王旭辉

18100549550 | wxh.simoo@gmail.com
工作年限: 7 年 | 17届本科 | 后端开发工程师

专业技能

- 了解RAG、Prompt、向量检索、个性化问答
- 熟悉 Go、PHP ，具备扎实的编码、调试与工程化能力
- 熟悉 Kafka、RabbitMQ 的应用与实践
- 熟悉 MySQL、Redis 的使用与优化，具备性能调优经验
- 熟悉 插件化体系架构设计，包括插件生命周期管理、状态机、调度控制与运行时隔离
- 熟悉 事件驱动架构，具备基于 Saga 模式 实现跨系统最终一致性事务的经验
- 了解 Kubernetes、Docker 在应用容器化与部署中的使用

工作经历

深圳复临信息科技有限公司（ ONES ） 2020.10 - 2025.02

后端开发工程师 开放平台

公司专注研发 ONES 企业研发管理工具（对标 Jira），已在国内市场处于行业领先地位

开放平台旨在基于 ONES 系统构建 插件体系与开放能力，为有定制化需求的客户提供灵活扩展方案，通过插件满足个性化场景，显著提升成单率与客户满意度

开放平台官网： https://developer.ones.com/

主要工作：

- 参与设计搭建初版插件体系，实现核心系统与扩展功能的解耦，支持第三方开发者灵活定制和扩展
- 参与设计搭建插件调度系统，实时监控插件状态并自动纠偏，确保插件实际状态与期望状态一致
- 设计搭建事件订阅系统， 支持灵活订阅与实时推送，提升合作伙伴系统的集成效率和响应速度
- 参与设计搭建OpenAPI体系，统一对外开放服务接口，促进第三方集成与生态构建
- 参与设计搭建开放体系，将业务流程抽象为标准化功能扩展点，结合契约接口设计，支持功能动态加载与热插拔

成就：

- 推动公司转型为插件交付模式，显著提升交付效率与产品灵活性
- 参与并推动平台从零到一的设计与搭建，构建了基础架构和核心能力

深圳市啪啪运动科技有限公司 2018.09 - 2020.09

后端开发工程师

专注智慧体育数字化解决方案，服务全国多家大型场馆与政府机构

主要工作：

- 与前端、产品、测试团队紧密协作，推动产品迭代

项目经历

婴幼儿辅食RAG智能问答系统（个人项目）

鉴于每日手动查阅小红书辅食帖子的低效，所以开发RAG智能问答系统，以更快、更智能地生成宝宝的辅食安排

主要工作：

- 系统采用经典的DDD四层架构，通过依赖注入实现控制反转。各层职责清晰（Domain定义接口，Infrastructure实现，Application组装业务，Presentation初始化并对外暴露），严格遵循分层依赖规则，奠定了高质量、可扩展的代码基础
- 离线索引构建：支持批量处理多文件路径，基于 LangChain 封装的 DocumentLoaders 实现多格式文档加载；通过 LangChain 封装的DashScopeEmbeddings 提供高效向量化处理；并结合 LangChain 封装的 FAISS 类完成多样化向量存储管理
- 检索生成：按照文档检索 → 提示词构建 → 答案生成 → 后处理流程来完成检索生成：在文档检索阶段，结合用户问题与档案信息进行个性化查询和结果重排序；在提示词构建阶段，融合系统角色、用户档案、对话上下文与参

考资料，生成上下文感知的提示词；在答案生成阶段，以专业身份生成科学、安全、实用的回答；在后处理阶段，对答案进行结构化提取、营养规则校验与风险提示，确保结果准确可靠

插件调度系统

初版插件体系在生命周期管理、状态同步与容错机制等核心环节面临显著挑战，制约了系统整体稳定性与生态发展

主要工作：

- 系统架构：参与设计并实现高可用插件化系统，采用控制平面（Plugin-Platform + Plugin-Runtime-Manager）与数据平面（Plugin-Host）分离架构。通过声明式 API 管理插件全生命周期，由 Plugin-Host-Boot 负责数据平面节点进程的生命周期管理，并依托 Node.js VM 沙箱与ZeroMQ+Protobuf 高性能通信，保障插件安全隔离与系统稳定扩展
- 插件生命周期管理：构建基于有限状态机的插件生命周期管理体系（安装、启用、停用、卸载），设计期望状态、实例状态与过程状态三层状态模型，确保状态流转严格有序、可预测且全程可追溯。
- 分布式最终一致性事务： 构建基于事件驱动架构的编排式 Saga 事务系统，用于协调涉及多个外部系统的插件业务操作（如能力注册）。通过持久化工作队列与协调器协程，驱动具备可重试、可补偿特性的分布式事务流程，确保跨系统操作的最终一致性、幂等性与故障恢复能力
- 调度控制： 借鉴 Kubernetes 控制器模式，为核心插件实例的生命周期（启用/停用）设计独立的状态调谐循环。通过监听期望状态与实际状态差异，并经由事件驱动触发相应的控制指令，实现实例状态的自动化对齐、自我修复与高效调度
- 安全与隔离：在 Plugin-Host 层基于 Node.js vm 模块实现插件运行时沙箱隔离，严格限制插件对宿主资源的访问，确保多租户环境下的安全性与稳定性

成果：

- 成功支撑插件稳定运行，系统可用性与可靠性显著提升
- 引入基于事件驱动与Saga事务调度的异步协调机制，有效保障插件生命周期操作的最终一致性、可重试性与自动容错，显著提升系统在分布式环境下的可靠性
- 系统具备企业级可扩展性与最终一致性，依托插件级运行时安全隔离与故障快速自愈机制，为开放平台生态提供高可靠底层支撑

事件订阅系统

当前插件采用轮询方式监听数据变化，产生大量无效请求，导致服务器压力大、资源浪费和响应延迟。为提升效率，所以引入了事件订阅系统，仅在数据变更时推送通知，显著降低了不必要的轮询开销，提高了系统性能和实时性

主要工作：

- 事件转换模块：从Kafka消费原始事件数据，并依据事件订阅模块所管理的动态规则，进行格式标准化、数据过滤与防腐化处理，最终将规范事件写入统一事件池（Kafka Topic），完成事件的初步清洗与汇聚
- 事件路由模块：从统一事件池消费消息后，为每个事件异步启动协程任务，通过HTTP协议同步发送至Plugin-Platform平台。此过程内置了重试机制、熔断器（防止故障扩散）以及死信队列（隔离异常消息）等可靠性保障机制，确保事件投递的最终一致性。Plugin-Platform平台再负责将事件分派给相应的业务插件
- 事件订阅模块：负责对所有事件的订阅关系及其插件回调地址进行统一管理和配置，为事件转换与路由提供动态决策依据，实现了处理规则的可控与灵活变更

成果：

- 通过用事件推送替代原有轮询机制，有效减少无效查询请求，显著降低服务端负载和数据库访问压力，系统资源利用率得到优化
- 事件路由模块中协程异步任务与Kafka消费者组的架构设计，使系统可通过增加实例数快速扩展事件处理能力，更好应对业务流量波动
- 事件订阅模块提供统一、动态的规则管理界面，支持新插件快速订阅所需事件，插件接入周期缩短，系统灵活性和可维护性大幅增强