QCon[上海站]
全球软件开发大会2016

映客直播
服务端架构优化之路

SPEAKER

王振涛

International Software
Development Conference

主办方 Geekbang 极客邦科技 InfoQ

目录

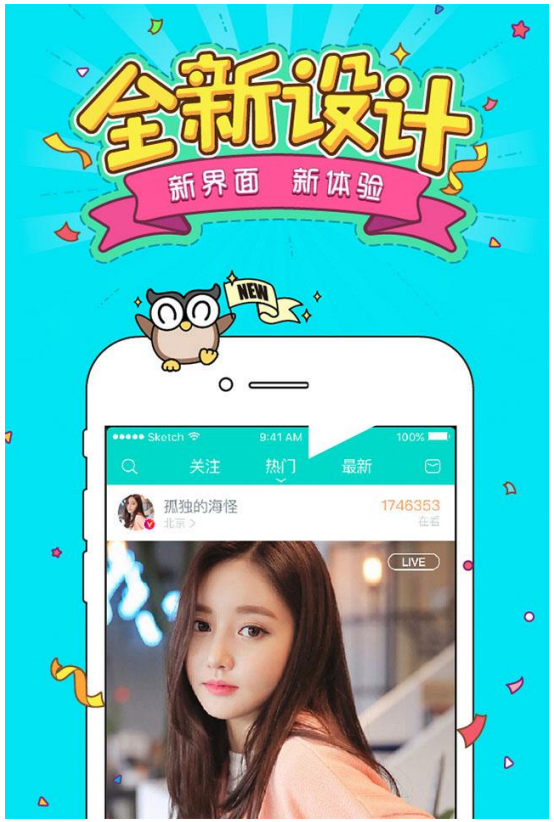
- 映客直播发展历程
- 服务端整体架构演进
- “洪荒之力”引发的技术变迁
- 业务挑战vs系统稳定性
- 结束语

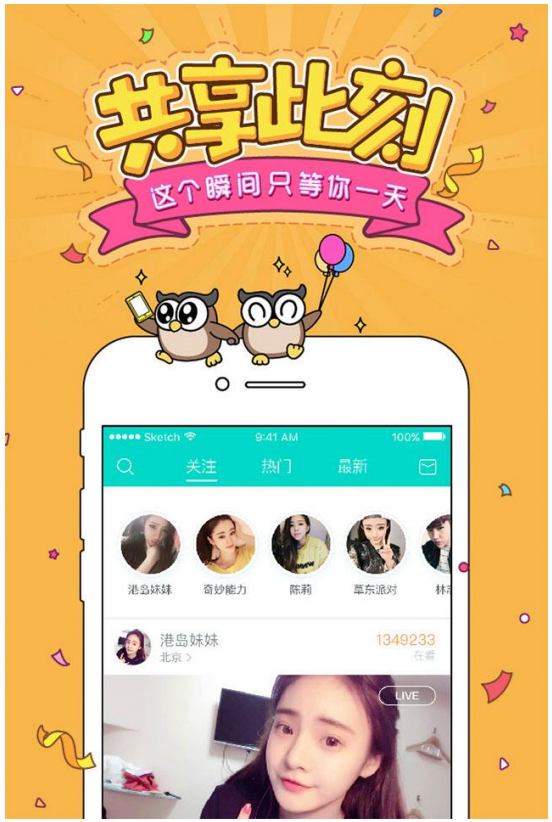
目录

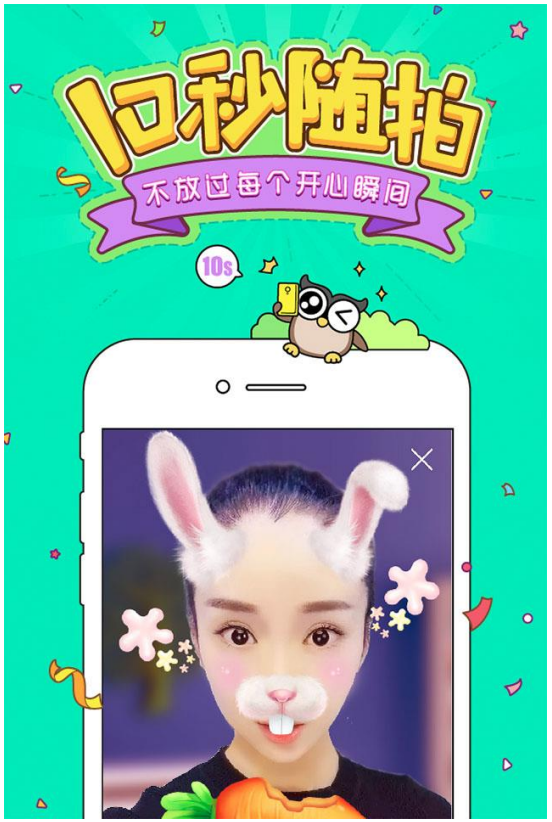
映客直播发展历程


下午3:15
28.8K/s
41%
关注
热门
附近
吕洲
小映读书
STUDENT
爱心接力
10月20日 20:00
郑湫泓JadeCheng
北京市
14913
在看
直播

映客直播发展历程







扩 展 中

映客直播发展历程

- 2015.05　映客直播APP发布
DAU 200
- 2015.10　DAU 10w+
- 2015.12　DAU 100w+
- 在持续爆发性增长

目录

- 映客直播发展历程
- 服务端整体架构演进
- 业务场景引发技术变迁
- 业务挑战vs系统稳定性
- 结束语

服务端整体架构演进

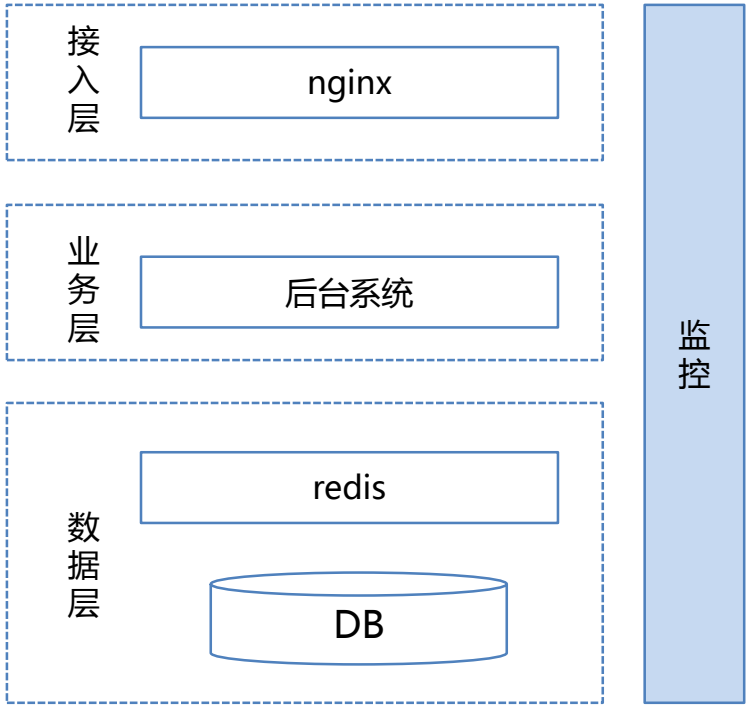
业务起步：映客诞生

- ✓ IDC：8台虚拟机
  - ✓ 服务6台：接入层、业务服务、基础服务、长连接服务
  - ✓ 存储2台：mysql/redis
- ✓ 服务
  - ✓ 接入层：nginx
  - ✓ 业务服务：node.js
  - ✓ 基础服务：nginx/c++、tornado/python
  - ✓ 长连接服务：socket.io
- ✓ 存储
  - ✓ Mysql：master/slave，单实例，多个业务库
  - ✓ Redis：master/slave，4个实例，3个业务1个MQ
  - ✓ Mysql和redis共用2台虚拟机
- ✓ 人力
  - ✓ 开发2人，运维1人

服务端整体架构演进

技术架构：1.0

- ✓ 快速迭代
- ✓ 服务稳定
- ✓ 成本节约



服务端整体架构演进

业务增长：规模化

- ✓ 日活：10w-100w+
- ✓ 流量爆发式增长
- ✓ 与时间赛跑

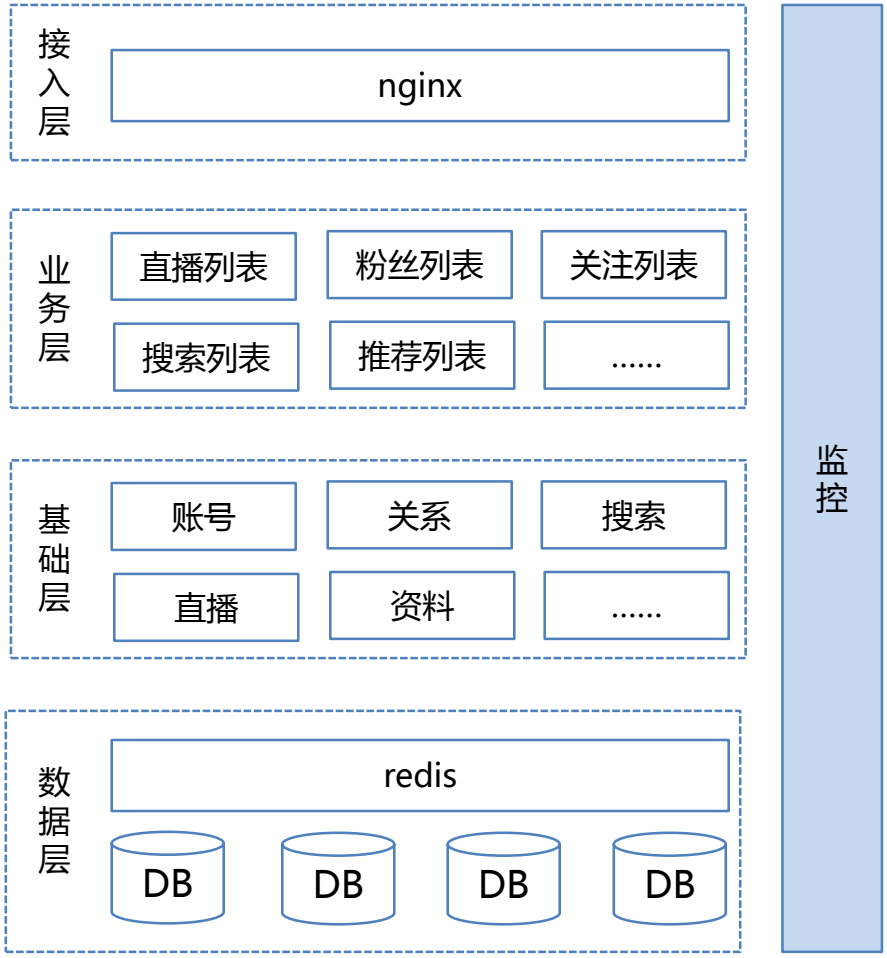
带来的问题

- ✓ 业务混部，复杂度越来越高
- ✓ 数据耦合，性能相互影响
- ✓ 重复开发，成本高
- ✓ 服务扩容难，性能跟不上

服务端整体架构演进

技术架构：2.0

- ✓ 业务拆分
- ✓ 数据拆分
- ✓ 模块化复用
- ✓ 多级cache
- ✓ 服务上云



服务端整体架构演进

服务上云：初体验

- ✓ Ddos
  - ✓ 云主机免费防DDos
  - ✓ 代理层部署到云
- ✓ 明星活动，瞬时压力大
  - ✓ 云主机按量付费
  - ✓ 长连接层部署到云

服务上云：整体迁移

- ✓ 存储
  - ✓ 同步
  - ✓ 切换
- ✓ 服务
  - ✓ 隔离
  - ✓ 镜像

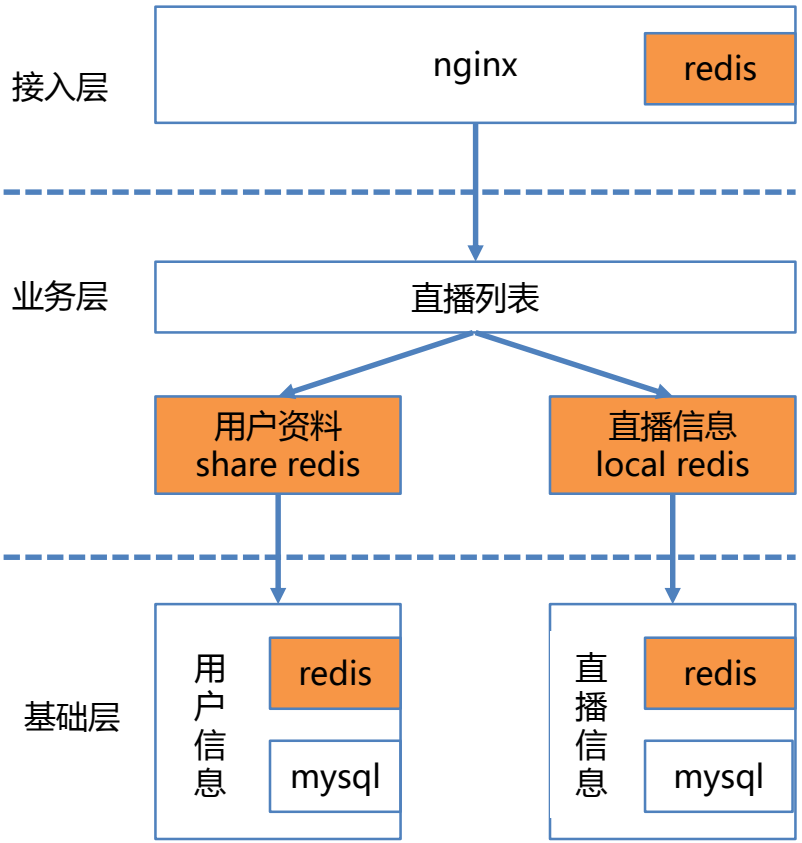
服务上云：水土不服

- ✓ mysql连接数限制
  - ✓ 上限2000，多进程服务框架
- ✓ 外网带宽限制
  - ✓ 上限200M，长连接消息广播跑满带宽
- ✓ 内网带宽限制
  - ✓ 上限500M，长连接消息转发服务跑满带宽
- ✓ 云主机稳定性
  - ✓ 资源抢占，宕机

服务端整体架构演进

多级cache：抗流量

- ✓ 直播/用户信息实时变化
- ✓ 高峰瞬时流量大
- ✓ 活动扩服务器扩带宽



服务端整体架构演进

业务持续发展：爆发

- ✓ 日活迈向千万级别
- ✓ 产品迭代快
- ✓ 运营活动不断

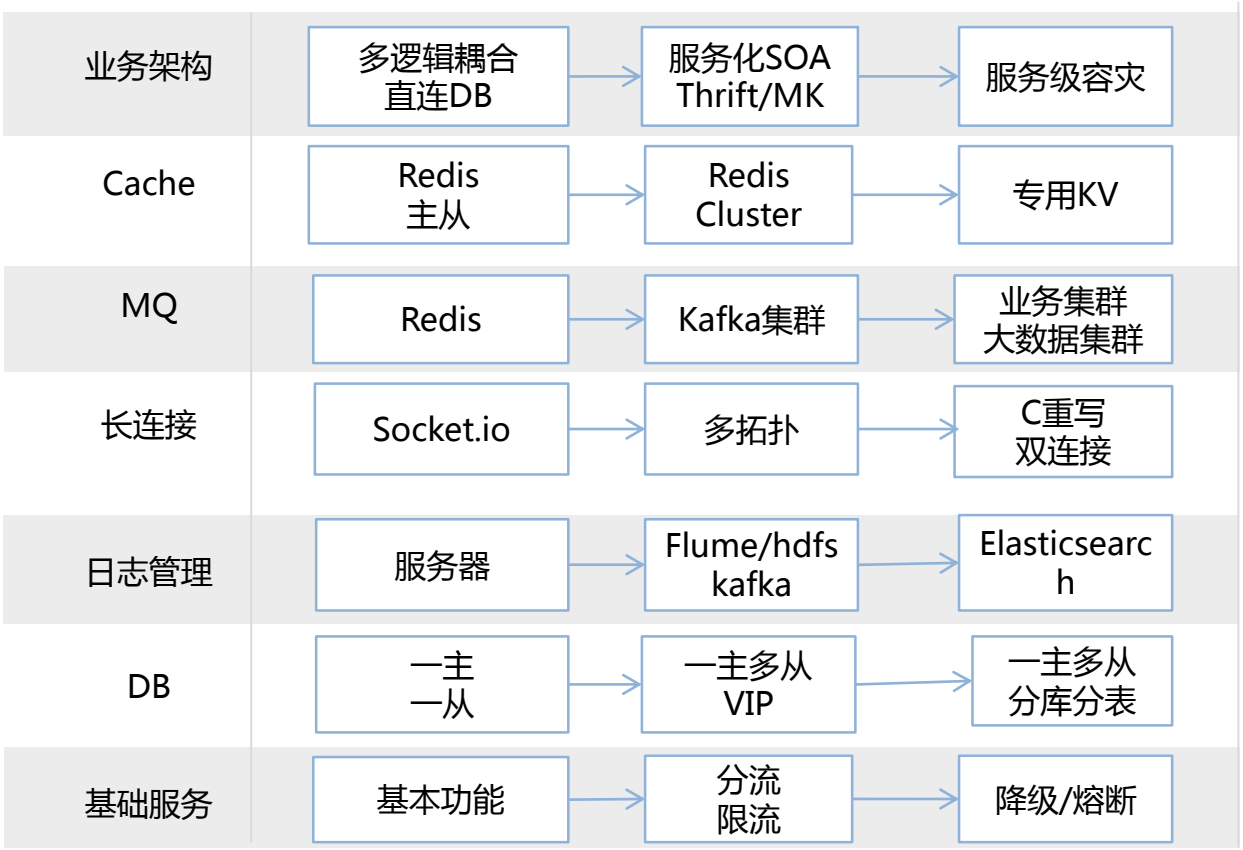
带来的问题

- ✓ 业务间依赖，相互影响
- ✓ 业务和数据耦合，存储无法扩展
- ✓ 产品需求叠加，业务系统越来越复杂
- ✓ 性能不稳定，扩容难
- ✓ 运维困难

服务端整体架构演进

持续演进

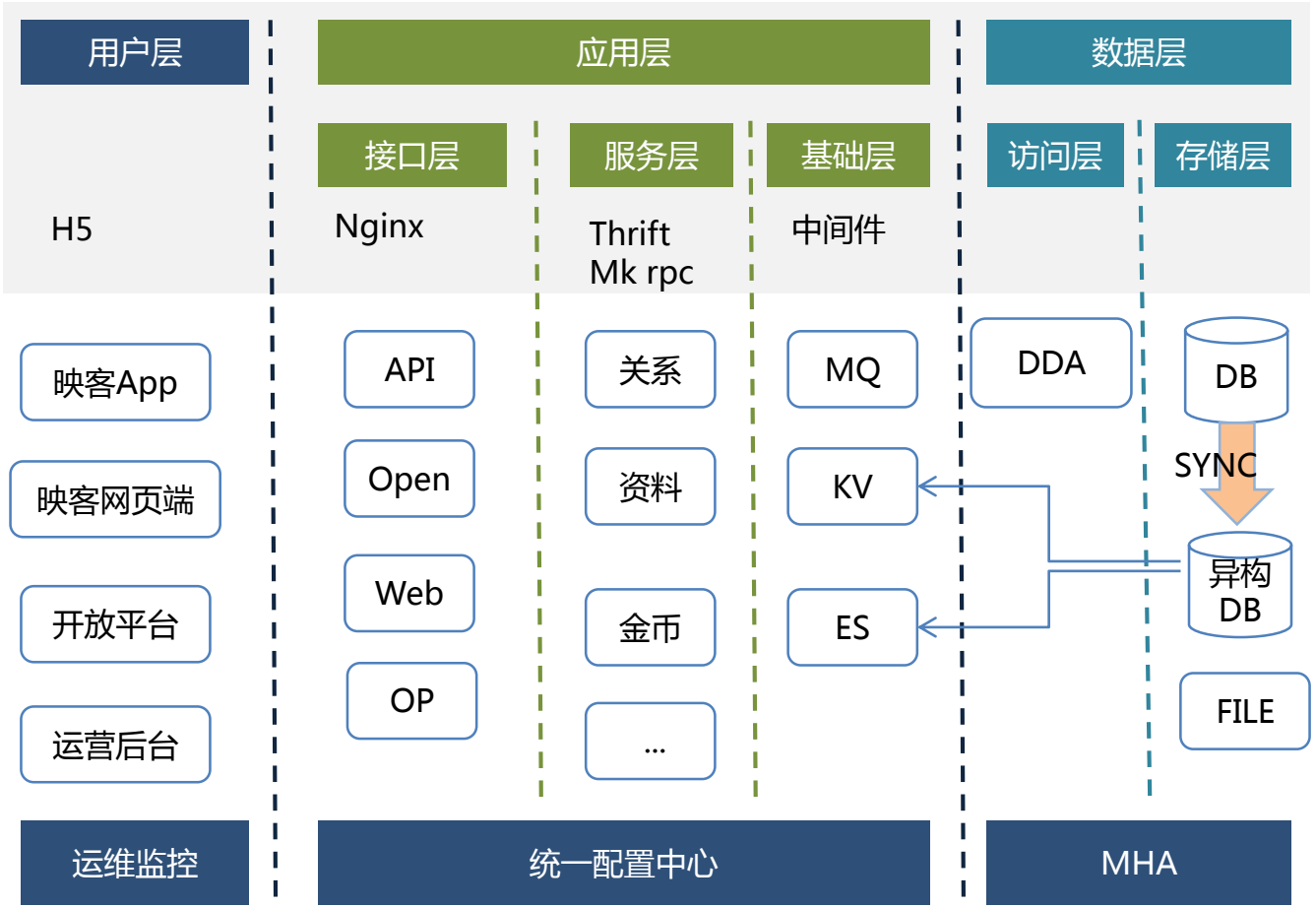
- ✓ 服务化
- ✓ 存储优化
- ✓ 中间件
- ✓ KV/MQ
- ✓ 长连接



服务端整体架构演进

技术架构：3.0

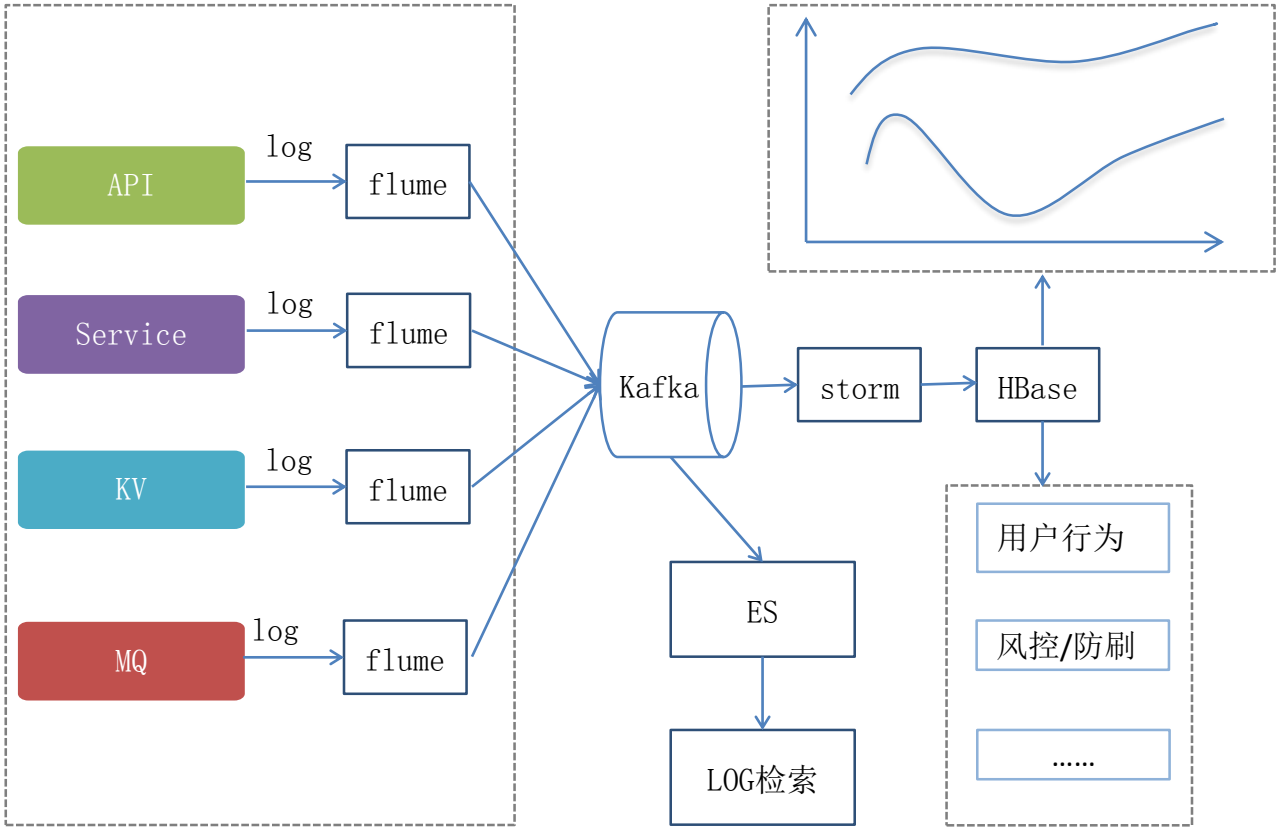
- ✓ 服务化重构
  - ✓ 解耦和
  - ✓ 存储和计算隔离
  - ✓ 服务限流/降级
  - ✓ 可伸缩部署
- ✓ 系统级容错
  - ✓ 高可用设计
  - ✓ 独立部署
- ✓ 存储优化
  - ✓ 分库分表
  - ✓ 分布式事务框架
  - ✓ 异步优化
- ✓ 中间件
- ✓ 长连接重构
- ✓ Jenkins自动化部署
- ✓ 完善监控体系



服务端整体架构演进

建立完善的监控体系

- ✓ 性能监控
  - ✓ 服务器状态
  - ✓ 服务性能
    - ✓ 接口QPS
    - ✓ TOPN响应时间
  - ✓ DB/KV/MQ状态
    - ✓ 读写QPS
    - ✓ 慢查询/死锁
    - ✓ Cache命中率
- ✓ 业务监控
  - ✓ 业务可用性
  - ✓ 全链路准实时巡检
- ✓ 日志分析
  - ✓ error/warning
  - ✓ 大数据行为分析
- ✓ 健康度分析
  - ✓ 同比/环比/趋势



目录

“洪荒之力”引发的技术变迁

傅园慧直播

- ✓ 奥运热点事件
- ✓ 瞬时集聚人气
- ✓ 房间极其活跃

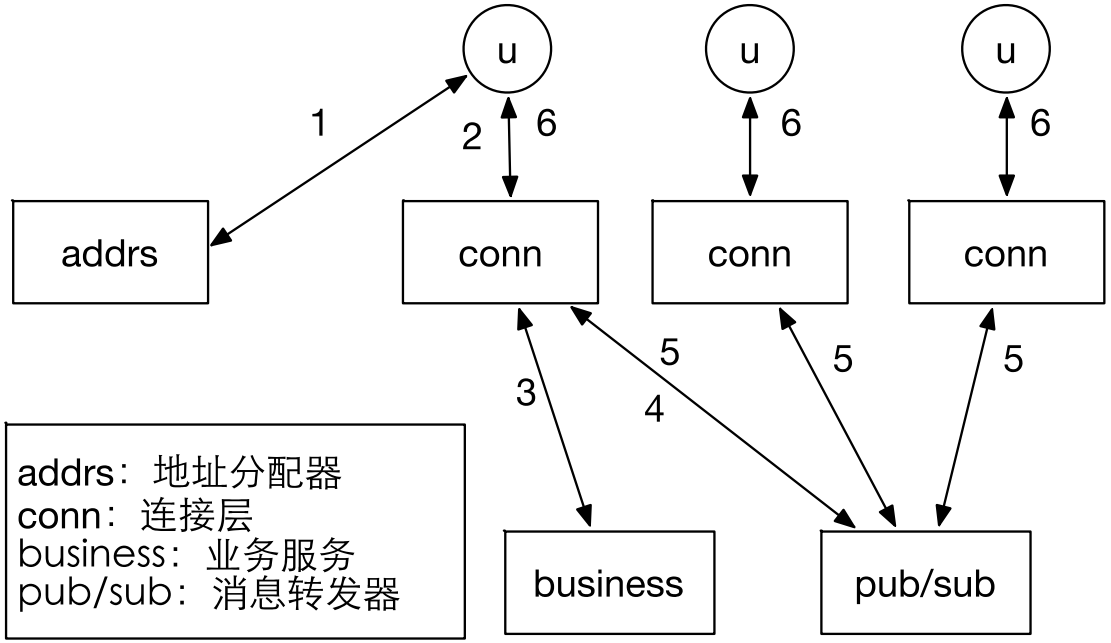
技术角度

- ✓ 长连接服务
  - ✓ 消息推送
  - ✓ 广播

“洪荒之力”引发的技术变迁

长连接服务：1.0

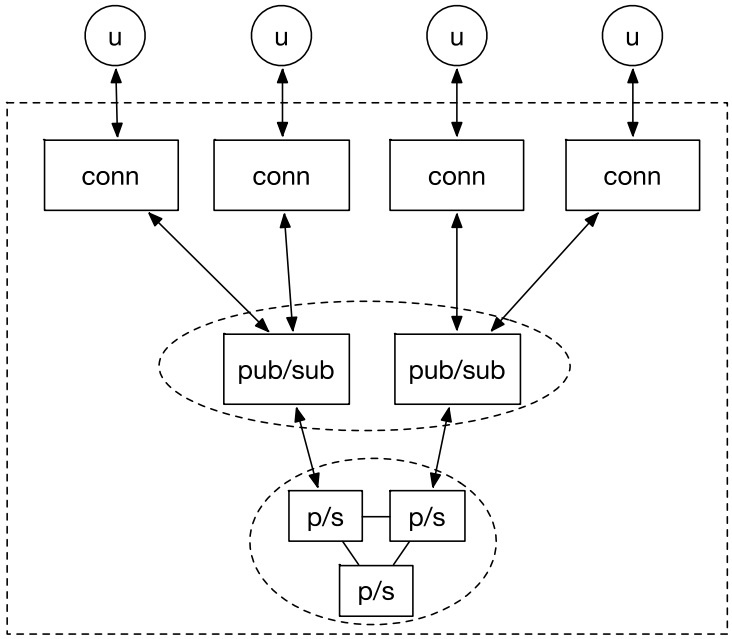
- ✓ 地址分配器
  - ✓ 分配连接层服务地址
- ✓ 连接层
  - ✓ 维护用户连接及用户状态
- ✓ 业务服务
  - ✓ 业务逻辑处理
- ✓ 消息订阅/广播服务
  - ✓ 消息转发

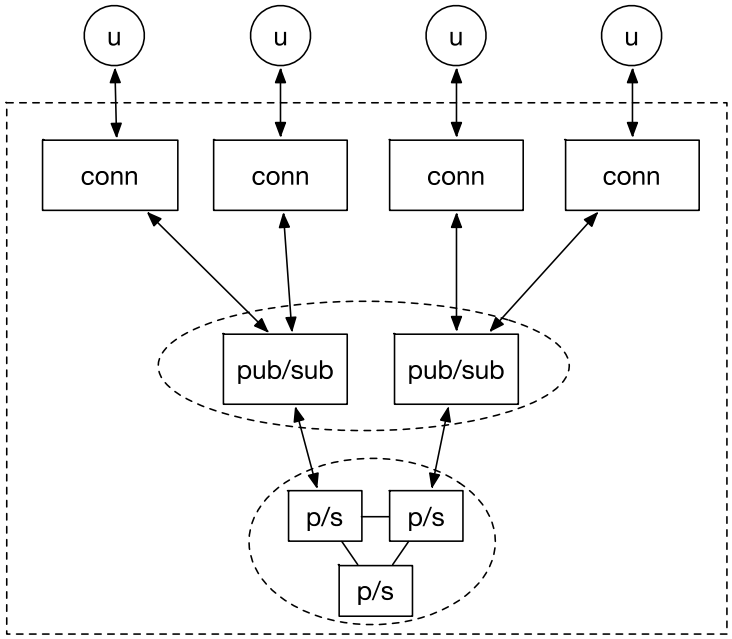


“洪荒之力”引发的技术变迁

长连接服务：2.0

- ✓ 订阅服务分层
- ✓ 单拓扑升级多拓扑

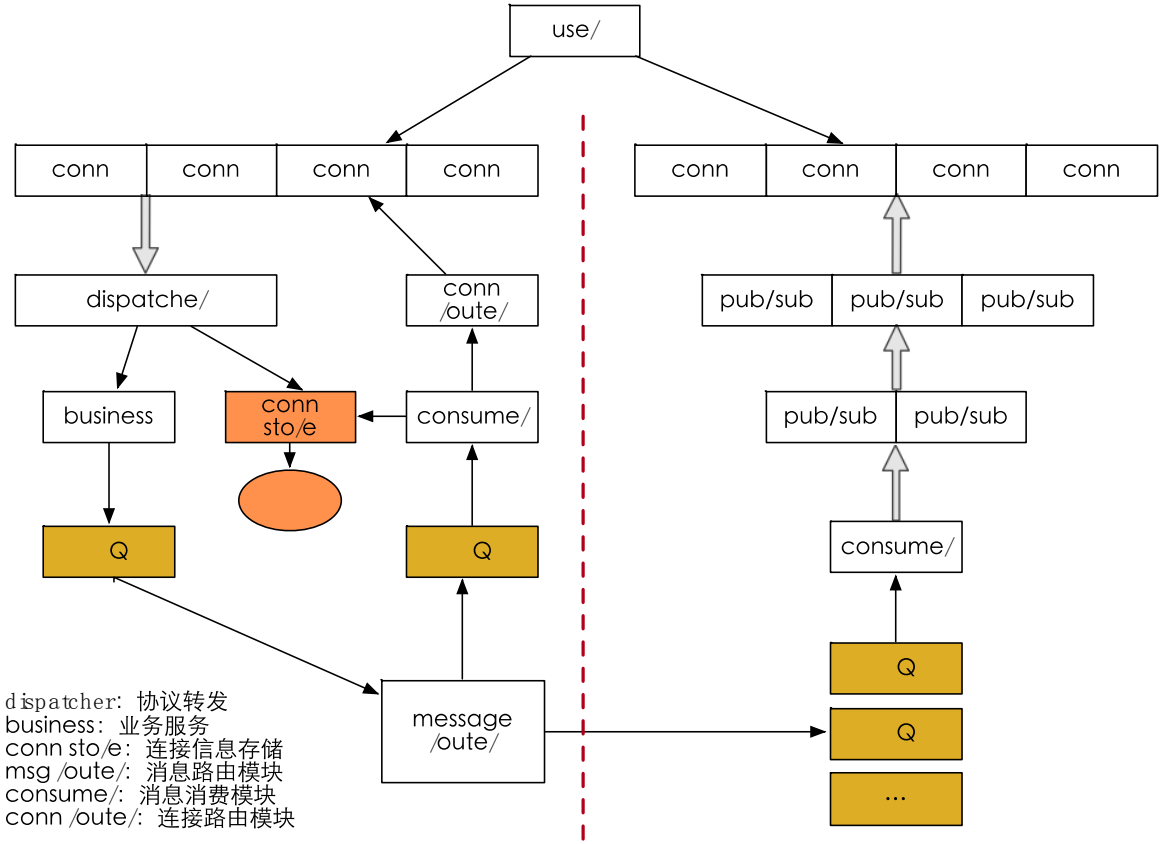




“洪荒之力”引发的技术变迁

长连接服务：3.0

- ✓ 双连接，上下行数据分离
- ✓ C重构替换node.js



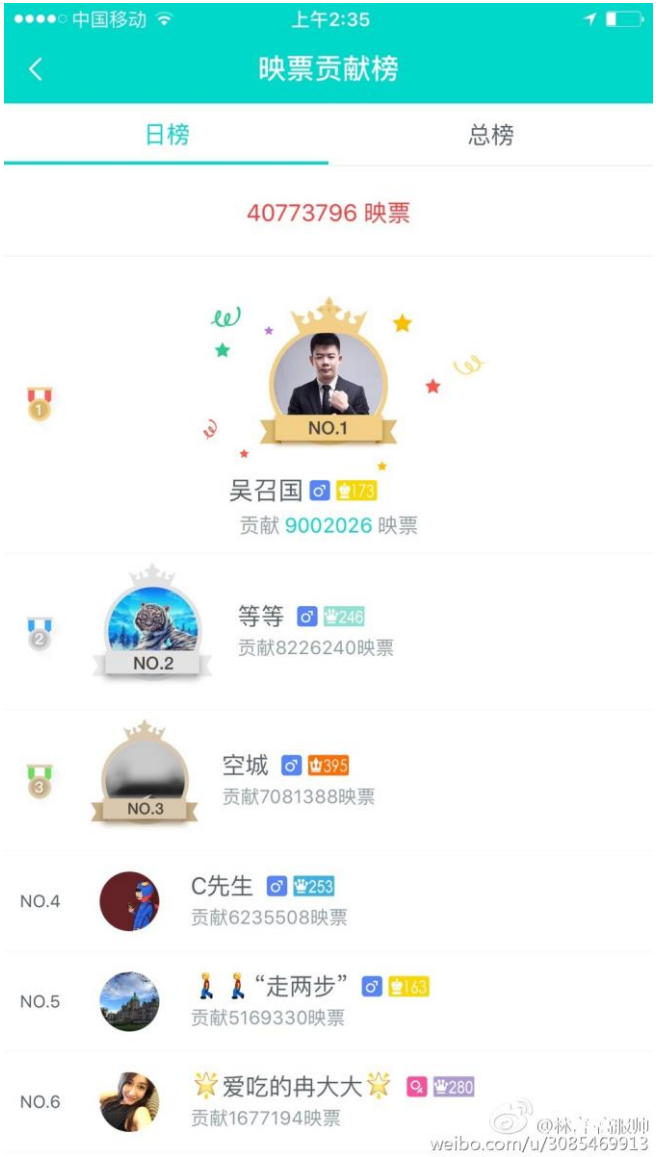
“洪荒之力”引发的技术变迁

主播林言

- ✓ 一天收获40773796个映票
- ✓ 小礼物连送，1个小礼物1映票
- ✓ 粉丝刷不停

技术角度

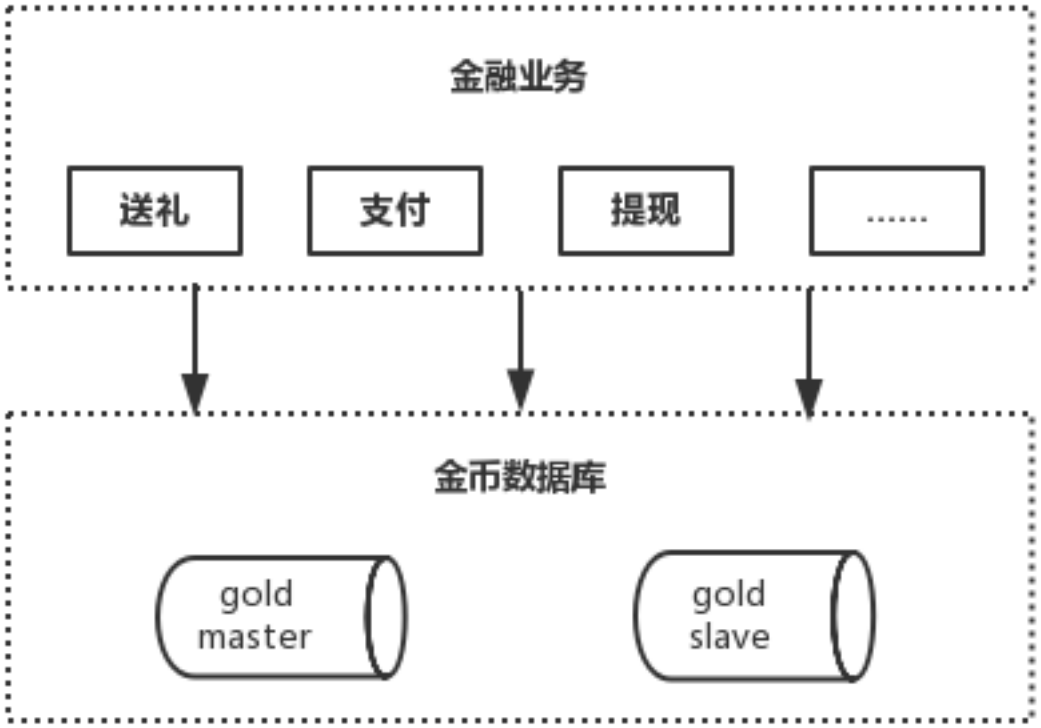
- ✓ 金融相关系统
  - ✓ 高并发
  - ✓ 强一致性
  - ✓ 实时性



“洪荒之力”引发的技术变迁

金币系统：1.0

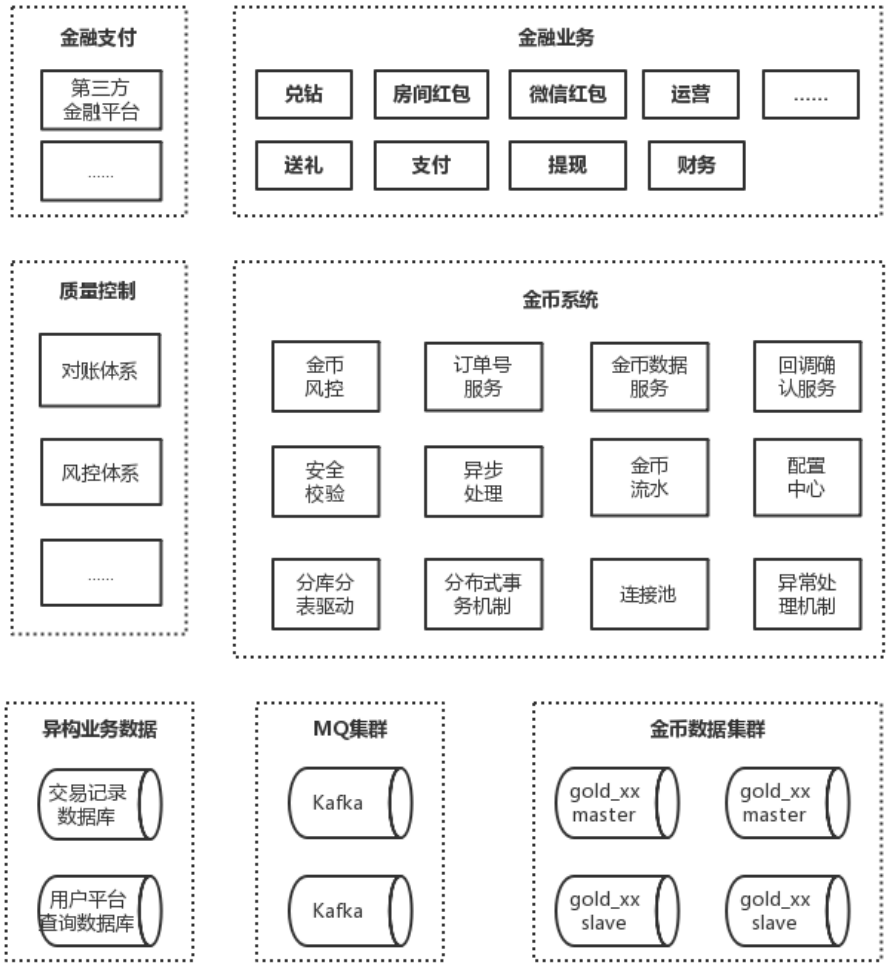
- ✓ 业务离散
- ✓ 直接操作金币库



“洪荒之力”引发的技术变迁

金币系统：2.0

- ✓ 访问入口收拢
- ✓ 数据权限控制
- ✓ 存储优化
  - ✓ 分库分表
  - ✓ 异步处理
  - ✓ 分布式事务
- ✓ 金币流水统一规范
- ✓ 建立风控体系
  - ✓ 事后、事中、事前



“洪荒之力”引发的技术变迁

主播社会你球姐

- ✓ 自带人气
- ✓ 粉丝数多

技术角度

- ✓ 关系服务
  - ✓ 高并发
  - ✓ 大容量
  - ✓ 实时性
  - ✓ 一致性

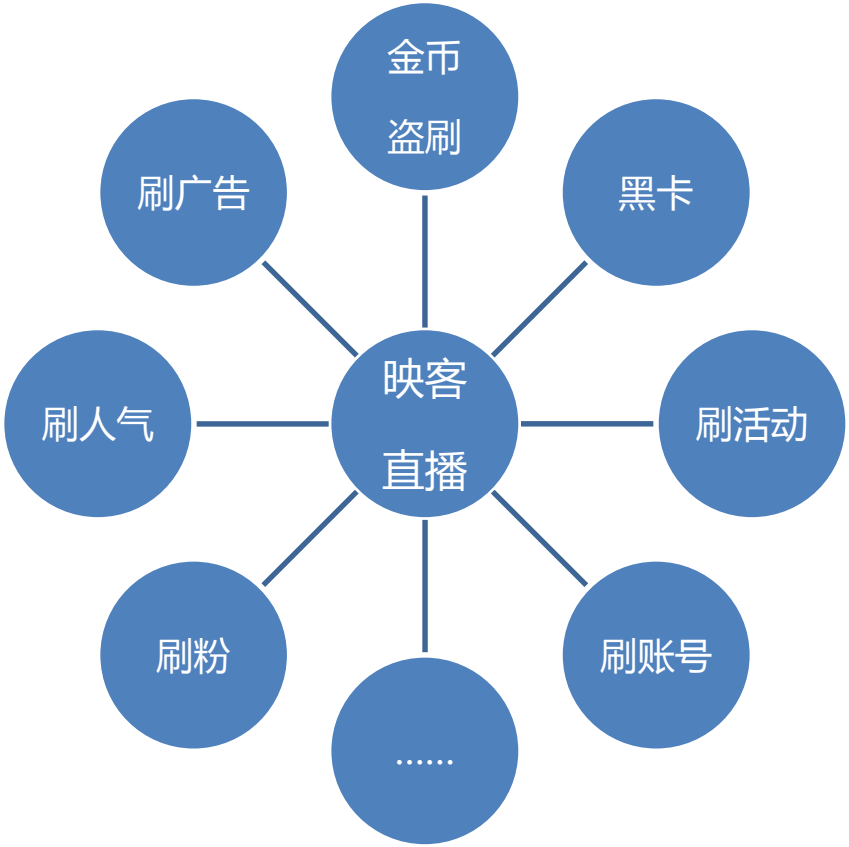
“洪荒之力”引发的技术变迁

关系服务技术优化点

- ✓ 微服务化
  - ✓ 收拢访问入口
  - ✓ 数据与业务隔离
  - ✓ 核心服务独立部署
  - ✓ 服务熔断/降级机制
  - ✓ 可平滑扩展
- ✓ 存储优化
  - ✓ 读写分离
  - ✓ 写异步化
  - ✓ 数据拆分
    - ✓ 分库分表
    - ✓ 在线离线存储分离
    - ✓ 大V做key拆分
- ✓ 推拉结合

“洪荒之力”引发的技术变迁

流量大了，刷子来了



“洪荒之力”引发的技术变迁

防刷

- ✓ 金融防刷
  - ✓ 异常识别：虚假手机号、越狱机
  - ✓ 大数据分析：金币流水日志、用户行为
- ✓ 账号防刷
  - ✓ 模拟器识别
  - ✓ 接口加固
- ✓ 广告/敏感词防刷
  - ✓ 模型 + 规则
  - ✓ 用户行为分析
  - ✓ 审核联动

目录

业务挑战

发展快，产品版本迭代快，流量持续爆发性增长

版本频繁	•引入BUG，稳定性风险 •需求并行，版本兼容问题
项目周期短	•架构优化排不上期 •技术欠债
项目流程不规范	•信息不对称 •沟通协调成本大
新人多	•业务新同学不了解业务细节 •技术新同学不了解实现细节
监控难度大	•指标覆盖不全 •规则变化快

系统稳定性

稳定性度量：可用性（APP->系统->服务->接口）

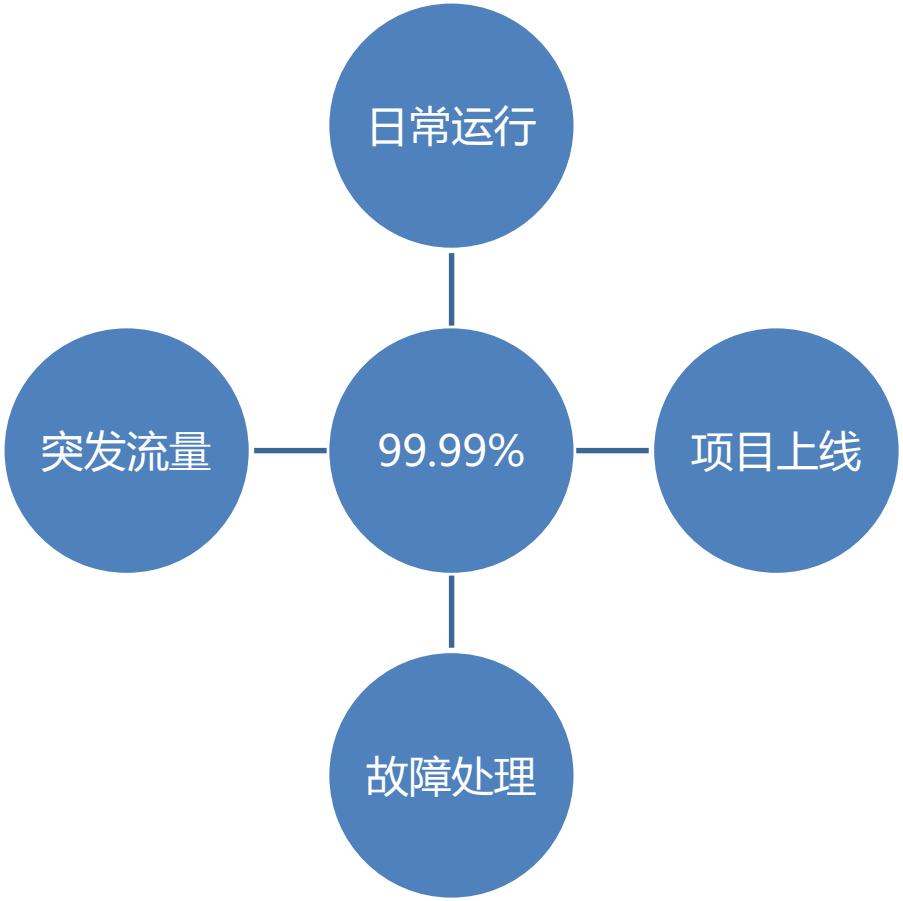
可用性级别	指标	不可用时间/年	不可用时间/天
弱可用	90%	36.5天	2.4小时
基本可用	99%	3.65天	14分钟
较高可用	99.9%	8.76小时	86秒
高可用	99.99%	52.6分钟	8.6秒
极高可用	99.999%	5.25分钟	0.86秒
强高可用	99.9999%	31.5秒	8.6毫秒

业务挑战VS系统稳定性

面对多重的业务挑战，如何保证系统的稳定性

业务挑战VS系统稳定性

面对多重的业务挑战，如何保证系统的稳定性



业务挑战VS系统稳定性

日常运行

- ✓ 完善的监控体系，及时响应
- ✓ 定期业务梳理，全链路巡检
- ✓ 建立业务和人员备份

业务上线

- ✓ 自动化测试
- ✓ 关键代码review机制
- ✓ 灰度/AB Test
- ✓ 验证checklist流程化

故障处理

- ✓ 及时止损
- ✓ 保护用户体验
- ✓ 力保关键路径
- ✓ 故障总结

突发流量

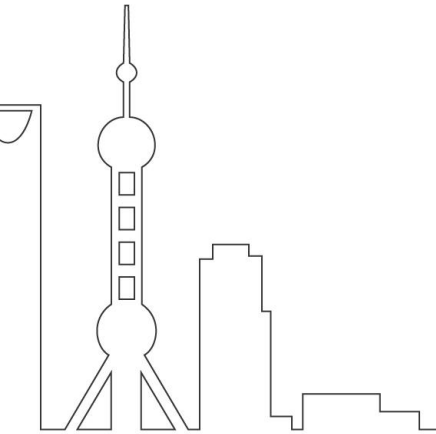
- ✓ 备好不同场景预案
- ✓ 服务限流/降级
- ✓ 根本原因分析

目录

结束语

- 保持对技术的敏感度
- 技术需要理解业务
- 在路上

在找技术牛er，有兴趣请扫码



Thanks!

International Software Development Conference

主办方 Geekbang. 极客邦科技 InfoQ