[三思笔记] MySQL Proxy 应用入门

2011-2-16 君三思

http://www.5ienet.com/

MySQL Proxy(MySQL 代理)是一个通过 MySQL 网络协议，提供 MySQL 服务器与客户端之间连接的应用工具，在基本配置条件下，MySQL 代理仅是传递客户端发出的查询请求到 MySQL 服务器端，而后返回 MySQL 服务器的响应到客户端。

由于 MySQL 代理使用的 MySQL 网络协议，因此所有 MySQL 兼容的客户端(包括 mysql 命令行、调用 mysql 命令行的类库、以及支持 MySQL 网络协议的应用)均可无需修改连接代理。

通过配置，MySQL 代理同样能够监测及修改客户端与服务器端的通讯，这样 DBA 可以控制客户端提交的查询，比如调整查询的结果集，甚至可以跳过 MySQL 数据库，直接返回数据给客户端。

本文档基于 MySQL 代理 0.8.0 版本。

提示：
MySQL 代理当前仍为 alpha 版本，不建议在产品环境下使用。

MySQL 代理预编译版本支持的平台还算广泛，包括 Linux(含 RedHat,Fedora,Debian,SuSE 等)，Mac OS X，FreeBSD，IBM AIX，Sun Solaris，Microsoft Windows(xp,vista,server2003/2008)等均可支持。

一、安装 MySQL 代理

有下列几种安装方式可选：

- 采用预编译的二进制版本：
- 使用源码编译方式安装：

1.1 二进制方式安装

这种方式操作比较简单，基本上就是解压缩，然后修改 pach 环境变量，加入 MySQL Proxy 命令行的路径即可，简述步骤如下：

```
# tar zvxf mysql-proxy-0.8.x-os.tar.gz
# export PATH=$PATH:mppath/sbin
```

1.2 源码方式安装

源码统计的话，下列依赖包需要首先被安装：

- libevent 1.x or higher (1.3b or later is preferred)
- lua 5.1.x or higher
- glib2 2.6.0 or higher

- pkg-config
- libtool 1.5 or higher
- MySQL 5.0.x or higher developer files

而后解压缩下载到的源码包，并执行 configure 进行配置

```
shell> tar zxf mysql-proxy-0.7.2.tar.gz
shell> cd mysql-proxy-0.7.2
shell> ./configure
```

执行 make 进行编译

```
shell> make
```

执行 make check 检查编译的情况

```
shell> make check
```

执行 make install 进行安装

```
shell> make install
```

默认情况下 mysql-proxy 会被安装到/usr/local/sbin/mysql-proxy 中。

Linux 下源码方式安装，详细操作步骤可见：http://www.5ienet.com/note/html/stmp08/index.shtml

二、MySQL Proxy 配置选项

启动 MySQL Proxy 对应的命令行正是 mysql-proxy，位于安装路径/bin 目录下，mysql-proxy 命令行也提供了一些参数，用来实现不同的功能，在启动 MySQL Proxy 前必须对部分参数进行设置，比如监控服务器地址，名称，端口等。

直接执行 mysql-proxy 命令，附加--help-all 参数，可以查看到该命令支持的所有参数，及调用语法：

```
[root@rhel5u3 ~]# /usr/local/mysql-proxy/bin/mysql-proxy --help-all
Usage:
  mysql-proxy [OPTION...] - MySQL Proxy

Help Options:
  -h, --help                                  Show help options
  --help-all                                  Show all help options
  --help-admin                                Show options for the admin-module
  --help-proxy                                Show options for the proxy-module

admin-module
  --admin-address=<host:port>                 listening address:port of the admin-server
(default: :4041)
```

```
      --admin-username=<string>                          username to allow to log in
      --admin-password=<string>                          password to allow to log in
      --admin-lua-script=<filename>                      script to execute by the admin plugin

    proxy-module
      -P, --proxy-address=<host:port>                        listening address:port of the proxy-server
(default: :4040)
      -r, --proxy-read-only-backend-addresses=<host:port>    address:port of the remote slave-server (default:
not set)
      -b, --proxy-backend-addresses=<host:port>              address:port of the remote backend-servers
(default: 127.0.0.1:3306)
      --proxy-skip-profiling                             disables profiling of queries (default: enabled)
      --proxy-fix-bug-25371                              fix bug #25371 (mysqld > 5.1.12) for older libmysql
versions
      -s, --proxy-lua-script=<file>                      filename of the lua script (default: not set)
      --no-proxy                                         don't start the proxy-module (default: enabled)
      --proxy-pool-no-change-user                          don't use CHANGE_USER to reset the connection
coming from the pool (default: enabled)

    Application Options:
      -V, --version                                      Show version
      --defaults-file=<file>                             configuration file
      --verbose-shutdown                                 Always log the exit code when shutting down
      --daemon                                           Start in daemon-mode
      --user=<user>                                      Run mysql-proxy as user
      --basedir=<absolute path>                          Base directory to prepend to relative paths in the
config
      --pid-file=<file>                                  PID file in case we are started as daemon
      --plugin-dir=<path>                                path to the plugins
      --plugins=<name>                                   plugins to load
      --log-level=(error|warning|info|message|debug)     log all messages of level ... or higher
      --log-file=<file>                                  log all messages in a file
      --log-use-syslog                                   log all messages to syslog
      --log-backtrace-on-crash                           try to invoke debugger on crash
      --keepalive                                        try to restart the proxy if it crashed
      --max-open-files                                   maximum number of open files (ulimit -n)
      --event-threads                                    number of event-handling threads (default: 1)
      --lua-path=<...>                                   set the LUA_PATH
      --lua-cpath=<...>                                  set the LUA_CPATH
```

从语法可以看得出来，基本上该命令行的使用是非常简单的，常规需求仅通过命令行+不同参数的组合即可实现。MySQL Proxy 与 MySQL 系出同门，那么参数的指定方式自然也很类型，上述命令行中显示的参数，也可以指定在配置文件中，这个配置文件与 mysql 的 my.cnf 很类似，甚至完全可以将参数就放在 my.cnf 中，在调用 mysql-proxy 命令时通过

--defaults-file 指定参数文件，参数文件中参数配置规则也与 mysql 相同，例如：

```
[mysql-proxy]
admin-address = host:port
admin-user = root
admin-pass = verysafe
.........
```

最上方的[mysql-proxy]标识参数的作用域，参数文件中指定参数不需要加--前缀。

下面就各个参数的不同意义来逐个说明，mysql-proxy 支持的参数可以分成四类：

2.1 帮助相关参数

与帮助相关的参数共有四个：

- --help：显示常用的帮助选项；
- --help-all：显示全部帮助选项；
- --help-admin：显示管理模块的帮助选项；
- --help-proxy：显示代理模块的帮助选项；

2.2 管理相关参数

- --admin-address=host:port：指定管理员主机及服务端口，默认值为 localhost:4041。
- --admin-lua-script=script_file：指定管理模块的 lua 脚本文件。
- --admin-username=user：指定登录到 mysql-proxy 管理界面的用户名。
- --admin-password=pass：指定登录到 mysql-proxy 管理界面的用户密码。

2.3 代理相关参数

- --proxy-address=host:port：简写形式-P

指定监听服务的主机名(或 IP 地址)及服务端口，默认端口号为 4040。

- --proxy-backend-addresses=host:port，简写形式-b

指定监听的主机名(或 IP 地址)及端口，指定的主机为代理实现连接的 MySQL 服务器。可以通过本参数同时指定多个服务器的方式，实现 mysql 的负载轮循。如果监听的 mysql 服务有多个，MySQL 代理会自动按照循环方式分配客户端连接到后台的 mysql 服务中。比如说当前设置了 a 和 b 两项 mysql 服务，当第一个客户端发起连接请求时，会连接到服务 a，第二个连接请求则连接到服务 b，而第三个连接请求又连接服务 a，以此循环。

需要注意本参数在命令行与参数文件中使用是稍有差异，主要表现中同时指定多个服务的情况下。

当使用命令行模式调用本参数时，参数后只能跟一个服务，如果要代理的服务有多个，那么必须同时指定多个参数，比如说：

```
# mysql-proxy --proxy-backend-addresses 192.168.0.1:3306 --proxy-backend-addresses 192.168.0.2:3306
```

如果是使用配置文件指定本参数的话，就可以在一个参数中指定多个值了，参数值之间以,(逗号)分隔即可，例如：

```
proxy-backend-addresses = 192.168.0.1:3306, 192.168.0.2:3306
```

- --proxy-read-only-backend-addresses=host:port：简写形式-r

指定监听的主机名(或 IP 地址)及端口，该服务器将仅用于提供只读服务。

注意：

只有当服务器对应的内部结构(详见 proxy.global.backends)进行了相关配置，可以通过检查 backend 的 type 列确定其连接的类型时，设置本参数才有效。因此，本选项仅适用于指定 lua 脚本文件，允许使用不同 backend 类型的情况。

- --proxy-lua-script=filename：简写形式-s

指定加载的 lua 脚本文件路径，注意这个脚本文件并非 mysql-proxy 启动时即加载和解析，而是直到第一个连接创建时才解析，而后每次连接均会重新加载，也就是说在 mysql-proxy 运行期间可以动态修改 lua 的脚本文件，保存后会在下次连接创建时即时生效。

- --proxy-pool-no-change-user

当从连接池(proxy-backend-addresses 列表)中再次获取连接时，禁止调用 MySQL CHANGE_USER 接口。默认情况下允许。

- --proxy-fix-bug-25371

修复 bug#25371 (mysqld > 5.1.12)的旧 libmysql 问题。

- --proxy-skip-profiling

禁止分析查询(跟踪时间统计信息)，默认情况下允许。

- --no-proxy：禁用代理模块。

2.4 调用相关的参数

- --basedir：指定 mysql-proxy 程序运行的基础目录，本参数指定的路径必须是绝对路径，如果指定相对路径的话，mysql-proxy 执行时会抛出异常信息。
- --daemon：以守护进程模式运行。
- --defaults-file：指定参数文件路径，如果不指定本参数，则表示参数由命令行执行时指定。
- --event-threads：指定事件处理的线程数，默认为 1。
- --keepalive：创建一个守护进程，当发现 mysql-proxy 进程崩溃则自动重新启动，windows 平台下该参数无效。
- --log-backtrace-on-crash：当 mysql-proxy 进程崩溃时调用 debug 生成跟踪信息。
- --log-file：指定日志文件，记录 mysql-proxy 运行过程中的信息。
- --log-level：日志记录级别，有 error/warning/info/message/debug 几种选择。
- --log-use-syslog：记录 mysql-proxy 错误日志到系统日志中，仅用于 linux/unix 平台。
- --lua-cpah：设置 LUA_CPATH 环境变量

- --lua-path：设置 LUA_PATH 环境变量
- --max-open-files：指定 mysql-proxy 最大能打开的文件数
- --pid-file：指定存储 pid 信息的文件(仅用于守护模式)
- --plugin-dir：指定插件所在路径；
- --plugins：指定加载的插件；
- --user：指定运行 mysql-proxy 的用户。
- --version：简写形式-V，显示 mysql-proxy 的版本信息。

三、使用 MySQL Proxy

MySQL Proxy 的使用其实是非常简单的，启动 MySQL Proxy 服务对应的命令行是 mysql-proxy，如上节所示该命令行的参数就那么几类，常用的参数更是屈指可数。

一般来讲，下列几个参数是必须指定的：

- admin-username：指定管理员用户
- admin-password：指定管理员密码
- admin-lua-script：指定管理模块处理脚本
- daemon：指定以守护模式运行
- proxy-backend-addresses：指定代理实际连接的 MySQL 数据库

例如，执行 mysql-proxy 命令并指定参数：

```
[root@rhel5u3 ~]# /usr/local/mysql-proxy/bin/mysql-proxy --admin-username=jss --admin-password=123
--admin-lua-script=/usr/local/mysql-proxy/lib/mysql-proxy/lua/admin.lua -b 172.16.1.113:3307 --daemon
```

而后可以通过 netstat -lnt 查询相应端口是否已经处于监听状态：

```
[root@rhel5u3 ~]# netstat -lnt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 127.0.0.1:2208              0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:11111               0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:4040                0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:4041                0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:3306                0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:16851               0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:21                  0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:631               0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:666                 0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:2207              0.0.0.0:*                   LISTEN
tcp        0      0 :::1521                     :::*                        LISTEN
tcp        0      0 :::60918                    :::*                        LISTEN
tcp        0      0 :::22                       :::*                        LISTEN
tcp        0      0 ::1:631                     :::*                        LISTEN
[root@rhel5u3 ~]#
```

可以看到默认的管理端口 4041 和监听端口 4040 均已启动。服务启动无误，下面可以通过这台代理服务器来连接 172.16.1.113:3307 数据库了。

3.1 初次连接

首先登录到目标数据库，创建相应用户并授予权限，为了能够清晰了表现 mysql-proxy 的工作，我们在创建用户时，指定仅允许 MySQL Proxy 所在服务器连接：

```
[root@mysqldb2 ~]# mysql -uroot -p'verysafe' -S /data/mysqldata/3307/mysql.sock
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 5.1.51-junsansi-edition-log Source distribution

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> grant all on jssdb.* to jss@'172.16.1.110' identified by 'jss';
Query OK, 0 rows affected (0.00 sec)
```

如上所示，我们在创建用户时仅允许来自 172.16.1.110(即 mysql-proxy 所在服务器)的服务器连接，这个时候，我们转到一台 client 端执行连接，首先直接连接目标数据库服务器(由于服务器端 jss 用户限制了连接 IP，直接连接应该是连接不上的)：

```
[root@mysqldb1 ~]# mysql -ujss -pjss -h 172.16.1.113 -P 3307
ERROR 1045 (28000): Access denied for user 'jss'@'172.16.1.112' (using password: YES)
```

连接果然被拒绝，而后再尝试通过 MySQL-Proxy 代理连接 MySQL 服务器：

```
[root@mysqldb1 ~]# mysql -ujss -pjss -h 172.16.1.110 -P 4040
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.1.51-junsansi-edition-log Source distribution

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| jssdb              |
| test               |
+--------------------+
```

```
3 rows in set (0.00 sec)
```

顺利连接，表明 mysql-proxy 代理的工作正常。

3.2 管理 mysql-proxy

Mysql-proxy 自己维护了一个可通过 mysql 命令行模式管理的操作界面，连接时按照 mysql 命令行格式指定用户名/密码/服务器 IP，而后指定代理服务器的 4040 端口访问，例如：

```
[root@mysqldb1 ~]# mysql -ujss -p123 -h 172.16.1.110 -P 4041
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.0.99-agent-admin

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

那么这个界面是怎么来的，支持的命令有哪些，功能是什么，应该如何扩展？这些都不是问题，看一下启动 mysql-proxy 命令时指定的 lua 脚本文件即可明了，如下：

```
[root@rhel5u3 ~]# more /usr/local/mysql-proxy/lib/mysql-proxy/lua/admin.lua
..........
..........
function set_error(errmsg)
        proxy.response = {
                type = proxy.MYSQLD_PACKET_ERR,
                errmsg = errmsg or "error"
        }
end

function read_query(packet)
        if packet:byte() ~= proxy.COM_QUERY then
                set_error("[admin] we only handle text-based queries (COM_QUERY)")
                return proxy.PROXY_SEND_RESULT
        end

        local query = packet:sub(2)

        local rows = { }
        local fields = { }

        if query:lower() == "select * from backends" then
                fields = {
```

```
                { name = "backend_ndx",
                  type = proxy.MYSQL_TYPE_LONG },

                { name = "address",
                  type = proxy.MYSQL_TYPE_STRING },
                { name = "state",
                  type = proxy.MYSQL_TYPE_STRING },
                { name = "type",
                  type = proxy.MYSQL_TYPE_STRING },
                { name = "uuid",
                  type = proxy.MYSQL_TYPE_STRING },
                { name = "connected_clients",
                  type = proxy.MYSQL_TYPE_LONG },
        }

        for i = 1, #proxy.global.backends do
                local states = {
                        "unknown",
                        "up",
                        "down"
                }
                local types = {
                        "unknown",
                        "rw",
                        "ro"
                }
                local b = proxy.global.backends[i]

                rows[#rows + 1] = {
                        i,
                        b.dst.name,          -- configured backend address
                        states[b.state + 1], -- the C-id is pushed down starting at 0
                        types[b.type + 1],   -- the C-id is pushed down starting at 0
                        b.uuid,              -- the MySQL Server's UUID if it is managed
                        b.connected_clients  -- currently connected clients
                }
        end
elseif query:lower() == "select * from help" then
        fields = {
                { name = "command",
                  type = proxy.MYSQL_TYPE_STRING },
                { name = "description",
                  type = proxy.MYSQL_TYPE_STRING },
        }
```

```
                rows[#rows + 1] = { "SELECT * FROM help", "shows this help" }
                rows[#rows + 1] = { "SELECT * FROM backends", "lists the backends and their state" }
        else
                set_error("use 'SELECT * FROM help' to see the supported commands")
                return proxy.PROXY_SEND_RESULT
        end

        proxy.response = {
                type = proxy.MYSQLD_PACKET_OK,
                resultset = {
                        fields = fields,
                        rows = rows
                }
        }
        return proxy.PROXY_SEND_RESULT
end
```

这个脚本是 mysql-proxy 自带文件，功能非常简单，语法对于初接触者会感到有些生疏，但逻辑结构还是比较容易理解的，从上述的这段脚本可以看出，当前仅支持 select * from help/backends 两条语句，其中 select * from help 会返回两条记录，显示支持的语句及简述，select * from backends 则显示提供服务的后台数据库服务器列表，及各服务器的状态等信息。

3.3 lua 脚本的应用

MySQL Proxy 主要基于 lua 脚本来扩展代理服务器的功能，并且从设定上就将 lua 分成两类：

- 一类负责管理模块的控制，对应参数 admin-lua-script，我们上小节看到的脚本就属于这类。
- 另一类负责代理模块的控制，对应参数 proxy-lua-script。

两类脚本的编码规则完全相同，只是对应功能有差异，管理模块侧重于代理服务器相关状态的控制，代理模块则侧重于客户端的 CRUD 操作。

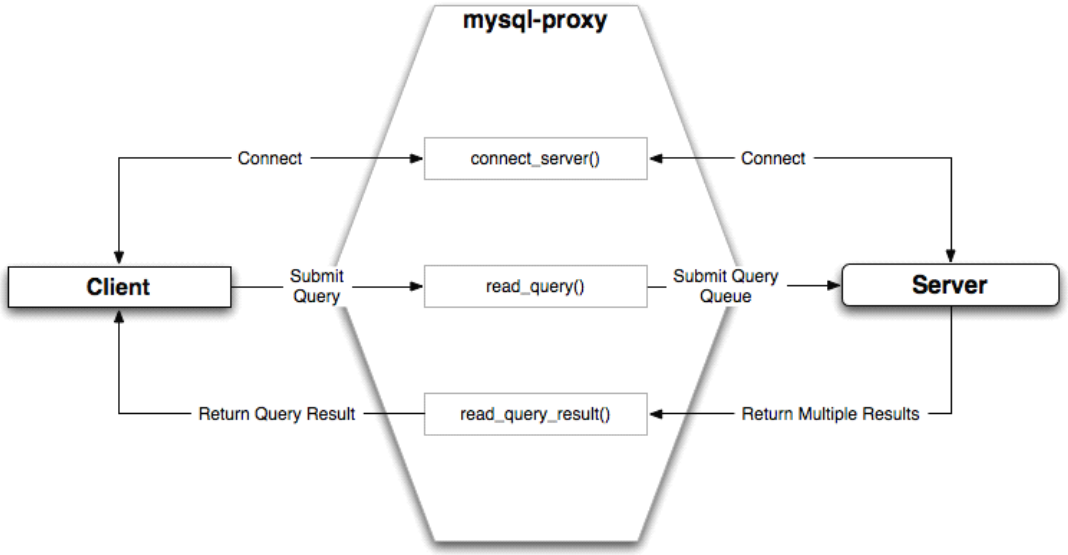
MySQL Proxy 支持通过 LUA 脚本定制下列几个函数，来处理客户端与 mysql 数据库之间的交互：

- connect_server()：当客户端发起连接请求到 mysql 代理时，就会调用该函数，DBA 可以定制该函数实现负载均衡，以及更复制的客户端-服务器的连接条件。
- read_handshake()：当服务器返回握手(handshake)信息时调用该函数，DBA 可以定制该函数提供附加返回信息，或者在验证用户身份前首先执行自定义的其它检查。
- read_auth()：当客户端提交认证包(含用户名、密码、默认数据库)到服务器时调用该函数。
- read_auth_result()：当服务器返回认证包给客户端时会调用该函数。
- read_query()：当客户端发起查询请求到 mysql 服务器时会调用该函数，DBA 可以通过定制该函数以修改和维护客户端要执行的查询，甚至跳过 mysql 数据库直接向客户端返回结果。

- read_query_result()：当 mysql 服务器返回查询结果到客户端时调用该函数，DBA 可以通过定制该函数，修改或过滤结果集。

我感觉非常重要的一点在于，这几个函数是用来控制客户端与 mysql 服务器之间的交互过程，如果不设定完全可以，则按照默认逻辑执行查询或返回记录到在客户端与 mysql 服务器之间，而通过这些函数的定制，则基本可以比较随意的控制查询结果，如前面所说，甚至可以不发查询语句到 mysql 服务器，而直接返回记录给客户端。

Mysql-proxy 代理在客户端与 mysql 服务器之间的交互控制逻辑图如下：



更多关于 lua 脚本方面的内容参考 lua 官网：http://www.lua.org，mysql-proxy 处理脚本的内部结构与脚本调用示例可参考 mysql 官网：http://dev.mysql.com/doc/refman/5.1/en/mysql-proxy-scripting.html。

3.4 测试性能

Mysql 的性能测试范围太广，这里仅是使用 mysql 自带的 mysqlslap 命令行工具进行简单测试，测试流程为使用 mysqlslap 命令指定相同参数，分别连接代理和 mysql 服务器，对比响应时间。

测试前要先创建测试用户，连接到目标数据库执行下列语句。

```
mysql> grant all on testdb.* to test@'172.16.1.%' identified by 'test';
Query OK, 0 rows affected (0.00 sec)
```

接下来就可以执行测试了，先是直接连接 mysql 服务器，执行命令如下：

```
[root@mysqldb1 ~]# mysqlslap -utest -ptest -h 172.16.1.113 -P 3306 --create-schema=testdb -x 2 -y 3 -a -c 20 -i 50
Benchmark
        Average number of seconds to run all queries: 0.424 seconds
        Minimum number of seconds to run all queries: 0.354 seconds
        Maximum number of seconds to run all queries: 0.605 seconds
```

```
        Number of clients running queries: 20
        Average number of queries per client: 0
```

上述命令执行时将模拟 20 个用户的并发连接，模拟创建 5 列(2 列字符型，3 列数值型)的表，每个用户各执行指定的语句 50 次，执行的语句由 mysqlslap 自动生成。

几个生疏参数的简要说明：

--create-schema：指定运行测试的 schema，建议单独指定一个，不要与现有业务库混淆。

-x：--number-char-cols 的简写形式，与-a 配合使用，用来指定测试语句中含有的 varchar 类型列的数量。

-y：--number-int-cols 的简写形式，与-a 配合使用，用来指定测试语句中含有的 int 类型列的数量

-a：--auto-generate-sql 的简写形式，当没有通过命令行选项或附加文件指定要执行的操作时，就自动生成 SQL 语句。与此相关的还有其它一些参数，可用来设定执行语句的查询数量，写入条件，执行次数等等，详见官方文档 mysqlslap 命令小节。

-c：--concurrency 的简写形式，指定模拟的并发用户数。

-i：--iterations 的简写形式，指定执行测试语句的次数。

而后再尝试通过 mysql 代理连接 mysql 服务器，执行命令如下：

```
    [root@mysqldb1 ~]# mysqlslap -utest -ptest -h 172.16.1.110 -P 4040 --create-schema=testdb -x 2 -y 3 -a -c 20
-i 50
    Benchmark
        Average number of seconds to run all queries: 0.428 seconds
        Minimum number of seconds to run all queries: 0.344 seconds
        Maximum number of seconds to run all queries: 1.385 seconds
        Number of clients running queries: 20
        Average number of queries per client: 0
```

首先必须要强调一条，因为 mysqlslap 测试的环节较有为限，因此上述测试结果并不严谨，结果仅供参考。从有限的结果可以看出，应用代理后，一般的处理操作会受到些许影响，从前后数值对比来看，影响还是比较轻微。

不过考虑到官方文档开篇的那段话：MySQL Proxy is currently an Alpha release and should not be used within production environments. 因此三思建议在正式应用 Mysql Proxy 前还是需要做更严谨和完善的测试，以确保整个系统性能的高效和稳定。

附：源码方式编译安装 MySQL Proxy

MySQL Proxy 是一个通过 MySQL 网络协议，提供 MySQL 服务器与客户端之间连接的应用工具，所有 MySQL 兼容的客户端(包括 mysql 命令行，调用 mysql 命令行的类库，以及支持 MySQL 网络协议的应用)均无需任何修改即可直接连接 Mysql Proxy。

本文记录 ENTERPRISE LINUX5u3 版本下源码编译安装 MySQL Proxy 0.8 的步骤。

1、安装依赖包

源码编译方式安装 MySQL Proxy 前，下列依赖包必须首先安装：

- libevent 1.x or higher (1.3b or later is preferred)
- lua 5.1.x or higher
- glib2 2.16.0 or higher (官方文档说是 2.6.0 或更高，实际安装时，mysql-proxy 0.8 版本时提示不能低于 2.16.0 的)
- pkg-config
- libtool 1.5 or higher
- MySQL 5.0.x or higher developer files

A>. 安装 libevent

RHEL5.3 版本自带的 libevent 版本较低，为 1.1 版，不符合需求，直接下载更高版本安装，操作步骤如下：

```
# wget http://monkey.org/~provos/libevent-1.4.14b-stable.tar.gz
# tar xvfz libevent-1.4.14b-stable.tar.gz
# cd libevent-1.4.14b-stable
# ./configure --prefix=/usr/local/libevent-1.4
# make && make install
```

B>. 安装 lua 脚本语言包

Lua 是一个极轻量级的脚本语言，MySQL Proxy 通过该语言进行功能扩充。

首先到其官网下载最新版本 5.1.4，而后解压安装，依次执行命令如下：

```
# wget http://www.lua.org/ftp/lua-5.1.4.tar.gz
# tar xvfz lua-5.1.4.tar.gz
# cd lua-5.1.4
# vi Makefile
```

修改 MakeFile 文件，将：

INSTALL_TOP= /usr/local

修改为：

INSTALL_TOP= /usr/local/lua

这样做的目的，是为了将 lua 相关的文件放在同一目录内，便于查找和应用。

Lua 的编译和安装详见目录内的 INSTALL 文件，三思这里是在 linux 下安装，直接执行下列命令即能完成编译和安装：

```
# make linux install
```

C>. 安装 glib2

官方文档说是需要 2.6.0 或更高版本，实际在安装 mysql-proxy 0.8 时，提示 glib2 的版本不能低于 2.16.0，这里为了后面安装进程顺利执行，首先升级系统的 glib2，步骤如下。

```
# wget http://ftp.gnome.org/pub/gnome/sources/glib/2.22/glib-2.22.5.tar.gz
# tar xvfz glib-2.22.5.tar.gz
# cd glib-2.22.5
# ./configure --prefix=/usr/local/glib-2.2
# make && make install
```

D>. 安装 MySQL 开发包

主要需要用到 mysql_config 等应用，mysql 的开发包可以直接到其官网下载：http://dev.mysql.com/downloads/mysql/5.1.html

Rpm 包的安装比较简单，执行命令如下：

```
# rpm -ivh MySQL-devel-community-5.1.51-1.rhel5.x86_64.rpm
```

2、安装 MySQL Proxy

接下来，终于轮到正主上场，三思这里安装的是 MySQL Proxy 0.8.1 版本，也可以直接到其官网下载：http://dev.mysql.com/downloads/mysql-proxy/

```
# tar xvfz mysql-proxy-0.8.1.tar.gz
# cd mysql-proxy-0.8.1
# ./configure LDFLAGS="-lm -ldl" LUA_CFLAGS="/usr/local/lua/bin/lua -I/usr/local/lua/include" LUA_LIBS="/usr/local/lua/lib -llua" --prefix=/usr/local/mysql-proxy --with-lua
# make
# make install
```

执行 mysql-proxy 命令验证：

```
[root@rhel5u3 mysql-proxy]# /usr/local/mysql-proxy/bin/mysql-proxy -V
mysql-proxy 0.8.1
  chassis: mysql-proxy 0.8.1
  glib2: 2.22.5
  libevent: 1.4.14b-stable
  LUA: Lua 5.1.4
    package.path: /usr/local/mysql-proxy/lib/mysql-proxy/lua/?.lua
    package.cpath: /usr/local/mysql-proxy/lib/mysql-proxy/lua/?.so
-- modules
  admin: 0.8.1
  proxy: 0.8.1
```

编译过程看起来简单，仅只是几条命令，但这个安装颇不顺利，折腾了我整整一天时间，也许是三思运气不好，当然更多应该还是水平有限，实际执行编译的过程中遇到了很多错误，错误信息及解决方案见下，希望对同样遇到该问题的同学有所帮助：

操作系统版本如下：

```
[root@rhel5u3 ~]# cat /etc/issue
Enterprise Linux Enterprise Linux Server release 5.3 (Carthage)
Kernel \r on an \m
```

错误 1：

```
checking for LUA... no
```

... checked for Lua via pkg-config: No package 'lua' found. retrying with lua5.1

看错误信息是说找不到 lua 包。

解决方案：

A>. 参数 1.2 步骤中所示，编译安装 lua；

B>. 设置环境变量：

```
export LUA_CFLAGS="-I/usr/local/lua/include"
export LUA_LIBS="-L/usr/local/lua/lib -llua -ldl"
```

注意变量中的路径应为 lua 实际安装路径，而后重新执行 configure 配置编译程序。

如果执行上述操作故障依旧，可以尝试将 lua 安装路径下的 etc/lua.pc 文件复制到 glib2 安装路径中的 lib/pkgconfig/文件夹内。

错误 2：

checking for GLIB... configure: error: Package requirements (glib-2.0 >= 2.16.0) were not met:

Requested 'glib-2.0 >= 2.16.0' but version of GLib is 2.12.3

说明默认识别到的 glib2 版本不正确。

解决方案：

A>. 参照 1.3 步骤所示，编译安装适当版的 glib；

B>. 设置环境变量：

```
export GLIB_CFLAGS="-I/usr/local/glib-2.2/include/glib-2.0"
export GLIB_LIBS="-L/usr/local/glib-2.2/lib/glib-2.0"
export GMODULE_CFLAGS="-I/usr/local/glib-2.2/include"
export GMODULE_LIBS="-L/usr/local/glib-2.2/lib"
export GTHREAD_CFLAGS="-I/usr/local/glib-2.2/include"
export GTHREAD_LIBS="-L/usr/local/glib-2.2/lib"
```

特别强调，GMODULE 和 GTHREAD 的相关变量必须设置，否则会遇到(gmodule-2.0 >= 2.16.0) were not met 或(gthread-2.0 >= 2.16.0) were not met 之类错误。

变量值中的路径应为 glib2 的实际安装路径，而后重新执行 configure 配置编译程序。

错误 3:

```
checking for event_init in -levent... no
configure: error: libevent is required
```

没有找到 libevent 依赖包。

解决方案：

A>. 参照 1.1 步骤中所示，编译安装 libevent。

B>. 设置环境变量：

```
export LDFLAGS="-L/usr/local/libevent-1.4/lib -lm"
```

而后重新执行 configure 配置编译程序

错误 4

/usr/local/lua/lib/liblua.a: could not read symbols: Bad value

看起来是 lua 调用出错，相关文档提示是提示：64bit 环境可能遇到该现象。

解决方案：

A>. 编辑 lua 的 src/MakeFile 文件，修改下列参数：

```
CFLAGS= -O2 -Wall $(MYCFLAGS)
```

修改为：

```
CFLAGS= -O2 -Wall -fPIC $(MYCFLAGS)
```

B>. 按照 1.2 步骤中所示，编译安装 lua。如果之前已经编译过，记得要执行 make clean 清除已编译配置。

错误 5

```
/usr/local/glib-2.22/include/glib-2.0/glib/gtypes.h:34:24: error: glibconfig.h: No such file or directory
```

解决方案：

```
# cp /usr/local/glib-2.2/lib/glib-2.0/include/glibconfig.h /usr/local/glib-2.2/include/glib-2.0/
```

而后重新执行编译。

参考文档：

https://answers.launchpad.net/mysql-proxy/+question/58950

http://www.6xuan.com/read.php?414

附：三思笔记系列文章快速链接：

[三思笔记]我想对初学 ORACLE 的朋友说
http://www.5ienet.com/note/html/stdstep/how-to-learn-in-oracle.shtml

[三思笔记]Linux 环境下源码编译安装 MySQL5.1
http://www.5ienet.com/note/html/stmysql51/index.shtml

[三思笔记]ORACLE9i 数据库优化案例
http://www.5ienet.com/note/html/tuning/index.shtml

[三思笔记]MySQL 数据库权限体系入门
http://www.5ienet.com/note/html/mysql_priv/index.shtml

[三思笔记]ORACLE RAC 数据库配置 Dataguard 环境
http://www.5ienet.com/note/html/dgrac/index.shtml

[三思笔记]RMAN 管理 RAC 数据库的备份与恢复
http://www.5ienet.com/note/html/rman_rac/index.shtml

[三思笔记]Linux 平台 Oracle10gR2 RAC 数据库安装补丁集
http://www.5ienet.com/note/html/racupgrade/index.shtml

[三思笔记]ORACLE10gR2 RAC 增加及删除节点
http://www.5ienet.com/note/html/sracnode/index.shtml

[三思笔记]学用 ORACLE ASH&AWR 特性
http://www.5ienet.com/note/html/ash_awr/index.shtml

[三思笔记]Linux5 版本安装 Oracle11gR2
http://www.5ienet.com/note/html/st11g/index.shtml

[三思笔记]全面学习 Oracle Scheduler 特性
http://www.5ienet.com/note/html/scheduler/index.shtml

[三思笔记]全面学习 Oracle Flashback 特性
http://www.itpub.net/1019082.html

[三思笔记]手把手教你用 VMware 在 linux 下安装 oracle10g RAC
http://www.5ienet.com/note/html/srac/index.shtml

[三思笔记]全面学习分区表及分区索引
http://www.itpub.net/996554.html

[三思笔记]单条 SQL 语句实现复杂逻辑几例
http://www.itpub.net/970868.html

[三思笔记]一步一步学 Dataguard
http://www.itpub.net/958526.html

[三思笔记]使用可传输表空间的特性复制数据
http://www.itpub.net/926949.html

[三思笔记]日期时间及数字的格式化参数大全
http://www.itpub.net/913307.html

[三思笔记]RMAN 高级应用之 Duplicate 复制数据库
http://www.itpub.net/906598.html

[三思笔记]RHEL AS4 下升级 oracle10g 到 10.2.0.3
http://www.itpub.net/896394.html

[三思笔记]RHEL AS4 下安装 32 位 oracle10g
http://www.itpub.net/884137.html

[三思笔记]Statspack 初步学和用
http://www.itpub.net/857807.html

[三思笔记]oracle 著名及非著名函数介绍
http://www.itpub.net/843333.html

[三思笔记]一步一步学 rman
http://www.itpub.net/810100.html

[三思笔记]学习动态性能表
http://www.itpub.net/782892.html