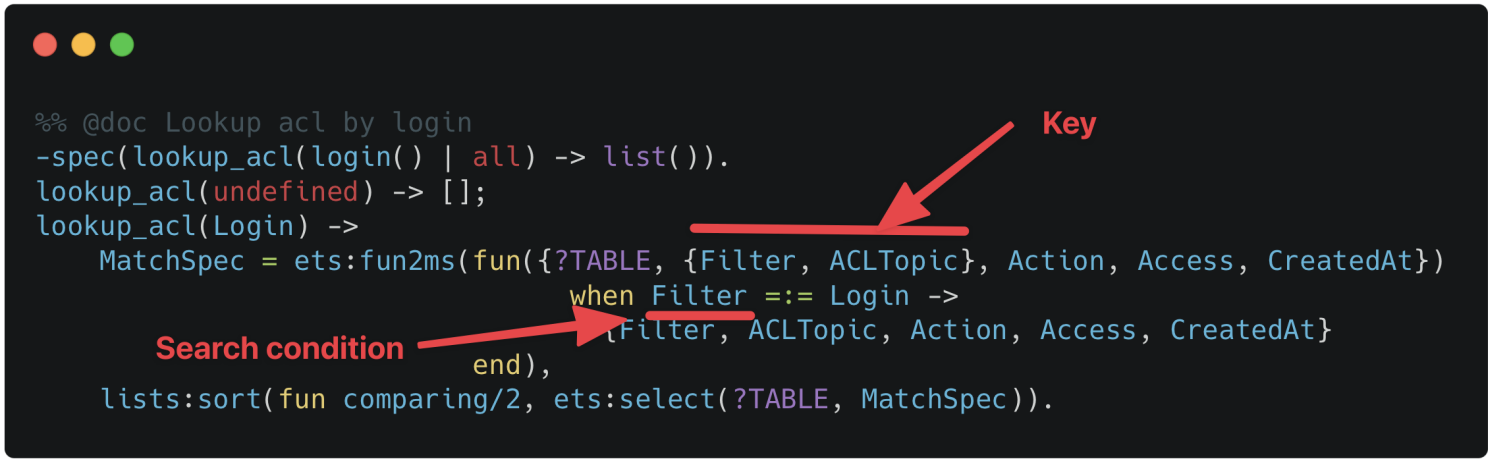
Steps to reproduce

- Write a convenience script to create MQTT users and their ACLs
- Create 5000 MQTT users with ACLs
- Run `emqtt-bench` with `-c 5000 -i 0`

Steps to reproduce



So what's the problem?



So what's the problem?

- We have **bag** table with `{UserId, Topic}` key
- We search by `UserId`, i.e. part of the key
- So full `ets` scan is done every time

So what's the problem?

```
(emqx@127.0.0.1)17> timer:tc(fun() -> emqx_acl_mnesia:check_acl(#{clientid => <<"cid">>,
username => <<"u1">>}, publish, <<"/a/2/a">>, undefined, undefined) end).
{79289,ok}

(emqx@127.0.0.1)18> timer:tc(fun() -> emqx_acl_mnesia:check_acl(#{clientid => <<"cid">>,
username => <<"u1">>}, publish, <<"/a/1/a">>, undefined, undefined) end).
{74343,{stop,deny}}

(emqx@127.0.0.1)19> timer:tc(fun() -> emqx_acl_mnesia:check_acl(#{clientid => <<"fake">>,
username => <<"fake">>}, publish, <<"/some/topic">>, undefined, undefined) end).
{76000,ok}
```

How to fix?

- Since table type is **bag** we can't use partial scans to take advantage of ordering
- Since there actually may be duplicates we can't somehow change `ets` type for the table

How to fix?
We need to change schema

- We should have `UserId` as key, like in 5.x or 4.2 branch
- The fix is simple
- But...

How to fix?

We need to change schema on the fly

- We should introduce optimization during cluster upgrade as well
- There may be present new nodes and old nodes in the cluster
- ACL rules can be added through `emqx_ctl` or API on an **old** node, and **new** nodes should take them in account
- ACL rules can be added through `emqx_ctl` or API on a **new** node, and **old** nodes should take them in account

The plan
Migration part

We introduce a new `migrator` process which is started during cluster upgrade or when a new node starts.

- It creates new `emqx_acl2` table to eventually hold all ACLs from old `emqx_acl` table
- Starts to check constantly wether **all nodes** of the cluster have `migrator` processes
- When this is true, it **puts a tombstone** (a special record) into old `emqx_acl` table
- Starts to move records from `emqx_acl` to `emqx_acl2`, each one in transaction. Several nodes may be doing this simultaneously
- Finally checks that the old `emqx_acl` table is empty (has only the tombstone) and hibernates

The plan

Perspective of ACL database client module

ACL database client module does not know about migration process, it knows only about the tombstone. While there is **no tombstone**, ACL database client

- Stores new ACL rules **both** into old `emqx_acl` and new `emqx_acl2` tables
- Removes ACL rules from **both** tables
- Retreives ACL rules from **both** tables and combines them

We need to have duplication because at this point there still may be nodes in the cluster knowing nothing about the new table.

The plan

Perspective of ACL database client module

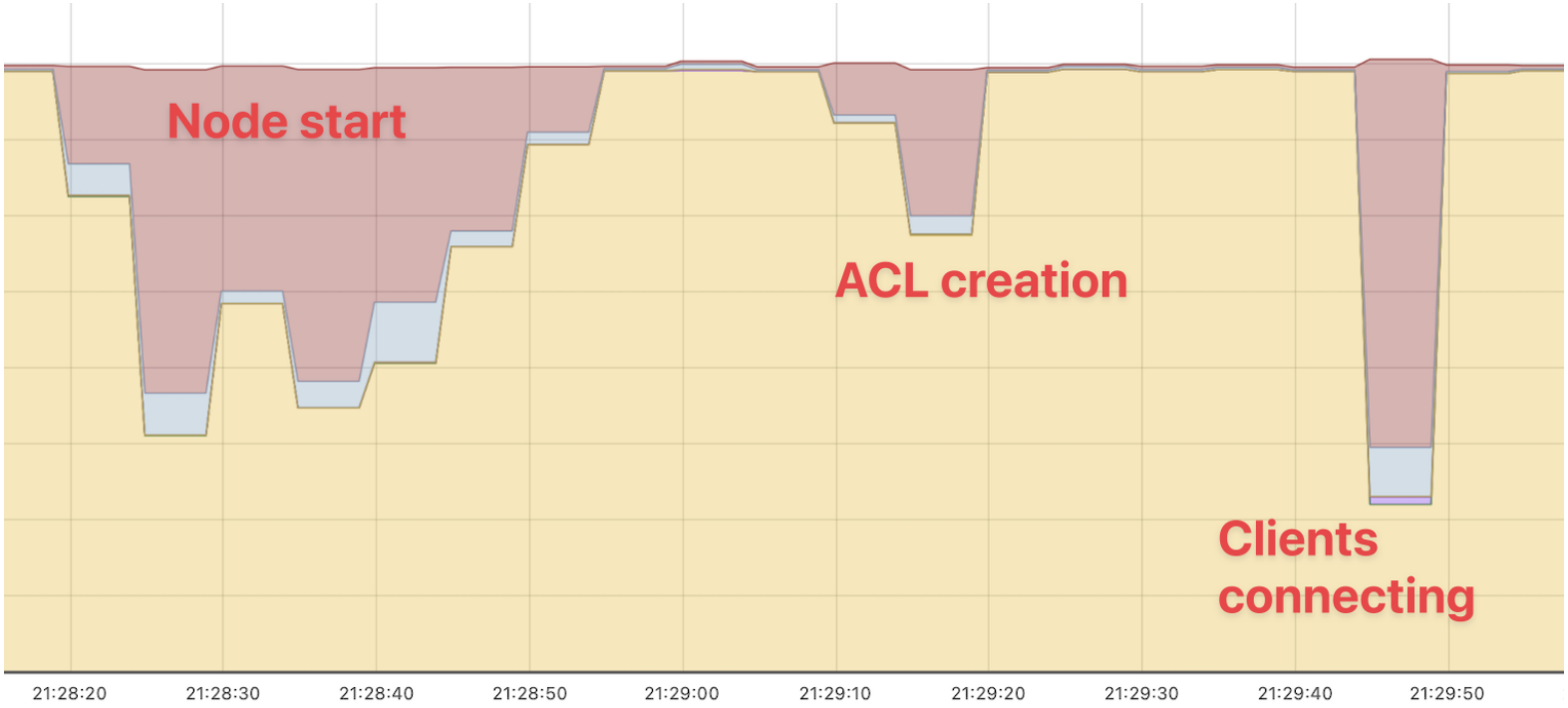
When there is the **tombstone**, ACL database client

- Stores new ACL rules into the **new** `emqx_acl2` only. Now there are no old nodes not knowing about `emqx_acl2`.
- Removes ACL rules from both tables. This is no-op after the migration is over.
- Retrieves ACL rules from both tables and combines them. After the migration is over old table has no records, and this is also a no-op.

The plan
appup

```
{<<"4.3.[0-3]">>, [
     {add_module,emqx_acl_mnesia_db},
     {add_module,emqx_acl_mnesia_migrator, [emqx_acl_mnesia_db]},
     {update, emqx_auth_mnesia_sup, supervisor},
     {apply, {emqx_acl_mnesia_migrator, start_supervised, []}},
     {load_module,emqx_auth_mnesia_api, brutal_purge,soft_purge,[]},
     {load_module,emqx_acl_mnesia, brutal_purge,soft_purge,[]},
     {load_module,emqx_acl_mnesia_api, brutal_purge,soft_purge,[]},
     {load_module,emqx_acl_mnesia_cli, brutal_purge,soft_purge,[]}
   ]},
```

Local test

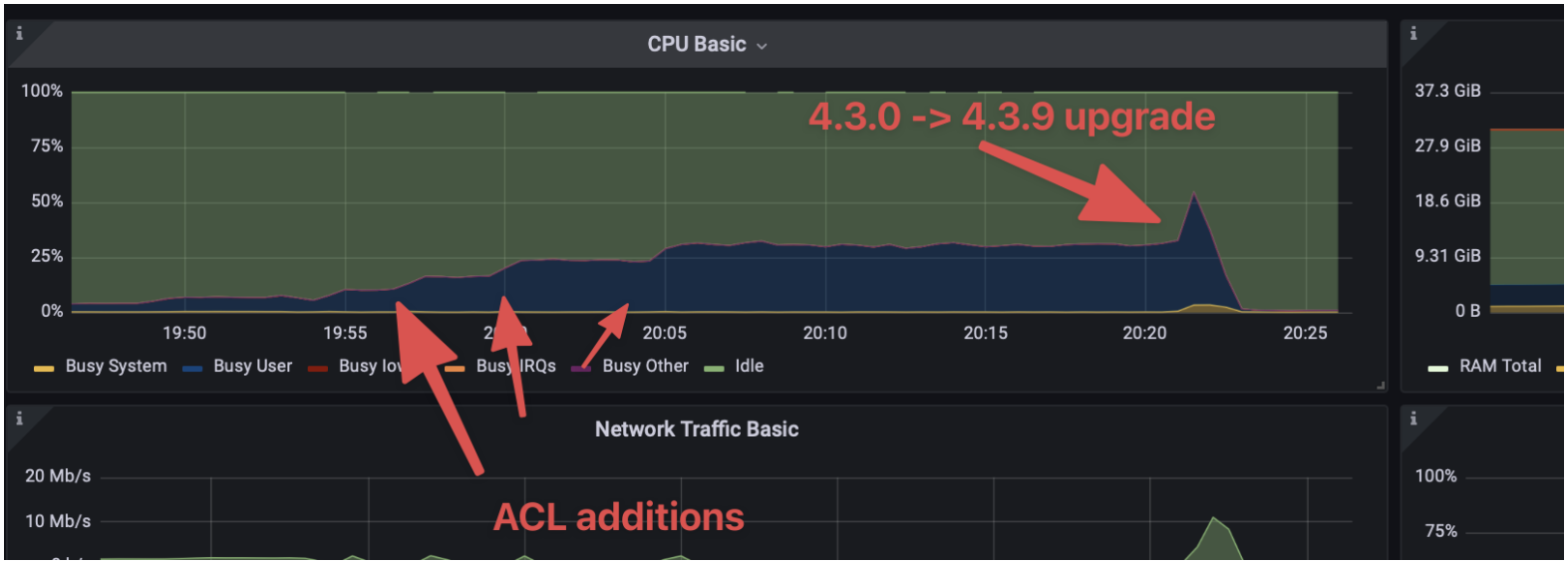


AWS Cluster test

https://github.com/savonarola/emqx-acl-migration-test

- Virtual environment created with EMQX CDK
- 5 nodes
- 100000 connected clients
- 100000 ACL records

AWS Cluster test



AWS Cluster test

- The more ACL records we have, the more CPU is consumed by pubs, even though ACL records are not related to the publishing clients
- We have a CPU consumption peak during migration
- CPU usage is greatly reduced after the migration

Thank you!