关键技术点

MySQL数据库编程、单例模式、queue队列容器、C++11多线程编程、线程互斥、线程同步通信和unique_lock、基于CAS的原子整形、智能指针shared_ptr、lambda表达式、生产者-消费者线程模型

项目背景

为了提高MySQL数据库（基于C/S设计）的访问瓶颈，除了在服务器端增加缓存服务器缓存常用的数据之外（例如redis），还可以增加连接池，来提高MySQL Server的访问效率，在高并发情况下，大量的**TCP三次握手、MySQL Server连接认证、MySQL Server关闭连接回收资源和TCP四次挥手**所耗费的性能时间也是很明显的，增加连接池就是为了减少这一部分的性能损耗。

在市场上比较流行的连接池包括阿里的druid，c3p0以及apache dbcp连接池，它们对于短时间内大量的数据库增删改查操作性能的提升是很明显的，但是它们有一个共同点就是，全部由Java实现的。

那么本项目就是为了在C/C++项目中，提供MySQL Server的访问效率，实现基于C++代码的数据库连接池模块。

连接池功能点介绍

连接池一般包含了数据库连接所用的ip地址、port端口号、用户名和密码以及其它的性能参数，例如初始连接量，最大连接量，最大空闲时间、连接超时时间等，该项目是基于C++语言实现的连接池，主要也是实现以上几个所有连接池都支持的通用基础功能。

初始连接量（initSize）：表示连接池事先会和MySQL Server创建initSize个数的connection连接，当应用发起MySQL访问时，不用再创建和MySQL Server新的连接，直接从连接池中获取一个可用的连接就可以，使用完成后，并不去释放connection，而是把当前connection再归还到连接池当中。

最大连接量（maxSize）：当并发访问MySQL Server的请求增多时，初始连接量已经不够使用了，此时会根据新的请求数量去创建更多的连接给应用去使用，但是新创建的连接数量上限是maxSize，不能无限制的创建连接，因为每一个连接都会占用一个socket资源，一般连接池和服务器程序是部署在一台主机上的，如果连接池占用过多的socket资源，那么服务器就不能接收太多的客户端请求了。当这些连接使用完成后，再次归还到连接池当中来维护。

最大空闲时间（maxIdleTime）：当访问MySQL的并发请求多了以后，连接池里面的连接数量会动态增加，上限是maxSize个，当这些连接用完再次归还到连接池当中。如果在指定的maxIdleTime里面，这些新增加的连接都没有被再次使用过，那么新增加的这些连接资源就要被回收掉，只需要保持初始连接量initSize个连接就可以了。

连接超时时间（connectionTimeout）：当MySQL的并发请求量过大，连接池中的连接数量已经到达maxSize了，而此时没有空闲的连接可供使用，那么此时应用从连接池获取连接无法成功，它通过阻塞的方式获取连接的时间如果超过connectionTimeout时间，那么获取连接失败，无法访问数据库。

该项目主要实现上述的连接池四大功能，其余连接池更多的扩展功能，可以自行实现。

MySQL Server参数介绍

mysql> show variables like 'max_connections';
该命令可以查看MySQL Server所支持的最大连接个数，超过max_connections数量的连接，MySQL Server会直接拒绝，所以在使用连接池增加连接数量的时候，MySQL Server的max_connections参数也要适当的进行调整，以适配连接池的连接上限。

功能实现设计

ConnectionPool.cpp和ConnectionPool.h：连接池代码实现
Connection.cpp和Connection.h：数据库操作代码、增删改查代码实现

连接池主要包含了以下功能点：
1.连接池只需要一个实例，所以ConnectionPool以单例模式进行设计
2.从ConnectionPool中可以获取和MySQL的连接Connection
3.空闲连接Connection全部维护在一个线程安全的Connection队列中，使用线程互斥锁保证队列的线程安全
4.如果Connection队列为空，还需要再获取连接，此时需要动态创建连接，上限数量是maxSize
5.队列中空闲连接时间超过maxIdleTime的就要被释放掉，只保留初始的initSize个连接就可以了，这个功能点肯定需要放在独立的线程中去做
6.如果Connection队列为空，而此时连接的数量已达上限maxSize，那么等待connectionTimeout时间如果还获取不到空闲的连接，那么获取连接失败，此处从Connection队列获取空闲连接，可以使用带超时时间的mutex互斥锁来实现连接超时时间
7.用户获取的连接用shared_ptr智能指针来管理，用lambda表达式定制连接释放的功能（不真正释放连接，而是把连接归还到连接池中）
8.连接的生产和连接的消费采用生产者-消费者线程模型来设计，使用了线程间的同步通信机制条件变量和互斥锁

开发平台选型

有关MySQL数据库编程、多线程编程、线程互斥和同步通信操作、智能指针、设计模式、容器等等这些技术在C++语言层面都可以直接实现，因此该项目选择直接在windows平台上进行开发，当然放在Linux平台下用g++也可以直接编译运行。

压力测试

验证数据的插入操作所花费的时间，第一次测试使用普通的数据库访问操作，第二次测试使用带连接池的数据库访问操作，对比两次操作同样数据量所花费的时间，性能压力测试结果如下：

数据量	未使用连接池花费时间	使用连接池花费时间
1000	单线程：1891ms 四线程：497ms	单线程：1079ms 四线程：408ms
5000	单线程：10033ms 四线程：2361ms	单线程： 5380ms 四线程：2041ms
10000	单线程：19403ms 四线程：4589ms	单线程：10522ms 四线程：4034ms

MySQL数据库编程

MySQL的windows安装文件云盘地址如下（development开发版，mysql头文件和libmysql库文件）：
链接：https://pan.baidu.com/s/1Y1l7qvpdR2clW5OCdOTwrQ
提取码：95de

这里的MySQL数据库编程直接采用oracle公司提供的MySQL C/C++客户端开发包，在VS上需要进行相应的头文件和库文件的配置，如下：
1.右键项目 - C/C++ - 常规 - 附加包含目录，填写mysql.h头文件的路径
2.右键项目 - 链接器 - 常规 - 附加库目录，填写libmysql.lib的路径
3.右键项目 - 链接器 - 输入 - 附加依赖项，填写libmysql.lib库的名字
4.把libmysql.dll动态链接库（Linux下后缀名是.so库）放在工程目录下

MySQL数据库C++代码封装如下：

```
#include <mysql.h>
#include <string>
using namespace std;
#include "public.h"

// 数据库操作类
class MySQL
{
public:
    // 初始化数据库连接
    MySQL()
    {
        _conn = mysql_init(nullptr);
    }
    // 释放数据库连接资源
    ~MySQL()
    {
        if (_conn != nullptr)
            mysql_close(_conn);
    }
    // 连接数据库
    bool connect(string ip, unsigned short port, string user, string password,
        string dbname)
    {
        MYSQL *p = mysql_real_connect(_conn, ip.c_str(), user.c_str(),
            password.c_str(), dbname.c_str(), port, nullptr, 0);
        return p != nullptr;
    }
    // 更新操作 insert、delete、update
    bool update(string sql)
    {
        if (mysql_query(_conn, sql.c_str()))
        {
            LOG("更新失败:" + sql);
            return false;
        }
        return true;
    }
    // 查询操作 select
    MYSQL_RES* query(string sql)
    {
        if (mysql_query(_conn, sql.c_str()))
        {
            LOG("查询失败:" + sql);
            return nullptr;
        }
        return mysql_use_result(_conn);
    }
private:
    MYSQL *_conn; // 表示和MySQL Server的一条连接
};
```