

潘兆丰



求职意向: 后端开发工程师 (Golang业务开发方向) | 首选工作地: 江浙沪 / 海外
工作经验: 1年4个月 | 出生年月: 1998/12
手机号码: 15988788890 (微信同号) / 18868846460
邮箱: issacpan98@gmail.com

教育经历

浙江大学 硕士 软件工程	2021.09–2024.06
浙江工业大学 本科 信息与计算科学	2017.09–2021.07

工作经历

滴滴出行 实习生 基础平台-文件存储	2022.11–2024.05
滴滴出行 后端开发工程师 基础平台-文件存储	2024.07–2025.10

部门介绍:	• 部门产品为分布式文件存储服务, 为公司内部各业务线提供分布式对象/文件存储能力
离线 EC 迁移	2024.07–2024.10

背景:	• 传统多副本存储占用大量资源, 存储成本高且无法条带化读取, 需安全高效地迁移 PB 级冷数据至 EC(纠删码)存储
工作:	• 负责开发离线数据迁移模块, 按分组粒度将数据从多副本迁移至 EC 存储
挑战:	• 单分组包含 10 亿+小文件, 需在保证数据一致性和业务无感知的前提下, 实现高吞吐、零错误的迁移
解决:	• 采用类似 Map-Reduce 的架构设计迁移模块: 多个 Master 将待迁移数据 Hash 至多个 Migrate 中间表, 多个 Worker 抢占任务执行迁移, 最终迁移速度能打满磁盘 • 将任务拆分为数据上传、校验、元数据更新等流水线阶段, 提高并发与吞吐率的同时保证正确性

BS2.0 存储引擎-研发后台任务及调度框架	2025.02–2025.07
------------------------	-----------------

背景:	• 存储底座 BS 2.0 架构升级为大集群模式, 同时管理数百亿个副本, 需构建统一的调度框架和数据保障体系以提升可靠性和运维效率
工作:	• 设计并开发 BS 后台任务, 包含数据正确性任务 (数据校验、数据修复) 以及并发垃圾回收任务 • 为以上后台任务设计调度框架 BG, 包含任务创建、分发、异常任务重启、限流等操作
挑战:	• 海量副本调度: 管理 100 亿+副本的后台任务, 需设计高效且安全的任务调度框架 • 前后台隔离: 控制后台任务流量, 需兼顾前台延迟与后台吞吐
解决:	• 配置模块 RS 通过将数据分区的方式减少单次调度处理的数据量, 同时通过存储模块 BS 心跳检查副本数是否完整, 分担调度模块 BG 扫描压力 • 调度模块 BG 由多台机器组成, 每个节点不断尝试抢占分布式锁从 RS 拉取分区信息生成调度任务, 通过考虑机器负载、不同任务间的互斥性和优先级, 生成调度任务 • 后台任务限流从按磁盘类型的硬限制, 演进为独立统计前后台流量、定期扫描自动调整前后台限流比例, 同时通过预测前台延迟, 在高负载时直接拒绝后台请求 • 将垃圾回收分为存量和增量阶段, 进行两阶段回收, 尽可能避免影响客户端写入

项目经历

Go实现消息队列	2025.07–至今
----------	------------

项目地址:	• https://github.com/issac1998/go-queue
项目描述:	• 使用 Go 实现的分布式强一致持久化消息队列, 采用分区+段的日志型文件存储, 通过 Raft 算法保障一致性, 支持 Etcd 服务发现与 Broker 动态扩容 • 采用 Multi-Raft 架构管理元数据及数据, 每个 Partition 是独立的 Raft Group, Group 内 Leader 处理写请求, Follower 处理读请求 • Controller 作为特殊的 Raft Group, 管理集群元数据、分配 Partition 并定期通过迁移 Leader 实现负载均衡 • 实现 Exactly-Once、Consumer Group、Dead Letter Queue、延时队列、顺序消息、生产者/消费者事务, 支持消息压缩与去重

秒杀系统设计	2025.09–至今
--------	------------

项目地址:	• https://github.com/issac1998/mall
项目描述:	• 搭建高并发秒杀系统 (Go/Gin + Redis Cluster + Kafka + MySQL), 涵盖用户登录、下单流程, 实现秒杀库存扣减及订单创建, 预扣减阶段达 2K QPS, 订单创建阶段达 500 QPS • 利用 Redis Lua 原子预扣与库存分片, 结合 Local Quota (分配部分可直接扣减的库存) 分摊 Redis 开销 • Kafka 异步生产 + Worker Pool 消费, 通过 Redis Pub/Sub 机制兜底 TCC 异步消息丢失引发的悬挂及少卖问题 • 实现多级限流 (本地令牌桶 + Redis 全局限流) 与动态降级策略, 保障核心链路稳定性与弹性恢复 • 工程化实践: Singleflight 防缓存击穿、Snowflake 分布式ID、Redis Pub/Sub 缓存同步等