ProxySQL

What is ProxySQL ?

- ProxySQL is a open-source high-performance SQL aware proxy. It runs as a daemon watched by a monitoring process.
- ProxySQL seats between application and db servers.
- The daemon accepts incoming traffic from MySQL clients and forwards it to backend MySQL servers.

A few most commonly used features are :

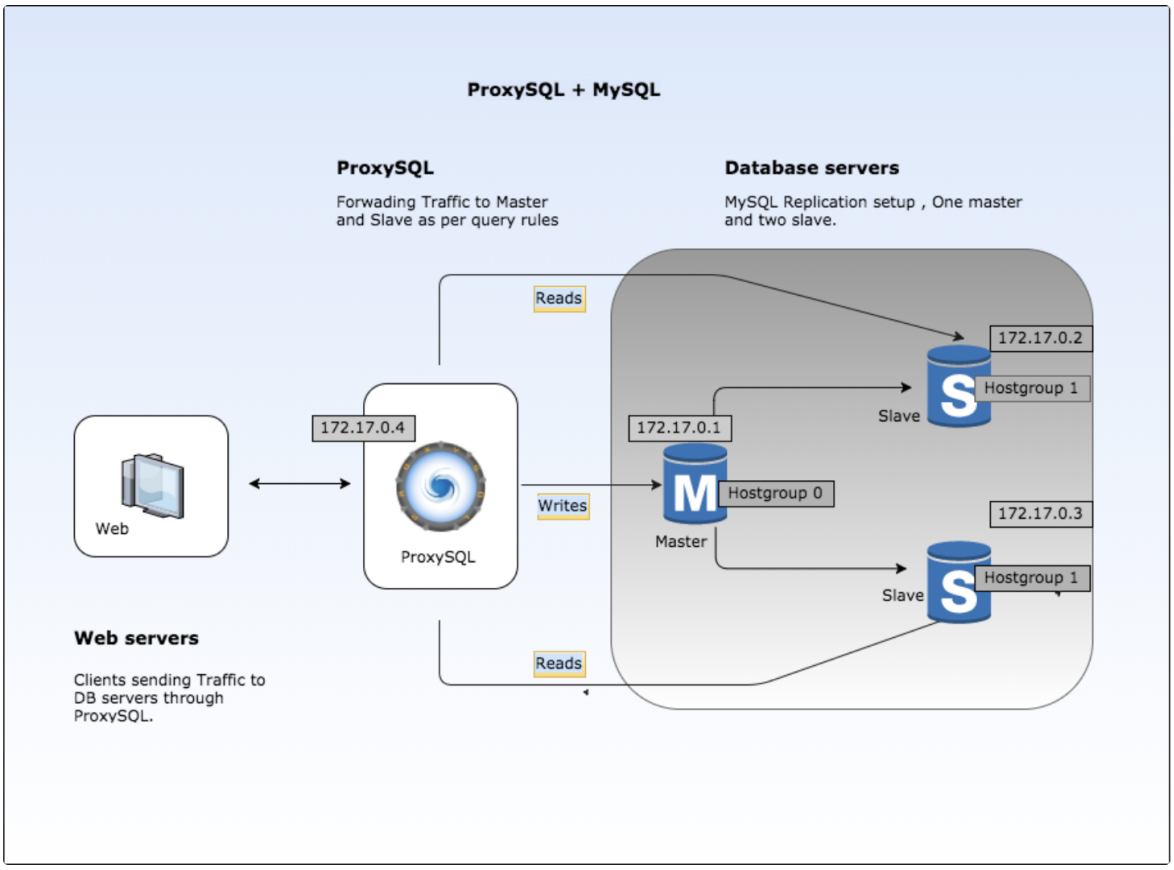
- Read/write split
- On-the-fly queries rewrite
- Query caching
- Real time statistics on web console
- Seamless replication switchover
- Query mirroring
- Monitoring

So the main advantages of using ProxySQL is, it is designed to run continuously without needing to be restarted. Most configuration can be done at runtime using queries similar to SQL statements and it is more light weight.

Let us explore the basic Query Routing (Read/Write split) for the effective load sharing. We have set up 4 nodes to make the architecture .

Let us explore the basic Query Routing (Read/Write split) for the effective load sharing. We have set up 4 nodes to make the architecture .

- node1 (172.17.0.1) , Master
- node2 (172.17.0.2) , Slave
- node3 (172.17.0.3) , Slave
- node4 (172.17.0.4) , ProxySQL



ProxySQL on Single Master and Two Slaves.

Note: *By default, ProxySQL binds with two Ports 6032 and 6033.*
6032 is admin port and 6033 is the one which accepts incoming connections from clients.

MySQL Replication setup :

Configuring MySQL's master-slave replication is outside the scope of this tutorial, we already have nodes with replication running.

Before entering to admin interface of ProxySQL , create one application user with all privileges required to your application and one monitoring user at every MySQL DB server.

```
mysql> CREATE USER 'sysbench'@'172.17.0.%' IDENTIFIED BY 'sysbenc
mysql> GRANT ALL PRIVILEGES on *.* TO 'sysbench'@'172.17.0.%';

mysql> CREATE USER  'monitor'@'172.17.0.%' IDENTIFIED BY 'monitor'
mysql> GRANT USAGE,REPLICATION CLIENT on *.* TO 'monitor'@'172.17.

mysql> FLUSH PRIVILEGES;
```

Install and start ProxySQL :

For Installation kindly refer : https://github.com/sysown/proxysql/wiki

```
 $ service proxysql start
Starting ProxySQL: Main init phase0 completed in 0.000491 secs.
Main init global variables completed in 0.000675 secs.
Main daemonize phase1 completed in 0.00015 secs.
DONE!
```

Now connect to ProxySQL admin interface to start with configuration :

```
 $ mysql -u admin -padmin -h 127.0.0.1 -P6032
```

Configure Backends :

ProxySQL uses the concept of hostgroup. A hostgroup is a group of host with logical functionalities.

In this setup , we have used just need 2 hostgroups:

hostgroup 0 for the master [Used for Write queries]
hostgroup 1 for the slaves [Used for Read Queries]

Apart from this we can also have one analytical server as slave of same master and we can assign new hostgroup id for the server and redirect all analytical related queries (long running) at this host.

```
Admin> INSERT INTO
mysql_servers(hostgroup_id,hostname,port) VALUES
(0,'172.17.0.1',3306);

Admin> INSERT INTO
mysql_servers(hostgroup_id,hostname,port) VALUES
(1,'172.17.0.2',3306);

Admin> INSERT INTO
mysql_servers(hostgroup_id,hostname,port) VALUES
(1,'172.17.0.3',3306);

Admin> INSERT INTO  mysql_replication_hostgroups VALUES
(0,1,'production');

Admin > SELECT * FROM mysql_replication_hostgroups;

+------------------+------------------+------------+
| writer_hostgroup | reader_hostgroup | comment    |
+------------------+------------------+------------+
| 0                | 1                | production |
+------------------+------------------+------------+

Admin> LOAD MYSQL SERVERS TO RUNTIME;


Admin> SAVE MYSQL SERVERS TO DISK;
```

> **Note:** When we load *MYSQL SERVERS* , Our writer host also get configured in reader hostgroup automatically by ProxySQL to handle all those queries which are redirected to reader hostgroup in case no slaves are online.
>
> So bonus point here is we can decrease the weightage assigned to master servers inside **mysql_server** table for reader hostgroup , so that our most of read queries will go on server which has higher weight.

```
UPDATE mysql_servers SET weight=200 WHERE hostgroup_id=1
AND hostname='172.17.0.1';

Admin> SELECT hostgroup_id,hostname,port,status,weight
FROM mysql_servers;
+--------------+-----------+------+--------+---------+
| hostgroup_id | hostname  | port | status | weight  |
+--------------+-----------+------+--------+---------+
| 0            | 172.17.0.1 | 3306 | ONLINE | 1000    |
| 1            | 172.17.0.2 | 3306 | ONLINE | 1000    |
| 1            | 172.17.0.3 | 3306 | ONLINE | 1000    |
| 1            | 172.17.0.1 | 3306 | ONLINE | 200     |
+--------------+-----------+------+--------+---------+
```

Configure User :

`monitor` user will continuously check the status of backend in specified interval.

`sysbench` is user created for the application.

```
Admin> UPDATE global_variables SET
variable_value='monitor' WHERE
variable_name='mysql-monitor_password';

Admin> LOAD MYSQL VARIABLES TO RUNTIME;
Admin> SAVE MYSQL VARIABLES TO DISK;

Admin> INSERT INTO
mysql_users(username,password,default_hostgroup)
VALUES (sysbench,sysbench,1);

Admin> SELECT
username,password,active,default_hostgroup,default_schema
,max_connections,max_connections FROM mysql_users;
```

```
+----------+----------+--------+------------------+-----
    | username | password | active | default_hostgroup |
default_schema | max_connections   | max_connections   |

+----------+----------+--------+------------------+-----
    | sysbench | sysbench | 1      |  0               |
  NULL          | 10000              | 10000           |

+----------+----------+--------+------------------+-----

Admin> LOAD MYSQL USERS TO RUNTIME;

Admin> SAVE MYSQL USERS TO DISK;
```

Configure monitoring :

ProxySQL constantly monitors the servers it has configured. To do so, it is important to configure some interval and timeout variables (in milliseconds).

```
Admin> UPDATE global_variables SET variable_value=2000
WHERE variable_name IN
('mysql-monitor_connect_interval','mysql-monitor_ping_int
erval','mysql-monitor_read_only_interval');

Admin> UPDATE global_variables SET variable_value = 1000
where variable_name = 'mysql-monitor_connect_timeout';

Admin> UPDATE global_variables SET variable_value = 500
where variable_name = 'mysql-monitor_ping_timeout';

Admin> LOAD MYSQL VARIABLES TO RUNTIME;

Admin> SAVE MYSQL VARIABLES TO DISK;
```

Monitor module regularly checks replication lag (using seconds_behind_master) if a server has max_replication_lag set to a non-zero value.

With below configuration, servers will only be shunned in case replication delay exceeds 60 seconds (1 min) behind master

```
Admin> UPDATE mysql_servers SET max_replication_lag=60;
Query OK, 1 row affected (0.00 sec)
```

Configure Query Rules :

To send all `SELECT` queries on slaves (based on Regex).

```
Admin> INSERT INTO mysql_query_rules (active,
match_digest, destination_hostgroup, apply)
VALUES (1, '^SELECT.*', 1, 0);

Admin> INSERT INTO mysql_query_rules (active,
match_digest, destination_hostgroup, apply)
VALUES (1, '^SELECT.*FOR UPDATE', 0, 1);

Admin> SELECT active, match_pattern,
destination_hostgroup, apply FROM mysql_query_rules;

Admin> SELECT rule_id, match_digest,destination_hostgroup
hg_id, apply FROM mysql_query_rules WHERE active=1;

+---------+----------------------+-------+-------+
| rule_id | match_digest         | hg_id | apply |
+---------+----------------------+-------+-------+
| 1       | ^SELECT .            | 1     | 0     |
| 2       | ^SELECT.*FOR UPDATE$ | 0     | 1     |
+---------+----------------------+-------+-------+
```

```
Admin> LOAD MYSQL QUERY RULES TO RUNTIME;

Admin> SAVE MYSQL QUERY RULES TO DISK;
```

When the Query Processor scans the query rules trying to find a match with no success and it reaches the end, it will apply the default_hostgroup for the specific user according to mysql_users entry.

In our case, user `sysbench` has a `default_hostgroup=0`, therefore any query not matching the above rules [Eg ALL WRITES] will be sent to hostgroup 0 [Master].Below stats tables are used to validate if your query rules getting used by incoming traffic.

```
SELECT rule_id, hits, destination_hostgroup hg FROM
mysql_query_rules NATURAL JOIN stats_mysql_query_rules;

+---------+-------+----+
| rule_id | hits  | hg |
+---------+-------+----+
| 1       | 17389 | 1  |
| 2       | 234   | 0  |
+---------+-------+----+
```

We can also redirect some specific pattern queries by using `digest` in `stats_mysql_query_digest`

<u>Validate the DB Connection :</u>

Application will connect to `6033` port on host `172.17.0.4` of ProxySQL to send DB traffic.

```
ProxySQL-Host$ mysql -u sysbench -psysbench -h 127.0.0.1
-P6033 -e "SELECT @@server_id"

+-------------+
| @@server_id |
+-------------+
|           2 |
+-------------+
```

Check Backend Status :

It shows ProxySQL is able to successfully connect to all backends.

```
mysql> select * from monitor.mysql_server_ping_log order
by time_start_us desc limit 3;

+------------+------+------------------+----------------
| hostname | port |time_start_us |ping_success_time_us |
ping_error |
+------------+------+------------------+----------------
| 172.17.0.1 | 3306 | 1516795814170574 | 220
| NULL       |
| 172.17.0.2 | 3306 | 1516795814167894 | 255
| NULL       |
| 172.17.0.3 | 3306 | 1516795804170751 | 259
| NULL       |
+------------+------+------------------+----------------
```

Below table of ProxySQL shows number of queries executed per host.

```
Admin > select
hostgroup,srv_host,status,Queries,Bytes_data_sent,Latency
_us from stats_mysql_connection_pool where hostgroup in
(0,1);
```

```
+-----------+-----------+---------+---------+----------
| hostgroup |srv_host |status | Queries | Bytes_data_sent
| Latency_us |
+-----------+-----------+---------+---------+----------
| 0         | 172.17.0.1 | ONLINE   | 12349   | 76543232
| 144       |
| 1         | 172.17.0.2 | ONLINE   | 22135   | 87654356
| 190       |
| 1         | 172.17.0.3 | ONLINE   | 22969   | 85344235
| 110       |
| 1         | 172.17.0.1 | ONLINE   | 1672    | 4534332
| 144       |
+-----------+-----------+---------+---------+----------
```

If any of your server goes unreachable from any hostgroup , status gets changed from `ONLINE` to `SHUNNED`.

It means ProxySQL wont send any queries to that host until it comes back to `ONLINE`.

We can also take any server offline for maintenance. To disable a backend server it is required to change its status to `OFFLINE_SOFT` (Gracefully disabling a backend server) or `OFFLINE_HARD`(Immediately disabling a backend server.)

In this case no new traffic will be sent to the node.

```
Admin> UPDATE mysql_servers SET status='OFFLINE_SOFT'
WHERE hostname='172.17.0.2';

Query OK, 1 row affected (0.00 sec)
```