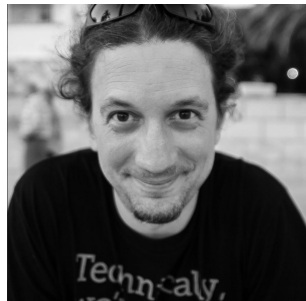




PERCONA
LIVEONLINE
MAY 12 - 13th
2021

Pandemic ! A tale of 25x growth

Hello!



I am Miklos “Mukka” Szel

Senior MySQL Architect @ Edmodo

miklos.szel@edmodo.com

Hello!



I am Natarajan Chidhambharam (Nat)

Infrastructure engineer (Database) @ Edmodo

nat@edmodo.com

Edmodo



- Edmodo is a **global education network** that
 - gives teachers the tools to share engaging lessons,
 - keeps parents updated,
 - builds a vibrant classroom community.
- Founded in 2008.
- Edmodo has more than 140 million users.
- [edmodo](https://edmodo.com)

Infrastructure setup @ Edmodo



- Hosted on AWS.
- DB - Percona Server with async replication.
- 200.000 Queries Per Second peak (Back to school period)
- 13 MySQL clusters and 34 MySQL EC2 instances (13 Primaries, 21 replicas)

r5.large	2
r5.xlarge	4
r5.2xlarge	18
r5.4xlarge	8
c5.4xlarge	2

- App servers (Dockerized) in AWS Auto Scaling Groups



*We've seen a significant increase in usage this week. [..] At a guess, this is probably due to closures and restrictions imposed due to the **ongoing viral outbreak in the world.***

Chris Dunn (Director of data/ops/sec)

Growth - traffic in March 2020

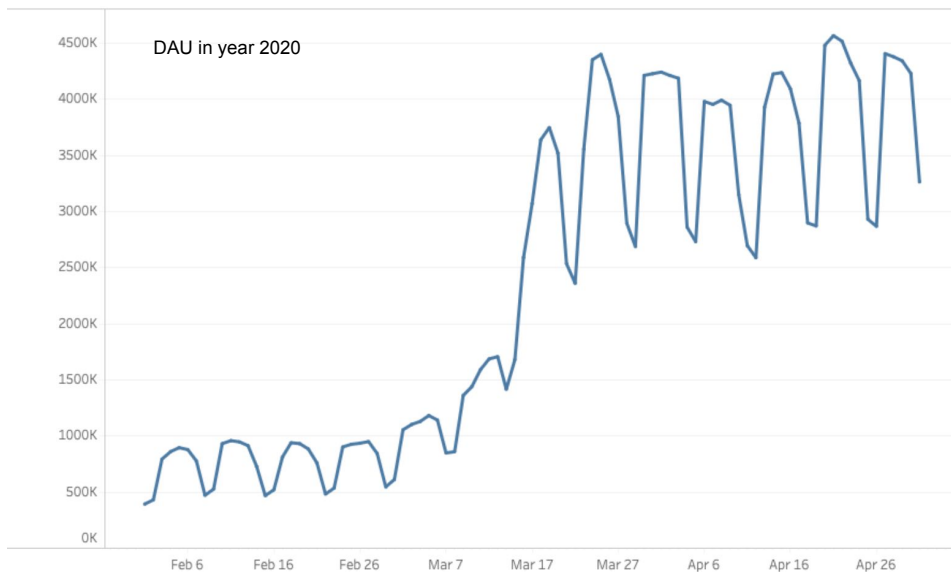
Before March 2020

- DAU - 1.1 million
- 250M req/ day
- Peak - 216K req/min
- DB Peak - 200K QPS

End of March 2020

- DAU - 4.6 million
- 2.5B req/ day
- Peak - 3M req/min
- DB Peak - 5M QPS

This required a 15x increase in capacity!



Beginning - DB Monitoring

- On premise monitoring
 - Percona Monitoring and Management 2.x
 - Nagios
- VividCortex
- pt-stalk
- AWS cloudwatch



VividCortex

Nagios[®]



Amazon CloudWatch

Beginning - Config Management

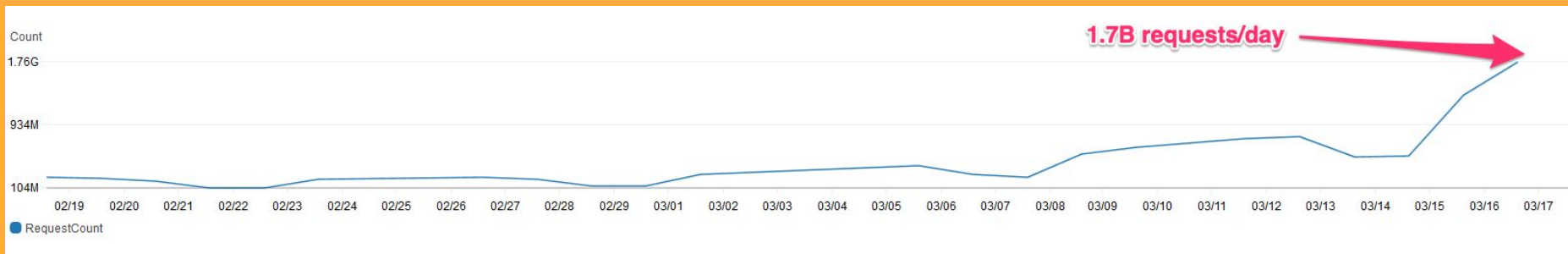
- Ansible

- Dynamic inventory based on ec2 tags
- Security and OS tuning
- ec2/ebs deployment and config
- MySQL, user/replication setup
- Restore from different sources (s3 object, xtrabackup stream from a donor machine)
- PMM/nagios/pt-stalk setup
- Slack Notification



ANSIBLE

Traffic is skyrocketing !



What did we do about it ?

Scaling for reads

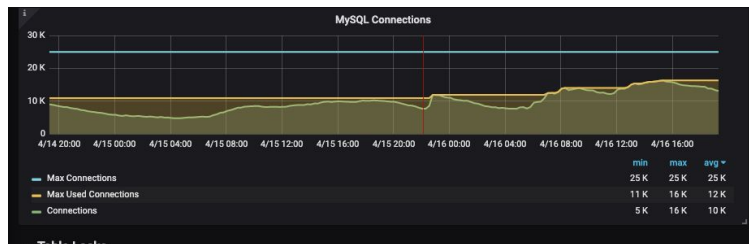
- Started adding new readers on a daily basis
- Deployment with Ansible, warmup manually
- This helped for a couple of days

Connection pooling

- Backend is written in Ruby On Rails
- It overwrites the default wait_timeout

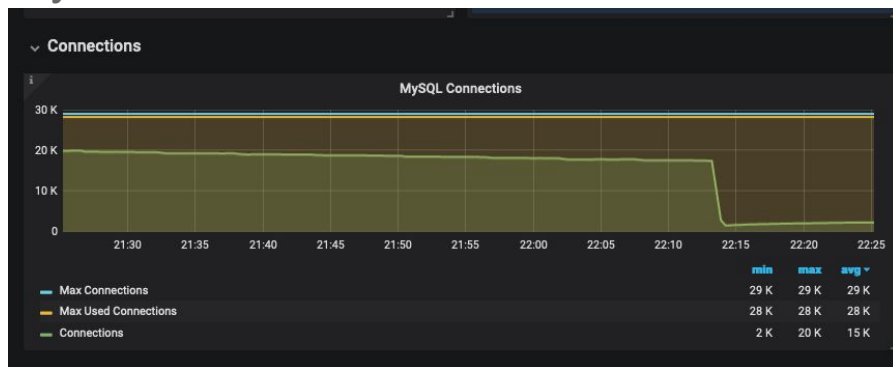
```
SET NAMES utf8mb4, @@SESSION.sql_auto_is_null = 0, @@SESSION.wait_timeout = 2147483,  
@@SESSION.sql_mode = 'STRICT_ALL_TABLES';
```

- Increase max_connections on all clusters
- Periodically killing the connections where WHERE command = 'Sleep' and time >600

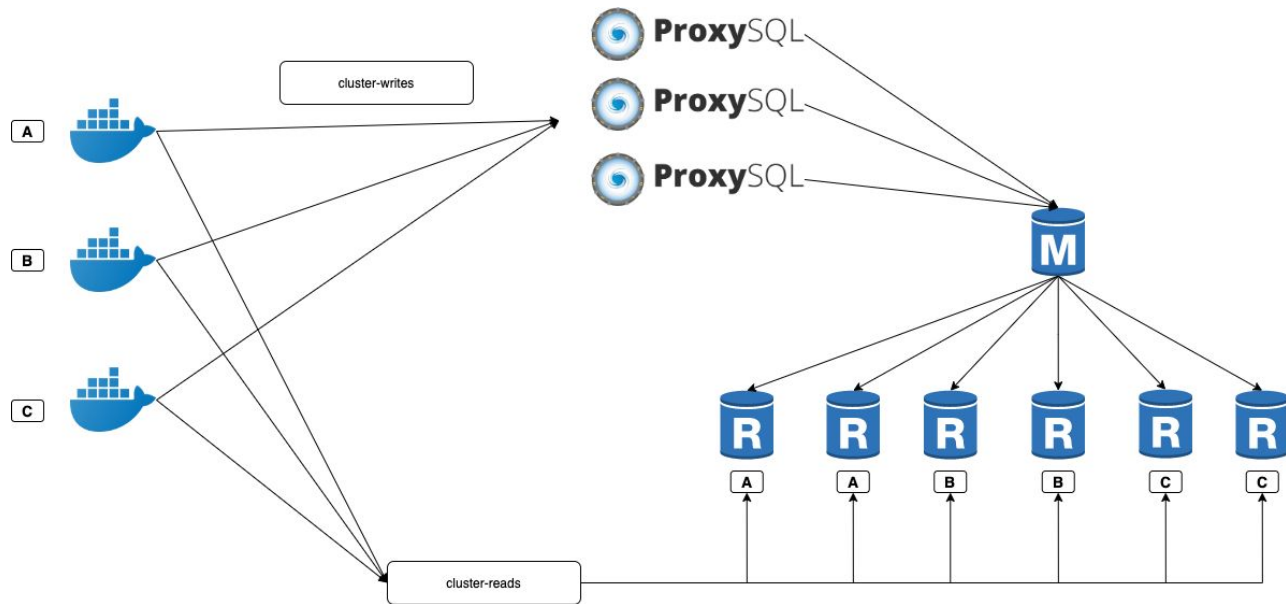


Connection pooling - contd.

- At around 2 million QPS increasing the max_connection is not an option anymore
- Source servers start crashing
- Adding ProxySQL a little before the last moment



Setup



Config management

- Support for deploying ProxySQL and ProxyWeb
- Added several tweaks to our Ansible playbook
 - We backup the buffer pool and restore it during deployment
- Machines are fully production ready if the deployment is successful (buffer is warm, no replication lag)
- Restore from s3 objects (start as many readers as we want)
- Restore/warm up time for a 2TB instance is between 3-4 hours

ProxyWeb

- Our recently open sourced internal tool to visualize ProxySQL

The screenshot displays the ProxyWeb web interface. At the top, a blue navigation bar contains the 'ProxyWeb' logo and several tabs: 'disk', 'main', 'monitor', 'stats', 'stats_history', 'Misc', and 'proxysql_standalone'. The 'main' tab is currently selected. Below the navigation bar, a dropdown menu is open, listing various configuration categories such as 'global_variables', 'mysql_query_rules', 'mysql_replication_hostgroups', 'mysql_servers', 'mysql_users', 'proxysql_servers', 'runtime_checksums_values', 'runtime_global_variables', 'runtime_mysql_query_rules', 'runtime_mysql_replication_hostgroups', 'runtime_mysql_servers', 'runtime_mysql_users', 'runtime_proxysql_servers', and 'runtime_scheduler'. To the left of the dropdown, there is a 'SUBMIT' button and a 'Show' dropdown set to 'All' with 'entries' listed below it. On the right side of the interface, the breadcrumb 'main - global_variables' is shown next to a settings icon. Below this, a search bar is present. The main content area features a table with two columns: 'variable_name' and 'variable_value'. The table lists several variables, including 'admin-admin_credentials', 'admin-checksum_admin_varia', 'admin-checksum_mysql_query_rules', and 'admin-checksum_mysql_servers', all of which have a value of 'true'.

variable_name	variable_value
admin-admin_credentials	
admin-checksum_admin_varia	
admin-checksum_mysql_query_rules	true
admin-checksum_mysql_servers	true

Query tuning

- Queries doing otherwise OK started misbehaving
 - Growing traffic
 - Nature of the traffic
 - Table growth
 - Uneven distribution of data
 - Attacks - we started getting more attention
- New - not so well tested - queries due to new feature rollout!
- The (in)famous OPS-7082 ticket with 45 optimization related subtasks

Query tuning - indexing

- The readers' resource usage has been reduced by ~10% by fixing indices.

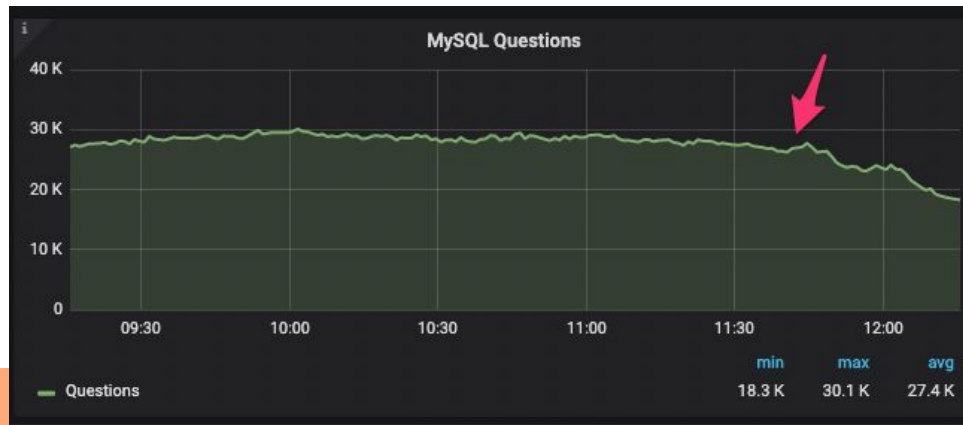


Query tuning - caching

- ProxySQL helps removing connector generated commands like com_ping
- Cached frequent and aggregating queries in the memcached or with ProxySQL
- Stopped running billions of queries this way, mostly aggregating ones
- Traced back 16% of all writes on the most important cluster to a single query:
 - Decreased the frequency of the UPDATE on the frontend by 6x

Query tuning - rerouting

- Offloads select from the writers to the readers - ProxySQL
- Added actual readers to some previously `1 leg` clusters (1 writer-1 failover)
- In case of the backend, offloaded 70% of the reads to the readers
- Did the same for all clusters watching the effect, made scaling for writes possible



Query tuning - rewriting

- Devs worked on application logic changes to:
 - Optimize queries
 - Remove unnecessary code paths
 - Optimize cronjobs started causing issues due to the high concurrency
- Used ProxySQL to rewrite some problematic queries

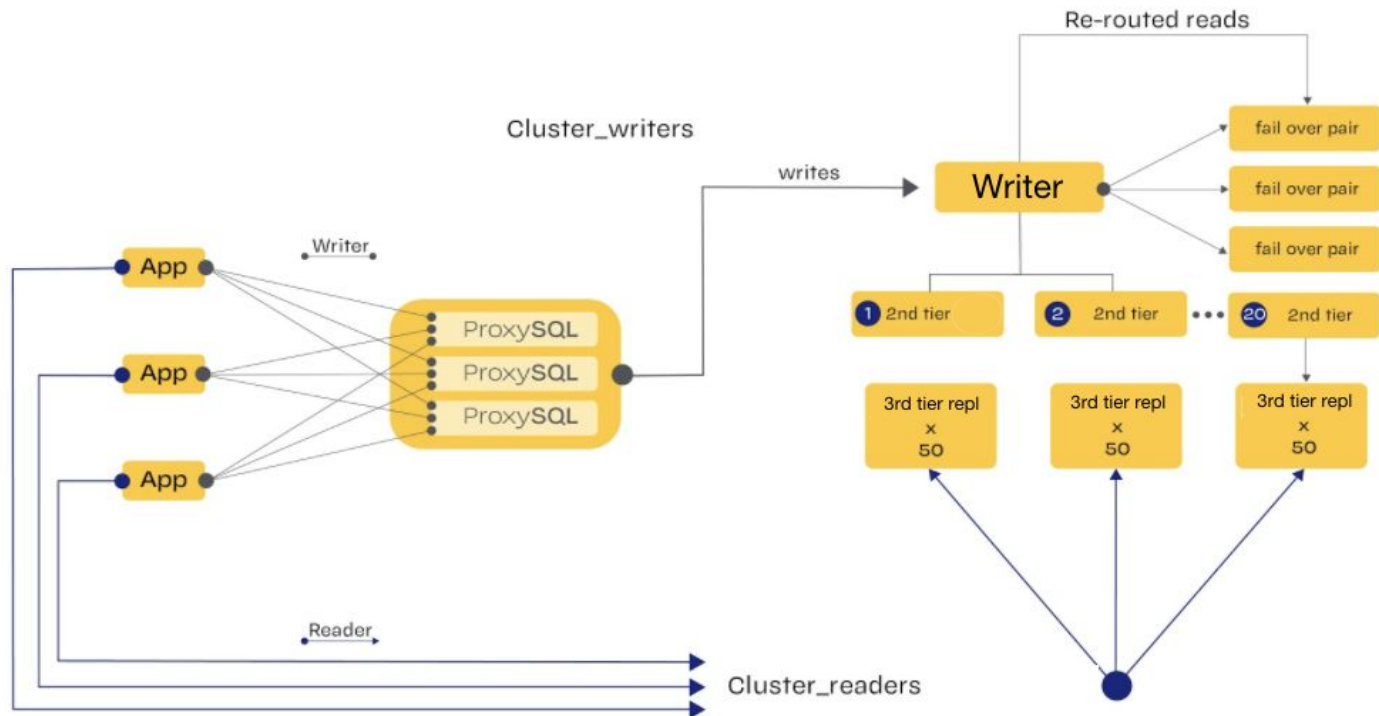
Scaling for writes - split

- Traffic breakdown with ProxySQL's stats_mysql_query_digest
- Moved the most heavily written dbs to their separate clusters using:
 - ProxySQL
 - Chain replication

Replication redesign

- Using bigger instances in order to:
 - Avoid network saturation
 - More RAM - hot data fits - less IO
 - Less gp2 disk needed
- Scaled up to 70 replicas then we moved to 3 tier replication (Ansible, MHA, Nagios changes)
- Adjusting the disks' size on the instances online(EBS) or during the deployment

Final DB flow



Miscellaneous

- Upgraded the HW under PMM several times, then added a new instance
- Prepared for handling 6x more traffic by mid April
- 450 db servers
 - 200 r5.8xlarge in the biggest cluster, 3 tier replication
- Used more than 1PB gp2 disk at the end
- Keeping the traffic within the zones for reads
- Reduced the ProxySQL CPU usage by 90% with running it with `--idle-threads`
- Reduced the `threads_connected` even further with fine tuning `mysql_auto_increment_delay_multiplex`

Current status

- Tens of millions of new users
- New site for a country in a new DC to support the remote learning of 20 million students
- Traffic on the Global site is still 5x of what it used to be

Thanks!



You can find us at:

- <https://www.linkedin.com/in/natarajanct/nat@edmodo.com>
- <https://linkedin.com/in/miklos.szel/miklos.szel@edmodo.com>

THANK YOU !



PERCONA
LIVEONLINE
MAY 12 - 13th
2021