Fortify your MySQL data security in AWS using ProxySQL and Firewelling

Marco Tusa
Percona
About Me

• Open source enthusiast
• Principal Consultant
• Working in DB world over 25 years
• Open source developer and community contributor
Hello, Attendees!

Why use ProxySQL with Aurora or AWS solutions

What can be done to make your AWS/Aurora safer and more stable
What is ProxySQL (in 1 slide)

• ProxySQL has an advanced multi-core architecture.
• It's built from the ground up to support hundreds of thousands of concurrent connections, multiplexed to potentially hundreds of backend servers.
• Query filtering by design
• Query caching
• Embedded configuration distribution (cluster)
• Design to scale (the largest ProxySQL deployment spans several hundred proxies).
• … and more
What is AWS Aurora (in 1 slide)

• Amazon Aurora is a MySQL and PostgreSQL compatible relational database built for the cloud
• Features a distributed, fault-tolerant, self-healing storage system that auto-scales up to 64TB per database instance
• Delivers high performance and availability with up to 15 low-latency read replicas, point-in-time recovery, continuous backup to Amazon S3, and replication across three Availability Zones
• fully managed by Amazon Relational Database Service (RDS)
• … and more
Use ProxySQL version 2 (The problem)

ProxySQL deal with backend servers using:

- Replication Hostgroup
  - Async replication
- Scheduler
  - PXC, NDB etc

AWS Aurora do not use READ_ONLY but INNODB_READ_ONLY

Use ProxySQL version 2 (Solution)
October 2017, this issue was opened (https://github.com/sysown/proxysql/issues/1195)

```
MYHGM_MYSQL_REPLICATION_HOSTGROUPS "CREATE TABLE mysql_replication_hostgroups
(writer_hostgroup INT CHECK (writer_hostgroup>=0) NOT NULL PRIMARY KEY ,
reader_hostgroup INT NOT NULL CHECK (reader_hostgroup<>writer_hostgroup AND
reader_hostgroup>=0) , check_type VARCHAR CHECK (LOWER(check_type) IN
('read_only','innodb_read_only','super_read_only')) NOT NULL DEFAULT 'read_only' ,
comment VARCHAR NOT NULL DEFAULT '' , UNIQUE (reader_hostgroup))"

mysql> select * from mysql_replication_hostgroups;
+------------------+------------------+------------------+------------+
| writer_hostgroup | reader_hostgroup | check_type       | comment    |
+------------------+------------------+------------------+------------+
| 70               | 71               | innodb_read_only | aws-aurora |
+------------------+------------------+------------------+------------+
1 row in set (0.00 sec)
```
Use ProxySQL version 2 (How to implement)

First rollout your Aurora setup
• Identify the Endpoint for EACH instance
  • `aws rds describe-db-instances`
  • Web interface

```sql
INSERT INTO mysql_servers (hostname,hostgroup_id,port,weight,max_connections) VALUES ('proxysqltestdb.c7wzm8xxmrze.eu-central-1',70,3306,1000,2000);
VALUES ('proxysqltestdb.eu-central-1',71,3306,1000,2000);
VALUES ('proxysqltestdb2.eu-central-1',71,3306,1000,2000);
VALUES ('proxysqltestdb-eu-central-1b.eu-central.1',71,3306,1,2000);
```

```sql
INSERT INTO mysql_replication_hostgroups(writer_hostgroup,reader_hostgroup,comment,check_type) VALUES (70,71,'aws-aurora','innodb_read_only');
```

LOAD MYSQL SERVERS TO RUNTIME; SAVE MYSQL SERVERS TO DISK;
But WHY I should use it?

GOOD QUESTION
## Why

<table>
<thead>
<tr>
<th>Queries/Sec</th>
<th>Aurora</th>
<th>ProxySQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read only</td>
<td>14486</td>
<td>18560</td>
</tr>
<tr>
<td>Write only</td>
<td>1329</td>
<td>1359</td>
</tr>
<tr>
<td>RW</td>
<td>14486</td>
<td>18560</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Read</th>
<th>Write</th>
<th>RW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Aurora</td>
<td>ProxySQL</td>
<td>Aurora</td>
</tr>
<tr>
<td>AVG Events</td>
<td>90696</td>
<td>98709</td>
<td>686</td>
</tr>
<tr>
<td>AVG max C latency ns</td>
<td>12217913</td>
<td>10973011</td>
<td>24530078</td>
</tr>
<tr>
<td>AVG min C latency ns</td>
<td>105</td>
<td>70</td>
<td>910</td>
</tr>
<tr>
<td>AVG exec time ms</td>
<td>110</td>
<td>66</td>
<td>1943</td>
</tr>
<tr>
<td>Select/s</td>
<td>18100</td>
<td>19700</td>
<td>0</td>
</tr>
<tr>
<td>writes/s</td>
<td>0</td>
<td>0</td>
<td>260</td>
</tr>
<tr>
<td>update/s</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>delete/s</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Why

Higher is better
Why

When using Java connector, the only connection latency is in the range of nanoseconds. The chart shows the connection latency for different databases:

- **Aurora**
  - Read: 12217913 ns
  - Write: 24530078 ns
- **ProxySQL**
  - Read: 10973011 ns
  - Write: 19871523 ns
- **Aurora (RW)**
  - Read: 21855254 ns
  - Write: 23189711 ns

The table below provides an overview of the average maximum and minimum connection latency in nanoseconds for each database:

<table>
<thead>
<tr>
<th>Database</th>
<th>Read</th>
<th>Read</th>
<th>Write</th>
<th>Write</th>
<th>Read</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aurora</td>
<td>12217913</td>
<td>10973011</td>
<td>24530078</td>
<td>19871523</td>
<td>21855254</td>
<td>23189711</td>
</tr>
<tr>
<td>ProxySQL</td>
<td>10973011</td>
<td>10973011</td>
<td>19871523</td>
<td>19871523</td>
<td>21855254</td>
<td>23189711</td>
</tr>
<tr>
<td>Aurora (RW)</td>
<td>21855254</td>
<td>21855254</td>
<td>23189711</td>
<td>23189711</td>
<td>23189711</td>
<td>23189711</td>
</tr>
</tbody>
</table>

The connection latency is significantly lower in Aurora compared to ProxySQL, especially for read operations.
Why

Latency while running the tests

Left Aurora
Right ProxySQL
Why it happens

ProxySQL can redirect the queries as you like and to the instance you want.

How do we read this graph? From left to right:
- read_only test with an Aurora cluster endpoint
- read_only test with ProxySQL
- write_only with an Aurora cluster endpoint
- write_only with ProxySQL
- read and write with an Aurora cluster endpoint
- read and write with ProxySQL
Why it happens

The Cluster endpoint is an endpoint for an Aurora DB cluster that connects to the current primary instance for that DB cluster. Each Aurora DB cluster has a cluster endpoint and one primary instance. That endpoint receives the read and write request and sends them to the same instance. The main use for it is to perform failover if needed.

Each Aurora DB cluster has a reader endpoint. If there is more than one Aurora Replica, the reader endpoint directs each connection request to one of the Aurora Replicas. The reader endpoint only load balances connections to available Aurora Replicas in an Aurora DB cluster. It does not load balance specific queries.

If you want to load balance specific queries to distribute the read workload for a DB cluster, you need to manage that in your application and use instance endpoints to connect directly to Aurora Replicas to balance the load.
Aaah That’s why

• Native AWS Cluster endpoints and Reader endpoints are limited in what they offer

• With ProxySQL you can very granularly choose how to use each instance, without the need to have the application modify how it works

• Using ProxySQL will allow the use of additional elements like
  • Query Cache
  • Query rewrite
  • Blocking/firewalling
Now What?

Secure all around
• Secure access to RDS (account, IASM, Roles)

• Secure network access (limit to local range/Port, VPN, etc)

• Secure MySQL user/password/location/access + grants

• Secure ProxySQL (user/password + encrypted )
Time to relax?
No is not!

Enemies and dangers are around you
  • Your own application
  • Developers
  • Your DBA/OPS
  • Yourself
We can do more

• Queries with no filtering (Where)
• Over complicated queries
• Not indexed Queries
• Jobs that are not suppose to hit main production
• Reports on writer node/instance

Currently the only way to manage some of the above is to use SP and limit all users to execute
Fields of action

Secure by limiting actions on the db

Use Proxy as Firewall
Secure by limiting actions on the db

MySQL Query Rules Table

Filter by:
- username
- schemaname
- client_addr
- proxy_addr
- proxy_port
- digest
- match_digest
- match_pattern
Secure by limiting actions on the db: Targets

**Destination_hostgroup**
Pointing to different HG

**Replace_pattern**
Rewriting the Query

Or if firewalling blocking the query
Secure limiting actions on the DB

Limit queries by (user/ip/ports)

insert into mysql_query_rules (rule_id,client_addr,username,destination_hostgroup,active,retries,match_digest,apply)
values(24,'192.168.1.50','app_test',101,1,3,'^SELECT.*$',1);

insert into mysql_query_rules (rule_id,client_addr,username,destination_hostgroup,active,retries,match_digest,apply)
values(38,'192.168.1.51','app_test',200,1,3,'.',1);

client_addr: 192.168.1.50
proxy_addr: NULL
proxy_port: NULL

client_addr: 192.168.1.51
proxy_addr: NULL
proxy_port: NULL
Secure limiting actions on the DB

Block queries not filtered, without where (I am a dummy)

```sql
insert into mysql_query_rules
     (rule_id,match_digest,error_msg,active,apply)
values(1,'^SELECT\s((?!\swhere).)*$','Bad Idea to performa SELECT without a WHERE ... change the syntax and I will let you PASS',1,1);
```

```sql
mysql> select count(*) from wmillAUTOINC;
ERROR 1148 (42000): Bad Idea to perform SELECT without a WHERE ... change the syntax and I will let you PASS

mysql> select count(*) from wmillAUTOINC where millid=365;
<table>
<thead>
<tr>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>393</td>
</tr>
</tbody>
</table>
```
Secure limiting actions on the DB

Block and transform query by type Select/update/inserts

```sql
use windmills; select count(*) a, tbl1.b from wmillAUTOINC aa, (select count(*) b from wmillMID) tbl1 where millid=364
| a   | b      |
| 418 | 164577 |
```

No where in subquery

**match_pattern:**

```sql
select count\((\*)\) a, tbl1\.b from wmillAUTOINC aa, (select count\((\*)\) b from wmillMID \) tbl1 where millid=(\d*)
```

**replace_pattern:**

```sql
select count(*) a, tbl1.b from wmillAUTOINC aa, (select count(*) b from wmillMID where millid=\1) tbl1 where millid=\1
| a   | b   |
| 418 | 407 |
```

https://github.com/sysown/proxysql/issues/1556
Use ProxySQL as firewall

- block all
- block stupid checks (SELECT 1)
- let pass something by regexp
- let pass only specific queries
- Make it efficient
## IPtables

```
[root@galera1h1n5 ~]# iptables -L
Chain INPUT (policy ACCEPT)
 target     prot opt source               destination
 ACCEPT     all  --  anywhere             anywhere
 ACCEPT     tcp  --  10.0.0.0/24          anywhere             tcp dpt:mysql
 ACCEPT     tcp  --  10.0.0.0/24          anywhere             tcp dpt:tram
 ACCEPT     tcp  --  10.0.0.0/24          anywhere             tcp dpt:bmc-reporting
 ACCEPT     tcp  --  10.0.0.0/24          anywhere             tcp dpt:krb524
 ACCEPT     udp  --  10.0.0.0/24          anywhere             udp dpt:tram
 ACCEPT     tcp  --  192.168.1.0/24       anywhere             tcp dpt:42000
 ACCEPT     tcp  --  192.168.1.0/24       anywhere             tcp dpt:42002
 ACCEPT     tcp  --  10.0.0.0/24          anywhere             tcp dpt:6033
 ACCEPT     tcp  --  10.0.0.0/24          anywhere             tcp dpt:6032
 ACCEPT     icmp --  anywhere             anywhere            state RELATED,ESTABLISHED
 ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
 ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:mysql
 REJECT     all  --  anywhere             anywhere            reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT)
 target     prot opt source               destination
 REJECT     all  --  anywhere             anywhere            reject-with icmp-port-unreachable

Chain OUTPUT (policy ACCEPT)
 target     prot opt source               destination
```
### ProxySQL way

<table>
<thead>
<tr>
<th>rule_id</th>
<th>active</th>
<th>username</th>
<th>proxy_port</th>
<th>Schema</th>
<th>destination_HG</th>
<th>match_digest</th>
<th>apply</th>
<th>FlagOUT</th>
<th>FlagIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>100:</td>
<td>1</td>
<td>pxc_test</td>
<td>6034:windmills</td>
<td>52:Select MAX(id) as id FROM wmllIMUID</td>
<td>1:NULL</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101:</td>
<td>1</td>
<td>pxc_test</td>
<td>6035:windmills</td>
<td>52:Select MAX(id) as id FROM wmllAUTOINC</td>
<td>0:1000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>102:</td>
<td>1</td>
<td>pxc_test</td>
<td>6036:windmills</td>
<td>52:Select MAX(id) as id FROM wmllMID</td>
<td>0:1000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1003:</td>
<td>1</td>
<td>pxc_test</td>
<td>6036:windmills</td>
<td>50:NULL</td>
<td>1:NULL</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1001:</td>
<td>1</td>
<td>pxc_test</td>
<td>6036:windmills</td>
<td>50:^SELECT.*,FOR UPDATE</td>
<td>1:NULL</td>
<td>1000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1002:</td>
<td>1</td>
<td>pxc_test</td>
<td>6036:windmills</td>
<td>50:^SELECT.*,@@</td>
<td>1:NULL</td>
<td>1000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1003:</td>
<td>1</td>
<td>pxc_test</td>
<td>6036:windmills</td>
<td>52:SELECT</td>
<td>1:NULL</td>
<td>1000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **incoming SQL**
- **Matching Rule with apply=1** will exit QueryProcessor
- **Matching Rule** Bypass the firewall
Filter stupid ping

Select 1;
Million of times

```sql
insert into mysql_query_rules
(rule_id,match_digest,ok_msg,active,apply)
values(500,'SELECT 1','Ok',1,1);
```
ProxySQL way (apply NOW)

```sql
SELECT
    wmillAUTOINC.id, wmillAUTOINC.millid, wmillAUTOINC.location
FROM wmillAUTOINC
WHERE wmillAUTOINC.millid=300 and wmillAUTOINC.active=1
;

insert into mysql_query_rules
    (rule_id, proxy_port, username, destination_hostgroup, schema name, active, retries, apply, flagout, match_digest)
values(101, 6033, 'pxc_test', 52, 'windmills', 1, 3, 1, null,
    'SELECT wmillAUTOINC .id, wmillAUTOINC .millid, wmillAUTOINC .location FROM wmillAUTOINC
    WHERE wmillAUTOINC.millid=\d* and wmillAUTOINC.active=.*');
```
ProxySQL way (apply Later)

Select MAX(millid) as millid ,MAX(active) as active FROM wmillMID;
insert into mysql_query_rules
(rule_id,proxy_port,schemaname,username,destination_hostgroup,active,retries,match_digest,apply,flagout)
values(999,6033,'windmills','pxc_test',50,1,3,' Select MAX(millid) as millid ,MAX(active) as active FROM wmillMID',0,1000);

insert into mysql_query_rules
(rule_id,proxy_port,schemaname,username,destination_hostgroup,active,retries,match_digest,apply,flagin)
values(1042,6033,'windmills','pxc_test',52,1,3,'^SELECT.*$',1,1000);
Eehm ... How much it cost?

With nothing enable 10 µ
Eehm ... How much it cost?
Can you give me a discount?
ProxySQL way (apply now)

```sql
select hostgroup, schemaname, count_star, digest, replace(replace(digest_text, '.', '\.'), '?', '.*') as QR from stats_mysql_query_digest where schemaname='windmills' order by count_star desc;
```

<table>
<thead>
<tr>
<th>hostgroup</th>
<th>schemaname</th>
<th>count_star</th>
<th>digest</th>
<th>QR</th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
<td>windmills</td>
<td>573331</td>
<td>0x52A98085A233E516</td>
<td></td>
</tr>
</tbody>
</table>

```sql
SELECT wmillAUTOINC\.id, wmillAUTOINC\.millid, wmillAUTOINC\.location FROM wmillAUTOINC WHERE wmillAUTOINC\.millid=.* and wmillAUTOINC\.active=.*
```
ProxySQL way (apply NOW digest)

```
SELECT
    wmillAUTOINC.id, wmillAUTOINC.millid, wmillAUTOINC.location
FROM wmillAUTOINC WHERE wmillAUTOINC.millid=300 and
    wmillAUTOINC.active=1');

insert into mysql_query_rules
    (rule_id, proxy_port, username, destination_hostgroup, schema
    name, active, retries, apply, flagout, digest)
values(101, 6033, 'pxc_test', 52, 'windmills', 1, 3, 1, null, '0xDB3A841EF5443C35');
```
ProxySQL way (apply later)

```sql
select
hostgroup,schemaname,count_star,digest,replace(replace(digest_text,'.','\.'),'?','.*') QR
from stats_mysql_query_digest
where schemaname='windmills'
order by count_star desc;
```

<table>
<thead>
<tr>
<th>hostgroup</th>
<th>schemaname</th>
<th>count_star</th>
<th>digest</th>
<th>QR</th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
<td>windmills</td>
<td>139</td>
<td>0x839B1DCE7A8B247A</td>
<td></td>
</tr>
</tbody>
</table>

Select MAX(millid) as millid ,MAX(active) as active FROM wmillAUTOINC
ProxySQL way (apply Later digest)

Select MAX(millid) as millid, MAX(active) as active FROM wmillMID;

insert into mysql_query_rules
(rule_id, proxy_port, schemaname, username, destination_group, active, retries, digest, apply, flagout)
values(999, 6033, 'windmills', 'pxc_test', 50, 1, 3, '0x839B1DCE7A8B247A', 1, 1000);

insert into mysql_query_rules
(rule_id, proxy_port, schemaname, username, destination_group, active, retries, match_digest, apply, flagin)
values(1042, 6033, 'windmills', 'pxc_test', 52, 1, 3, '^SELECT.*$', 1, 1000);
Eehm ... How much it cost Now?
Eehm ... How much it cost?
Deal

Nothing enable  : $\sim 10 \mu$

Using match_digist  : $\sim 30 \mu$

Using digest  : $\sim 4 \mu$
Done! (or conclusions if you like)

- ProxySQL is now (v2.x and above) supporting ASW/Aurora
- ProxySQL is better than native connector
- Your security at SQL level sucks (look at the mirror before complain with others)
- ProxySQL allow you implement SQL control and a way to correct things (while you fix them for real)
- It has a cost (nothing is for nothing)
- At the moment we do not have it (digest solution) automated (Can you develop it? Help the community !)
But must done right

Performance can be affected (by Tibor Korocz)

(https://www.percona.com/blog/2017/04/10/proxysql-rules-do-i-have-too-many/)
Thank You
Rate My Session

Accelerating Application Development with Amazon Aurora

Description
In this hands-on tutorial, you'll learn how to leverage the unique features of Amazon Aurora to build faster, more scalable database applications optimized for the cloud. We discuss architectural best practices and features designed to help you develop applications faster and reach the widest possible audience, including Aurora Serverless, read replica auto scaling, cross-region replicas, backtracking, fast database cloning, and Performance Insights. You'll understand how to best take advantage of the Aurora platform's capabilities to effectively accelerate application development.
We’re Hiring

Percona’s open source database experts are true superheroes, improving database performance for customers across the globe.

Our staff live in nearly 30 different countries around the world, and most work remotely from home.

Discover what it means to have a Percona career with the smartest people in the database performance industries, solving the most challenging problems our customers come across.
Contact Me

To Contact Me:

Marco.tusa@percona.com

tusamarco@gmail.com

To Follow Me:

http://www.tusacentral.net/

http://www.percona.com/blog/

https://www.facebook.com/marco.tusa.94

@marcotusa

http://it.linkedin.com/in/marcotusa/

Consulting = No mission refused!