



链滴

Communications link failure during rollback(). Transaction resolution unknown

作者: [ymxf1](#)

原文链接: <https://ld246.com/article/1635126604348>

来源网站: [链滴](#)

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



- 错误描述如图：

```
2021-10-20 15:53:09.775 ERROR 39597 --- [in-11000-exec-8] com.alibaba.druid.util.JdbcUtils : close connection error

java.sql.SQLNonTransientConnectionException: Communications link failure during rollback(). Transaction resolution unknown.
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQL_Error.java:110) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQL_Error.java:97) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQL_Error.java:89) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQL_Error.java:63) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.mysql.cj.jdbc.ConnectionImpl.rollback(ConnectionImpl.java:1866) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.mysql.cj.jdbc.ConnectionImpl.realClose(ConnectionImpl.java:1726) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.mysql.cj.jdbc.ConnectionImpl.close(ConnectionImpl.java:720) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.alibaba.druid.filter.FilterChainImpl.connection_close(FilterChainImpl.java:180) ~[druid-1.1.10.jar:1.1.10]
    at com.alibaba.druid.filter.stat.StatFilter.connection_close(StatFilter.java:261) ~[druid-1.1.10.jar:1.1.10]
    at com.alibaba.druid.filter.FilterChainImpl.connection_close(FilterChainImpl.java:181) ~[druid-1.1.10.jar:1.1.10]
    at com.alibaba.druid.filter.FilterAdapter.connection_close(FilterAdapter.java:776) ~[druid-1.1.10.jar:1.1.10]
    at com.alibaba.druid.filter.FilterChainImpl.connection_close(FilterChainImpl.java:181) ~[druid-1.1.10.jar:1.1.10]
    at com.alibaba.druid.filter.stat.StatFilter.connection_close(StatFilter.java:261) ~[druid-1.1.10.jar:1.1.10]
    at com.alibaba.druid.filter.FilterChainImpl.connection_close(FilterChainImpl.java:181) ~[druid-1.1.10.jar:1.1.10]
    at com.alibaba.druid.proxy.jdbc.ConnectionProxyImpl.close(ConnectionProxyImpl.java:115) ~[druid-1.1.10.jar:1.1.10]
    at com.alibaba.druid.util.JdbcUtils.close(ConnectionImpl.java:72) ~[druid-1.1.10.jar:1.1.10]
    at com.alibaba.druid.pool.DruidDataSource.disconnectConnection(DruidDataSource.java:1308) [druid-1.1.10.jar:1.1.10]
    at com.alibaba.druid.pool.DruidDataSource.handleFatalError(DruidDataSource.java:1500) [druid-1.1.10.jar:1.1.10]
    at com.alibaba.druid.pool.DruidDataSource.handleConnectionException(DruidDataSource.java:1540) [druid-1.1.10.jar:1.1.10]
    at com.alibaba.druid.pool.DruidPooledConnection.handleException(DruidPooledConnection.java:133) [druid-1.1.10.jar:1.1.10]
    at com.alibaba.druid.pool.DruidPooledConnection.setAutoCommit(DruidPooledConnection.java:715) [druid-1.1.10.jar:1.1.10]
    at org.springframework.jdbc.datasource.DataSourceTransactionManager.doBegin(DataSourceTransactionManager.java:281) [spring-jdbc-5.1.2.RELEASE.jar:5.1.2.RELEASE]
    at org.springframework.transaction.support.AbstractPlatformTransactionManager.getTransaction(AbstractPlatformTransactionManager.java:170) [spring-tx-5.1.2.RELEASE.jar:5.1.2.RELEASE]
    at org.springframework.transaction.interceptor.TransactionAspectSupport.createTransactionIfNecessary(TransactionAspectSupport.java:674) [spring-tx-5.1.2.RELEASE.jar:5.1.2.RELEASE]
    at org.springframework.transaction.interceptor.TransactionAspectSupport.invokeWithinTransaction(TransactionAspectSupport.java:291) [spring-tx-5.1.2.RELEASE.jar:5.1.2.RELEASE]
    at org.springframework.transaction.interceptor.TransactionInterceptor.invoke(TransactionInterceptor.java:99) [spring-tx-5.1.2.RELEASE.jar:5.1.2.RELEASE]
```

- 错误背景：

使用SpringBoot + Druid + mysql开发应用服务，使用Nginx做4层代理转发到Mysql服务端。服务启动后再swagger上第一次访问接口是正常的，但是过了几十秒后再访问就报错了，后面就再也发不通。一直提示无法建立连接

```
java.sql.SQLNonTransientConnectionException: Create breakpoint : Could not create connection to database server. Attempted reconnect 3 times. Giving up.
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQL_Error.java:110) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQL_Error.java:97) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQL_Error.java:89) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQL_Error.java:63) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQL_Error.java:73) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.mysql.cj.jdbc.ConnectionImpl.connectWithRetries(ConnectionImpl.java:985) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.mysql.cj.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:836) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.mysql.cj.jdbc.ConnectionImpl.<init>(ConnectionImpl.java:455) ~[mysql-connector-java-8.0.13.jar:8.0.13]
    at com.mysql.cj.jdbc.ConnectionImpl.getInstance(ConnectionImpl.java:240) ~[mysql-connector-java-8.0.13.jar:8.0.13]
```

Communications link failure. The last packet successfully received from the server was 177,731

milliseconds ago. The last packet sent successfully to the server was 177,735 milliseconds ago.

- 初步分析:

作为一名合格的程序员，必须有独立解决问题的能力（当然在我们部门，我也没有可以求救的人了）遇到不认识的异常就要会百度、Google~ 但是我左搜右搜上搜下搜，都找不到正确的解决方案。无，只能靠自己了。仔细看了下异常信息，是从com.alibaba.druid里抛出来的，难道是连接池的问题于是我把Druid连接池换成了HikariCP，结果是可以正常访问数据库了，赶紧提交代码，别耽误其他事的开发。

数据库不可访问的危机是解除了，但是日志里还是会打印一堆warn警告。如图：

我是一个略有强迫症的程序员，解决问题必须要知其然且知其所以然，还有warn警告，说明问题并没有从根源上解决。于是我又仔细分析了下警告提示（从服务器成功接收到的最后一个数据包是177,731毫秒前。最后一个成功发送到服务器的数据包是177,735毫秒之前）。可以考虑使用更短的maxLifetime值难道是配置有问题？我将 max-lifetime值调小，发现并没有任何作用。

“Communications link failure是通讯链路故障...接收数据包...发送数据包”，猜想问题可能出在通链路上。由于我的mysql是在内网服务器上的，没有直接将mysql服务器暴露到外网上，所以需要在nginx服务器做一个4层代理。我开始怀疑是不是nginx配置有问题呢，以下是我nginx的配置。

```
stream {
    upstream redis{
        server
    }
    upstream mysql{
        server
    }
    server{
        listen 33336 so_keepalive=1;
        proxy_connect_timeout 10s;
        proxy_read_timeout 10s;
        proxy_pass mysql;
    }
}
```

proxy_connect_timeout是nginx连接上游服务超时时间，如果配置多个上游服务节点，且nginx路到的第一个上游服务不可用时，连接超过设置的超时时间，nginx会自动将请求路由到其他上游服务点。

proxy_timeout 是如果指定时间内，客户端没有请求发送过来，nginx会断开与上游节点的链接，以省资源（查了以后才知道的）。

Mysql客户端与服务端之间是TCP长链接，在Mysql上是可以看到连接数的。于是我登录了mysql，行了连接数的查询。show status like 'Threads%';

```
mysql> show status like 'Threads%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Threads_cached | 0     |
| Threads_connected | 1    |
| Threads_created | 1     |
| Threads_running | 1     |
+-----+-----+
4 rows in set (0.00 sec)
```

当我服务停止时，连接数为1，当我启动服务并进行一次接口调用后（会访问数据库），连接数变成了。说明数据库根据我代码里配置的 `minimum-idle` 为5初始化了5个连接。

```
mysql> show status like 'Threads%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Threads_cached | 0     |
| Threads_connected | 6    |
| Threads_created | 6     |
| Threads_running | 1     |
+-----+-----+
4 rows in set (0.00 sec)
```

10s后再次查看连接数变成了1，说明数据库认为客户端连接断开了，释放了5个连接。

```
mysql> show status like 'Threads%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Threads_cached | 5     |
| Threads_connected | 1    |
| Threads_created | 6     |
| Threads_running | 1     |
+-----+-----+
4 rows in set (0.00 sec)
```

再次访问接口变成了6，说明hikariCP默认实现了自动重连，数据库根据配置又重新创建了5个连接。

```
mysql> show status like 'Threads%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Threads_cached | 0     |
| Threads_connected | 6    |
| Threads_created | 6     |
| Threads_running | 1     |
+-----+-----+
4 rows in set (0.01 sec)
```

停止服务又变成了1，说明数据库识别到客户端断开后，释放了5个连接。

```
mysql> show status like 'Threads%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Threads_cached | 5     |
| Threads_connected | 1    |
| Threads_created | 6     |
| Threads_running | 1     |
+-----+-----+
4 rows in set (0.00 sec)
```

我们可以通过 `show processlist;`命令看一下数据库进程信息的状态。

最开始连接数为0，没有任何连接信息，只有执行 `show processlist`命令的查询进程。

```
mysql> show processlist;
+----+-----+-----+-----+-----+-----+-----+-----+
| Id | User | Host      | db | Command | Time | State | Info |
+----+-----+-----+-----+-----+-----+-----+-----+
| 14 | root | localhost | NULL | Query   | 0    | starting | show processlist |
+----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

服务启动后，数据库不会立即创建连接进程，调用一次查询后出现了5个进程，并且可以看到用户及其B。Time为进程存在时间

```
mysql> show processlist;
+----+-----+-----+-----+-----+-----+-----+-----+
| Id | User      | Host          | db          | Command | Time | State | Info |
+----+-----+-----+-----+-----+-----+-----+-----+
| 14 | root      | localhost     | NULL        | Query   | 0    | starting | show processlist |
| 25 | mbankdemo | 192.168.1.220:48028 | mbankdemo | Sleep   | 1    |          | NULL |
| 26 | mbankdemo | 192.168.1.220:48030 | mbankdemo | Sleep   | 2    |          | NULL |
| 27 | mbankdemo | 192.168.1.220:48032 | mbankdemo | Sleep   | 1    |          | NULL |
| 28 | mbankdemo | 192.168.1.220:48038 | mbankdemo | Sleep   | 1    |          | NULL |
| 29 | mbankdemo | 192.168.1.220:48040 | mbankdemo | Sleep   | 0    |          | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+
```

```
mysql> show processlist;
+----+-----+-----+-----+-----+-----+-----+-----+
| Id | User      | Host          | db          | Command | Time | State | Info |
+----+-----+-----+-----+-----+-----+-----+-----+
| 14 | root      | localhost     | NULL        | Query   | 0    | starting | show processlist |
| 30 | mbankdemo | 192.168.1.220:48062 | mbankdemo | Sleep   | 6    |          | NULL |
| 31 | mbankdemo | 192.168.1.220:48064 | mbankdemo | Sleep   | 7    |          | NULL |
| 32 | mbankdemo | 192.168.1.220:48066 | mbankdemo | Sleep   | 6    |          | NULL |
| 33 | mbankdemo | 192.168.1.220:48068 | mbankdemo | Sleep   | 6    |          | NULL |
| 34 | mbankdemo | 192.168.1.220:48070 | mbankdemo | Sleep   | 5    |          | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+
```

可以看到Time增长到10s后，会自动消失。

```
mysql> show processlist;
+----+-----+-----+-----+-----+-----+-----+-----+
| Id | User      | Host          | db          | Command | Time | State | Info |
+----+-----+-----+-----+-----+-----+-----+-----+
| 14 | root      | localhost     | NULL        | Query   | 0    | starting | show processlist |
| 19 | mbankdemo | 192.168.1.220:47964 | mbankdemo | Sleep   | 10   |          | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+
```

总结

经过上面验证过程后，确定连接断开时间是与Nginx的proxy_timeout保持一致的，于是我把proxy_timeout配置取消后，观察连接是正常的，可持续到我们配置的interactive_timeout值。

(1)interactive_timeout:

参数含义：服务器关闭交互式连接前等待活动的秒数。交互式客户端定义为在mysql_real_connect()使用CLIENT_INTERACTIVE选项的客户端。

参数默认值: 28800秒 (8小时)

(2)wait_timeout:

参数含义: 服务器关闭非交互连接之前等待活动的秒数。

在线程启动时, 根据全局wait_timeout值或全局interactive_timeout值初始化会话wait_timeout值取决于客户端类型(由mysql_real_connect()的连接选项CLIENT_INTERACTIVE定义)。

参数默认值: 28800秒 (8小时)