欢迎大家来到第五阶段课程

《分布式流媒体》实训项目

TNV DAY09

复习课

预习内容

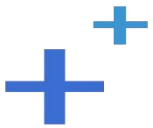
客户机（8）

客户机（8）

客户机类(client_c)的属性

- 成员变量
  - 连接池管理器：s_mngr
  - 跟踪服务器地址表：s_taddrs
  - 存储服务器连接数上限：s_scount

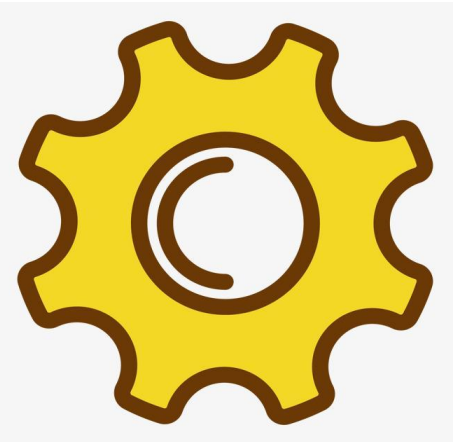
知识讲解



客户机类(client_c)的方法

知识讲解

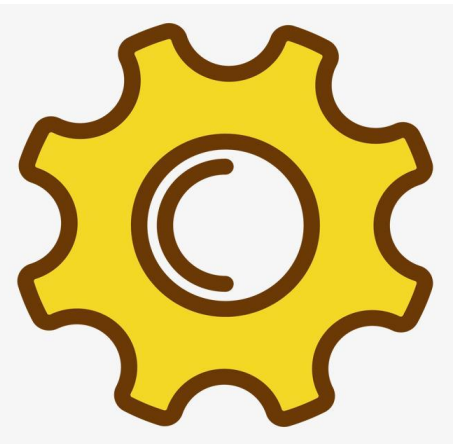
- 初始化：init
  - 检查并拆分跟踪服务器地址表
  - 检查跟踪服务器连接数上限
  - 检查存储服务器连接数上限
  - 创建连接池管理器
  - 初始化跟踪服务器连接池
- 终结化：deinit
  - 销毁连接池管理器
  - 清空跟踪服务器地址表



客户机类(client_c)的方法

知识讲解

- 从跟踪服务器获取存储服务器地址列表： saddrs
  - 随机抽取一台跟踪服务器地址，获取相应的连接池
    - 失败：尝试下一台跟踪服务器
  - 从针对该跟踪服务器的连接池中获取一个空闲连接
    - 失败：尝试下一台跟踪服务器
  - 通过该连接从跟踪服务器获取存储服务器地址列表
    - 套接字通信错误
      - 关闭连接并尝试下一个连接
    - 否则
      - 成功：释放连接
      - 失败：关闭连接
  - 返回处理结果

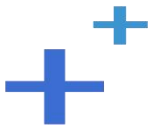


附录：程序清单

TNV/src/05_client/07_client.h

知识讲解

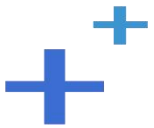
```
//客户机
//声明客户机类
//
#pragma once

#include <vector>
#include <string>
#include <lib_acl.hpp>
//
//客户机类
//
class client_c {
public:
        //初始化
        static int init(char const* taddrs,
```



TNV/src/05_client/07_client.h

```
        int tcount = 16, int scount = 8);
//终结化
static void deinit(void);

//从跟踪服务器获取存储服务器地址列表
int saddrs(char const* appid, char const* userid,
        char const* fileid, std::string& saddrs);
//从跟踪服务器获取组列表
int groups(std::string& groups);

//向存储服务器上传文件
int upload(char const* appid, char const* userid,
        char const* fileid, char const* filepath);
//向存储服务器上传文件
int upload(char const* appid, char const* userid,
```

知识讲解



TNV/src/05_client/07_client.h

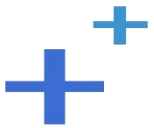
```
            char const* fileid, char const* filedata, long long filesize);
        //向存储服务器询问文件大小
        int filesize(char const* appid, char const* userid,
            char const* fileid, long long* filesize);
        //从存储服务器下载文件
        int download(char const* appid, char const* userid,
            char const* fileid, long long offset, long long size,
            char** filedata, long long* filesize);
        //删除存储服务器上的文件
        int del(char const* appid, char const* userid, char const* fileid);

private:
        static acl::connect_manager* s_mngr; //连接池管理器
        static std::vector<std::string> s_taddrs; //跟踪服务器地址表
        static int s_scount; //存储服务器连接数上限
};
```

知识讲解

TNV/src/05_client/08_client.cpp

知识讲解

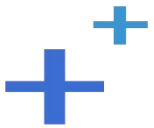
```
//客户机
//实现客户机类
//
#include <lib_acl.h>
#include "01_types.h"
#include "03_util.h"
#include "01_conn.h"
#include "03_pool.h"
#include "05_mngr.h"
#include "07_client.h"

#define MAX_SOCKERRS 10 //套接字通信错误最大次数

acl::connect_manager* client_c::s_mngr = NULL;
std::vector<std::string> client_c::s_taddrs;
```



TNV/src/05_client/08_client.cpp

知识讲解

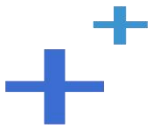
```
int client_c::s_scount = 8;

//初始化
int client_c::init(char const* taddrs,
        int tcount /* = 16 */, int scount /* = 8 */) {
        if (s_mngr)
                return OK;

        //跟踪服务器地址表
        if (!taddrs || !strlen(taddrs)) {
                logger_error("tracker addresses is null");
                return ERROR;
        }
        split(taddrs, s_taddrs);
        if (s_taddrs.empty()) {
```



TNV/src/05_client/08_client.cpp

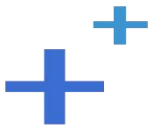
```
        logger_error("tracker addresses is empty");
        return ERROR;
}

//跟踪服务器连接数上限
if (tcount <= 0) {
        logger_error("invalid tracker connection pool count %d <= 0",
                tcount);
        return ERROR;
}

//存储服务器连接数上限
if (scount <= 0) {
        logger_error("invalid storage connection pool count %d <= 0",
                scount);
```

知识讲解



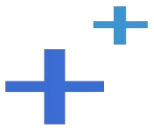
TNV/src/05_client/08_client.cpp


```
        return ERROR;
    }
    s_scount = scount;

    //创建连接池管理器
    if (!(s_mngr = new mngr_c)) {
        logger_error("create connection pool manager fail: %s",
                acl_last_serror());
        return ERROR;
    }

    //初始化跟踪服务器连接池
    s_mngr->init(NULL, taddrs, tcount);

    return OK;
```



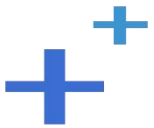
TNV/src/05_client/08_client.cpp


```
}

//终结化
void client_c::deinit(void) {
        if (s_mngr) {
                delete s_mngr;
                s_mngr = NULL;
        }

        s_taddrs.clear();
}

//从跟踪服务器获取存储服务器地址列表
int client_c::saddrs(char const* appid, char const* userid,
        char const* fileid, std::string& saddrs) {
```



TNV/src/05_client/08_client.cpp


```
if (s_taddrs.empty()) {
        logger_error("tracker addresses is empty");
        return ERROR;
}

int result = ERROR;

//生成有限随机数
srand(time(NULL));
int ntaddrs = s_taddrs.size();
int nrand = rand() % ntaddrs;

for (int i = 0; i < ntaddrs; ++i) {
        //随机抽取跟踪服务器地址
        char const* taddr = s_taddrs[nrand].c_str();
```



TNV/src/05_client/08_client.cpp

知识讲解

```
nrand = (nrand + 1) % ntaddrs;

//获取跟踪服务器连接池
pool_c* tpool = (pool_c*)s_mngr->get(taddr);
if (!tpool) {
        logger_warn("tracker connection pool is null, taddr: %s",
                taddr);
        continue;
}

for (int sockerrs = 0; sockerrs < MAX_SOCKERRS; ++sockerrs) {
        //获取跟踪服务器连接
        conn_c* tconn = (conn_c*)tpool->peek();
        if (!tconn) {
                logger_warn("tracker connection is null, taddr: %s",
```

TNV/src/05_client/08_client.cpp

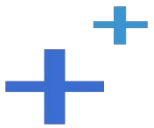
知识讲解

```
                taddr);
        break;
}

// 从跟踪服务器获取存储服务器地址列表
result = tconn->saddrs(appid, userid, fileid, saddrs);

if (result == SOCKET_ERROR) {
        logger_warn("get storage addresses fail: %s",
                tconn->errdesc());
        tpool->put(tconn, false);
}
else {
        if (result == 0)
```

TNV/src/05_client/08_client.cpp

```
                tpool->put(tconn, true);
            else {
                logger_error("get storage addresses fail: %s",
                        tconn->errdesc());
                tpool->put(tconn, false);
            }
            return result;
        }
    }
}

return result;
}
```

知识讲解



下节课见