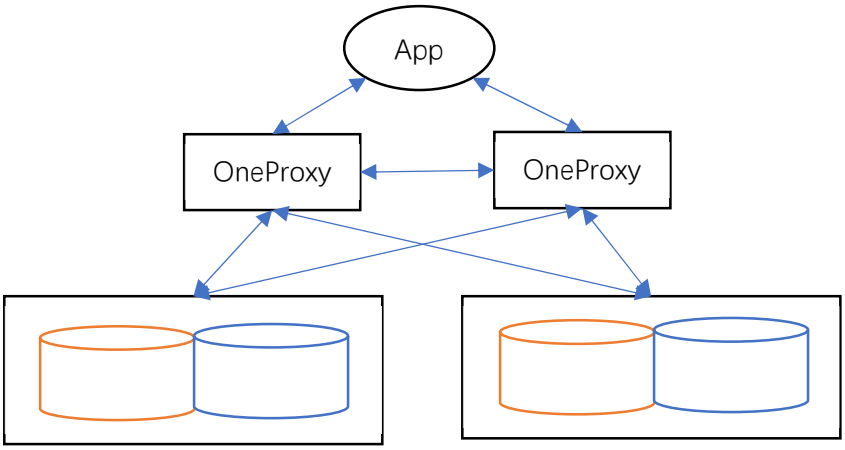
OneProxy 是一个高效稳定的 MySQL 协议层代理软件，可以透明地支持 MySQL 架构的横向扩展。它具备以下令人心动的功能特点：

- 超 10 个用户已经稳定使用长达 5 年之久，商业软件品质保障。
- 低时延，每个查询增加不到 100us 的时延，远超其他同类 Proxy 软件。
- 稳定性，软件采用 C&C++语言编写，无内存 GC 问题，多个用户场景验证超过一年不用重起。
- 高性能，代理需要解释 SQL 语句，是 CPU 消耗型的操作，每个核每秒能处理超 25000 的查询转发，良好的单机扩展能力，也可构建 Proxy 集群做横向扩展。
- 后端连接池功能，可以有效控制到 MySQL 节点上的总连接数。
- 支持 MySQL Group Replication 和 Percona Xtradb Cluster 集群架构，自动快速识别后端节点切换识别主节点。
- 支持读写分离等多种流量转发策略，无缝实现读扩展。
- 支持分库分表，并有跨节点的结果集聚合操作支持能力，对应用的透明度较高。
  - 支持 Int、Big Int、Char、Date、Timestamp 类型的单列分区。
  - 支持 Range、Hash、List 分区方式，支持二级子分区。
  - 支持分区信息冗余到不同的字段，以更好地进行分区过滤。
  - 支持跨结点的结果集合并及排序（order by）。
  - 支持跨节点的结果集汇总操作（count/sum/max/min）。
  - 支持跨节点的结果集的分组汇总（group by）。
  - 支持跨节点的结果集分页操作（limit/offset）。
  - 支持基于分片的并行查询操作（parallel query）。
- 支持结果集缓存，透明提升查询操作性能。
- 支持前后端密码分离，确保数据库密码不外泄。
- 支持 IP 白名单，支持表级安全设置，支持 SQL 防火墙，阻止 SQL 注入式攻击。
- 内置 HA 机制，无须安装配置第三方软件，实现代理节点的快速故障转移。
- 具备丰富的 SQL 性能统计信息，内置 Http 服务，可以轻松查看性能数据。
- 内置序列号生成器，可以高效生成单向增长的序列值。
- 支持 MySQL 8 协议，支持最新版本的 MySQL JDBC 驱动程序。

今天大量的企业已在使用 MySQL，也熟悉了分库分表操作，一个好的数据访问层（DAL）可以起到事半功倍的作用，OneProxy 就是一个经得起考验的优秀数据访问层软件。

要理解 OneProxy 的关键配置项，就需要了解它的基本布署架构，如下图所示：



由于要支持分库分表，所以需要支持挂载多个 MySQL 主备或 MGR 集群，每个集群需要有唯一的名字，可以称之为组名（Group Name）。每个集群可以是一主多从多个结点，也可以是 MGR 或 PXC 集群。怎么样来添加所有的节点？只要提供三个关键的信息：

- MySQL 节点 IP 地址
- MySQL 节点端口号
- 节点所在集群的名字（Group Name）

配置节点可需要到以下两个选项：

- proxy-master-addresses.[1-256] = 节点 IP:端口@集群的名字
- proxy-slave-addresses.[1-256] = 节点 IP:端口@集群的名字

需要注意的是，这里并不需要提供 MySQL 数据库的名字。MySQL 数据库名字在登录信息中提供，需要包含以下 4 个信息：

- 集群的名字（Group Name）
- 登录用户名
- 登录密码（用自带 mysqlpwd 加密，MySQL 8 需要用 mysql_native_password 认证）
- 登录数据库名字

配置登录信息需要用到以下两个选项：

- proxy-user-list.[1-256] = 用户名/加密后口令@数据库名字
- proxy-user-group.[1-256] = 集群的名字:用户名/加密后口令@数据库名字

前者指定了应用连接 OneProxy 的登录信息，后者指定了某个集群的登录信息。如果对同一个用户名，在两个地方都指定了信息，则起到前后端密码分离的作用。

许多HA软件(包括MGR)都会通过设置“read_only”变量来标识节点是读写节点(Primary)还是只读节点（Slave），可以在 OneProxy 中设置以下变量值为 1 来自动识别读写状态，以设置准确的节点类型:

- proxy-auto-readonly = {0|1}

OneProxy 会以 50ms 的频率去检查后端节点状态，基本上是应用无感的秒切。只要 HA 或 MGR 集群能保证同一时间点只有一个节点会关闭“read_only”变量,就不会存在双写情况。这样可以省去通知前端应用写库切换的工作，在大规模应用布署下，这个通知机制还是比较复杂的，使用 OneProxy 则简洁多了。

只要再提供其他几个关键的选项，就可以启动 OneProxy 来进行测试了。如下所示:

- mysql-version = 虚拟版本号（应当是所有节点中最小的 MySQL 版本号）
- event-threads = 允许的 CPU 核数（一般不超过 16，平均每核 2.5 万 QPS 转发）
- proxy-address = 监听地址（默认是“:3307”，等同于 MySQL 的 Listener 地址）
- proxy-group-policy = 集群的名字:流量策略（master-only 或 read-balance）

接下来就可以来准备一个完整的配置文件（conf/proxy.conf）了，如下所示:

```
[oneproxy]
#proxy-address              = :3307
mysql-version               = 8.0.27
event-threads               = 2

proxy-auto-readonly         = 1
proxy-group-policy          = default:read_balance
proxy-master-addresses.1    = 127.0.0.1:3306@default
proxy-user-list.1           = default:test/password@test
```

OneProxy 会自动启用一个守护进程，以保证单实例的高可用，如下所示:

```
[root@RH6SRV1 ~]# ps -ef | grep oneproxy
root      6545     1  0 Mar29 ?        00:00:00 /data/oneproxy/bin/oneproxy
root      6546  6545  3 Mar29 ?        00:30:15 /data/oneproxy/bin/oneproxy
root      8603  8587  0 00:41 pts/2    00:00:00 grep oneproxy
```

接下来就可以开启压测了，如查你用 sysbench 工具，则需要加上“--db-ps-mode=disable”选项（因为在分库分表的情况下，Prepare Statement 的管理过于复杂，并且 MySQL 目前并不能从 Prepare Statement 模式得到多少性能提升，其 jdbc 驱动默认也未开启)。

现在可以将 OneProxy 当成 MySQL 节点，将连接信息配置成它的监听地址即可。

OneProxy 在设计时充分考虑了安全性，可以在整个 Proxy 实例层面、集群层面、表层面指定安全级别。可以通过以下几个选项来灵活控制：

- proxy-security-level = level
- proxy-group-security = 集群的名字:level
- proxy-table-security = 表名:level

这里“level”是一个整数，不同的标志位表示不同的含义，其定义如下表所示：

标志位	含义
0x01	禁止 DDL 操作
0x02	针对 Update 和 Delete 操作，必须有明确的 Where 条件
0x04	不允许 Delete 操作
0x08	不允许 DML 操作，只允许查询语句

集群级别的默认值是“1”，即禁止 DDL 操作，除非在可信的 IP 范围内（一般情况下应用代码中是不会有 DDL 操作的，不排除少数应用代码中有 Truncate 操作）。默认情况下本地连接（“127.0.0.1”和“unix_socket”）为可信连接，远程连接可以通过如下参数进行设置：

- proxy-secure-client.[1-255] = IP 地址

比如 DBA 常用的登录机器，可以设置为可信地址，以便日常管理。还可以通过开启 IP 白名单功能来进行 IP 维度的访问控制。可以使用以下两个选项：

- proxy-enable-ipfirewall = {0|1}
- proxy-firewall-ip = IP 列表文件（一行一个 IP 地址，在起动时加载）

也可以临时禁止某个已经访问过的 IP 地址的访问，将其设置为拒绝模式，可以登录 OneProxy 管理端口（默认用户名 admin，默认密码 OneProxy，默认端口 4041，都可以定制）使用以下两个命令进行控制：

- set blackip '192.168.0.1'，禁止某个 IP 的后续登录。
- set greenip '192.168.0.1'，允许某个 IP 的后续登录。

可以通过”list ipqos”命令来查看有哪些 IP 地址登录过 OneProxy，如下图所示：

```
mysql> list ipqos;
+-------------+------+------+------+------+------+------+------+------+------+------+------+------+
| ADDRESS     | DENY | Sess | QoS  | T01  | T02  | T03  | T04  | T05  | T06  | T07  | T08  | T09  | T10  |
+-------------+------+------+------+------+------+------+------+------+------+------+------+------+------+
| 127.0.0.1   |    0 |    1 |    0 |    0 |    0 |    0 |    0 |    0 |    0 |    0 |    0 |    0 |    0 |
| unix_socket |    0 |    0 |    0 |    0 |    0 |    0 |    0 |    0 |    0 |    0 |    0 |    0 |    0 |
+-------------+------+------+------+------+------+------+------+------+------+------+------+------+------+
2 rows in set (0.00 sec)
```

第一列表示是否禁止登录，第二列表示有多少个连接，后续字段是过去 10 个秒级时间点的实时 QPS 情况。

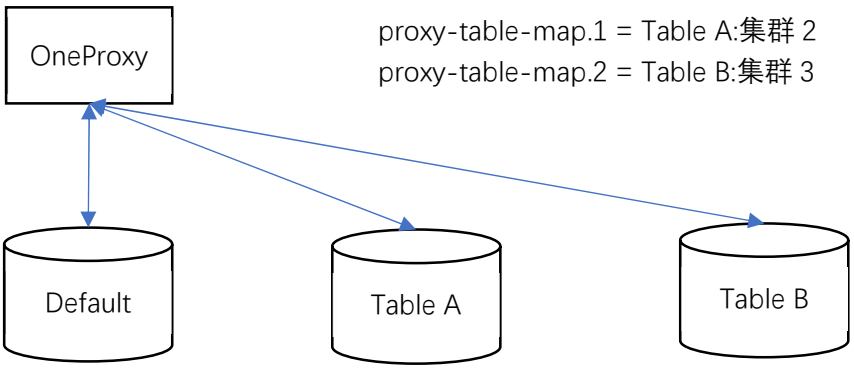
OneProxy 在设计时做了一些功能取舍，以便更简洁地在一个软件中同时支持读写分离和分库分表，也就是对应用还是有一些要求，如果不满足则无法使用 OneProxy 来做数据访问层了。主要的取舍如下：

- 不支持 Prepare Statement 接口。
  - 对于 JDBC，默认未使用此接口，可以运行。
  - 对于 PHP PDO，可以加上“PDO::ATTR_EMMULATE_PREPARE”连接属性。
  - 对于 Go 的 MySQL 驱动，可以加上“interpolateParams”连接属性。
  - 对于 MySQL 的 C 客户端，不要使用“mysql_stmt_*”等函数来操作数据。
- 不支持 Set 语句（直接返回成功，但不做任何事情），因此不能设置用户级变量，也不能更改会话级的设置。因为后端的连接池机制需要保证连接不带状态属性。但”set autocommit = {0|1}”可以安心使用，不用担心 OneProxy 不支持。
- 不支持”USE”命令来切换当前数据库。从 OneProxy 角度来看，不同的集群相当于原先 MySQL 实例的数据库，“USE”命令也用来切换默认集群，而不是切换后端的默认数据库。
- 不支持分布式事务语及跨集群的 SQL 语句。虽然 OneProxy 后面能同时挂载多个集群，但其核心本质是一个功能较强、稳定和高效的流量路由软件，没有完整的 SQL 优化器和执行层，目前还不是一个真正的分布式数据库。

虽然 OneProxy 不支持一个 SQL 跨多个集群的操作，但仍可以用于垂直拆分，将无关联的表透明地移到其他集群，并且标记一下此表所在的集群名字，OneProxy 就可以实现灵活的表级路由控制，将后端多个集群虚拟化成一个大集群。可以通过如下选项：

- proxy-table-map.[1-255] = 表名:集群名字

分库分表情况下，也是如此，会自动维扩每个分表所在的集群名字，自动建立对应关系。



OneProxy 使用自己打点的方式来检测主从时延，在单主模式的流量策略下会自动创建心跳表（包含 Proxy UUID 和时间点两个字段，表名为：oneproxy_replication_timestamp），并以 100ms（可配置）的频率进行更新和查询，以计算主从时延。心跳表信息如下：

```
mysql> select * from oneproxy_replication_timestamp;
+--------------------------+------------------+
| proxy_uuid               | proxy_stamp      |
+--------------------------+------------------+
| ANGV-FXAA-GVJV-AAJQ-JVXA | 1648677892621686 |
+--------------------------+------------------+
1 row in set (0.00 sec)
```

通过以下两个选项来控制，更新和查询的频率：

- checkpoint-interval = 100 查询 Slave Binlog 时间点的频率
- checkpoint-confirm = 100 更新 Master Binlog 时间点的频率

采用自己打点的原因是“show slave status”里的 SBM 的信息不足够准确，并且只有秒级精度，而 OneProxy 希望能做到 100ms 级别的精度。当从节点的时延超过 60 秒（可配置）时，在读写分离时就不会向从节点分配流量。目前这个配置项是整个 OneProxy 实例级别的，没有实现集群级别的设置。控制项如下所示：

- binlog-maxdelay = 60 超过此值则自动踢除，低于此值则自动加回

由于 MySQL 上存在读写干扰情况，为了保证有一个节点能够及时跟上主库，可以设置节点为 HA 类型节点，这类节点在读写分离时不会承担读流量，以保证其恢复速度。可通过如下选项进行设置：

- proxy-haonly-addresses.[1-255] = 主机名:端口

这里需要你的 HA 机制确保主节点切换操作只在 HA 类型节点之间发生。除了主从时延检测之外，OneProxy 还实现了更强的一致性读的功能，当某个表被更新后的 100ms（通过“global-rwlatency”进行配置）内，对此表的所有查询都会自动在主节点上处理，以确保更新后的查询（和更新不在同一个事务内）可以被看到，避免更新后看不到数据的现象。

所有的流量策略可以在管理端口执行“list policy”来获取，常见的有如下几类：

- master_only 读写主节点
- read_failover 读写主节点，主节点不可用则读从节点
- read_slave 写主节点，随机读 Slave 类型节点
- read_other 写主节点，随机读 Master 之外的节点
- read_balance 写主节点，随机读所有节点

OneProxy 丰富的流量策略和全局一致性功能，可以让你轻松应对各种业务场景。