👨‍💻赵勇臻

（+86）181-6810-0075 | zhaoyzzz@outlook.com

出生年月：2002.3.22 | 籍贯：甘肃庆阳

教育背景

南京大学 – 硕士 – 软件工程专业（2024.09-2026.07）（推免）

南京大学 – 本科 – 软件工程专业（2020.09-2024.07）

专业技能

- **基础知识**：熟悉计算机网络、操作系统和编译原理等方面基础知识。
- **编程语言**：熟悉 **Java**， **Python** 以及 **C++**。
- **后端框架**：熟悉 **SpringBoot**。也能快速上手其他 Web 框架。
- **技术分享**：在博客 https://rzyn2020.github.io/ 中进行一些技术分享。
- **语言能力**：英语（CET6），能熟练阅读英文文档和书籍，观看英文技术视频。

实习经历

腾讯 – CSIG – TRTC – 音视频 sdk（2025.06 – ）

腾讯实时音视频（Tencent Real-Time Communication，简称 TRTC） 是腾讯云提供的一款核心的实时音视频通信 PaaS 产品。我主要负责音视频 sdk 开发。

网易 – 伏羲 – 后端系统开发 –（2023.06 – 2023.09）

AOP（Agent-Oriented-Programming，面向智能体编程）是网易伏羲设计的一套全新的编程范式，用户可以通过该框架调用智能体服务。我主要负责项目自用序列化框架 DDL 模块的开发，并采用**测试驱动开发（TDD）**和**类型驱动开发**等技术，确保了代码的高质量和可维护性。主要工作如下：

- **功能开发：**扩展 DDL 模块的功能，新增对多种复杂数据类型的支持同时，为项目引入 type hints， 结合 **mypy** 进行静态类型检查，有效提升了代码的可读性、可维护性和类型安全性，减少了运行时错误的发生概率。
- **性能分析：**开发性能分析脚本，利用火焰图进行性能分析，定位了关键性能瓶颈。
- **完善测试：**基于 **Property-Based Random Testing** 思想，完善了 DDL 模块的测试，将覆盖率提升至 **90%**。
- **性能优化：** 采用了多种优化方法提升了 DDL 的性能。在优化之前 DDL 序列化平均耗时在 protobuf 的**50倍**左右，优化后在大部分典型场景下性能与原生 protobuf 相差在**2倍**以内，部分场景为 protobuf 的**四分之一**不到。

项目经历

求是社区 – （2025.04 – 2025.07）

实现了一个面向哲学与社会学爱好者的前后端分离社区平台，引入 **AI 虚拟哲学家**参与深度讨论。我负责项目的技术选型与整体框架搭建，并采用 **ADD (Architecture Driven Design)** 和 **DDD (Domain-Driven Design)** 方法进行架构设计与功能实现。

技术栈：Spring Boot、MyBatis-Plus、MySQL、Redis、MongoDB、Elasticsearch、RabbitMQ、Docker

核心技术：

- 利用 **RabbitMQ** 将用户评论、点赞、收藏及系统消息进行异步解耦，有效提升系统并发处理能力。
- 通过 **Redis ZSET** 实现了**用户活跃度实时排行**，并采用先写 MySQL，再删除 Redis 缓存的策略，在高并发场景下有效保证了**缓存一致性**。
- 为满足社区在高并发场景下业务 ID 的唯一性与可追溯性，设计并实现了一套基于**雪花算法**的**分布式 ID 生成方案**，显著**降低了 ID 生成延迟**。
- 结合 **RAG** 和 **Agent** 技术，引入了 **AI 虚拟哲学家**，使其能理解并参与用户的哲学讨论，极大丰富了社区互动体验。

Transformer-LLM – （2025.03 – 2025.04）

- 实现了一个基于 **Transformer** 架构的**大语言模型**，并在此基础上进行性能优化以及微调，对齐工作。
- 深入研究并实践了 LLM 的性能剖析（**Profiling**） 以及**性能优化**技术。
- 通过该项目，掌握了 LLM 从模型设计、训练、优化到部署的**端到端流程和关键技术**。

SysY-RISCV 编译器 – （2024.07 – 2024.09）

- 使用 **C++17** 完成了一个**高性能编译器**，其生成的代码性能表现达到 **GCC O2 水准**。
- 基于 **SSA IR** 实现了多种**编译器优化技术**，如**死代码消除**、**常量折叠**、**循环优化**，寄**存器分配**，显著提升了目标代码执行效率。
- 负责编译器后端开发并担任**组长**，锻炼了团队协作和领导能力，加深了对编程语言的认识。

miniOS 操作系统 – （2022.03 – 2022.06）

- 使用**C语言**实现了一个**支持多处理器的操作系统**。
- 在实现**内存分配器**的过程中，分别实现了基于**链表**的，基于**红黑树**的，和基于 **slab** 的内存分配器，深刻理解了不同内存管理策略的性能权衡，并将其融入到“**Fast Path，Slow Path**”相结合的系统设计原则中。
- 在实现内核多线程的过程中，深入理解了**并发编程的基本理论**，认识到在并发编程中“**防御式编程**”的重要性。