王日昱

微信: i664929402

手机: 189_0677_0895

邮箱: wangriyu1997@gmail.com

resume

> Links

- Github: https://github.com/wangriyu
- Homepage: https://home.wangriyu.wang
- Blog: https://blog.wangriyu.wang
- Resume: https://blog.wangriyu.wang/pages/resume

> Education

华北电力大学(保定) - NCEPU

计算机系 - 软件工程

本科 2015 - 2019

> Profile

- 17 年从事过前端开发，18 年转后端，19 年毕业进入杭州字节跳动
- 熟悉团队作业，熟悉产品设计、前后端交互之间的沟通和管理
- 习惯英语阅读，大学已过 CET6
- 代码书写规范，热爱编程，喜欢电子产品

> Skills

- **主力语言: Golang**, 副语言: javaScript
- 熟悉 Golang 的大部分中高级特性的使用方法，了解一些底层实现，比如切片、管道、内存管理、GC、map、sync.Map、http等， 熟悉单元测试、基准测试、pprof、trace 等工具
- 对网络协议栈略有了解，曾研究过**HTTP2、TLS(HTTPS)**
- 熟悉 Linux 环境和基本操作，有自己的服务器和建站经验（博客）
- 了解分布式相关概念，有一定高并发开发经验，熟悉多线程、锁、原子操作等，了解微服务架构和即时通讯架构
- 有部分基础架构经验；了解 RabbitMQ、Kafka 消息中间件

> Experience

- **2019.7 至今**，毕业后进入杭州字节跳动 EE 研发岗，前后参与过飞书管理后台维护、组织架构重构、权限中台搭建几个项目， 主要经历是参与权限中台从 0 到 1 的搭建过程，负责日常系统维护、业务需求对接、架构升级重构等事项

项目展示

权限 *1.0*

- 支持复杂架构节点授权。代表主客体的实体类型可以支持用户、部门、用户组、单位、租户等复杂类型表征
- 支持规则优先级和实体优先级冲突约束，比如部门链以子优先(范围由小到大)、用户优于用户组优于部门(粒度由细到粗)、同级以不能为准、动态优先级高于类型优先级等
- 支持动态生效条件，比如在规则的基础上增加约束条件: 某个时间段内生效
- 抽象规则模型，提供基础的授权服务和鉴权服务，业务方接入只需要按场景添加规则，即可通过鉴权实现各种细粒度管控
- 支持大型组织的复杂架构细粒度权限管控，上线组织架构可见性管控、名片页字段可见性管控、会话管控、搜索(黑名单)管控、管理员范围管控、应用通讯录范围管控

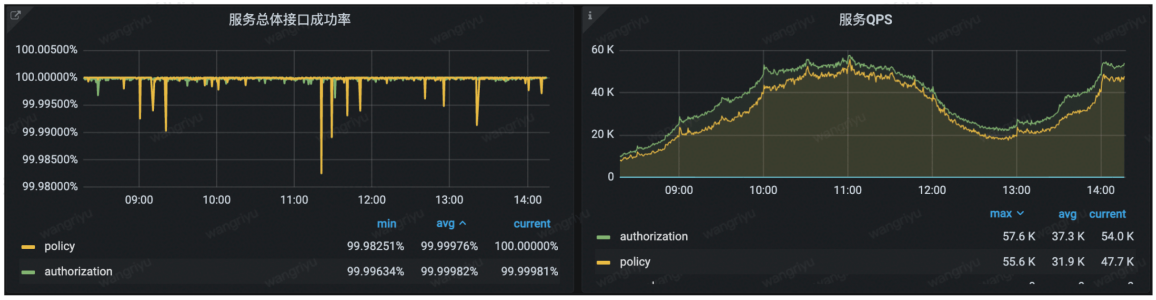
权限 *2.0*

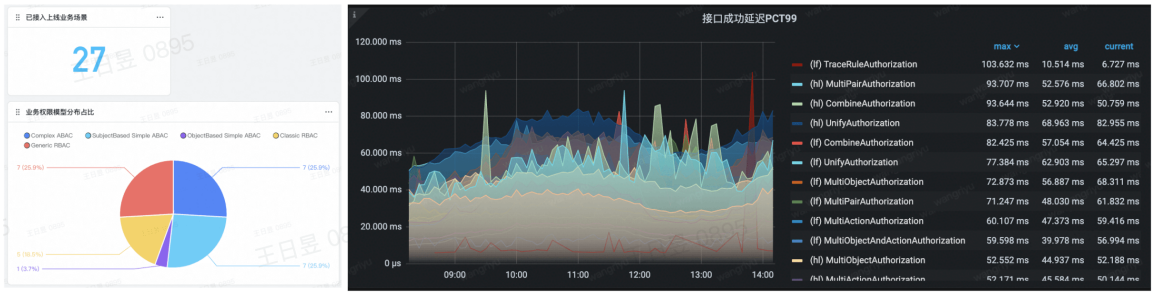
- 支持 C 端自授权和跨租户管控，支持离线点位推送和缓存
- 新增简化模型，支持海量规则配置需求，上线了对外沟通管控、联系人管控、密聊管控等

权限 *3.0*

- 权限模型扩展升级，支持 ABAC 和 RBAC 灵活管控；存储架构升级重构，可支撑百亿级规则(目前线上已有超 10 亿的规则)的高性能鉴权场景，支持水平扩展

- 完成历史业务数据平滑迁移和流量切换，升级后服务指标提升明显，性能和稳定性有大幅提升
- 接入包含文档权限管控和 IM 会话场景管控在内的 27 种权限管控场景，授权鉴权总计 QPS 10w+，PCT99 100ms





- **2018.8 ~ 2019.2** 在北京蓝城兄弟信息技术有限公司参与后端研发实习，主营项目是 Blued， 我所在的小组负责基础架构升级 & IM 服务重构，我主要参与了日志系统、CICD 流程的搭建以及离线消息推送模块重构

项目展示

1. **搭建一套完善的高可用可扩展日志系统**

- 架构实现: Filebeat + Kafka + ELK + Grafana
- ELK 使用 docker-compose 加自定义镜像和配置实现简易分发式部署，项目的基础模板分享到 docker-elk， 详细介绍参见博客
- 给 Grafana 添加企业微信报警设置并打包成镜像，项目分享到 docker-grafana，修改细节参照博客

2. **给 *Golang* 项目搭建一套 *CICD* 系统**

公司原本的项目以 PHP、Node 为主，架构组正在推进技术栈转型 Golang，旧项目像 Node 使用的是 "shipit + pm2" 实现一键部署和维护，Golang 项目当时没有类似的工具，调研测试后通过接入 Jenkins 为核心的系统实现 CICD 流程

3. **离线消息推送模块服务端重构**

离线消息推送是 IM 模块其中一个分支（PUSH）的下游，负责 APP 端上消息的离线推送通知， 此模块涉及到多个场景的事件推送，不同项目间通过 MQ 解耦，然后由 pns 服务消费事件并推送给第三方服务商， 原先项目由 Node 实现，重构过程用 Golang 重新设计和实现

- **PNS-Sender:** 端 SDK，提供推送 API，内部封装好消息格式，推给后端服务和 MQ

- **PNS:** 负责处理整个离线推送模块的业务逻辑， 从 RabbitMQ 消费推送事件， 解析不同场景的消息，并做格式转换，再过滤(对方关闭通知、屏蔽用户、黑名单等)，最后分发(华为、小米、OV、谷歌、苹果，不同平台走不同推送)
- **FCM:** 谷歌推送 SDK，华为和小米的推送直接在 PNS 中发送， 而谷歌推送因为特殊原因需要放到海外服务器上，单独拎出来发送，消息转换成 FCM 格式发送给谷歌
- **APNP:** 苹果推送 SDK，苹果推送和谷歌推送占推送中的大头， 推送服务单独维护

- **2017.6 ~ 2018.6** 在北京极智人科技有限公司旗下的技术猫编程俱乐部(微信小程序中可以搜索技术猫编程俱乐部)参与实习， 项目主要是内部一些尝试性项目，以探索总结为主，期间接触并尝试过 Flutter、Electron、Beego、EggJS、React、ReactNative 等框架。2017 年从事前端开发，技术栈以 react 为主；2018 年转后台，开始接触 Golang

> 项目展示
