Troubleshooting Relational Database Backed Applications

SWITCH

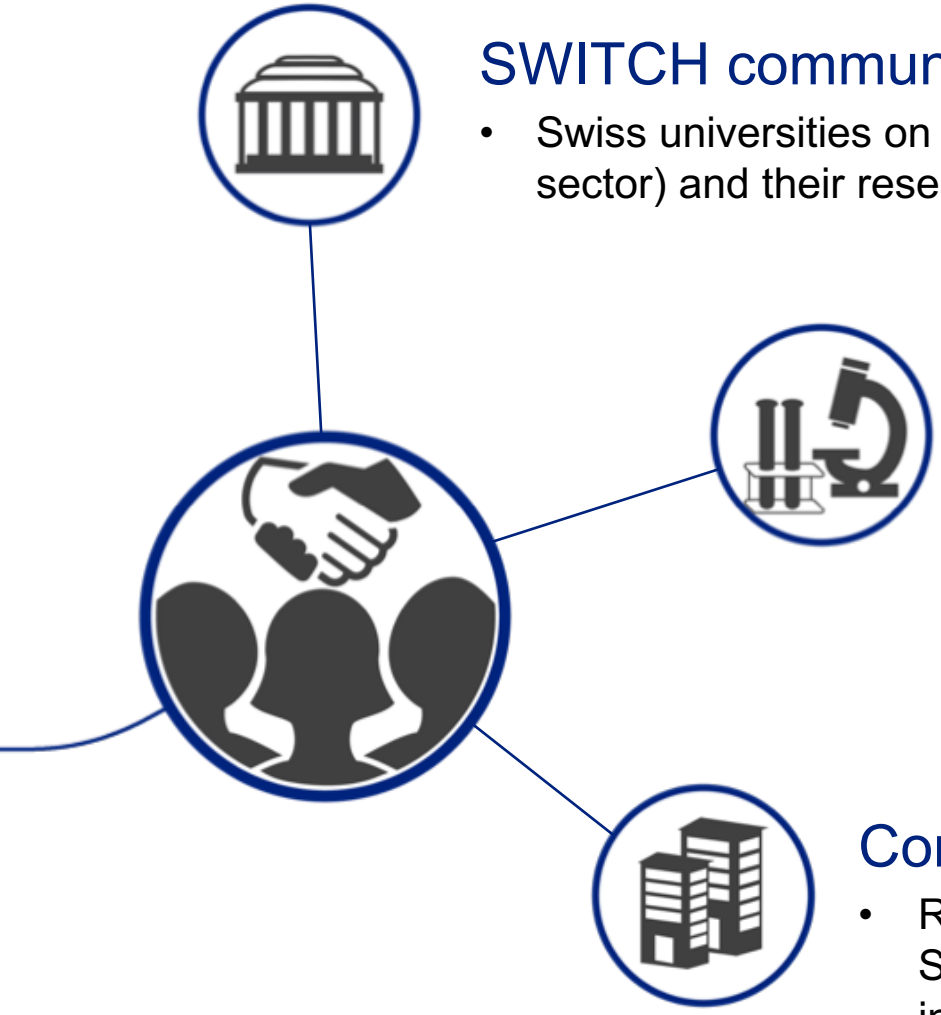
Greg Vernon
greg.vernon@switch.ch

Krakow, 30 January 2018

SWITCH

- We are the Swiss National Research and Education Network.
- We network the Institutes of Higher Education and Research to each other, and the rest of the world.
- We provide additional services such as Federated Authentication, Video, and File Sharing to our Educational customers.
- We manage the Top Level Domains for Switzerland (.ch) and Liechtenstein (.li).
- We provide SWITCH-CERT security service.

Our customers



SWITCH community

- Swiss universities on tertiary level (academic sector) and their research institutions

Extended community

- Other organizations involved in research or education

Commercial customers

- Registrars of .ch- and .li-Domain-Names, Swiss financial institutions, research-related industry and government

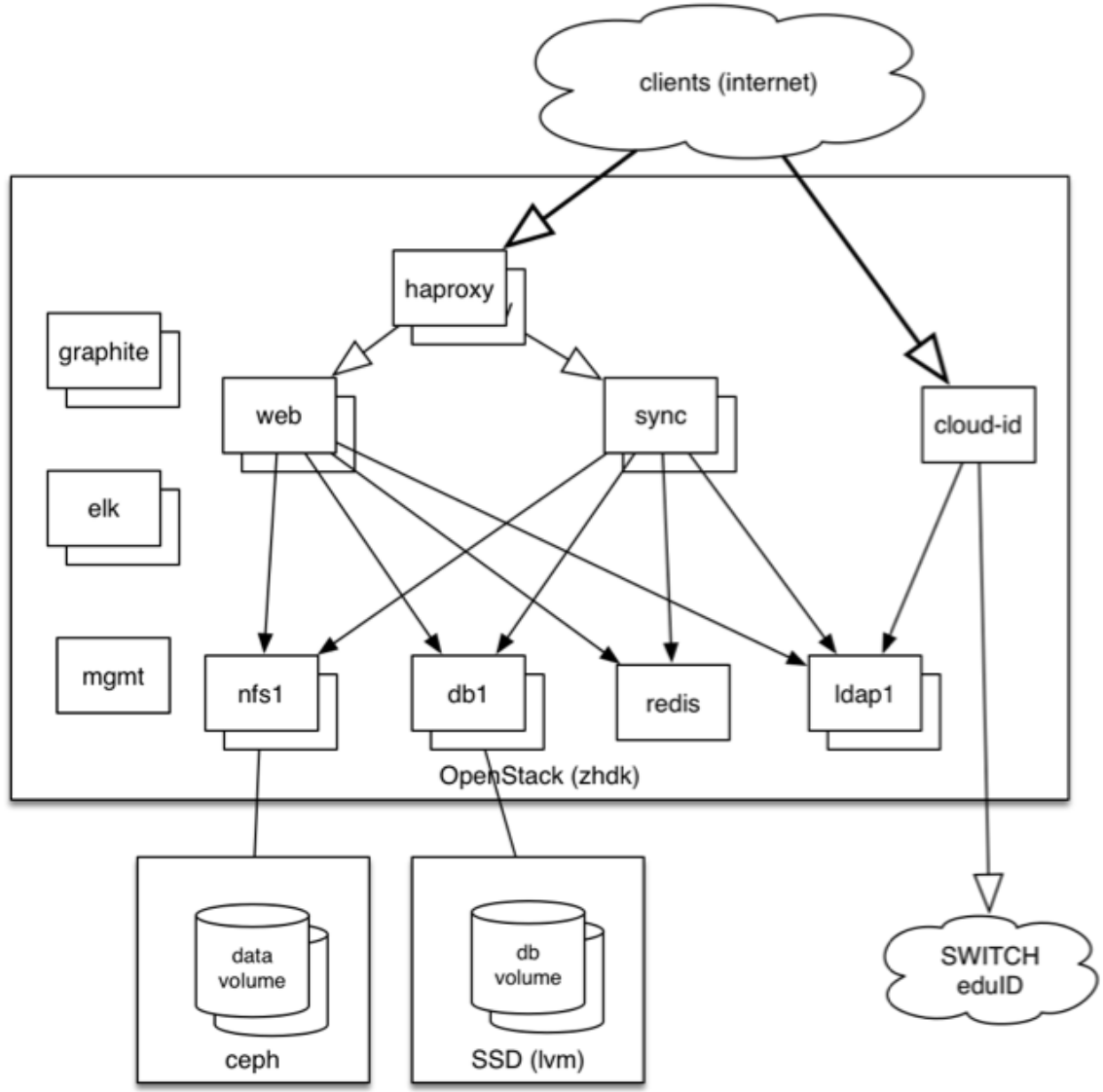
The Problem

- We were seeing corruption in our ownCloud instance.
- Our database servers were struggling. We had to add more read nodes to keep up with the load.
- We saw issues with MaxScale, and suspected that MaxScale had a role in the corruption we were seeing.
- The solution was a team effort between SWITCH and ownCloud.

SWITCHdrive

SWITCHdrive is our branded ownCloud offering. We had the following before the fix:

- About 30,000 Users
- 105,000,000 files
- 100,000,000 rows in our oc_filecache table
- 6 Mariadb servers in a Galera cluster
- 9 Apache Servers(4 Sync/4 Web/1 Management)
- Redis
- 3 LDAP Servers
- 6 NFS servers running atop CEPH
- 2 HAproxy load balancers
- Monitoring (Graphite, ELK)
- Runs atop SWITCHengines, our OpenStack offering
- Most services are Docker containers


clients (internet)
haproxy
graphite
web
sync
cloud-id
elk
mgmt
nfs1
db1
redis
ldap1
OpenStack (zhdk)
data volume
ceph
db volume
SSD (lvm)
SWITCH eduID

Database Environment

- MariaDB 10
- Galera Cluster
- 1 Write Node
- 5 Read Nodes
- MaxScale on the Web/Sync/Mgmt servers for DB connection channeling
- DB nodes are Docker containers

Galera Cluster

- Synchronous replication
- Multi-Master Capabilities, but we use single master with multiple read nodes to increase read bandwidth.
- Automatic node handling, when nodes join or leave the cluster.

MaxScale

- MariaDB Database Proxy
- We use it to split reads from writes.
- Automatic switchover when nodes fail/rejoin.
- All writes go to the master, all reads to the replicas.
- We use version 1.4.
- New version, 2.1, requires purchasing a license for 3 or more server instances in production
- ProxySQL might be our future direction.

Tools to diagnose DB problems

- MaxScale top queries
- Mysql performance_schema
- Explain
- Analyze
- Profiling

MySQL performance_schema

- These are performance tables.
- They contain a lot of information.
- You should truncate the tables as needed to reset the counters.
- Disabled by default in MariaDB 10.0.12, enable by putting the following into your my.cnf file:
  - performance_schema=on
- These views may show you problems that don't show up elsewhere.
- events_statements_summary_by_digest is very interesting.
- A good writeup is at: http://www.markleith.co.uk/2012/07/04/mysql-performance-schema-statement-digests/

MySQL performance_schema Top10 Queries

SWITCH

```
SELECT IF(LENGTH(DIGEST_TEXT) > 64, CONCAT(LEFT(DIGEST_TEXT, 30), ' ... ',
RIGHT(DIGEST_TEXT, 30)), DIGEST_TEXT) AS query,
    IF(SUM_NO_GOOD_INDEX_USED > 0 OR SUM_NO_INDEX_USED > 0, '*', '') AS
full_scan,
    COUNT_STAR AS exec_count,
    SUM_ERRORS AS err_count,
    SUM_WARNINGS AS warn_count,
    SEC_TO_TIME(SUM_TIMER_WAIT/1000000000000) AS exec_time_total,
    SEC_TO_TIME(MAX_TIMER_WAIT/1000000000000) AS exec_time_max,
    (AVG_TIMER_WAIT/1000000000) AS exec_time_avg_ms,
    SUM_ROWS_SENT AS rows_sent,
    ROUND(SUM_ROWS_SENT / COUNT_STAR) AS rows_sent_avg,
    SUM_ROWS_EXAMINED AS rows_scanned,
    DIGEST AS digest
  FROM performance_schema.events_statements_summary_by_digest
ORDER BY SUM_TIMER_WAIT DESC LIMIT 10 \G
```

(Source: http://www.markleith.co.uk/2012/07/04/mysql-performance-schema-statement-digests/)

MaxScale top queries

- MaxScale has a query summary as well
- You will then end up with a top 10 summary filter for the logged session.
- Documentation for the filter is at:
  - https://mariadb.com/kb/en/mariadb-enterprise/mariadb-maxscale-14/maxscale-top-filter-overview/

MaxScale top queries

To enable add the following to your /etc/maxscale.cnf file on your client systems:

```
# uncomment the following to lines to activate the top10Logger
router_options=running
filters=top10Logger
```

MaxScale top queries

- This is what we did
- The results were impressive

MaxScale identifying top queries

```
[root@drive-mgmt1 sessions]# cat top10.1181
Top 10 longest running queries in session.
=========================================
Time (sec) | Query
-----------+-----------------------------------------------------------------
    0.078 |  SELECT `oc_share`.`id`, `item_type`, `item_source`, `item_target`,`oc_share`.`parent`, `share_type`, `share_with`,
`uid_owner`,`file_source`, `path`, `file_target`, `oc_share`.`permissions`,`stime`, `expiration`, `token`, `storage`,
`mail_send`,`oc_storages`.`id` AS `storage_id`, `oc_filecache`.`parent` as `file_parent` FROM `oc_share` INNER JOIN `oc_filecache` ON
`file_source` = `oc_filecache`.`fileid`  AND `file_target` IS NOT NULL INNER JOIN `oc_storages` ON `numeric_id` = `oc_filecache`.`storage`
AND ((`share_type` in ('0', '2') AND `share_with` = 'username@switch.ch') ) AND `uid_owner` != 'username@switch.ch' ORDER BY
`oc_share`.`id` ASC
    0.040 |  INSERT INTO `oc_preferences` (`userid`, `appid`, `configkey`, `configvalue`) VALUES('username@switch.ch', 'user_ldap',
'homePath', '')
    0.018 |  COMMIT
    0.013 |  START TRANSACTION
    0.012 |  SELECT `fileid`, `storage`, `path`, `parent`, `name`, `mimetype`, `mimepart`, `size`, `mtime`,
                                                `storage_mtime`, `encrypted`, `etag`, `permissions`, `checksum`
                                        FROM `oc_filecache` WHERE `storage` = '4231' AND `path_hash` =
[...]
-----------+-----------------------------------------------------------------
Session started Wed Aug 30 06:14:49 2017
Connection from localhost_from_socket
Username        owncloud

Total of 14349 statements executed.
Total statement execution time      13.645 seconds
Average statement execution time     0.001 seconds
Total connection time               79.461 seconds
```

MaxScale identifying top queries

THIS was our problem query!

```
SELECT `oc_share`.`id`, `item_type`, `item_source`, `item_target`,`oc_share`.`parent`, `share_type`, `share_with`,
 `uid_owner`,`file_source`, `path`, `file_target`, `oc_share`.`permissions`,`stime`, `expiration`, `token`,
 `storage`, `mail_send`,`oc_storages`.`id` AS `storage_id`, `oc_filecache`.`parent` as `file_parent`
FROM `oc_share`
INNER JOIN `oc_filecache`
 ON `file_source` = `oc_filecache`.`fileid`
 AND `file_target` IS NOT NULL
INNER JOIN `oc_storages`
 ON `numeric_id` = `oc_filecache`.`storage`
 AND ((`share_type` in ('0', '2')
  AND `share_with` = 'username@switch.ch') )
 AND `uid_owner` != 'username@switch.ch'
ORDER BY `oc_share`.`id` ASC
```

But why is it our problem query? What is wrong with it? How can we fix it?

Explain

- The MariaDB EXPLAIN statement will give you an estimate on how your SQL statements will be run against the DB.
- EXPLAIN EXTENDED will give you a estimate as to what percentage of the table rows will be filtered by the condition.
- It will show you the steps used to get your result set, like:
  - How many rows are returned at each step
  - If sorts are being performed
  - If a filter is being used
  - If an index is being used
- There is also EXPLAIN FORMAT=JSON that gives JSON format back

Analyze

There is also an additional way to run Explain, formerly this was EXPLAIN ANALIZE, now it is just ANALIZE

- Can be run on any query you can run EXPLAIN on.
- Runs the optimizer.
- Runs the actual query.
- Returns results based on the actual query.
- Obviously slower than just running EXPLAIN, but gives more correct results on what is happening with the data.

Explain

To invoke an EXPLAIN plan just put EXPLAIN in front of the query, so for our problem query:

```
EXPLAIN SELECT `oc_share`.`id`, `item_type`, `item_source`, `item_target`,`oc_share`.`parent`, `share_type`,
`share_with`,
 `uid_owner`,`file_source`, `path`, `file_target`, `oc_share`.`permissions`,`stime`, `expiration`, `token`,
 `storage`, `mail_send`,`oc_storages`.`id` AS `storage_id`, `oc_filecache`.`parent` as `file_parent`
FROM `oc_share`
INNER JOIN `oc_filecache`
 ON `file_source` = `oc_filecache`.`fileid`
 AND `file_target` IS NOT NULL
INNER JOIN `oc_storages`
 ON `numeric_id` = `oc_filecache`.`storage`
 AND ((`share_type` in ('0', '2')
  AND `share_with` = 'username@switch.ch') )
 AND `uid_owner` != 'username@switch.ch'
ORDER BY `oc_share`.`id` ASC
```

Explain

And the output is:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	oc_share	index	file_source_index	PRIMARY	4	NULL	70305	Using where
1	SIMPLE	oc_filecache	eq_ref	PRIMARY, fs_storage_hash, fs_storage_mimetype, fs_storage_mimepart, fs_storage_size	PRIMARY	4	owncloud.oc_share.file_source	1	
1	SIMPLE	oc_storages	eq_ref	PRIMARY	PRIMARY	4	owncloud.oc_filecache.storage	1	

Explain

Our Problem is here:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	oc_share	index	file_source_index	PRIMARY	4	NULL	70305	Using where

What is the problem with this?

- We are using the PRIMARY KEY index to retrieve 70305 rows, and this is the wrong index.
- And then we are filtering on the output of this into a JOIN.
  - Note that the 'AND' operators in our join invokes an implicit 'WHERE' clause.
- This is also our top query that we see from MaxScale

Indexes

What is an index in an relational database?

- An index is basically a key-value store using one or more columns as a key to retrieve rows in a database table.
- PRIMARY KEY and UNIQUE constraints are enforced by indexes.
  - Be aware that MySQL/MariaDB do not handle NULLs as unique values in a UNIQUE constraint so unless You have UNIQUE and and NOT NULL constraints on the column you can still wind up with duplicate rows.
- A missing index can really make your database slow, but unused/seldom used indexes will also consume resources.
- Adding an index will lock your table during index creation.

Table Joins

- Without going into too much detail, you can join tables to get results based on a combination of tables. Usually you join based on common data in a column. Our query is doing this to find out who is sharing data.
- When joining tables, you want to be sure that you have an index on the column you are joining, or the join will be slow.
- You may also want indexes on the filters.
- An improper join may result in a cartesian product, basically returning a set of all possible combinations of data in both tables. Your filters might process this to be a correct result, but it will use a lot of resources.

Finding the missing index

Back to our problem query:

```
SELECT `oc_share`.`id`, `item_type`, `item_source`, `item_target`,`oc_share`.`parent`, `share_type`, `share_with`,
 `uid_owner`,`file_source`, `path`, `file_target`, `oc_share`.`permissions`,`stime`, `expiration`, `token`,
 `storage`, `mail_send`,`oc_storages`.`id` AS `storage_id`, `oc_filecache`.`parent` as `file_parent`
FROM `oc_share`
INNER JOIN `oc_filecache`
 ON `file_source` = `oc_filecache`.`fileid`
 AND `file_target` IS NOT NULL
INNER JOIN `oc_storages`
 ON `numeric_id` = `oc_filecache`.`storage`
 AND ((`share_type` in ('0', '2')
  AND `share_with` = 'username@switch.ch') )
 AND `uid_owner` != 'username@switch.ch'
ORDER BY `oc_share`.`id` ASC
```

We know that the problem table is oc_share, and we're filtering using a value for the column share_with, there is no index on share_with.

The fix

The fix to this problem is to add the missing index on the table oc_share using the column share_with:

```
CREATE INDEX share_with_index ON oc_share (share_with);
```

The results from this were amazing...

Explain after the index is added

Running explain on the query after the index is added:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	oc_share	ref	file_source_index, share_with_index	share_with_index	768	NULL	1	const
1	SIMPLE	oc_filecache	eq_ref	PRIMARY, fs_storage_hash, fs_storage_mimetype, fs_storage_mimepart, fs_storage_size	PRIMARY	4	owncloud.oc_share.file_source	1	
1	SIMPLE	oc_storages	eq_ref	PRIMARY	PRIMARY	4	owncloud.oc_filecache.storage	1	

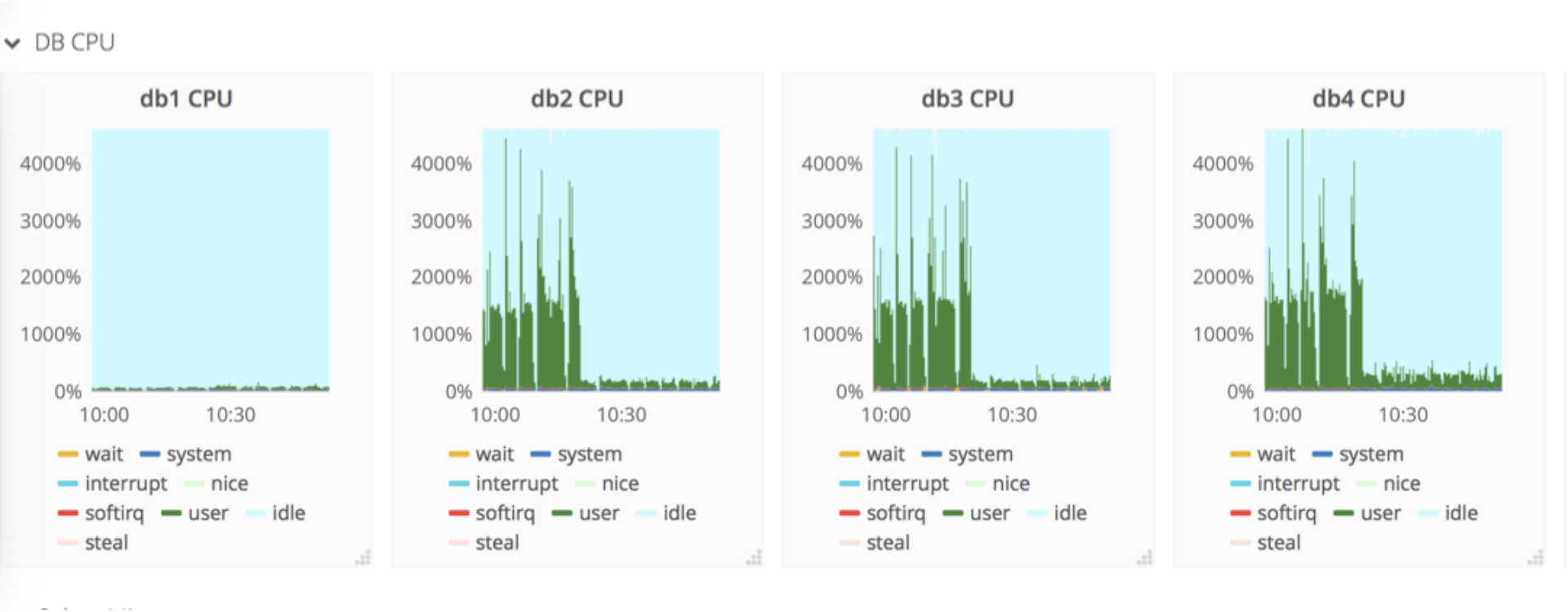
This time, our first part of query returns exactly ONE row instead of 70305

The results of implementing the fix

A Grafana chart of our average query response times after the change

The results of implementing the fix

And our Grafana chart of our CPU usage afterwards



The aftermath

- Database read node response time average went from 3 ms to about 0.15 ms.
- Database peak CPU usage went from 1200% to 200%.
- Our database infrastructure became more stable.
- Our owncloud cron jobs now run a bit faster.
- We have reduced our number of database servers to three. We could reduce that further, but keep three for failover.
- These three still use much fewer resources than they did before, with a database read node response time of about 0.2 ms.

SWITCH

Other database products?

PostgreSQL provides the pg_stat_statements extension for viewing SQL query summary information.

Oracle has the Oracle Diagnostics Pack and Oracle Tuning Pack as part of their Oracle Enterprise Manager tool.

Both PostgreSQL and Oracle also have EXPLAIN capabilities.

Profiling

- Query profiling in MariaDB can also be useful to see how your queries are running
- Set in a SQL session
  - ‘SET profiling = 1;
- You can then run your query and then run the SQL command SHOW PROFILES to see how long your queries take.
- More information at: https://mariadb.com/kb/en/library/show-profile

Profiling Example

```
MariaDB [owncloud]>SET profiling = 1;
Query OK, 0 rows affected (0.00 sec)


MariaDB [owncloud]>select count(*) from oc_share_external where
owner='gregory.vernon@switch.ch';
+----------+
| count(*) |
+----------+
|        0 |
+----------+
1 row in set (0.00 sec)

MariaDB [owncloud]>select count(*) from
oc_share_external;

+----------+
| count(*) |
+----------+
|      730 |
+----------+
1 row in set (0.00 sec)

MariaDB [owncloud]>SHOW PROFILES;
+----------+------------+-----------------------------------------------------------------------------------+
| Query_ID | Duration   | Query                                                                             |
+----------+------------+-----------------------------------------------------------------------------------+
|        1 | 0.00408854 | select count(*) from oc_share_external where owner='gregory.vernon@switch.ch' |
|        2 | 0.00382607 | select count(*) from oc_share_external                                            |
+----------+------------+-----------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

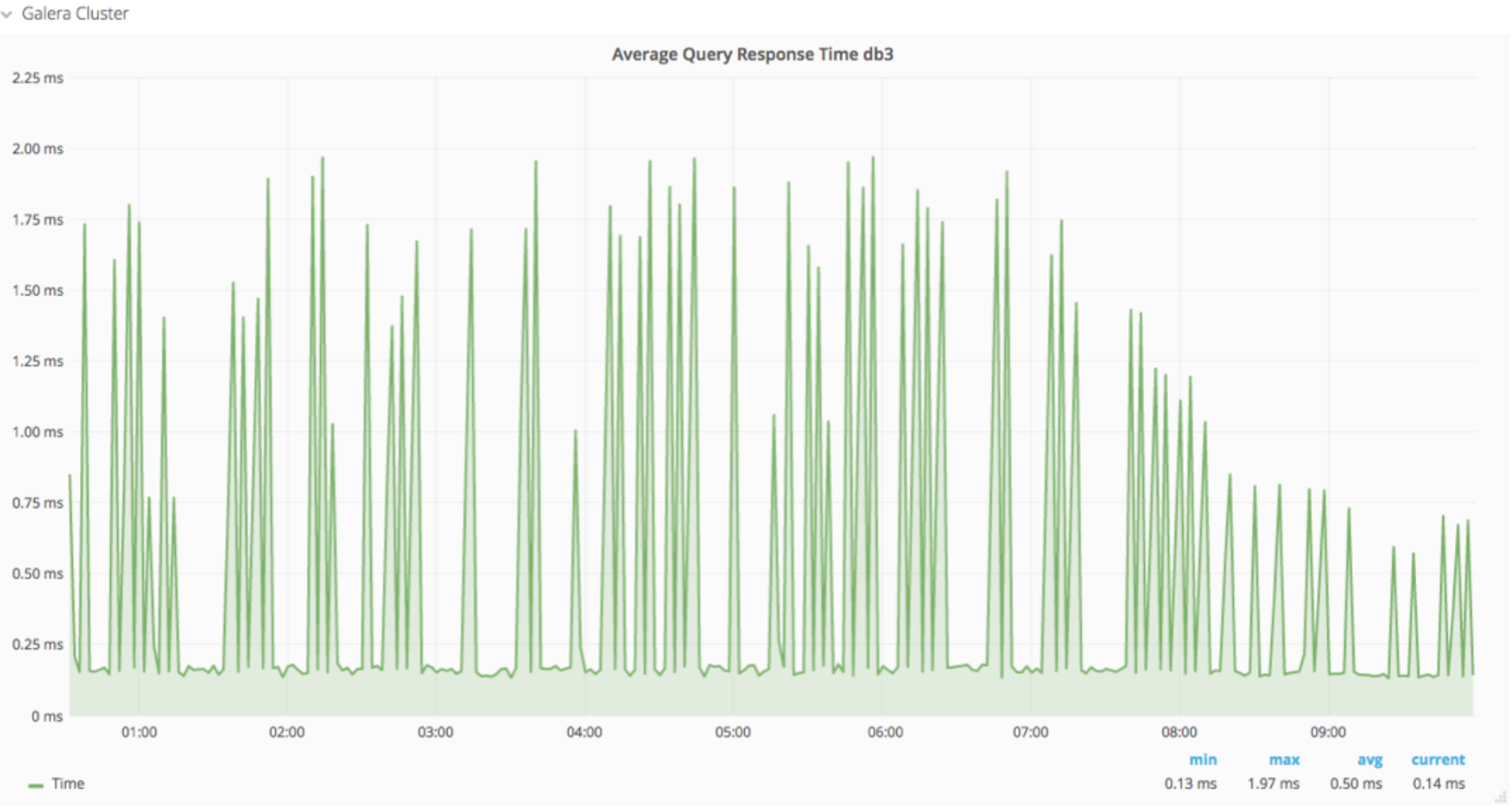
Summary

- Database issues can be hard to troubleshoot, if you don't know the tools.
- MySQL's optimizer won't always run your query the way you think it should. Sometimes you need to think like the optimizer.
- Your data is your data, so you may see different problems.
- Your problem query might not show up in the slow query log, it might just be a little inefficient.
- Missing indexes can cause you big problems, adding the right indexes can make your database much faster.
- It’s useful to see how long your query takes, and profiling can help you with this.

Questions

Email: greg.vernon@switch.ch

Just for fun: What are the spikes here?



SWITCH – an integral part of the Swiss academic community since 1987.