刘罡

(+86) 18707530177 | liugscho@163.com | github.com/Gangliu0036

教育背景

2023.09 ~ 2026.06(预计) 深圳大学 光电信息工程（硕士）

专业成绩：GPA 3.63 / 4.0

2019.09 ~ 2023.06 深圳大学 光电信息科学与工程（本科）

专业成绩：GPA 3.7 / 4.5

语言能力：大学英语六级（CET-6）,普通话二级甲等

获奖情况：研究生学业一等奖学金，第十九届“挑战杯”全国大学生课外学术科技作品竞赛（量子计算赛道）国家二等奖

专业技能

- 熟悉 C/C++，熟练使用指针操作及内存管理，掌握 C++面向对象特性（封装/继承/多态），熟悉 STL 常用容器与算法，熟练应用 C++11 特性（智能指针/移动语义等），了解 Go、Python 语言基础语法
- 熟悉常用数据结构（链表/栈/队列/二叉树/哈希表），掌握基础算法（排序/搜索/贪心/动态规划），具备 LeetCode 算法题解决能力
- 熟悉计算机网络体系结构（OSI 七层模型/TCP/IP 四层模型），深入理解 TCP/UDP 协议特性，掌握 TCP 连接管理（三次握手/四次挥手）、流量控制、拥塞控制等机制
- 熟悉 Linux 系统开发环境，掌握网络编程与 I/O 多路复用技术（select/poll/epoll），理解 Reactor 线程池模型实现原理
- 熟练使用 MySQL 数据库，熟悉索引优化（B+树结构）、事务管理（ACID 特性）、存储引擎（InnoDB/MyISAM）、锁机制（表锁/行锁）等核心机制
- 熟悉操作系统核心概念，包括进程通信（管道/消息队列/共享内存）、死锁预防（银行家算法）、内存管理（分页/分段/虚拟内存）等机制
- 具备 Cursor 等智能编程辅助工具使用经验和本地化部署大模型经验。在科研阶段曾基于 llama.cpp 部署本地大模型（支持 4-bit 量化），实现私有化文档问答系统，模型显存占用降低 60%。

项目经历

2025.03 ~ 2025.06 个人项目 基于异步日志的高并发文件服务系统

项目概述: 基于 libevent 网络库构建高并发文件服务系统，实现文件上传/下载/元数据管理核心功能，设计多级存储策略（热/温/冷数据分层存储），集成异步日志系统支持多线程并发写入（最高 128 线程）与自动滚动备份

技术实现:

- 采用单例模式+建造者模式实现可配置日志器，通过工厂方法创建分级日志对象（DEBUG/INFO/WARN 等级别）
- 设计双缓冲队列+条件变量实现异步日志处理，通过批量写入策略将 I/O 延迟降低至 2.3ms/op
- 基于 JSON Schema 规范管理文件元数据，采用读写锁保证内存哈希表的线程安全访问（并发读+独占写）
- 应用 C++17 Filesystem API 实现跨平台路径规范化处理，结合 RAII 机制与智能指针进行资源生命周期管理
- 基于 libevent 事件驱动架构实现 TCP 服务端，采用 Reactor 模式完成事件分发与请求解析，QPS 提升 42.6%
- 在 2 核 4G 云服务器环境下进行压力测试，实现单机峰值吞吐量 130MB

关键技术挑战:

- 多级存储介质性能优化：通过 LruCache 缓存热点数据，冷数据采用压缩存储（zlib level 5），降低 SSD 写入量 37%
- 高并发日志系统设计：实现无锁队列+定时刷新机制，在保证日志完整性的前提下将系统吞吐提升 2.8 倍
- 内存管理优化：采用对象池模式复用内存资源，通过 jemalloc 替代 glibc 内存分配器，降低内存碎片率 15%

技术成果:

- 系统掌握 libevent 事件循环核心机制，实现基于 Deferred Callback 的异步编程范式
- 深入理解多线程同步机制，熟练应用条件变量/原子操作/读写锁等并发控制技术
- 形成完整的服务端性能优化方法论，涵盖 I/O 模型选择（Reactor vs Proactor）、零拷贝传输、批处理策略
- 构建可扩展的日志系统架构，支持插件化存储后端（本地文件/AWS S3）、实时监控指标输出（Prometheus 格式）

2025.01 ~ 2025.03 个人项目 基于 **Raft** 共识算法的分布式 **KV** 存储数据库

项目概述: 实现基于 Raft 共识算法的分布式键值数据库，具备强一致性与分区容错性(P-CF 特性)，在(N/2)-1 个节点故障时仍可保持可用性。系统采用自主研发的 MprRpc 通信框架与 SkipListPro 跳表存储引擎，实现完整的分布式协调与数据持久化能力。

技术实现:

- 基于 protobuf 与自定义二进制协议实现 MprRpc 通信框架，支持节点间远程过程调用与高效数据传输
- 设计多级跳表存储结构 SkipListPro，实现 O(logN)时间复杂度的高效键值存取，支持快照持久化与内存压缩
- 实现 Raft 核心状态机：包括领导者选举(心跳超时检测、任期递增、选票收集)、日志复制(条目校验、一致性检查、提交传播)
- 构建强一致性模型：领导者节点通过并行流水线方式批量复制日志条目，在法定数节点持久化后提交状态变更
- 设计客户端会话协议：采用(IP+序列号)复合请求 ID 保证线性一致性，实现智能重试机制与幂等性控制

关键技术挑战:

- Raft 状态机与日志复制的原子性保证
- 强领导者模式下日志复制一致性保障
- 网络分区场景下的任期同步与状态回滚
- 跳表索引与 WAL 日志的协同持久化策略
- 水平扩展时读写性能优化

技术成果:

- 实现完整 Raft 共识协议，通过 10 节点集群验证强一致性，成功处理 200k+日志条目
- 通信框架达到 12k QPS 吞吐量，平均延迟低于 1.5ms（99th percentile）
- 跳表引擎实现 98% 空间利用率，支持 10M 级键值对的毫秒级查询
- 系统通过 Jepsen 测试验证线性一致性，支持自动故障转移与数据自愈

自我评价

1. 有较强的信息检索能力，擅长坚决疑难杂症，通过 Google、Github、StackOverflow 等国外论坛/文档解决技术问题.
2. 与时俱进，拥抱 AI，在开发过程中会使用大模型帮助理解难懂错误堆栈和调试思路，提高自身开发效率。
3. 在开发中对待问题具有认真求索的精神，够在短时间内解决问题并理解知识点，严格要求自己遵守编码规范，这使我少写了很多 bug。