a

教育经历

华东师范大学 985 2024年09月 - 2027年07月

计算机科学与技术 硕士 计算机科学与技术学院

广东工业大学 双一流 2020年09月 - 2024年07月

软件工程 本科 信息技术学院

获奖情况

算法竞赛

作为队长，带领队伍参加**国际大学生程序设计竞赛（ACM/ICPC）**，专注于C++算法设计与实现，获得**亚洲区域赛金奖3次**，银奖1次，**ECfinal(东亚总决赛)银奖1次，**多次创造校史最好成绩。蓝桥杯C++组国一，团队程序设计天梯赛团队国一等若干奖项。

解决算法难题超过2000道。通过竞赛，积累了丰富的C++编程、代码调试、性能优化和团队协作经验，为开发打下坚实基础。

实习经历

上海非凸智能科技有限公司 2025年08月 - 2025年12月

C++开发（高性能计算方向） 量化研究 上海

项目概述：从 0 到 1 搭建了一套美股高频行情处理系统，在 50 核并发环境下实现了对全市场每日 **5 亿条（5e8）** 原始数据的实时解析与计算。系统吞吐量达到 **1,000,000 msg/s**，单日全量行情回放及因子计算仅需 500 秒。

模块一：Order-Trade 到 Depth 的实时状态机重建

订单簿重建：基于共享内存（SHM）摄取的原始 **Order（报单）与 Trade（成交）** 增量数据，独立实现了一套高频行情状态机，通过增量更新算法实时维护全档位订单簿（Full Order Book）。

极致延迟优化：针对美股极高频的撤单与成交特性，优化了订单匹配与档位更新算法，将 **OT-to-Depth 的全链路处理延迟压低至 15,000ns (15μs)** 以内。

内存与架构设计：采用预分配内存池（Memory Pool）与连续内存布局，极大降低了动态内存分配开销与 Cache Miss；应用 Cache-line Padding 技术消除伪共享，确保在 50 核并发读取时的高吞吐性能。

模块二：Depth 到多维度因子的高性能计算引擎

大规模因子计算并行化：构建了支持 **300 个高频因子** 并发更新的计算引擎，单笔行情触发的全量因子累加计算延迟控制在 **80,000ns (80μs) 以内**。

Python 算子 C++ 工程化重构：负责将研究员的 Python 因子逻辑转化为高性能 C++ 实现。

精度对齐与压测验证：开发自动化的数值校验工具，确保 C++ 算子在极速计算环境下与 Python 原型实现达到 Bit-level 精度对齐。在 50 核环境下成功扛住 100 万 TPS 的瞬时流量冲击，验证了系统的极致稳定性。

专业技能

熟悉 C++ 语法，**掌握面向对象编程及 C++11 新特性，熟悉 Socket 网络编程与 IO 多路复用技术，了解 Linux 环境下多线程编程**；同时了解 Go、Python 语法。

精通 STL 常用容器与算法，**能够灵活根据场景选择数据结构，擅长二分查找、动态规划等算法。**

熟练使用Copilot等AI辅助编程工具。

了解操作系统基础知识（进程通信、内存管理）以及常见计算机网络协议（TCP/UDP）。

熟悉 Redis，MySQL。

项目经历

基于Raft共识算法的分布式KV存储数据库 2025年06月 - 2025年08月

本项目是基于Raft共识算法的分布式K-V数据库，具备线性一致性和分区容错性，在少于半数节点发生故障仍可正常对外提供服务。使用跳表数据库SKipListPro完成K-V存储功能。

主要工作:

1.Raft协议的心跳与选举机制：通过定时线程池触发心跳与选举任务，并维护集群的日志提交状态。

2.日志读写与提交：由领导节点处理客户端的读写请求，并将日志复制至跟随者节点，在超过半数节点复制成功后提交日志，应用命令至状态机并返回响应给客户端5.实现客户端协议，包括在客户端协议中加入由ip和请求序号组成的“请求id”以保证线性-致性，以及客户端重试等功能。

3.K-V存储：基于跳表数据结构实现跳表数据库SkipListPro完成K-V存储功能。

C++服务器框架 2025年03月 - 2025年05月

本项目在 Linux 环境下使用 C++ 从零开始开发了部分服务器框架，主要实现了协程库的编写，基于 ucontext_t 实现协程类，结合 epoll 和定时器实现了 N-M 协程调度器，支持 IO 事件和定时器事件的回调。通过协程调度，对常见 API 进行 hook 封装，实现异步调度。同时完成了日志功能、配置功能、读写锁等模块。

线程封装：封装了 pthread、互斥量、信号量、读写锁、自旋锁，实现范围锁。

协程实现：通过 ucontext_t 实现了非对称协程，设计三种协程状态，使子协程可以与线程主协程相互切换。

定时器：基于时间堆实现定时器功能，支持定时事件的添加、删除和更新。

协程调度：实现 N-M 协程调度器，支持 main 函数线程参与调度。在基本的协程调度器基础上结合 epoll 和定时器实现了 IO 协程调度，支持 IO 事件和定时事件的注册与回调。

Hook：基于 IO 协程调度器，对 sleep 系列、Socket IO 系列、文件描述符操作系列系统调用进行 hook 封装，实现阻塞调用的异步化。