



链滴

# 谈谈 PHP-FPM 模式下的 MySQL 持久连接

作者: [happyhacker](#)

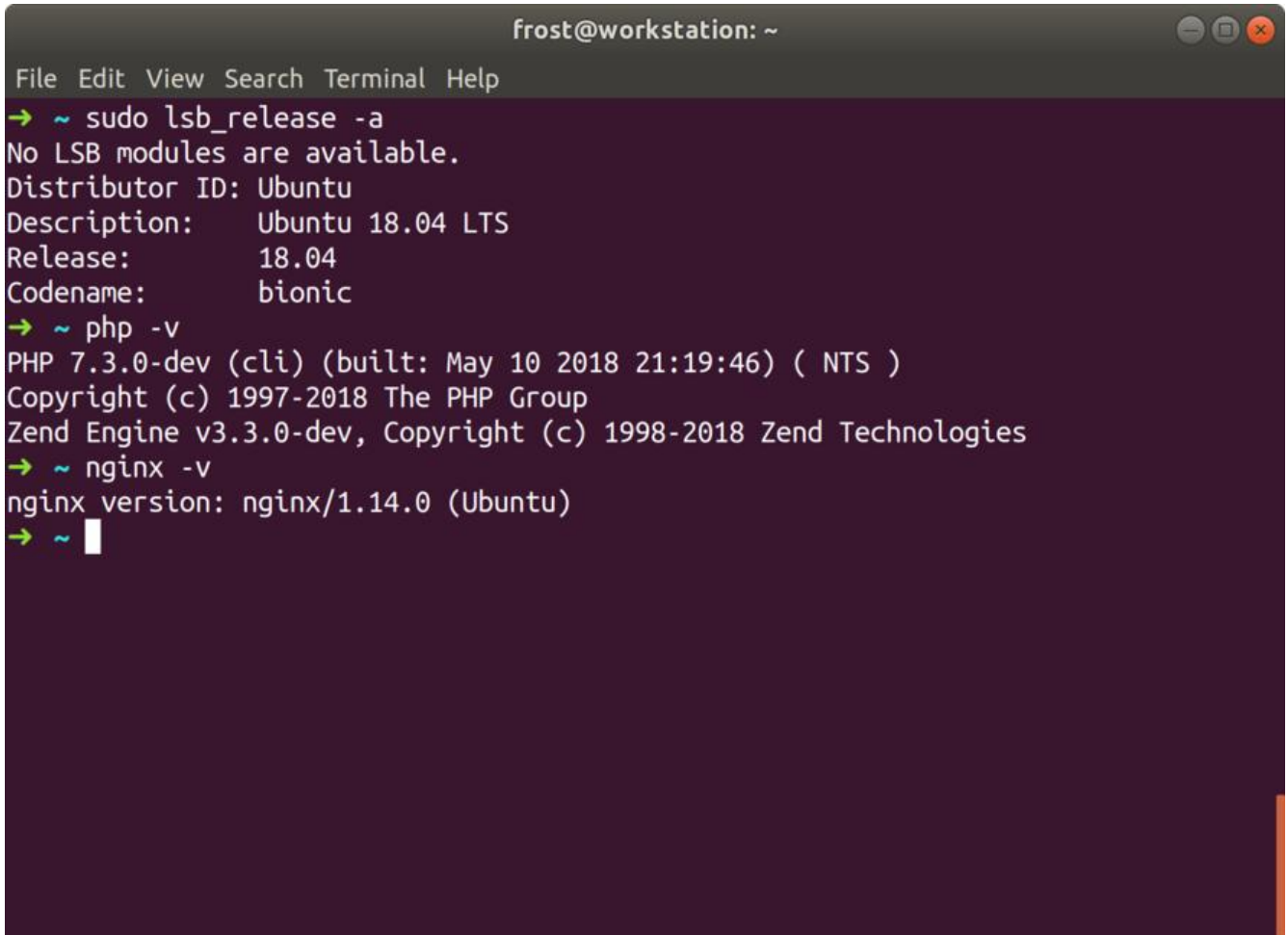
原文链接: <https://ld246.com/article/1526490593632>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

闲来无事研究一下PHP的MySQL持久连接问题。在mysql扩展的年代，应该用的是mysql\_pconnect可是那时候我还没有开始接触PHP，所以我们直接上PDO。

首先说一下本次测试用的环境。

A terminal window titled 'frost@workstation: ~' with a menu bar containing 'File Edit View Search Terminal Help'. The terminal shows the following commands and outputs:

```
→ ~ sudo lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04 LTS
Release:        18.04
Codename:       bionic
→ ~ php -v
PHP 7.3.0-dev (cli) (built: May 10 2018 21:19:46) ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.3.0-dev, Copyright (c) 1998-2018 Zend Technologies
→ ~ nginx -v
nginx version: nginx/1.14.0 (Ubuntu)
→ ~ █
```

关键还要看一下PHP的配置。

```

; Note: This value is mandatory.
pm = static

; The number of child processes to be created when pm is set to 'static' and the
; maximum number of child processes when pm is set to 'dynamic' or 'ondemand'.
; This value sets the limit on the number of simultaneous requests that will be
; served. Equivalent to the ApacheMaxClients directive with mpm_prefork.
; Equivalent to the PHP_FCGI_CHILDREN environment variable in the original PHP
; CGI. The below defaults are based on a server without much resources. Don't
; forget to tweak pm.* to fit your needs.
; Note: Used when pm is set to 'static', 'dynamic' or 'ondemand'
; Note: This value is mandatory.
pm.max_children = 1

; The number of child processes created on startup.
; Note: Used only when pm is set to 'dynamic'
; Default Value: min_spare_servers + (max_spare_servers - min_spare_servers) / 2
pm.start_servers = 1

; The desired minimum number of idle server processes.
; Note: Used only when pm is set to 'dynamic'
; Note: Mandatory when pm is set to 'dynamic'
pm.min_spare_servers = 1

; The desired maximum number of idle server processes.
; Note: Used only when pm is set to 'dynamic'
; Note: Mandatory when pm is set to 'dynamic'
pm.max_spare_servers = 1

```

注意其中的最重要的参数`pm.max_children=1`，这决定了只能有一个FPM的worker进程来处理所请求。这样把问题简化更容易发现特征。

我们知道，PHP的FPM有一个master进程和若干个worker进程，而worker进程并不是像最早的fastcgi一样每次处理完一个请求之后就销毁，下次再来请求需要重新启动。也就是说一个worker进程是以处理多个请求的。这给“持久连接”提供了理论基础。那么到底它能不能支持持久连接呢？如果支持持久连接，它的特征又是什么呢？下面直接上代码。

```
index.php (/var/www/fpm) - VIM
File Edit View Search Terminal Help
1 <?php
2 $dsn = 'mysql:dbname=fpmtest;host=127.0.0.1';
3 $user = 'fpm';
4 $password = 'fpm';
5
6 try {
7     $dbh = new \PDO($dsn, $user, $password, [
8         PDO::ATTR_PERSISTENT => true,
9     ]);
10    $stmt = $dbh->exec('insert into test set name = "happyhacker"');
11    $lastInsertId = $dbh->lastInsertId();
12
13    echo $lastInsertId, "\n";
14
15 } catch (PDOException $e) {
16     echo 'Connection failed: ' . $e->getMessage();
17 }
```

我在之前的文章里也提到过MySQL的`general_log`，按这里的描述配置一下就可以很清晰的看到哪个接在请求服务器了。

反复执行`curl http://fpm.org/index.php`（我给fpm.org配置了hosts），就可以看到不断变化的lastInsertId了。但这不是重点，要看两个现象。

用root账户登录mysql，执行`show processlist`可以看到类似如下的输出：

```
File Edit View Search Terminal Help
+-----+-----+-----+-----+-----+-----+-----+
| 12 | debian-sys-maint | localhost:52180 | NULL | Query | 0 | starting | show processlist |
| 14 | fpm              | localhost:53722 | fpmtest | Sleep | 39 |          | NULL              |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> show processlist;
+-----+-----+-----+-----+-----+-----+-----+
| Id | User          | Host          | db      | Command | Time | State   | Info              |
+-----+-----+-----+-----+-----+-----+-----+
| 12 | debian-sys-maint | localhost:52180 | NULL    | Query   | 0    | starting | show processlist |
| 14 | fpm            | localhost:53722 | fpmtest | Sleep   | 41   |          | NULL              |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> show processlist;
+-----+-----+-----+-----+-----+-----+-----+
| Id | User          | Host          | db      | Command | Time | State   | Info              |
+-----+-----+-----+-----+-----+-----+-----+
| 12 | debian-sys-maint | localhost:52180 | NULL    | Query   | 0    | starting | show processlist |
| 14 | fpm            | localhost:53722 | fpmtest | Sleep   | 1    |          | NULL              |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> 
```

可以看到，有两个客户端和服务端保持了连接。是谁呢？第一个是Ubuntu（其实是Debian）维护的MySQL包自带的默认管理员用户，其实表示的就是当前的这个MySQL cli连接。另外一个当然就是刚调用curl通过PHP的pdo创建的了。

```
frost@workstation: /var/www/fpm
File Edit View Search Terminal Help
→ fpm curl http://fpm.org/index.php
86
→ fpm curl http://fpm.org/index.php
87
→ fpm curl http://fpm.org/index.php
88
→ fpm curl http://fpm.org/index.php
89
→ fpm curl http://fpm.org/index.php
90
→ fpm curl http://fpm.org/index.php
91
→ fpm curl http://fpm.org/index.php
92
→ fpm curl http://fpm.org/index.php
93
→ fpm curl http://fpm.org/index.php
94
→ fpm curl http://fpm.org/index.php
95
→ fpm curl http://fpm.org/index.php
96
→ fpm 
```



现在再来看general\_log,

```
File Edit View Search Terminal Help

2018-05-16T16:53:12.644060Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:14.107533Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:14.744255Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:15.477043Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:17.939891Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:18.427685Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:18.823089Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:19.206052Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:19.573658Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:19.934893Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:20.310902Z      14 Query      insert into test set name = "happyhacker"
```

可以看到多次请求服务器端都是同一个线程ID (14) . 参考[这里](#)

那么怎么验证它是由当前这个FPM的worker维持的呢? 很简单, 重启一下FPM再看它有没有变化就知道了。

`sudo systemctl restart php-fpm.service`

```
→ fpm ps -ef | grep php-fpm | grep pool
www-data 16917 16903  0 00:15 ?          00:00:00 php-fpm: pool www
→ fpm sudo systemctl restart php-fpm.service
→ fpm ps -ef | grep php-fpm | grep pool
www-data 23223 23206  0 00:58 ?          00:00:00 php-fpm: pool www
```

可以看到一个新的worker已经在工作了。现在重新访问刚才的地址

先再去执行一下`show processlist;`

```
File Edit View Search Terminal Help
+-----+-----+-----+-----+-----+-----+-----+-----+
| 12 | debian-sys-maint | localhost:52180 | NULL | Query | 0 | starting | show processlist |
| 14 | fpm | localhost:53722 | fpmtest | Sleep | 41 | | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> show processlist;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Id | User | Host | db | Command | Time | State | Info |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 12 | debian-sys-maint | localhost:52180 | NULL | Query | 0 | starting | show processlist |
| 14 | fpm | localhost:53722 | fpmtest | Sleep | 1 | | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> show processlist;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Id | User | Host | db | Command | Time | State | Info |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 12 | debian-sys-maint | localhost:52180 | NULL | Query | 0 | starting | show processlist |
| 15 | fpm | localhost:58000 | fpmtest | Sleep | 72 | | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> 
```

可以看到14已经不在。因为维持14这个连接的fpm worker已经挂了，当然对应的服务器的线程也经销毁了。但15出现了。那么这几次请求用的是不是15呢？

当然是。

```
File Edit View Search Terminal Help

2018-05-16T16:53:12.644060Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:14.107533Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:14.744255Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:15.477043Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:17.939891Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:18.427685Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:18.823089Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:19.206052Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:19.573658Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:19.934893Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:53:20.310902Z      14 Query      insert into test set name = "happyhacker"
2018-05-16T16:59:24.443606Z      15 Connect    fpm@localhost on fpmtest using TCP/IP
2018-05-16T16:59:24.443785Z      15 Query      insert into test set name = "happyhacker"
2018-05-16T16:59:25.373642Z      15 Query      insert into test set name = "happyhacker"
2018-05-16T16:59:25.870505Z      15 Query      insert into test set name = "happyhacker"
2018-05-16T16:59:26.273708Z      15 Query      insert into test set name = "happyhacker"
2018-05-16T16:59:26.655643Z      15 Query      insert into test set name = "happyhacker"
2018-05-16T16:59:27.021722Z      15 Query      insert into test set name = "happyhacker"
2018-05-16T16:59:27.419617Z      15 Query      insert into test set name = "happyhacker"
2018-05-16T16:59:27.890997Z      15 Query      insert into test set name = "happyhacker"

```

所以结论很明显了，在FPM模式下是可以使用mysql持久化连接的。

所以理论上也可以实现MySQL连接池。有时间可以研究一下。想了想是没有办法实现连接池的，因为一个worker只能维持一个长连接，无法和别的worker共享，只能通过配置`pm.max_children`来让FPM持的长连接没有那么多不要超过MySQL的最大连接数。

不过这是一个危险操作，因为你也看到了，我在写这篇文章的过程中在没有手动重启FPM进程之前个长连接是一直保持的，而如果这个fpm进程是空闲的，那么这个连接就是被浪费的。这有可能导致量无用的连接占用MySQL的连接数，而连接数是有上限的，超过之后就无法再建立新的连接，导致续的连接失败。所以必须设置长连接数的上限，同时保证worker空闲一段时间后退后，（使用`pm.max_spare_servers`实现）或者再处理若干次请求之后重新启动（通过`pm.max_requests`实现），以保MySQL的正常连接数。