# 个人信息

- 姓名：王康；性别：男；生日：1989/12/24；婚否：已婚；
- 电话：18610842638； Email：focusj.x@gmail.com；
- 毕业院校：河北大学工商学院（本科）；毕业时间：2012/7/01；专业：信息管理与信息系统；
- 博客：http://www.jianshu.com/users/13542edebea3/

# 工作经历

## 1. 2019年3月~今：字节跳动

工作职责：

- 系统架构优化升级;
- 系统服务化重构改造；

## 1. 2017年11月~2019年3月：北京摩拜信息技术有限公司

工作职责：

- 负责从零搭建公司监控系统，包括：系统架构设计，子系统开发及项目进度把控。
- 推进监控系统在公司业务部门的对接和使用。

## 2. 2016年03月~2017年11月：北京第三石信息科技有限公司

工作职责：

- 后端核心开发，主要负责后端系统的设计和开发。
- 系统重构和微服务架构改造。

## 3. 2014年10月~2015年11月：ThoughtWorks

工作职责：

- 业务系统全栈开发

## 4. 2012年04月~2014年09月：北京尤尼信息科技有限公司

工作职责：

- 负责公司后端系统的开发和维护。
- 兼职项目经理，负责和产品经理对接和梳理需求，制定迭代计划。

# 技术总结

- 熟练使用Java，Scala，Golang，Python等语言，熟悉面向对象和函数式编程；良好的编码习惯，代码洁癖。

- 熟悉多线程编程和异步编程，了解多种并发模型。了解Java Executor框架，Akka等并发框架。
- 了解缓存，存储，消息队列等中间件的使用（MongoDB，MySQL，InfluxDB, Redis，Kafka，RabbitMQ等）。
- 了解微服务架构，服务网格。了解Spring Cloud Netflix生态，了解Istio，gRPC框架的使用。
- 了解SRE工程实践，对系统监控报警，Tracing等都较为熟悉。
- 了解DevOps，熟悉Docker和Kubernetes的使用。
- 熟悉敏捷项目管理，敏捷开发，极限编程，对CI/CD流程较为熟悉。

# 项目经验

## 字节跳动审核平台架构优化

*项目信息*

项目描述：负责字节跳动公司内审核平台的存储层重构，架构升级，以及服务化改造工作。

技术栈：Go，Python，Thrift，Kafka，MongoDB，Abase（KV），ES。

周期：2019/09 ~ 2020/06

*项目职责*

**实现DRC数据同步组件**

实现DRC数据同步组件，服务异构存储间数据同步。整个DRC分为三个模块：oplog收集模块负责从MongoDB拉取增量日志推送到kafka；数据拼装模块消费kafka并从db查询一次最新数据转送另一topic；sink模块消费拼装好的消息，将数据写入ES，Abase，HDFS等不同的数据源。

数据拼装模块是非常典型的生产者消费者模型。实现层面生产者占用主进程，负责从kafka拉取多条消息，下发到不同的worker中。每个Worker是独立的进程，同时每个worker中又有多个线程并发处理消息。这种多进程+多线程模型最大限度提升了消费能力（单纯用Python的进程和线程都会存在问题）。worker在接收到消息时也会作一遍去重操作，减少DB的查询次数。

在数据一致性上做到了程序层面的弱一致性保障：1. 尽可能保证消息在消费过程中不丢；2. 尽可能保证消息不乱。

项目收益：

- 数据同步实时性从原来几十秒提升至ptc99延时750ms；
- 数据一致性从原来的2个9提升至7个9+；
- 性能大幅提升，每月节省物理资源费用5000/月。

**存储层优化**

存储层优化，提升存储层稳定性和性能，构建统一数据服务。该项目主要为了解决平台在存储层面临的问题：1. 数据分区策略缺陷，导致热门业务数据分区过大，影响mongo稳定性；2. 由于业务特性数据周期生命周期一般小于1月，大量的过期数据严重影响了mongo的性能和稳定性；3. 平台内部引入了多套存储系统（ES，KV），但均未充分发挥作用，读写主要还集中在mongo。

基于上述痛点，主要提出以下改进方案：1. 冷热数据隔离，将处于业务完成态的数据迁移到KV系统，只在mongo中保存热数据；2. 充分发挥es的检索优势和kv的性能有事构造查询引擎，以减轻mongo的压力；3. 抽象数据服务，隔离所有的数据操作服务，提供统一数据出口。项目收益：

- 冷热数据分离，减少MongoDB数据量2/5。消除了数据库慢查，数据库整体稳定性大幅提升。同时为后续数据分区优化奠定基础；
- 充分利用es和kv的各自优势，构建查询引擎，查询接口延时平均下降20%。
- 构建数据中层服务，统一数据出口，同时，推动后续服务改造和数据中台建设。
- 将数据库原生oplog数据对外开发，提升其他系统使用平台数据的灵活性。

**任务引擎重构**

当前审核任务的实现方式是通过脚本拼凑来实现的，项目里到处都有处理任务状态流转的代码。导致的问题是，经常有任务状态流转出出错，任务跑到异常状态，影响审核员工作。这块新方案计划引入状态机来集中管理任务状态流转，预期达到效果：

- 保证任务状态流转的完备性，不会出现错误状态任务；
- 增加程序的健壮性，通过持久化Event的方式实现状态流转可重试，可回滚，可追溯；
- Event是和业务操作一一对应的，通过Flink消费Event流，计算目前系统上的实时指标。

**SRE和效能实践**

1. 在项目内引入持续集成实践，并在组内推广使用。同时培训项目成员如何写UT和TDD开发。
2. 梳理监控指标体系，构建系统SLA。从稳定性（服务在线时间）和服务质量（正确率和延时）两个维度梳理指标，并对指标进行埋点，构建监控看板。

## 审核平台GDPR改造

*项目信息*

项目描述：负责推进平台的GDPR改造，并推动业务方进行服务迁移。

技术栈：Python，Django，MongoDB

周期：2019/05~2019/08

*项目职责*

1. 完成系统技术改造，配置和生产代码分离，实现系统单元化部署能力；
2. 实现数据迁移系统，确保各业务线安全平稳安全迁移，最终实现上万队列的迁移；
3. 作为项目owner整体把控项目进度，确保项目如期交付；

## 摩拜监控系统

*项目信息*

项目描述：从零开始搭建摩拜内部监控系统，目前所有的核心服务都已接入监控服务。每天监控系统处理7亿+指标信息(100G)，发出有效报警上百条。

技术栈：Java，Go，Spring Cloud，Istio，Kubernetes，gRPC，InfluxDB，ES

周期：2018/02 ~ 2019/03

*项目职责*

1. 从零搭建公司业务监控系统，协助所有核心业务系统均已接入监控服务，现在每天处理7亿+指标量。
2. 使用Golang开发基于内存的实时报警策略计算模块，报警策略表达式可以支持配置阈值和同环比，同时还提供了降噪功能，防止瞬间抖动造成误报。实时性方面：一个异常指标从应用端上报到报警发出整个链路耗时可控制在1min之内。
3. 基于influx-proxy的做了InfluxDB双区多实例高可用方案。在原有开源项目上开发了动态更新数据节点配置的功能。
4. 优化数据消费服务指标判重逻辑。原有的实现方式是：直接查询MySQL数据库，程序使用线程池提升处理能力。但由于数据量太大导致了线程池队列积压，引发了OOM。通过引入布隆过滤器重新优化了这部分实现，不仅解决了原有问题还加快了系统处理速度。
5. 基于Kubernetes和Istio的ServiceMesh改造，并且基于Kubernetes和Istio API实现了灰度发布工具。
6. 使用Golang和Mux开发了内部短域名服务，上线半月的时间已经生成了7000多万个code。

*技术栈*：SpringCloud，Java8，Golang，gRPC，Istio，Kubernetes，InfluxDB，ElasticSearch，Grafana，Telegraf。

## 第三石订单系统

*项目信息*

项目描述：北京第三石在美国市场做C2C业务，主要是线下二手物品交易。为了让用户交易更安全方便，公司要做担保交易和物流。

技术栈： Java 8，Spring Boot，Spring StateMachine，JOOQ，Akka，Flyway，RabbitMQ，Redis，MySQL

周期：2016/09 ~ 2017/01

*项目职责*:

1. 引入Akka框架实现异步编程（将业务流程中的非关键流程异步话，比如发送push，订单扣费邮件等都可异步化），缩短服务的响应时间：订单接口从2~3s缩短到1s内。
2. 引入Spring StateMachine重构订单状态流转逻辑，增强了程序的可维护性。并通过其提供的回调机制实现订单历史状态持久化。
3. 基于Redlock算法实现了Java版本的Redis分布式锁。
4. 按照REST风格设计API，并通过swagger提供在线文档，降低前后端对接的难度。

## 第三石微服务重构

*项目信息*

项目描述：北京第三石是成立了三年多的创业公司，由于缺乏对后端服务重构和优化，所有的业务都耦合在一个系统里，导致系统臃肿复杂。开发维护成本越来越高。而且Python + Django也导致了严重的性能问题。为了解决这些问题，开始推进后端服务的拆分，以及技术栈转型.

技术栈：Java8，Vert.x，JOOQ，MySQL，Redis，Docker，etcd。

周期：2017/03 ~ 2018/10

*项目职责*

1. 系统边界划分以及微服务拆分规划。通过对现在业务的盘点和梳理，一共抽象了5个基础服务：用户，商品，订单，物流，聊天。实施方案如下：将现有系统中和基础服务相关的数据操作代码提取出来，包装为服务；接下来将剩余代码，按照不同的业务归属拆分为不同服务。
2. 技术调研和技术选型。用户、商品这些基础服务对性能要求比较高，综合对比了SpringBoot和Vertx，最终选择Vert.x作为基础服务框架。
3. 结合Redis缓存，最终用户服务完成后性能测试（4G/4Core）：读7000+QPS，写3000+QPS。

## 名称：业务系统

*项目信息*

项目描述：该项目是ThoughtWorks为某美国公司开发的业务处理系统。该公司主要负责为第三方公司提供财务审计，国际报税等业务。每项业务都由单独的系统处理。为了方便第三方客户查看及使用，需要作一个聚合的网站，把现有的业务聚合到一个网站内操作.

技术栈：.Net Web API，ReactJS，SQL Server

周期：2014/10 ~ 2015/11

*项目经历*:

- 引入Pact契约测试框架，提前暴露由接口协议改变而引发的微服务集成失败。
- 引入React封装前端通用组件，减少团队的重复性工作，保证网站体验的一致性。
- 丰富的敏捷开发实战体验：TDD，CodeReview，CI，CD等。