零声学院：Darren QQ 326873713
课程：https://ke.qq.com/course/420945?tuin=137bb271

1 池化技术

池化技术能够减少资源对象的创建次数，提高程序的响应性能，特别是在高并发下这种提高更加明显。使用池化技术缓存的资源对象有如下共同特点：

1. 对象创建时间长；
2. 对象创建需要大量资源；
3. 对象创建后可被重复使用

像常见的线程池、内存池、连接池、对象池都具有以上的共同特点。

2 什么是数据库连接池

定义：数据库连接池（Connection pooling）是程序启动时建立足够的数据库连接，并将这些连接组成一个连接池，由程序动态地对池中的连接进行申请，使用，释放。

大白话：创建数据库连接是一个很耗时的操作，也容易对数据库造成安全隐患。所以，在程序初始化的时候，集中创建多个数据库连接，并把他们集中管理，供程序使用，可以保证较快的数据库读写速度，还更加安全可靠。

这里讲的数据库，不单只是指Mysql，也同样适用于Redis。

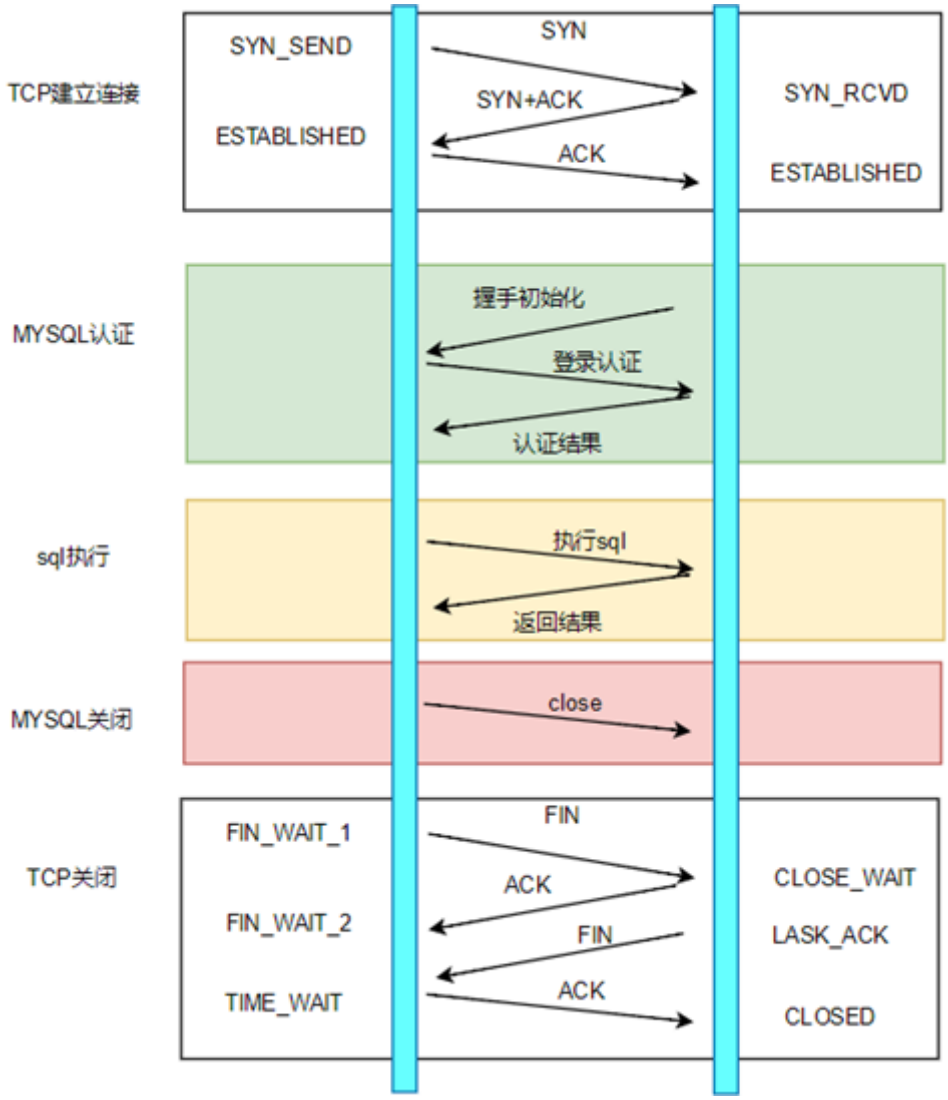
3 为什么使用数据库连接池

1. 资源复用
   由于数据库连接得到复用，避免了频繁的创建、释放连接引起的性能开销，在减少系统消耗的基础上，另一方面也增进了系统运行环境的平稳性（减少内存碎片以及数据库临时进程/线程的数量）。
2. 更快的系统响应速度
   数据库连接池在初始化过程中，往往已经创建了若干数据库连接置于池中备用。此时连接的初始化工作均已完成。对于业务请求处理而言，直接利用现有可用连接，避免了从数据库连接初始化和释放过程的开销，从而缩减了系统整体响应时间。
3. 统一的连接管理，避免数据库连接泄露
   在较为完备的数据库连接池实现中，可根据预先的连接占用超时设定，强制收回被占用连接。从而避免了常规数据库连接操作中可能出现的资源泄露。

3.1 不使用连接池

1. TCP建立连接的三次握手（客户端与MySQL服务器的连接基于TCP协议）
2. MySQL认证的三次握手
3. 真正的SQL执行
4. MySQL的关闭
5. TCP的四次握手关闭

可以看到，为了执行一条SQL，需要进行TCP三次握手，Mysql认证、Mysql关闭、TCP四次挥手等其他操作，执行SQL操作在所有的操作占比非常低。

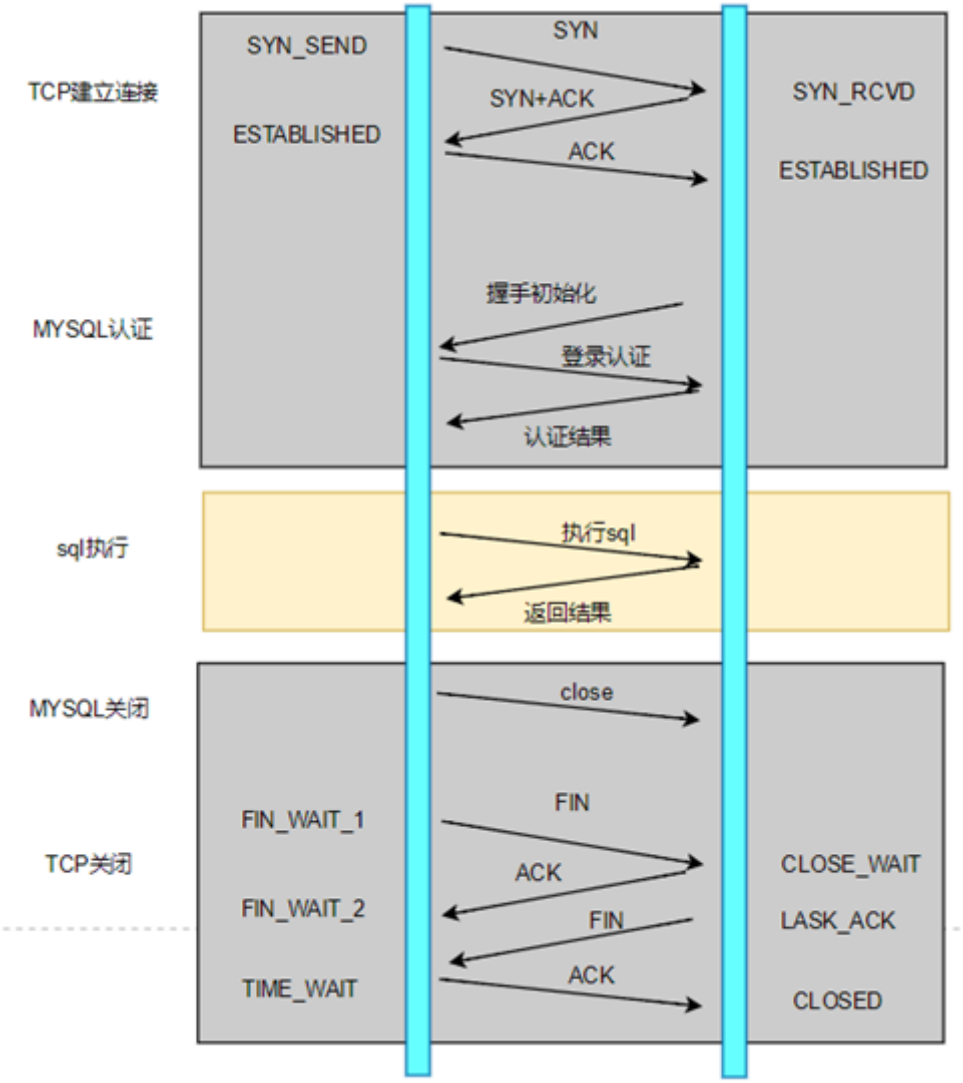


优点：实现简单
缺点：

- 网络IO较多
- 带宽利用率低
- QPS较低
- 应用频繁低创建连接和关闭连接，导致临时对象较多，带来更多的内存碎片
- 在关闭连接后，会出现大量TIME_WAIT 的TCP状态（在2个MSL之后关闭）

3.2 使用连接池

第一次访问的时候，需要建立连接。 但是之后的访问，均会复用之前创建的连接，直接执行SQL语句。



优点：

1. 降低了网络开销
2. 连接复用，有效减少连接数。
3. 提升性能，避免频繁的新建连接。新建连接的开销比较大
4. 没有TIME_WAIT状态的问题

缺点：

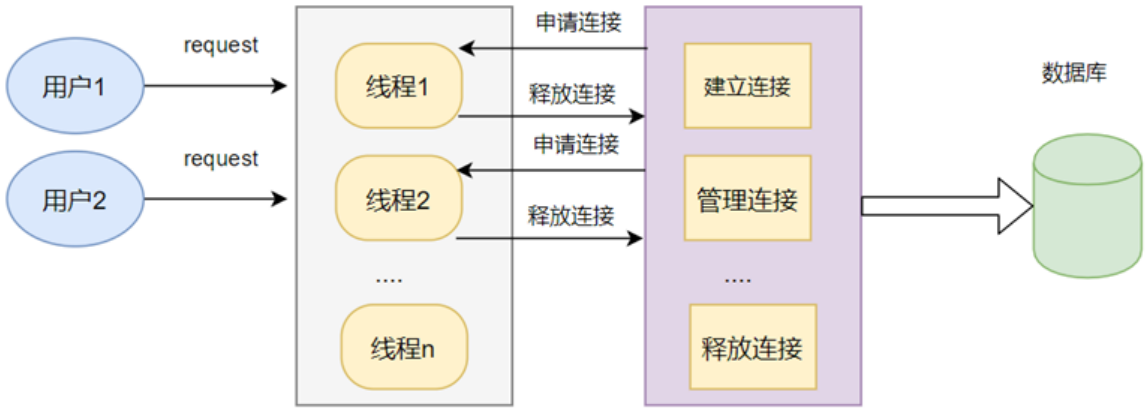
1. 设计较为复杂

3.3 长连接和连接池的区别

- 长连接是一些驱动、驱动框架、ORM工具的特性，由驱动来保持连接句柄的打开，以便后续的数据库操作可以重用连接，从而减少数据库的连接开销。
- 而连接池是应用服务器的组件，它可以通过参数来配置连接数、连接检测、连接的生命周期等。
- 连接池内的连接，其实就是长连接。

4 数据库连接池运行机制

从连接池获取或创建可用连接；
使用完毕之后，把连接返回给连接池；
在系统关闭前，断开所有连接并释放连接占用的系统资源；



5 连接池和线程池的关系

连接池和线程池的区别

- 线程池：主动调用任务。当任务队列不为空的时候从队列取任务取执行。
    - 比如去银行办理业务，窗口柜员是线程，多个窗口组成了线程池，柜员从排号队列叫号执行。
- 连接池：被动被任务使用。当某任务需要操作数据库时，只要从连接池中取出一个连接对象，当任务使用完该连接对象后，将该连接对象放回到连接池中。如果连接池中没有连接对象可以用，那么该任务就必须等待。
    - 比如去银行用笔填单，笔是连接对象，我们要用笔的时候去取，用完了还回去。

连接池和线程池设置数量的关系

- 一般线程池线程数量和连接池连接对象数量一致；
- 一般线程执行任务完毕的时候归还连接对象；

6 线程池设计要点

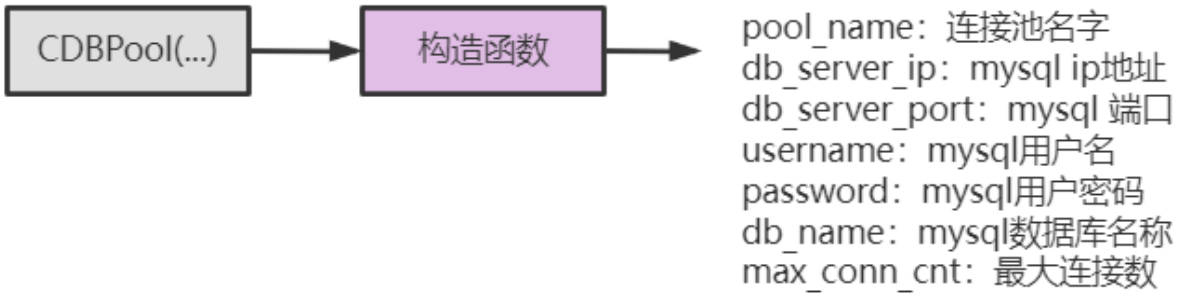
使用连接池需要预先建立数据库连接。

线程池设计思路：

1. 连接到数据库，涉及到数据库ip、端口、用户名、密码、数据库名字等；
    1. 连接的操作，每个连接对象都是独立的连接通道，它们是独立的
    2. 配置最小连接数和最大连接数
2. 需要一个队列管理他的连接，比如使用list；
3. 获取连接对象：
4. 归还连接对象；
5. 连接池的名字

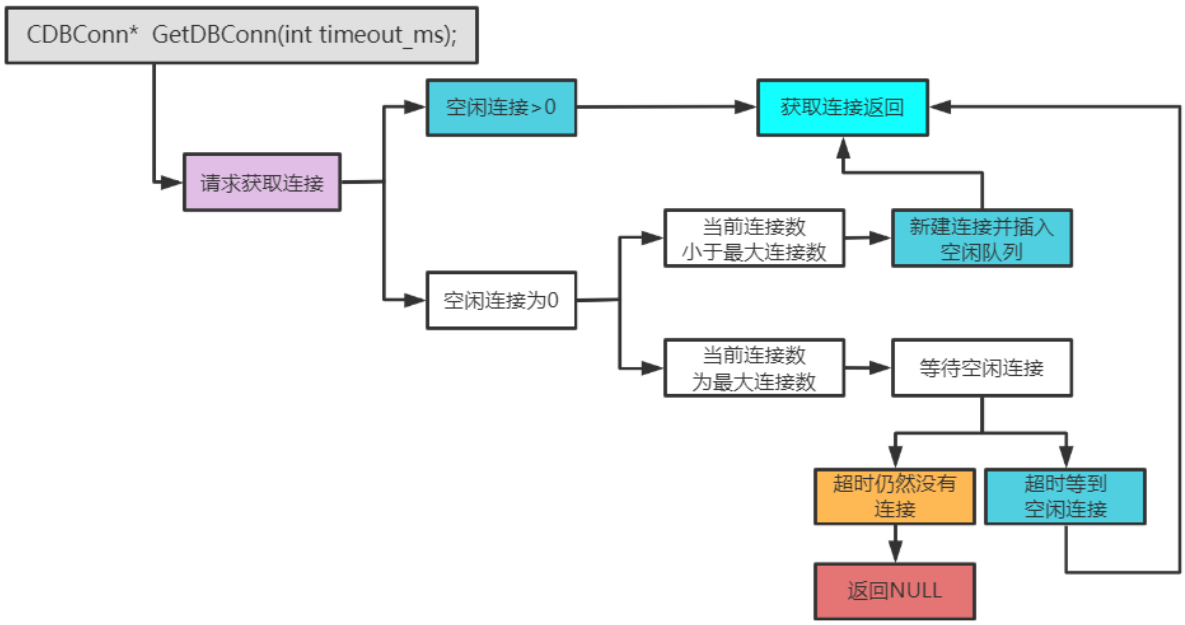
6.1 连接池设计逻辑

构造函数

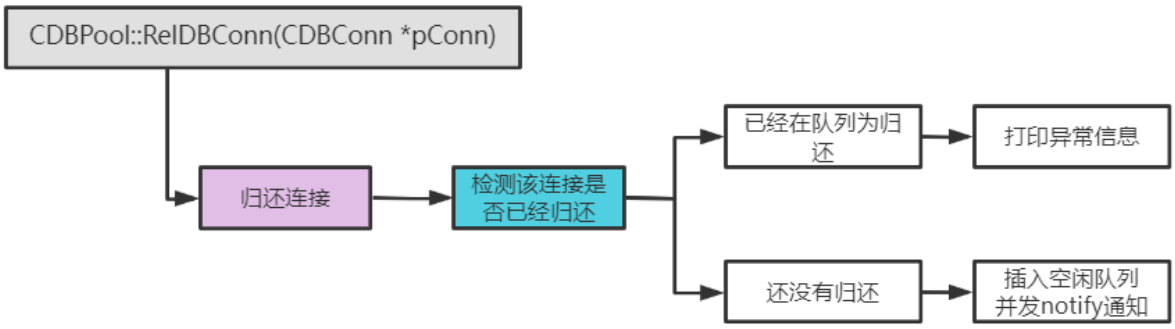


初始化

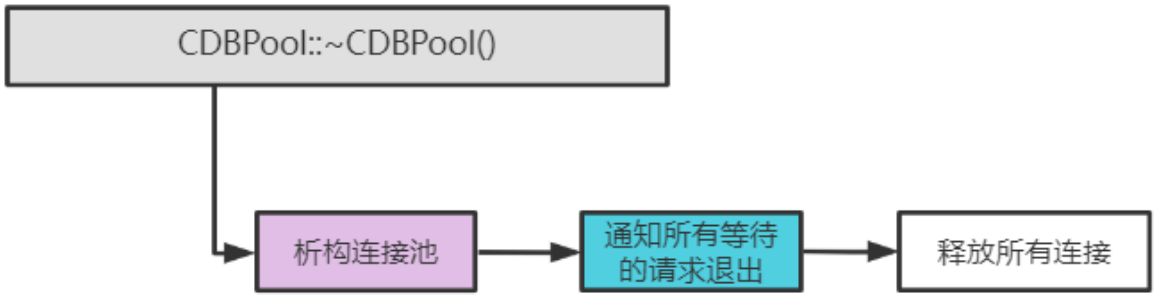
请求获取连接



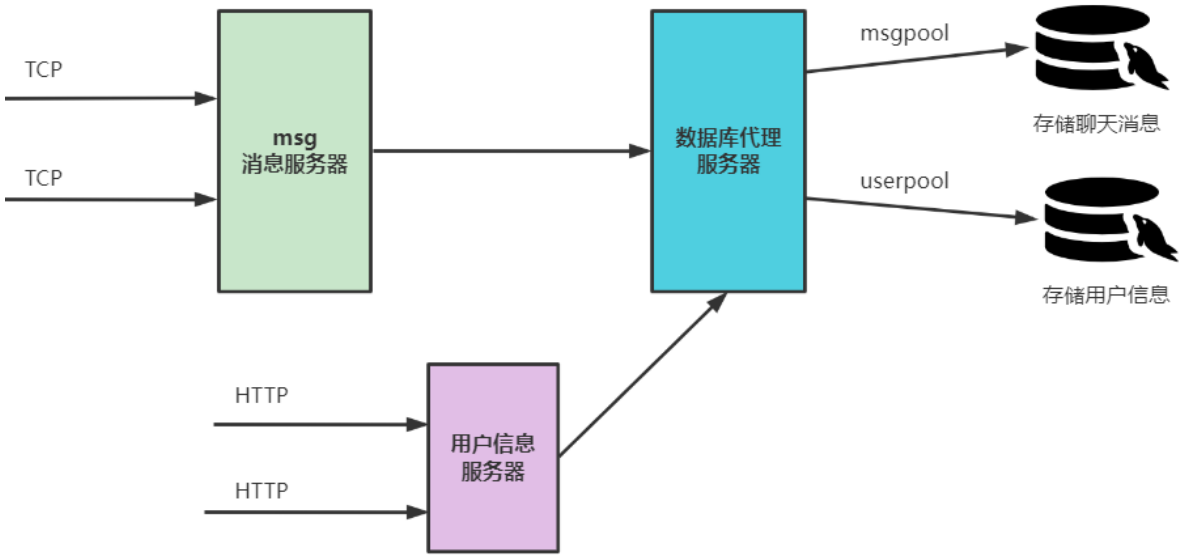
归还连接



析构线程池



连接池名



6.2 mysql连接重连机制

设置启用（当发现连接断开时的）自动重连

my_bool reconnect = true;

mysql_options(m_mysql, MYSQL_OPT_RECONNECT, &reconnect); // 配合mysql_ping实现自动重连

检测连接是否正常

int STDCALL mysql_ping(MYSQL *mysql);

描述：

检查与服务端的连接是否正常。连接断开时，如果自动重新连接功能未被禁用，则尝试重新连接服务器。该函数可被客户端用来检测闲置许久以后，与服务端的连接是否关闭，如有需要，则重新连接。

返回值：

连接正常，返回0；如有错误发生，则返回非0值。返回非0值并不意味着服务器本身关闭掉，也有可能是网络原因导致网络不通。

6.3 redis重连机制

1. 使用之前检测连接是否可用
2. 使用过程中出现连接异常则释放异常
3. 下一次使用该连接的时候如果发现连接不可用则重新初始化

redis的重连机制设计和mysql有区别。

7 连接池的具体实现

7.0 mysql常见命令

对数据库不熟悉的朋友参考：MySQL 教程 | 菜鸟教程 (runoob.com)

- CREATE DATABASE 数据库名; 创建数据库
- USE 数据库名; 选择数据库
- DROP DATABASE <数据库名>; 删除数据库
- CREATE TABLE table_name (column_name column_type); 创建表
- DROP TABLE table_name ; 删除表
- SELECT column_name,column_name 查询
  FROM table_name
  [WHERE Clause]
  [LIMIT N][OFFSET M]

查询实例读取数据表：
select * from runoob_tbl;

7.0 redis常见命令

对redis不熟悉的朋友参考Redis 教程 | 菜鸟教程 (runoob.com)

7.1 连接池实现代码

见代码：
mysql_pool
redis_pool

注意测试对应的 参数要和自己实际情况一致，通过宏定义设置

mysql_pool

```
#define DB_HOST_IP        "127.0.0.1"          // 数据库服务器ip
#define DB_HOST_PORT      3306
#define DB_DATABASE_NAME   "mysql_pool_test"     // 数据库对应的库名字, 这里需要自己提前用命令创建完毕
#define DB_USERNAME       "root"               // 数据库用户名
#define DB_PASSWORD       "123456"             // 数据库密码
#define DB_POOL_NAME      "mysql_pool"         // 连接池的名字，便于将多个连接池集中管理
#define DB_POOL_MAX_CON    4                   // 连接池支持的最大连接数量
```

redis_pool

```
#define DB_HOST_IP        "127.0.0.1"          // 数据库服务器ip
#define DB_HOST_PORT      6379
#define DB_INDEX          0                    // redis默认支持16个db
#define DB_PASSWORD       ""                   // 数据库密码，不设置AUTH时该参数为空
#define DB_POOL_NAME      "redis_pool"         // 连接池的名字，便于将多个连接池集中管理
#define DB_POOL_MAX_CON    4                   // 连接池支持的最大连接数量
```

7.1.1 msyql_pool测试

编译:

```
cd mysql_pool
mdkir build
cmake ..
make
```

编译后的执行文件在build目录

测试数据库的连接

1. test_curd.cpp对应源码
2. 修改主机的地址： #define DB_HOST_IP "127.0.0.1" // 数据库服务器ip
    1. 以及其他的参数，比如端口、用户名、密码等
3. 执行：./test_curd

本地：每个任务都重新创建连接

1. test_dbpool.cpp 对应workNoPool函数的执行
2. 修改主机的地址： #define DB_HOST_IP "127.0.0.1" // 数据库服务器ip
    1. 以及其他的参数，比如端口、用户名、密码等
3. 执行：./test_dbpool 4 4 0

本地：任务从连接池读取连接

1. test_dbpool.cpp 对应workUsePool函数的执行
2. 修改主机的地址： #define DB_HOST_IP "127.0.0.1" // 数据库服务器ip
    1. 以及其他的参数，比如端口、用户名、密码等
3. 执行：./test_dbpool 4 4 1

远程：每个任务都重新创建连接

1. test_dbpool.cpp 对应workNoPool函数的执行
2. 修改主机的地址： #define DB_HOST_IP "127.0.0.1" // 数据库服务器ip
    1. 以及其他的参数，比如端口、用户名、密码等
3. 执行：./test_dbpool 4 4 0

远程：任务从连接池读取连接

1. test_dbpool.cpp 对应workNoPool函数的执行
2. 修改主机的地址： #define DB_HOST_IP "114.215.169.66" // 数据库服务器ip
    1. 以及其他的参数，比如端口、用户名、密码等
3. 执行：./test_dbpool 4 4 1

7.1.2 redis_pool测试

编译

```
cd mysql_pool
mdkir build
cmake ..
make
```

编译后的执行文件在build目录

对应test_CachePoo.cpp

- testCacheUsePool 函数 测试使用同一连接重复set数据
- testCacheOneCmdPerConn函数测试 每次set都重新创建连接

对应test_dbpool.cpp

- ./test_dbpool 4 4 1 代表使用4线程，4连接，1使用连接池
- ./test_dbpool 4 4 0 代表使用4线程，4连接，0不用连接池

7.2 MySQL和Redis客户端连接编程

具体的mysql、redis客户端编程后续在 mysql专题、redis专题里讲解。
mysql api c客户端：https://www.yuque.com/linuxer/linux_senior/rcz4xl
hiredis的使用：https://www.yuque.com/linuxer/linux_senior/ofs3au

7.3 案例

问：4个连接池对象和4个线程使用4个连接池做同样的事情吗？还是区分每个线程做不同的事情。
答：连接池只是提供了连接对象，提供了一条连接通道，至于调用者要拿这个连接对象做什么业务是用调用者取决定的。出于业务解耦合的场景，也可以设置不同的线程池和不同的连接池应对不同的业务，比如即时通讯写入聊天记录和读取聊天记录采用不同的线程池和对象池。

8 连接池连接设置数量

连接数 = ((核心数 * 2) + 有效磁盘数)
按照这个公式，即是说你的服务器 CPU 是 4核 i7 的，那连接池连接数大小应该为 ((4*2)+1)=9

这里只是一个经验公式。还要和线程池数量以及具体业务结合在一起。

CPU总核数 = 物理CPU个数 * 每颗物理CPU的核数
总逻辑CPU数 = 物理CPU个数 * 每颗物理CPU的核数 * 超线程数

```
查看CPU信息（型号）
```

```
[root@AAA ~]# cat /proc/cpuinfo | grep name | cut -f2 -d: | uniq -c
       4  Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz


# 查看物理CPU个数
[root@AAA ~]# cat /proc/cpuinfo| grep "physical id"| sort| uniq| wc -l
2

# 查看每个物理CPU中core的个数(即核数)
[root@AAA ~]# cat /proc/cpuinfo| grep "cpu cores"| uniq
cpu cores    : 2

# 查看逻辑CPU的个数
[root@AAA ~]# cat /proc/cpuinfo| grep "processor"| wc -l
4
```