# 操作系统课程大作业实验报告：简化学术评审系统

### 1. 引言 (Introduction)

#### 1.1 项目背景与目标

现代学术会议或期刊广泛使用在线评审系统，例如 OpenReview 和 EasyChair。本课程项目旨在通过实现一个简化但功能完整的评审系统，帮助深入理解操作系统核心概念。项目重点涵盖以下技术领域：

* **文件系统**：设计并实现包含超级块、inode 表、数据块、空闲位图及目录结构的自定义文件系统。
* **并发控制**：基于线程池模型实现服务端对多客户端并发请求的处理。
* **缓存机制**：实现可配置的 LRU 块缓存，优化文件系统读写性能并提供统计数据。
* **网络通信**：构建基于 TCP 的 Client-Server 架构，通过自定义应用层协议完成交互。
* **鉴权与权限**：实现基于会话的身份认证与基于角色的访问控制（RBAC）。

本项目采用 C++17 标准开发，客户端为命令行交互界面，服务端维护自定义文件系统 VFS，将所有业务数据持久化存储于后端文件 data.fs 中。

#### 1.2 需求分析

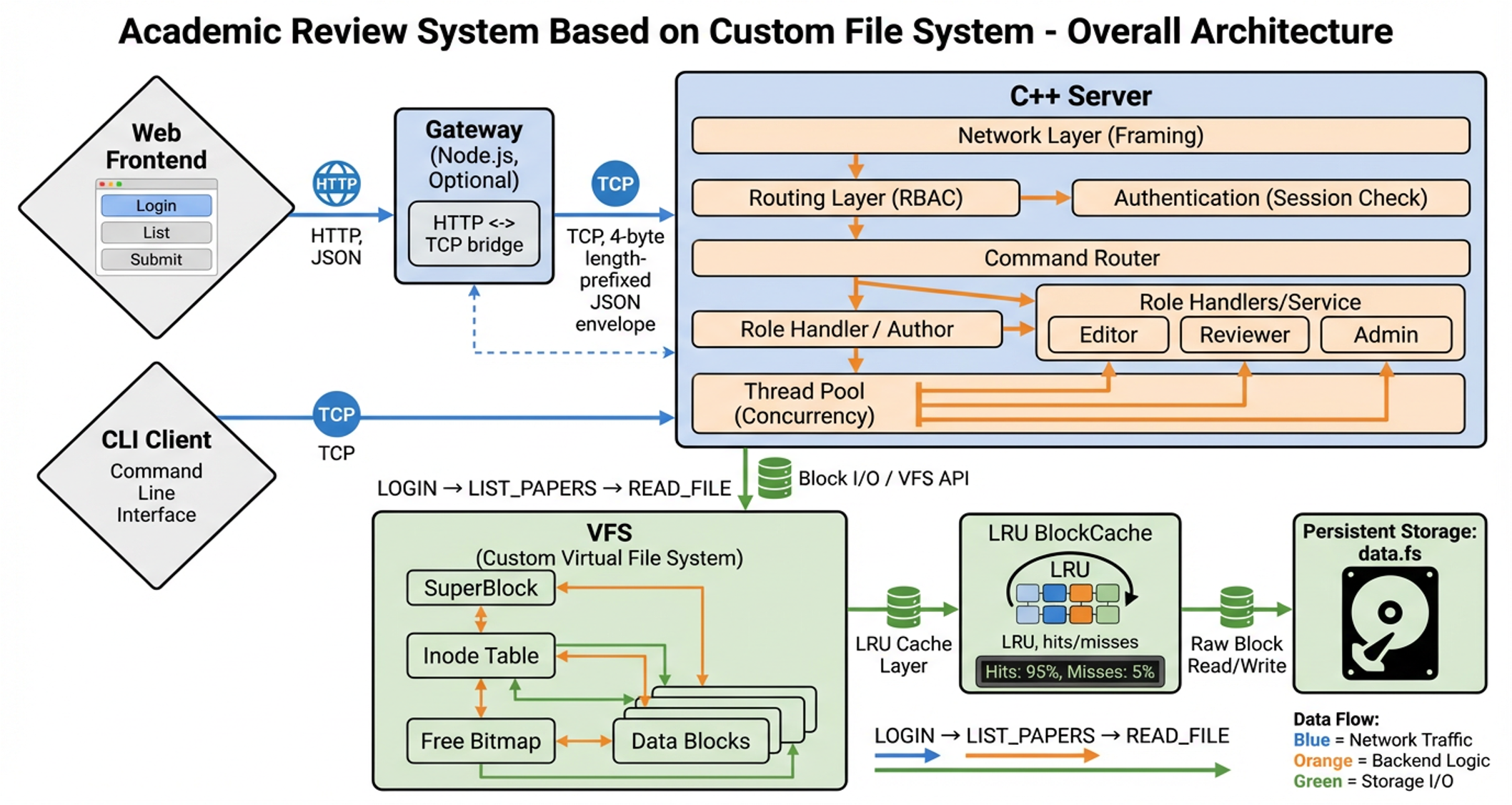
根据课程指引，系统需满足以下关键技术指标与业务需求：

* **C/S 架构**：所有业务操作由客户端发起，服务端解析请求并操作文件系统，需支持多客户端并发连接。
* **角色体系**：支持 Author（作者）、Reviewer（审稿人）、Editor（编辑）和 Admin（管理员）四种角色，各角色拥有独立的命令集与权限。
* **文件系统核心**：实现 block device 抽象，管理 superblock、inode table、data blocks 和 free bitmap；支持多级目录与路径解析；支持文件的创建、读取、写入与删除。
* **缓存优化**：引入 LRU 算法管理的块缓存，容量可配置，并能实时输出缓存命中率与替换统计。
* **数据备份**：支持全量备份与恢复机制，确保数据安全。
* **通信协议**：设计格式清晰、可扩展的应用层协议，支持文本与二进制数据的传输。

### 2. 实现细节 (Implementation Details)

#### 2.1 系统总体架构

系统采用经典的 Client-Server 架构。客户端负责解析用户命令并封装为协议报文；服务端负责业务逻辑处理、权限校验及底层文件系统操作。

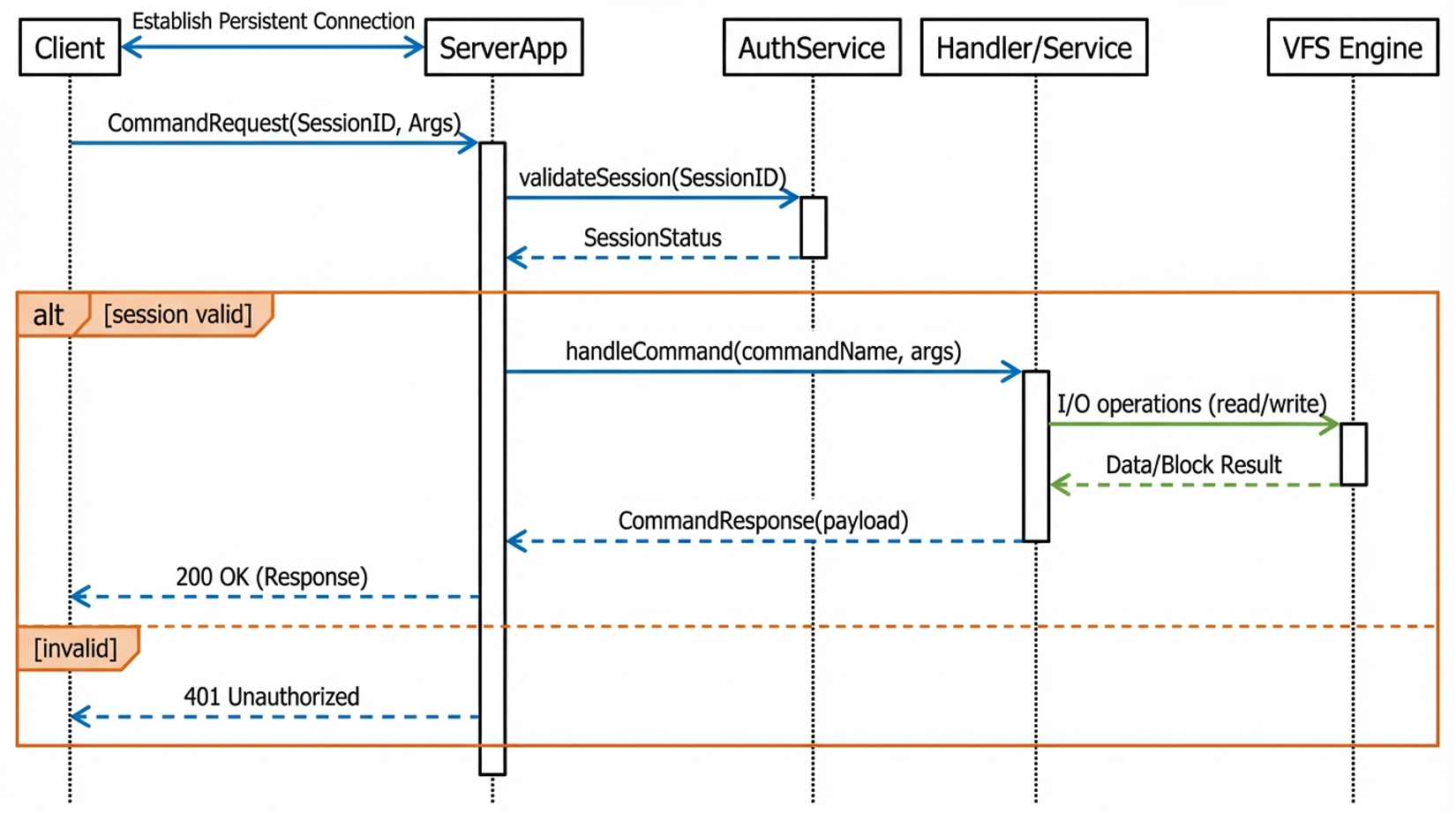


如图所示，系统核心组件包括：

* **Client CLI**：命令行交互终端，负责用户输入解析与结果展示。
* **TCP Network**：基于 TCP 协议的通信链路，保障数据传输的可靠性。
* **Server Application**：服务端主程序，包含请求路由与业务逻辑。
* **VFS**：虚拟文件系统层，屏蔽底层存储细节。
* **Data Persistence**：最终数据以二进制形式存储于宿主机的 data.fs 文件中。

#### 2.2 模块划分与类设计

为了实现高内聚低耦合的设计目标，项目采用了清晰的模块化结构。下图展示了系统主要类及其关系：

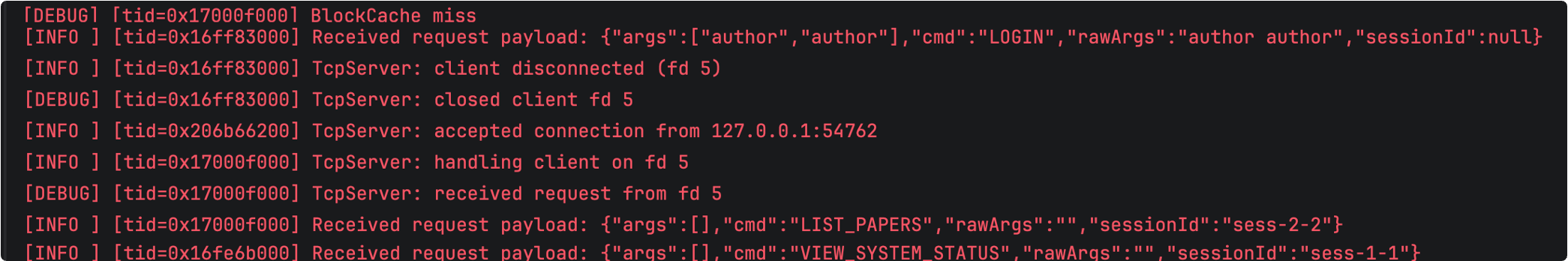


主要模块职责如下：

* **Common**：定义基础类型、日志工具及应用层协议结构。
* **Domain**：包含用户认证、权限矩阵及论文评审相关的领域模型。
* **Server/Net**：基于 TcpServer 类实现网络通信，内部集成 ThreadPool 以支持并发处理。
* **Server/Filesystem**：核心文件系统实现，包含 SuperBlock、Inode、BlockCache 及 Vfs 管理类。
* **Client**：封装 TcpClient 通信逻辑与 Cli 交互逻辑。

#### 2.3 并发模型

为满足多客户端并发访问需求，服务端采用了 Reactor 模式配合线程池的并发模型。



主线程负责监听端口并接受新连接，一旦建立连接，通过 accept 获取的 socket 描述符将被分发至 ThreadPool 中的工作线程。工作线程负责该连接后续的所有读写请求与业务处理，实现了连接级的并发控制。通过引入 std::mutex 互斥锁，确保了多线程环境下 VFS 与认证服务共享数据的安全性。

#### 2.4 通信协议设计

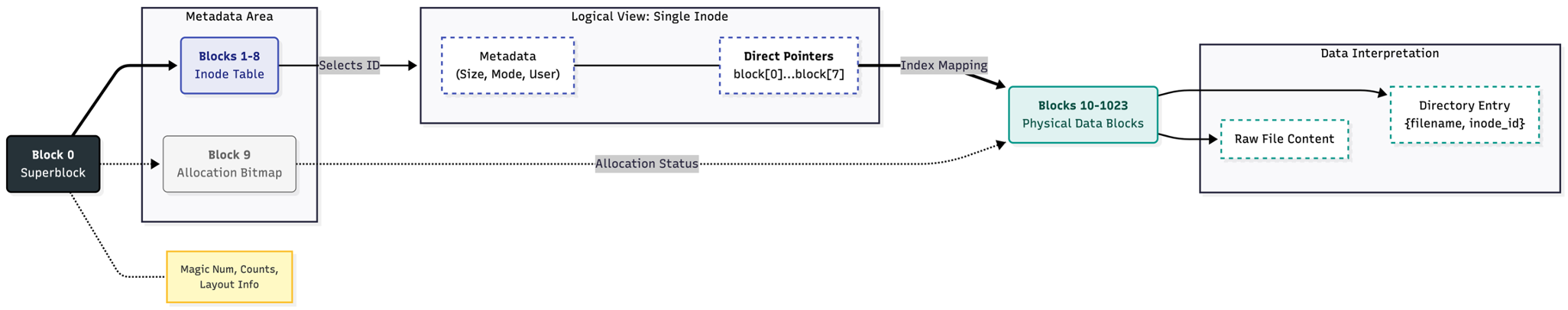
系统自定义了应用层通信协议，消息结构包含类型标识与负载数据。所有消息均被序列化为 JSON 格式字符串进行传输。

* **Type**：标识消息类型，如 CommandRequest、CommandResponse 或 Error。
* **Payload**：承载具体的请求参数或响应数据。
* **Framing**：传输层采用“4字节长度前缀 + 消息体”的帧格式，有效解决 TCP 粘包问题。



#### 2.5 自定义文件系统设计

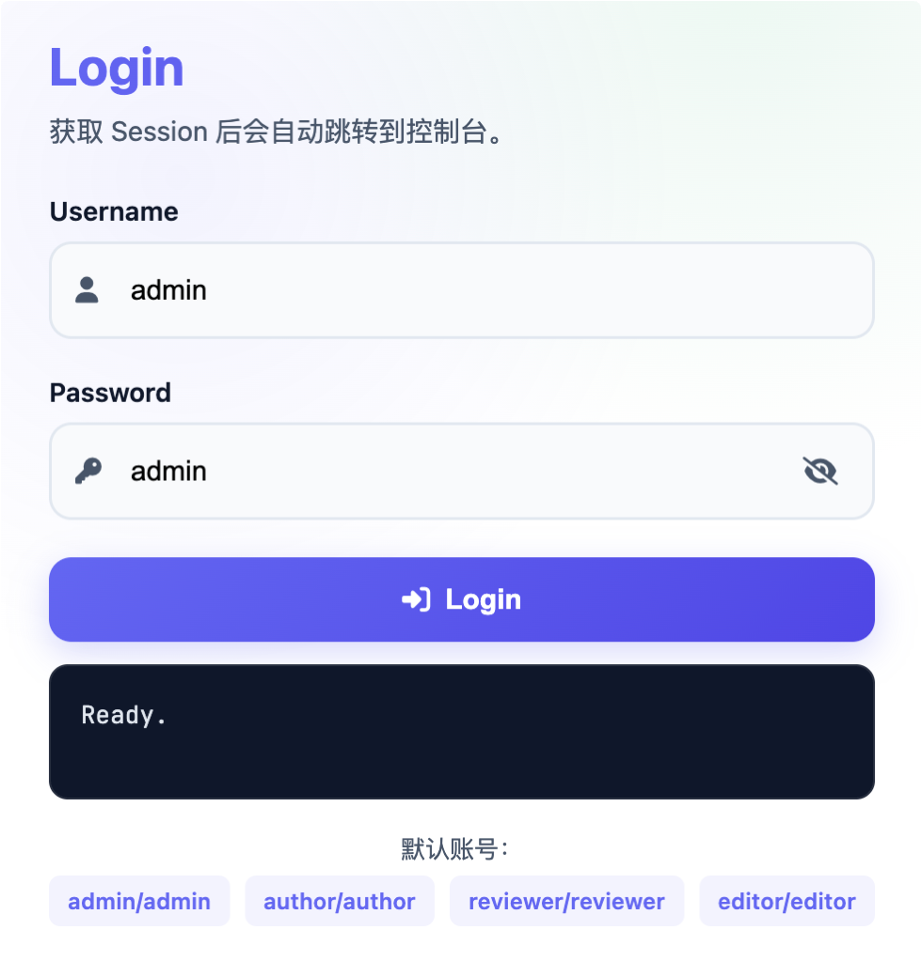
服务端启动时会挂载或格式化 data.fs 文件。文件系统采用了经典的 Unix 文件系统布局，分为超级块、Inode 表、空闲位图和数据块区域。



* **磁盘布局**：
  + **Superblock**：存储文件系统元数据（魔数、块大小、总块数）。
  + **Inode Table**：存储文件元信息（类型、大小、直接块索引）。
  + **Free Bitmap**：位图管理数据块分配。
  + **Data Blocks**：存储文件内容或目录项。
* **目录与路径解析**：  
  目录实现为存储 DirEntry 结构的文件。系统支持完整的多级路径解析，能将如 /papers/1/content.txt 的路径解析为对应的 Inode。
* **缓存机制**：  
  集成 BlockCache 模块，基于内存实现 LRU 缓存算法。系统提供统计接口，可实时查看缓存的命中与置换情况。

#### 2.6 鉴权与权限控制

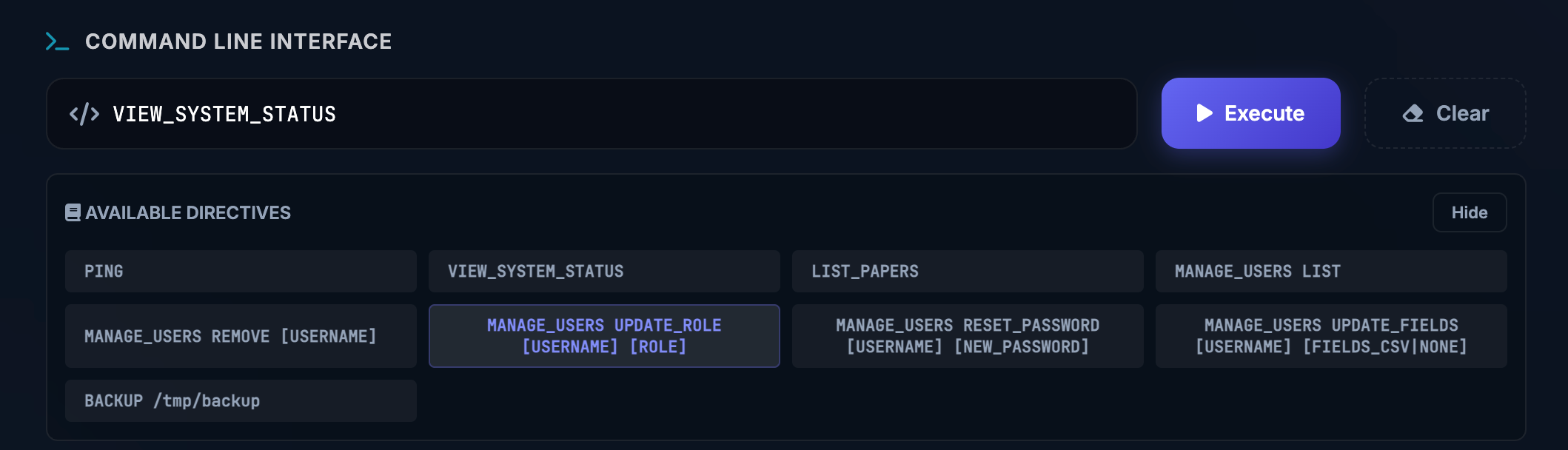
* **用户认证**：通过 AuthService 管理用户。登录后生成 sessionId，客户端后续请求自动携带该 ID 进行身份验证。
* **访问控制**：基于 RBAC 模型。Permission 模块定义权限矩阵，服务端根据会话角色（Author/Reviewer/Editor/Admin）校验操作权限。



#### 2.7 业务流程与交互

系统将论文评审业务逻辑映射为文件系统操作：

* **提交论文**：在 /papers/ 创建目录并写入元数据。
* **分配审稿**：Editor 修改 reviewers.txt。
* **提交评审**：Reviewer 在 reviews/ 目录写入评审文件。
* **最终决策**：Editor 更新元数据状态。
* **客户端交互**：CLI 提供动态菜单与快捷指令。



### 3. 执行说明 (Execution Instruction)

#### 3.1 系统构建

项目使用 CMake 进行构建管理。

1. **环境要求**：C++17 编译器 (GCC/Clang), CMake 3.10+
2. **构建命令**：

* mkdir build && cd build  
  cmake ..  
  cmake --build .

1. **产物**：
   * build/src/osproj\_server：服务端可执行文件
   * build/src/osproj\_client：客户端可执行文件

#### 3.2 运行说明

1. **启动服务端**：

* ./build/src/osproj\_server [port] [cacheCapacity]  
  # 示例：./build/src/osproj\_server 8080 1024

1. **启动客户端**：

* ./build/src/osproj\_client [serverIp] [serverPort]  
  # 示例：./build/src/osproj\_client 127.0.0.1 8080

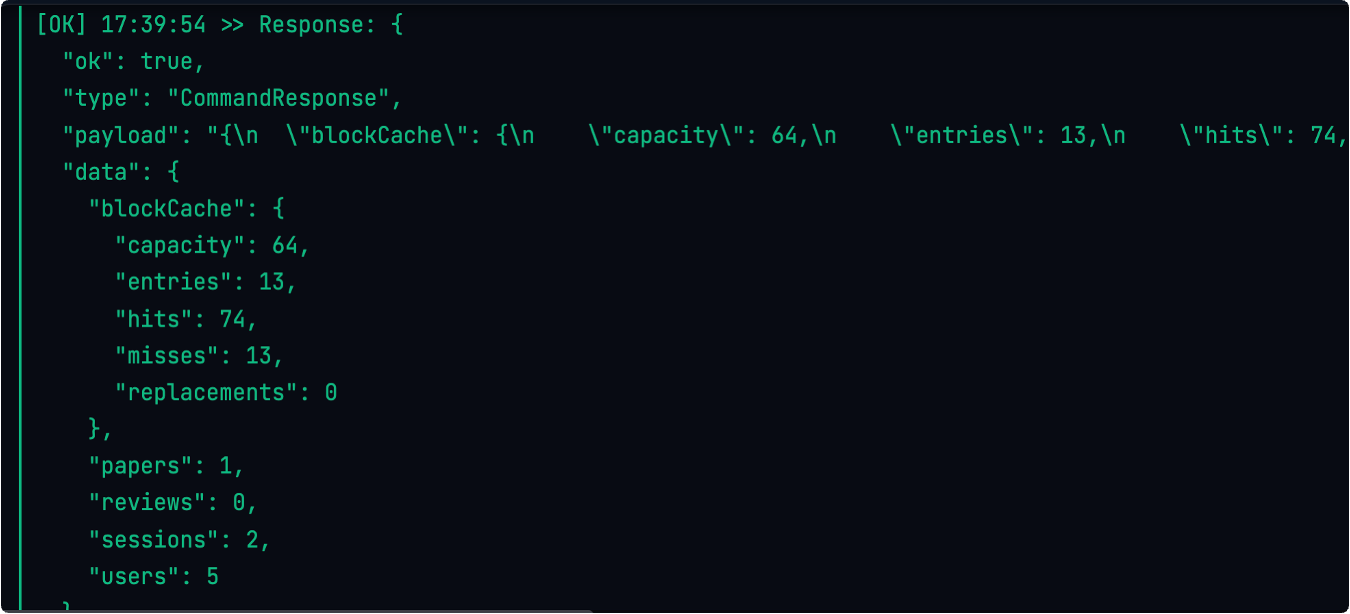
### 4. 结果展示 (Result Presentation)

#### 4.1 已完成功能清单

* **基础通信**：TCP 网络通信与 JSON 协议封装。
* **并发服务**：线程池并发处理模型。
* **文件系统**：Superblock、Inode、Bitmap 及文件的增删改查。
* **路径解析**：多级目录支持。
* **缓存系统**：LRU 缓存及统计。
* **业务逻辑**：论文提交、分配、评审、决策全流程。
* **权限控制**：基于角色的权限校验。

#### 4.2 系统运行效果

下图展示了系统在多终端环境下的端到端运行效果，验证了多角色协同工作及数据的持久化存储。



### 5. 团队分工

* **王梓衡**：负责总体架构设计、协议制定、服务端核心路由及自定义文件系统（Superblock、Inode、Bitmap）的实现。
* **宋曦**：负责前端系统架构设计、可视化交互实现及前后端通信模块封装。
* **张叶涵 & 李雨桐**：负责客户端 CLI 开发、业务流程逻辑实现、权限矩阵设计及备份恢复功能调研。

### 6. 附录

**核心命令参考**：

* **系统命令**：PING、LOGIN、ROLE\_HELP、VIEW\_SYSTEM\_STATUS
* **文件操作**：LIST、MKDIR、WRITE、READ、RM
* **业务操作**：SUBMIT、ASSIGN、REVIEW、DECISION