欢迎大家来到第五阶段课程

《分布式流媒体》实训项目

预习课

TNV DAY10

预习课

预习
内容

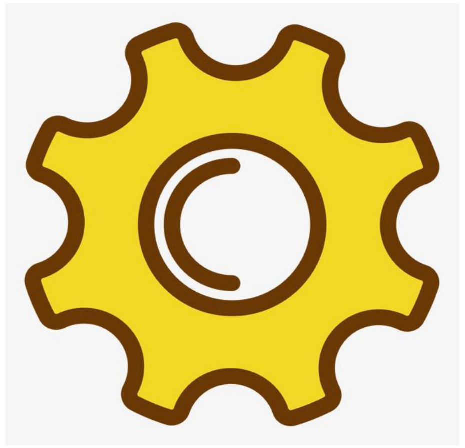
客户机（9）

客户机（9）

预习课

客户机类(client_c)的方法

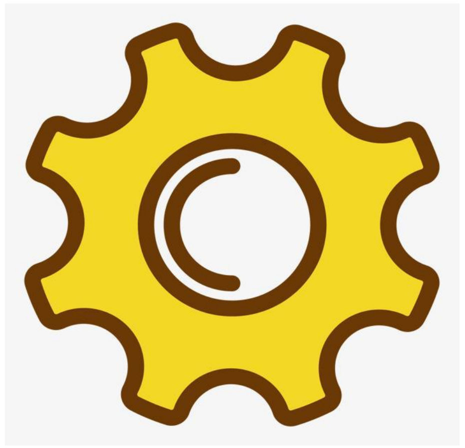
- 从跟踪服务器获取组列表：groups
  - 随机抽取一台跟踪服务器地址，获取相应的连接池
    - ➢ 失败：尝试下一台跟踪服务器
  - 从针对该跟踪服务器的连接池中获取一个空闲连接
    - ➢ 失败：尝试下一台跟踪服务器
  - 通过该连接从跟踪服务器获取组列表
    - ➢ 套接字通信错误
      - 关闭连接并尝试下一个连接
    - ➢ 否则
      - 成功：释放连接
      - 失败：关闭连接
  - 返回处理结果



知识讲解

客户机类(client_c)的方法

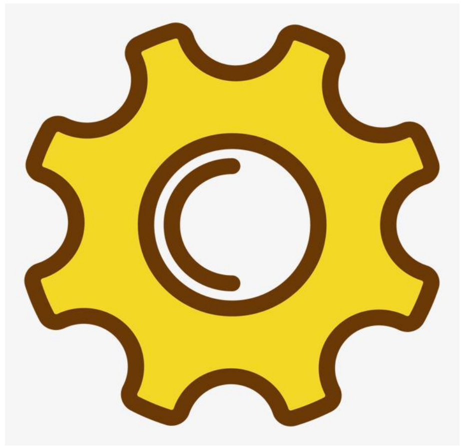
- 向存储服务器上传文件：upload
  - 从跟踪服务器获取存储服务器地址列表
  - 顺序抽取一台存储服务器地址，获取相应的连接池
    - 失败：创建针对该存储服务器的连接池并再次获取
      - 失败：尝试下一台存储服务器
  - 从针对该存储服务器的连接池中获取一个空闲连接
    - 失败：尝试下一台存储服务器
  - 通过该连接向存储服务器上传文件
    - 套接字通信错误
      - 关闭连接并尝试下一个连接
    - 否则
      - 成功：释放连接
      - 失败：关闭连接
  - 返回处理结果



预习课

客户机类(client_c)的方法

- 向存储服务器询问文件大小：filesize
  - 从跟踪服务器获取存储服务器地址列表
  - 顺序抽取一台存储服务器地址，获取相应的连接池
    - 失败：创建针对该存储服务器的连接池并再次获取
      - 失败：尝试下一台存储服务器
  - 从针对该存储服务器的连接池中获取一个空闲连接
    - 失败：尝试下一台存储服务器
  - 通过该连接向存储服务器询问文件大小
    - 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/08_client.cpp

知识讲解

```cpp
// 从跟踪服务器获取组列表
int client_c::groups(std::string& groups) {
        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) {
```

预习课

TNV/src/05_client/08_client.cpp

知识讲解

```
//随机抽取跟踪服务器地址
char const* taddr = s_taddrs[nrand].c_str();
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();
```

知识讲解

TNV/src/05_client/08_client.cpp

```
if (!tconn) {
        logger_warn("tracker connection is null, taddr: %s",
                taddr);
        break;
}

// 从跟踪服务器获取组列表
result = tconn->groups(groups);

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

预习课

TNV/src/05_client/08_client.cpp

知识讲解

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

        return result;
}

// 向存储服务器上传文件
```

预习课

TNV/src/05_client/08_client.cpp

```
int client_c::upload(char const* appid, char const* userid,
        char const* fileid, char const* filepath) {
        //检查参数
        if (!appid || !strlen(appid)) {
                logger_error("appid is null");
                return ERROR;
        }
        if (!userid || !strlen(userid)) {
                logger_error("userid is null");
                return ERROR;
        }
        if (!fileid || !strlen(fileid)) {
                logger_error("fileid is null");
                return ERROR;
        }
```

TNV/src/05_client/08_client.cpp

```
if (!filepath || !strlen(filepath)) {
        logger_error("filepath is null");
        return ERROR;
}

// 从跟踪服务器获取存储服务器地址列表
int result;
std::string ssaddrs;
if ((result = saddrs(appid, userid, fileid, ssaddrs)) != OK)
        return result;
std::vector<std::string> vsaddrs;
split(ssaddrs.c_str(), vsaddrs);
if (vsaddrs.empty()) {
        logger_error("storage addresses is empty");
        return ERROR;
```

TNV/src/05_client/08_client.cpp

```
}

result = ERROR;

for (std::vector<std::string>::const_iterator saddr =
        vsaddrs.begin(); saddr != vsaddrs.end(); ++saddr) {
        //获取存储服务器连接池
        pool_c* spool = (pool_c*)s_mngr->get(saddr->c_str());
        if (!spool) {
                s_mngr->set(saddr->c_str(), s_scount);
                if (!(spool = (pool_c*)s_mngr->get(saddr->c_str()))) {
                        logger_warn(
                                "storage connection pool is null, saddr: %s",
                                saddr->c_str());
                        continue;
```

TNV/src/05_client/08_client.cpp

知识讲解

```
		}
	}

	for (int sockerrs = 0; sockerrs < MAX_SOCKERRS; ++sockerrs) {
		// 获取存储服务器连接
		conn_c* sconn = (conn_c*)spool->peek();
		if (!sconn) {
			logger_warn("storage connection is null, saddr: %s",
				saddr->c_str());
			break;
		}

		// 向存储服务器上传文件
		result = sconn->upload(appid, userid, fileid, filepath);
```

预习课

TNV/src/05_client/08_client.cpp

```
        if (result == SOCKET_ERROR) {
                logger_warn("upload file fail: %s", sconn->errdesc());
                spool->put(sconn, false);
        }
        else {
                if (result == OK)
                        spool->put(sconn, true);
                else {
                        logger_error("upload file fail: %s",
                                sconn->errdesc());
                        spool->put(sconn, false);
                }
                return result;
        }
}
```

知识讲解

预习课

TNV/src/05_client/08_client.cpp

```
        }

        return result;
}

// 向存储服务器上传文件
int client_c::upload(char const* appid, char const* userid,
        char const* fileid, char const* filedata, long long filesize) {
        // 检查参数
        if (!appid || !strlen(appid)) {
                logger_error("appid is null");
                return ERROR;
        }
        if (!userid || !strlen(userid)) {
                logger_error("userid is null");
```

知识讲解

预习课

TNV/src/05_client/08_client.cpp

```
        return ERROR;
}
if (!fileid || !strlen(fileid)) {
        logger_error("fileid is null");
        return ERROR;
}
if (!filedata || !filesize) {
        logger_error("filedata is null");
        return ERROR;
}

// 从跟踪服务器获取存储服务器地址列表
int result;
std::string ssaddrs;
if ((result = saddrs(appid, userid, fileid, ssaddrs)) != OK)
```

TNV/src/05_client/08_client.cpp

```
        return result;
std::vector<std::string> vsaddrs;
split(ssaddrs.c_str(), vsaddrs);
if (vsaddrs.empty()) {
        logger_error("storage addresses is empty");
        return ERROR;
}

result = ERROR;

for (std::vector<std::string>::const_iterator saddr =
        vsaddrs.begin(); saddr != vsaddrs.end(); ++saddr) {
        //获取存储服务器连接池
        pool_c* spool = (pool_c*)s_mngr->get(saddr->c_str());
        if (!spool) {
```

TNV/src/05_client/08_client.cpp

```
		s_mngr->set(saddr->c_str(), s_scount);
		if (!(spool = (pool_c*)s_mngr->get(saddr->c_str()))) {
			logger_warn(
				"storage connection pool is null, saddr: %s",
				saddr->c_str());
			continue;
		}
	}

	for (int sockerrs = 0; sockerrs < MAX_SOCKERRS; ++sockerrs) {
		//获取存储服务器连接
		conn_c* sconn = (conn_c*)spool->peek();
		if (!sconn) {
			logger_warn("storage connection is null, saddr: %s",
				saddr->c_str());
```

TNV/src/05_client/08_client.cpp

```
        break;
}

// 向存储服务器上传文件
result = sconn->upload(
        appid, userid, fileid, filedata, filesize);

if (result == SOCKET_ERROR) {
        logger_warn("upload file fail: %s", sconn->errdesc());
        spool->put(sconn, false);
}
else {
        if (result == OK)
                spool->put(sconn, true);
        else {
```

TNV/src/05_client/08_client.cpp

```
                                        logger_error("upload file fail: %s",
                                                sconn->errdesc());
                                        spool->put(sconn, false);
                                }
                                return result;
                        }
                }
        }

        return result;
}

//向存储服务器询问文件大小
int client_c::filesize(char const* appid, char const* userid,
        char const* fileid, long long* filesize) {
```

TNV/src/05_client/08_client.cpp

```
//检查参数
if (!appid || !strlen(appid)) {
        logger_error("appid is null");
        return ERROR;
}
if (!userid || !strlen(userid)) {
        logger_error("userid is null");
        return ERROR;
}
if (!fileid || !strlen(fileid)) {
        logger_error("fileid is null");
        return ERROR;
}

//从跟踪服务器获取存储服务器地址列表
```

TNV/src/05_client/08_client.cpp

知识讲解

```
int result;
std::string ssaddrs;
if ((result = saddrs(appid, userid, fileid, ssaddrs)) != OK)
        return ERROR;
std::vector<std::string> vsaddrs;
split(ssaddrs.c_str(), vsaddrs);
if (vsaddrs.empty()) {
        logger_error("storage addresses is empty");
        return ERROR;
}

result = ERROR;

for (std::vector<std::string>::const_iterator saddr =
        vsaddrs.begin(); saddr != vsaddrs.end(); ++saddr) {
```

TNV/src/05_client/08_client.cpp

```
//获取存储服务器连接池
pool_c* spool = (pool_c*)s_mngr->get(saddr->c_str());
if (!spool) {
        s_mngr->set(saddr->c_str(), s_scount);
        if (!(spool = (pool_c*)s_mngr->get(saddr->c_str()))) {
                logger_warn(
                        "storage connection pool is null, saddr: %s",
                        saddr->c_str());
                continue;
        }
}

for (int sockerrs = 0; sockerrs < MAX_SOCKERRS; ++sockerrs) {
        //获取存储服务器连接
        conn_c* sconn = (conn_c*)spool->peek();
```

TNV/src/05_client/08_client.cpp

知识讲解

```
if (!sconn) {
        logger_warn("storage connection is null, saddr: %s",
                saddr->c_str());
        break;
}

// 向存储服务器询问文件大小
result = sconn->filesize(appid, userid, fileid, filesize);

if (result == SOCKET_ERROR) {
        logger_warn("get filesize fail: %s", sconn->errdesc());
        spool->put(sconn, false);
}
else {
```

预习课

TNV/src/05_client/08_client.cpp

知识讲解

```
                if (result == OK)
                    spool->put(sconn, true);
                else {
                    logger_error("get filesize fail: %s",
                        sconn->errdesc());
                    spool->put(sconn, false);
                }
                return result;
            }
        }
    }

    return result;
}
```

预习课

直播课见