

# 新型内容流服务架构 方式

范怀宇 @ 轻芒

# QCon

## 全球软件开发大会

### 北京·2019

更多技术干货分享，北京站精彩继续

提前参与，还能享受更多优惠

识别二维码  
查看了解更多

[2019.qconbeijing.com](http://2019.qconbeijing.com)



# 范怀宇

2009 年清华大学毕业，加入网易有道

2011 年加入豌豆荚，负责技术研发

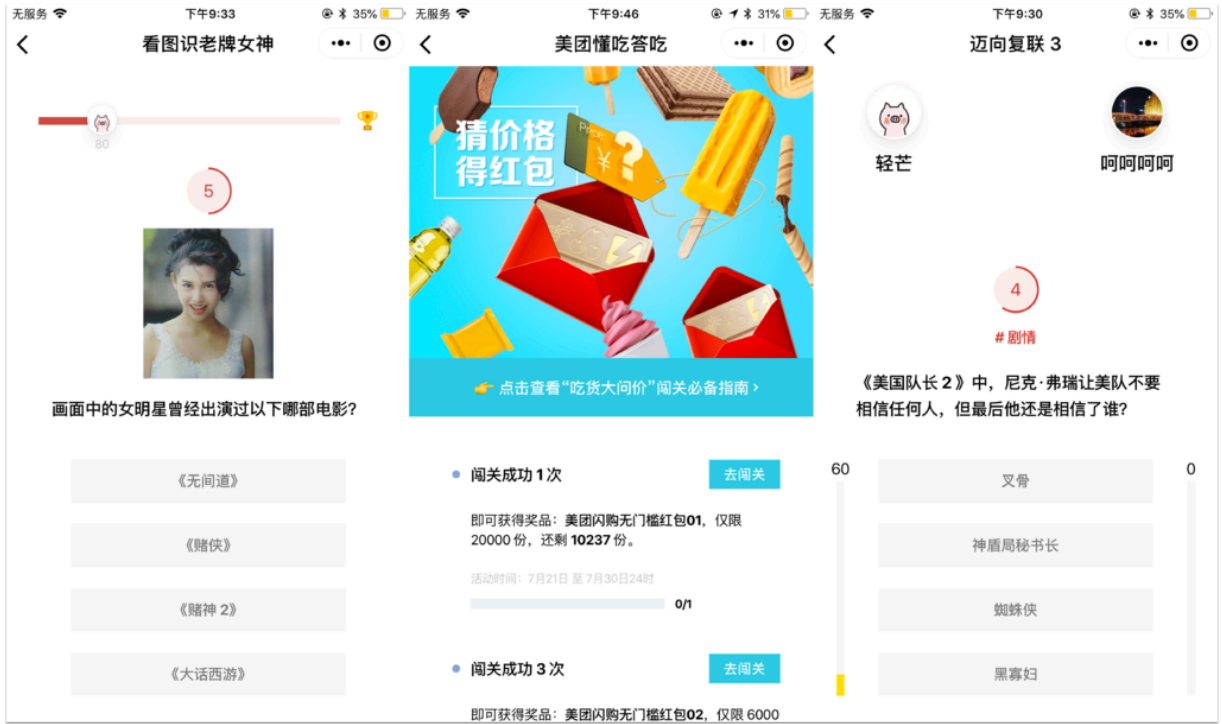
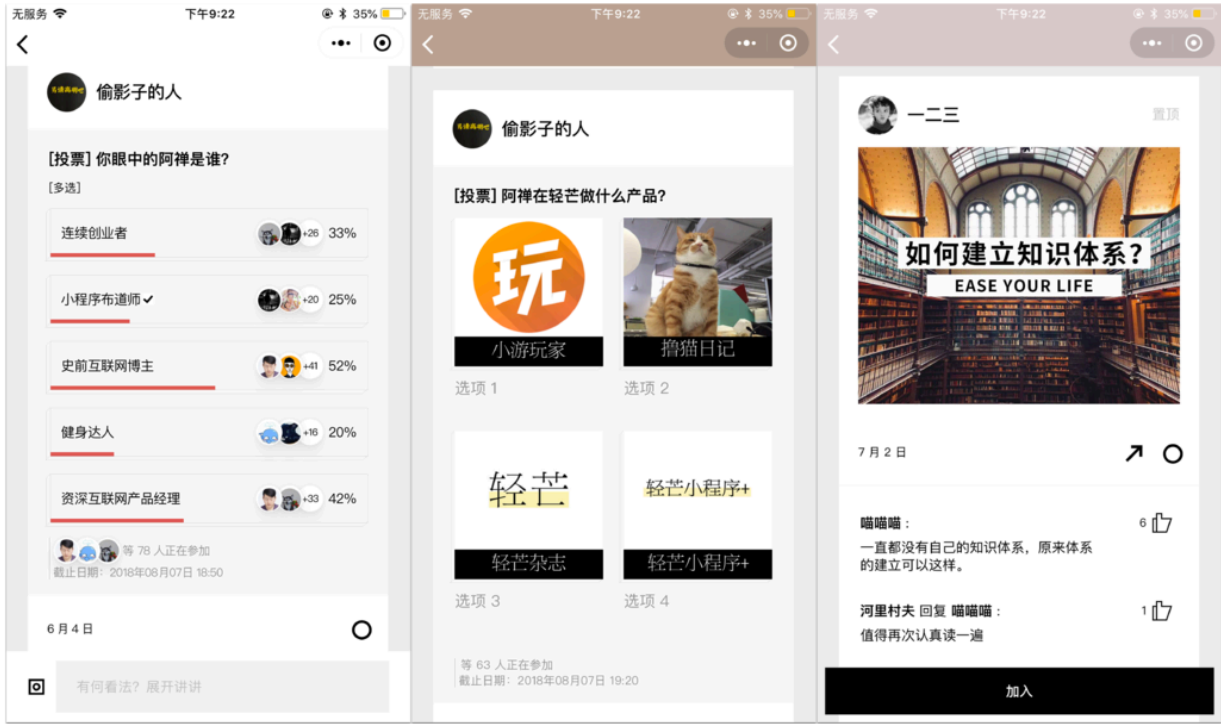
2016 年底作为联合创始人创办轻芒

面向读者——

轻芒杂志

面向内容创作者——

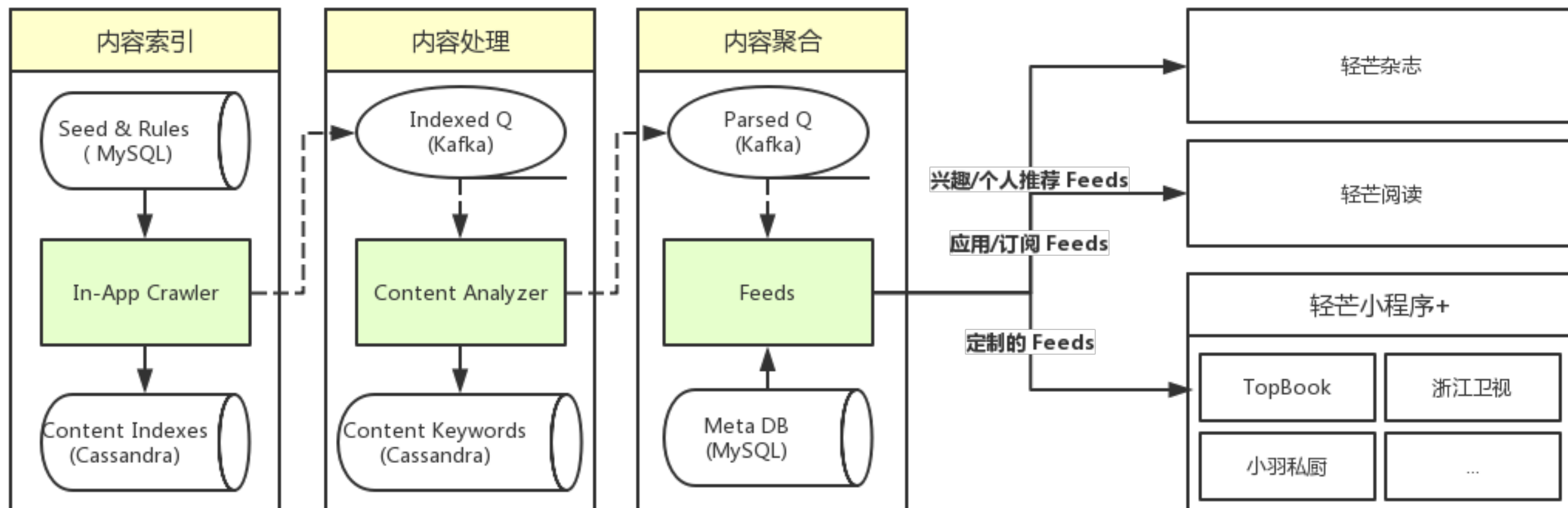
轻芒小程序+



# 提纲

- ◆ 轻芒内容服务的设计目标
- ◆ 基于 Scala + Akka 模型来构建内容服务
- ◆ 新架构带来的挑战和应对

# 设计目标



- ◆ 每天数十万的新内容被检索
- ◆ 需要计算数十万的应用、兴趣内容流
- ◆ 需要为上千万的用户计算内容流
- ◆ 为数万个小程序提供内容支撑



- ◆ 成熟的技术方案倾向于平衡
  - ◆ 在性能、稳定性、可扩展性、研发效率，可维护性等诸多方面相对平衡
- ◆ 最小的人力投入，保持最大的弹性
  - ◆ 研发效率：统一编程模型，减少中间件的种类，和熟悉的技术栈兼容
  - ◆ 弹性：降低代码规模，在用户规模变化下开发模式相对稳定
  - ◆ 性能和稳定性：可以妥协单机性能和稳定性，可以通过扩容来临时解决问题

# 基于 Scala + Akka 的 服务设计

# Scala

- ◆ JVM 方言
  - ◆ 可以无缝和 Java 生态相融合
- ◆ 混合了面向对象和函数式编程的特征
  - ◆ 比之 Java 更为灵活，简练
  - ◆ 过于灵活，学习曲线比较陡峭
- ◆ 常用在数据分析和处理领域
  - ◆ 被 Spark, Kafka, 等大型开源项目所使用
  - ◆ 被 Twitter, LinkedIn 等公司广泛应用



# Akka

- ◆ Akka 是围绕消息机制来构建的分布式开发框架
  - ◆ 对开发者屏蔽掉分布式的诸多细节
- ◆ Akka 是 Actor 模型在 JVM 上的实现
  - ◆ Actor 是分布式系统下的对象，异步通信、全局寻址
- ◆ 可以部署成一个去中心的集群
  - ◆ 同构的部署策略更利于压榨单机性能
- ◆ 改变了传统 Web Server 的编程模型
  - ◆ 以及，改变了服务部署的方式



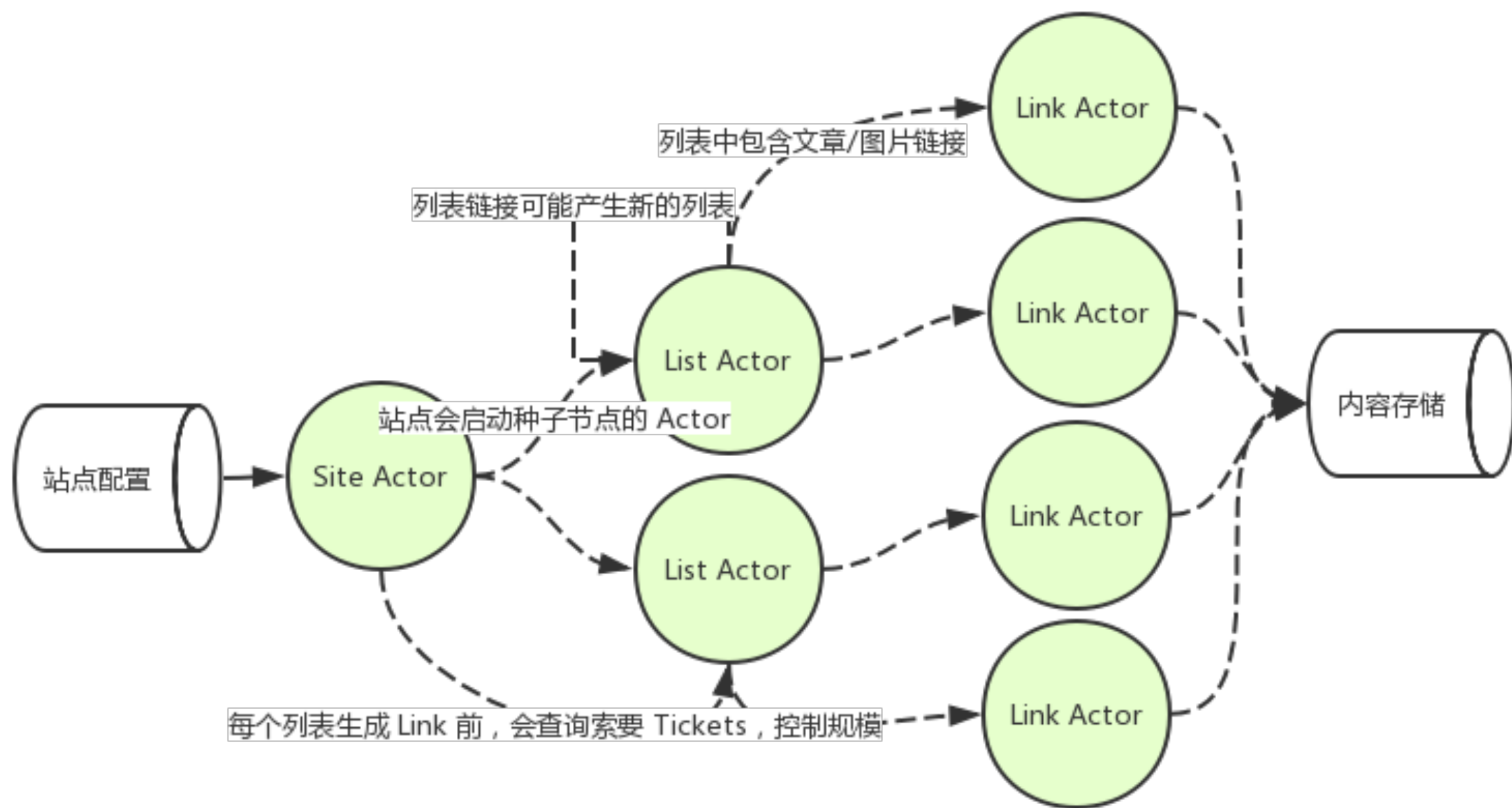
# 面向对象的编程

- ◆ 对象是对实体的抽象
  - ◆ 将数据和操作封装在一起
  - ◆ 让对象之间的关系清晰而明确
- ◆ 面向对象在分布式系统下碰到的问题
  - ◆ 数据同步
  - ◆ 寻址机制是单机化的
  - ◆ 同步的操作无法充分利用系统资源

# 统一的开发模型

- ◆ 定义系统中的 Actor
  - ◆ Actor 是分布式下的对象
  - ◆ 理解业务，对实体进行抽象，规划好 Actor 在分布式系统下的地址
- ◆ 初始化 Actor 中的数据
  - ◆ 初始化时把所需数据加载到内存
  - ◆ 后续操作基本在内存中，数据 + 操作一体化
- ◆ 处理消息
  - ◆ 基于消息驱动的异步模型

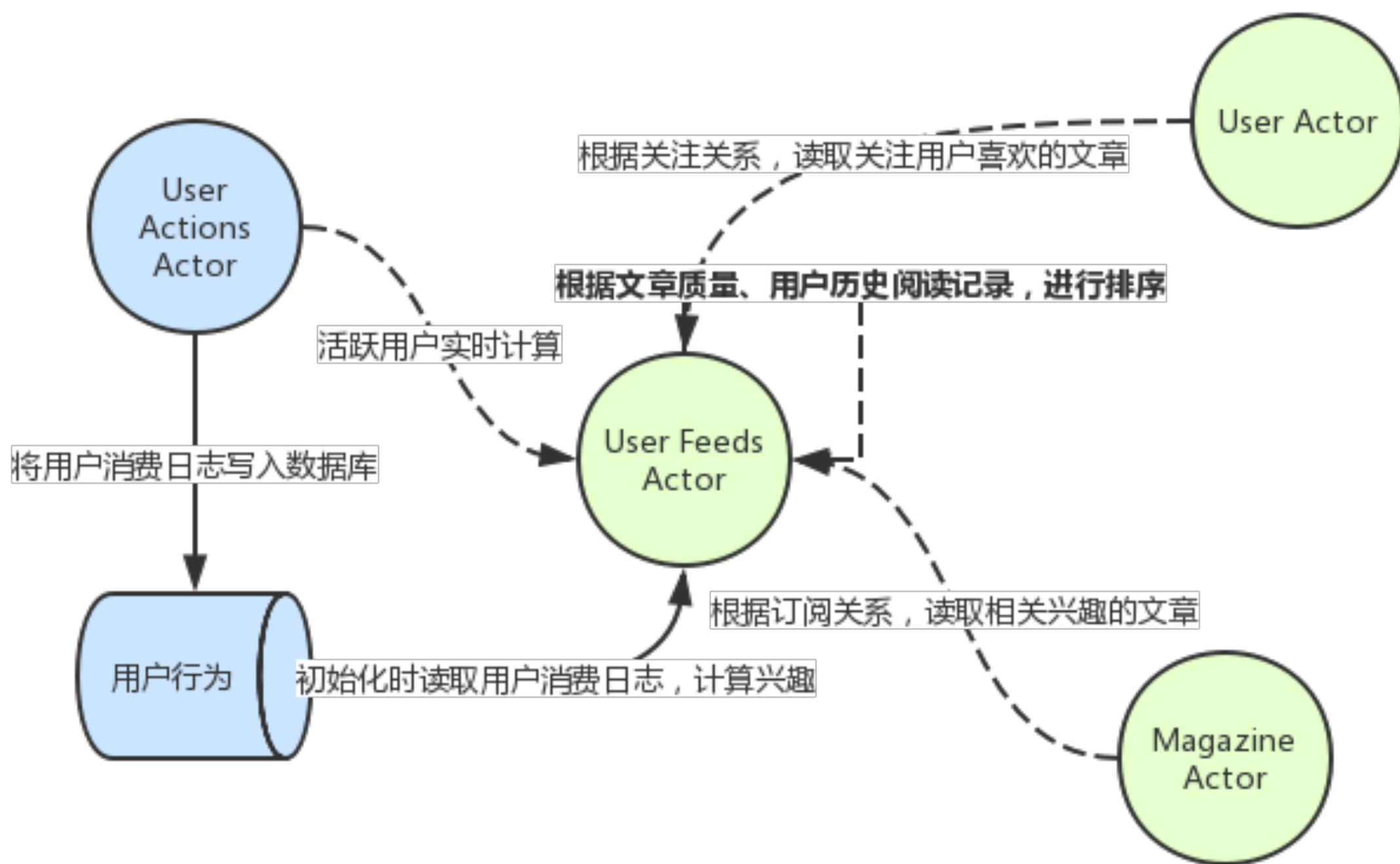
# 轻芒爬虫服务的实现





- ◆ 从流式的架构，变成了对象式的架构
  - ◆ 调度模块，解析模块，索引模块 …
  - ◆ 站点、列表、链接，很容易进行追踪和维护
- ◆ 解除了外部依赖
  - ◆ 没有中间的消息系统、存储服务之类的
  - ◆ 可轻松的单机部署，也可以部署成集群

# 轻芒内容流服务的实现



- ◆ 统一了用户 Feeds 流的实现
  - ◆ 推拉一体
  - ◆ 核心用户的 Feeds 会自然的停留在内存中提升效能
- ◆ 天生具有实时计算的能力
  - ◆ 初始化是从数据库中加载数据
  - ◆ 实时监听用户行为数据
  - ◆ 由于大量关联 Actor 位于内存中，实时计算速度较快

# 优点

- ◆ 编程模式统一
  - ◆ 对 Actor 的设计和抽象是核心
- ◆ 在分布式下的部署和开发变得简单
  - ◆ Akka 承包了很多细节
  - ◆ 全异步更易于分布式
- ◆ 机器会被充分利用
  - ◆ 缓存和计算一体，会充分利用内存和 CPU

# 新架构的挑战和应对

# Actor 的冷启动

- ◆ 冷启动的时长，影响整个系统的响应时长
  - ◆ Actor 的调用关系可能很繁杂，需要做好自己
- ◆ 降低 Actor 的抖动，把重要的 Actor 留在内存中
  - ◆ 根据业务特征来调整存活相关的参数
  - ◆ 扩大集群规模
- ◆ 通过临时缓存来降低再次启动的耗时
  - ◆ 个人 Feeds 会在退出的时候记住没有消费完的内容
  - ◆ 在下次启动的时候优先使用，异步初始化
- ◆ 建立有效的性能监控
  - ◆ 记录各个步骤的日志，汇总到阿里日志服务进行分析

# 系统资源的调度

- ◆ 分治经营

- ◆ 每个 Actor 控制存活时长、占用资源

- ◆ 集中调度

- ◆ 收集系统中全部 Actor 的数量和资源消耗情况
  - ◆ 根据时间、资源等对 Actor 进行调度

- ◆ 动态平衡还是很困难

- ◆ 比较容易导致雪崩
  - ◆ 高质量的实现和监控每个 Actor 是关键



# 部署的挑战

- ◆ 大集群 vs 微服务
  - ◆ Akka 本身就包括了集群的解决方案
  - ◆ 引入微服务会在开发模式上导致一定的不统一
- ◆ 包含全部业务的集群会在运维上带来挑战
  - ◆ 基于 Plan B 的发布和容灾方案
- ◆ 做到任何一个 Actor 都可以无损的重启才可以实现更轻松的发布

# Akka 集群的挑战

- ◆ 更频繁的序列化开销
  - ◆ 屏蔽了部署细节，会有更多的数据会被序列化
  - ◆ 单机性能的降低可以通过集群化来解决
- ◆ 无中心集群的稳定性
  - ◆ 心跳检测需要容忍度，在容忍度下服务可能不可用
  - ◆ 各种阈值的配置需要仔细把控，不然会引起抖动

# 总结

- ◆ 基于 Actor 的编程模型在很多场景下可以统一开发模型
  - ◆ 降低理解和设计的复杂度
  - ◆ 长期看，简化开发带来的价值是更大的
- ◆ 有效减少了中间件的使用，降低为运维复杂度
  - ◆ 但需要更熟练 Akka 集群的运维
- ◆ 新的开发模式，需要新的思维
  - ◆ 对业务的抽象比存储的设计更重要
  - ◆ 优化局部可以带来整体的优化
  - ◆ 围绕新的模式来设计日志和监控机制



加入我们

hello@qingmang.me

# 技术创新的浪潮接踵而来， 继续搬砖还是奋起直追？

云数据

AI

区块链

架构优化

高效运维

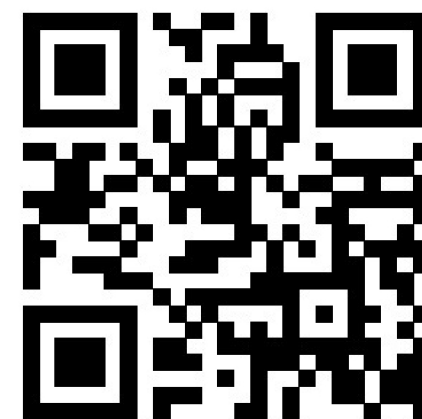
CTO技术选型

微服务

新开源框架

会议：2018年12月07-08日 培训：2018年12月09-10日

地址：北京·国际会议中心





# 极客时间VIP年卡

每天6元, 365天畅看全部技术实战课程

- 20余类硬技能, 培养多岗多能的混合型人才
- 全方位拆解业务实战案例, 快速提升开发效率
- 碎片化时间学习, 不占用大量工作、培训时间



Q & A