[mysql\_ping()以及MYSQL\_OPT\_RECONNECT](http://blog.chinaunix.net/uid-22957904-id-3594136.html) 2013-04-16 23:15:51

分类： LINUX

来源：http://www.felix021.com/blog/read.php?2102  
昨天@Zind同学找到我之前的[一篇blog](http://www.felix021.com/blog/read.php?2061)(已经修改)，里面提到了mysql\_ping和MYSQL\_OPT\_RECONNECT的一些事情。  
  
之 所以写那篇blog，是因为去年写的一些代码遇到了“2006:MySQL server has gone away”错误。这个问题是因为wait\_timeout这个参数的默认值是28800，也就是说，如果一个连接连续8个小时没有任何请求，那么 Server端就会把它断开。在测试环境中一个晚上没有请求很正常……于是第二天早上来的时候就发现这个错误了。  
  
其实我有考虑这个问题 的，真的……因为我知道php里面有个函数叫做mysql\_ping()，PHP手册上说：“mysql\_ping() 检查到服务器的连接是否正常。如果断开，则自动尝试连接。本函数可用于空闲很久的脚本来检查服务器是否关闭了连接，如果有必要则重新连接上。”  
  
回想起来，以前真是很傻很天真。根据MySQL官方C API里[mysql\_ping()的文档](http://dev.mysql.com/doc/refman/5.1/en/mysql-ping.html)："Checks whether the connection to the server is working. If the connection has gone down **and auto-reconnect is enabled** an attempt to reconnect is made. ... Auto-reconnect is disabled by default. To enable it, call mysql\_options() with the MYSQL\_OPT\_RECONNECT option"，也就是说，它实际上还依赖于MYSQL\_OPT\_RECONNECT这个配置，而这个配置默认（自5.0.3开始）是关闭的！  
  
虽然想起来很愤怒很蛋疼，不过看到 libmysql/client.c: mysql\_init() 里的注释就淡定了：

引用

By default we don't reconnect because it could silently corrupt data (after reconnection you potentially lose table locks, user variables, session variables (transactions but they are specifically dealt with in mysql\_reconnect()).  This is a change: < 5.0.3 mysql->reconnect was set to 1 by default.

好吧，既然有问题，那就正视它。解决办法是调用 mysql\_options ，将MYSQL\_OPT\_RECONNECT设置为1:

char value = 1;  
mysql\_options(mysql, MYSQL\_OPT\_RECONNECT, &value);

但 是!! 在mysql 5.0.19 之前，mysql->reconnect = 0 这一句是放在 mysql\_real\_connect() 里面的！也就是说，如果你不能像处理其他选项一样，而是必须在mysql\_real\_connect()之前设置 MYSQL\_OPT\_RECONNECT，坑爹啊！  
  
好吧好吧，总之，关于坑的问题暂告一段落，结论就是，不管是哪个版本，如果你想要启用自动重连，最好都是在mysql\_real\_connect()之后，反正不会错。  
  
然后这篇的重点来了（前面似乎太罗嗦了点）：MYSQL\_OPT\_RECONNECT的文档里头说了，这个选项是用来启用/禁用（当发现连接断开时的）自动重连，那么，MYSQL什么时候会发现链接断开呢？  
  
这个问题可能太大了，不过不妨先去追一下，mysql\_ping()做了啥。  
  
下载源码 <http://cdn.mysql.com/Downloads/MySQL-5.1/mysql-5.1.67.tar.gz> ，解压以后ctags -R，再vim -t mysql\_ping ，马上就定位到了，似乎太简单了点：

int STDCALL  
mysql\_ping(MYSQL \*mysql)  
{  
  int res;   
  DBUG\_ENTER("mysql\_ping");  
  res= simple\_command(mysql,COM\_PING,0,0,0);        //试着向服务器发送一个ping包  
  if (res == CR\_SERVER\_LOST && mysql->reconnect)    //如果server挂了，而mysql->reconnect为true  
    res= simple\_command(mysql,COM\_PING,0,0,0);      //再ping一次？？  
  DBUG\_RETURN(res);  
}

好吧，看来关键在于这个simple\_command了。ctrl+]，原来是这样：

#define simple\_command(mysql, command, arg, length, skip\_check) \  
  (\*(mysql)->methods->advanced\_command)(mysql, command, 0, 0, arg, length, skip\_check, NULL)

好吧，先去追一下MYSQL，里头有个 const struct st\_mysql\_methods \*methods ，再追一下 st\_mysql\_methods ....

typedef struct st\_mysql\_methods  
{  
  my\_bool (\*read\_query\_result)(MYSQL \*mysql);  
  my\_bool (\*advanced\_command)(MYSQL \*mysql, enum enum\_server\_command command,  
                  const unsigned char \*header, unsigned long header\_length,  
                  const unsigned char \*arg, unsigned long arg\_length,  
                  my\_bool skip\_check, MYSQL\_STMT \*stmt);  
  ......

坑爹啊！又是这种鸟代码！蛋疼的C语言！struct只有属性没有方法！没办法，只能暴力了：

引用

find -name '\*.c' -exec /bin/grep '{}' -Hne 'mysql->methods \*=' ';'  
./libmysql\_r/client.c:1907:  mysql->methods= &client\_methods;  
./sql-common/client.c:1907:  mysql->methods= &client\_methods;  
./libmysql/client.c:1907:  mysql->methods= &client\_methods;  
./libmysqld/libmysqld.c:120:  mysql->methods= &embedded\_methods;  
./sql/client.c:1907:  mysql->methods= &client\_methods;

果断追到client\_methods:

static MYSQL\_METHODS client\_methods=  
{  
  cli\_read\_query\_result,                      /\* read\_query\_result \*/  
  cli\_advanced\_command,                        /\* advanced\_command \*/  
  ...

也 就是说simple\_command最后调用了cli\_advanced\_command这个函数。前面的 simple\_command(mysql,COM\_PING,0,0,0) 相当于是调用了 cli\_advanced\_command(mysql, COM\_PING, 0, 0, 0, 0, 0, NULL) 。  
  
这个函数做了啥呢。。。其实也不复杂：  
1. 设置默认返回值为1 （意外出错goto时被返回）  
2. 设置sigpipe的handler（以便忽略它）  
3. 如果 mysql->net.vio == 0 ，那么调用mysql\_reconnect重连，失败的话就返回1  
4. mysql没准备好，返回1  
5. 清除之前的信息（错误码、缓冲区、affected\_rows）等等  
6. 调用net\_write\_command将命令发送给server，如果失败：  
    6.1 检查错误信息，如果是因为发送包太大，goto end  
    6.2 调用end\_server(mysql)关闭连接  
    6.3 调用mysql\_reconnect尝试重连，如果失败goto end  
    6.4 再次调用net\_write\_command将命令发送给server，失败则goto end  
7. 设置result = 0（发送成功）  
8. 如果参数中要求检查server的返回，则读取一个packet进行检查（失败的话就result=1）  
9. (end标签)   
10. 恢复sigpipe  
11. 返回result  
  
可 以看到，这里两次调用了mysql\_reconnect，但都是有条件的：第一次是在mysql->net.vio == 0的情况下，第二次是net\_write\_command失败且不是因为包太大的情况。vio相关的代码看得一头雾水，实在找不出头绪，于是决定暴力一 点：直接修改这个函数，加入一堆fprintf(stderr, ...)（具体加在哪里就不说了，反正使劲塞就是了），然后写了一个C代码：

#include   
#include   
#include   
  
void do\_err(MYSQL \*mysql) {  
    if (mysql\_errno(mysql)) {  
        fprintf(stderr, "%d:%s\n", mysql\_errno(mysql), mysql\_error(mysql));  
        exit(mysql\_errno(mysql));  
    }  
}  
  
int main()  
{  
    MYSQL \* mysql = mysql\_init(NULL);  
    do\_err(mysql);  
  
    mysql\_real\_connect(mysql, "127.0.0.1", "root", "123456", "test", 3306, NULL, 0);  
    do\_err(mysql);  
  
    char value = 1;  
    mysql\_options(mysql, MYSQL\_OPT\_RECONNECT, &value);  
      
    char cmd[1024] = "SELECT \* FROM t";  
    while (1) {  
        mysql\_query(mysql, cmd);  
        do\_err(mysql);  
  
        MYSQL\_RES \*result = mysql\_store\_result(mysql);  
  
        MYSQL\_ROW  row;  
        while ((row = mysql\_fetch\_row(result)) != NULL) {  
            int i, num\_fields = mysql\_num\_fields(result);  
            for (i = 0; i < num\_fields; i++)   
                printf("%s\t", row[i] ? row[i] : "NULL");   
            //注意上一句是不是二进制安全的，因为row里头可能包含\0，也可能末尾没有\0  
            printf("\n");  
        }  
  
        mysql\_free\_result(result);  
        printf("press enter..."); getchar();  
    }  
    mysql\_close(mysql);  
    return 0;  
}

运行输出：

引用

inside mysql\_real\_query  
mysql->net.vio = 0x90e760  
mysql->status = 0  
net write\_command  
after send\_query  
---  
1  
2  
press enter...//按回车之前先重启一下mysql server，下面这几句按照函数调用层次进行手动缩进了……  
inside mysql\_real\_query  
    mysql->net.vio = 0x90e760 //进入cli\_advanced\_command  
    mysql->status = 0  
    net\_write\_command  
    end\_server //说明net\_write\_command失败了  
        inside mysql\_reconnect //它会调用mysql\_real\_query  
            inside mysql\_real\_query  
                mysql->net.vio = 0x919990 //于是又回到了cli\_advanced\_command  
                mysql->status = 0  
                net\_write\_command //这次成功了  
            after send\_query  //这句我是写在mysql\_real\_query里面的  
        reconnect succeded  
    after reconnect: mysql->status = 0  
after send\_query //所以又来一次。。

根据fprintf的输出，发现在正常情况下，mysql->net.vio这个指针并不等于0，所以第一个mysql\_reconnect不会被调用。而net\_write\_command也是正确执行，第二个reconnect也没被调用。  
  
而 在执行完一个query，然后重启mysql server再执行query (mysql\_query => mysql\_real\_query => mysql\_send\_query => cli\_advanced\_command)，就会发现，mysql->net.vio仍然不等于0，但是net\_write\_command失败 了，于是先调用了end\_server()（这里面会将mysql->net.vio设置为0，不过不影响后面的流程...），然后调用了第二个 reconnect，这个reconnect会调用mysql\_init()以及mysql\_real\_query()执行一些初始化的命令，于是又回到 cli\_advanced\_command，再一步一步回溯。。。  
  
综上可知，如果设置了MYSQL\_OPT\_RECONNECT()，那 么mysql\_query()是可以完成自动重连的。实际上，由于cli\_advanced\_command会在必要情况下调用 mysql\_reconnect（实际上这个函数也只在这里被调用），因此，所有用到了cli\_read\_query\_result的地方（或者 simple\_command），也都可以完成自动重连。  
  
完结。  
  
//混蛋，这篇纯粹是为了凑一月至少一篇这个目标啊！

--

转载请注明出自 [**http://www.felix021.com/blog/read.php?2102**](http://www.felix021.com/blog/read.php?2102) ，如是转载文则注明原出处，谢谢:)  
Google Reader 订阅 [**点击这里**](https://www.google.com/reader/view/#stream/feed%2Fhttp%3A%2F%2Fwww.felix021.com%2Fblog%2Ffeed.php)，RSS地址: [**http://www.felix021.com/blog/feed.php**](http://www.felix021.com/blog/feed.php) 。