QCon

www.qconferences.com

Brought by InfoQ

美拍后端技术演进

<九个月实现破亿用户的可扩展架构>

美图 洪小军

@XiaoJunHong

Brought by InfoQ

产品介绍

- 国内最火和发展速度最快的短视频社区

2014年05月： IOS版正式上线

2014年06月： Android版上线

2015年01月： 用户总数突破1亿

技术选型

- 团队熟悉什么？团队做过什么？

- 更多的云服务可供选择
  - CDN、存储、推送、通讯……

- 更多的成熟技术可供选择
  - Mysql、Memcached、Redis、Kafka……

- 服务器具有更强大的性能
  - 有更高的硬件配置

大纲

- 快速发布上线阶段（2014.5 – 2014.7）
  - 极简化设计

- 产品快速迭代阶段（2014.8 – 2014.11）
  - 保持简单性设计

- 美拍社交化发展阶段（2014.12 – 2015.3）
  - 可扩展和可用性保证

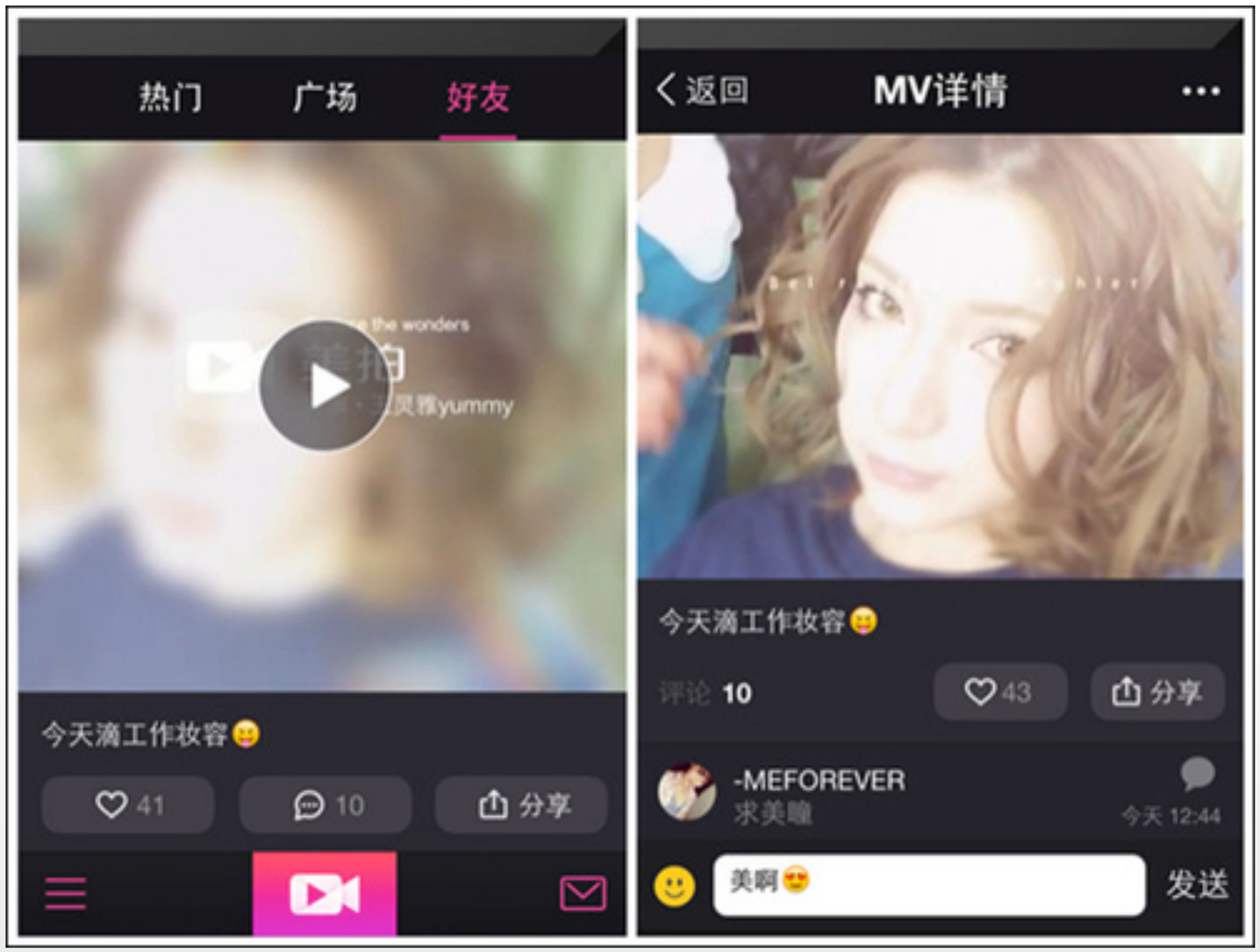
- 现阶段（2015.4 - ）
  - 高可扩展和高可用性保证

极简化设计阶段

- 快速验证产品可行性
- 技术选型：
  - 基于做过的“美陌”项目演变
  - 基于成熟的LAMP技术体系
  - 使用CDN和云存储服务
- 第一版：1位开发研发1周

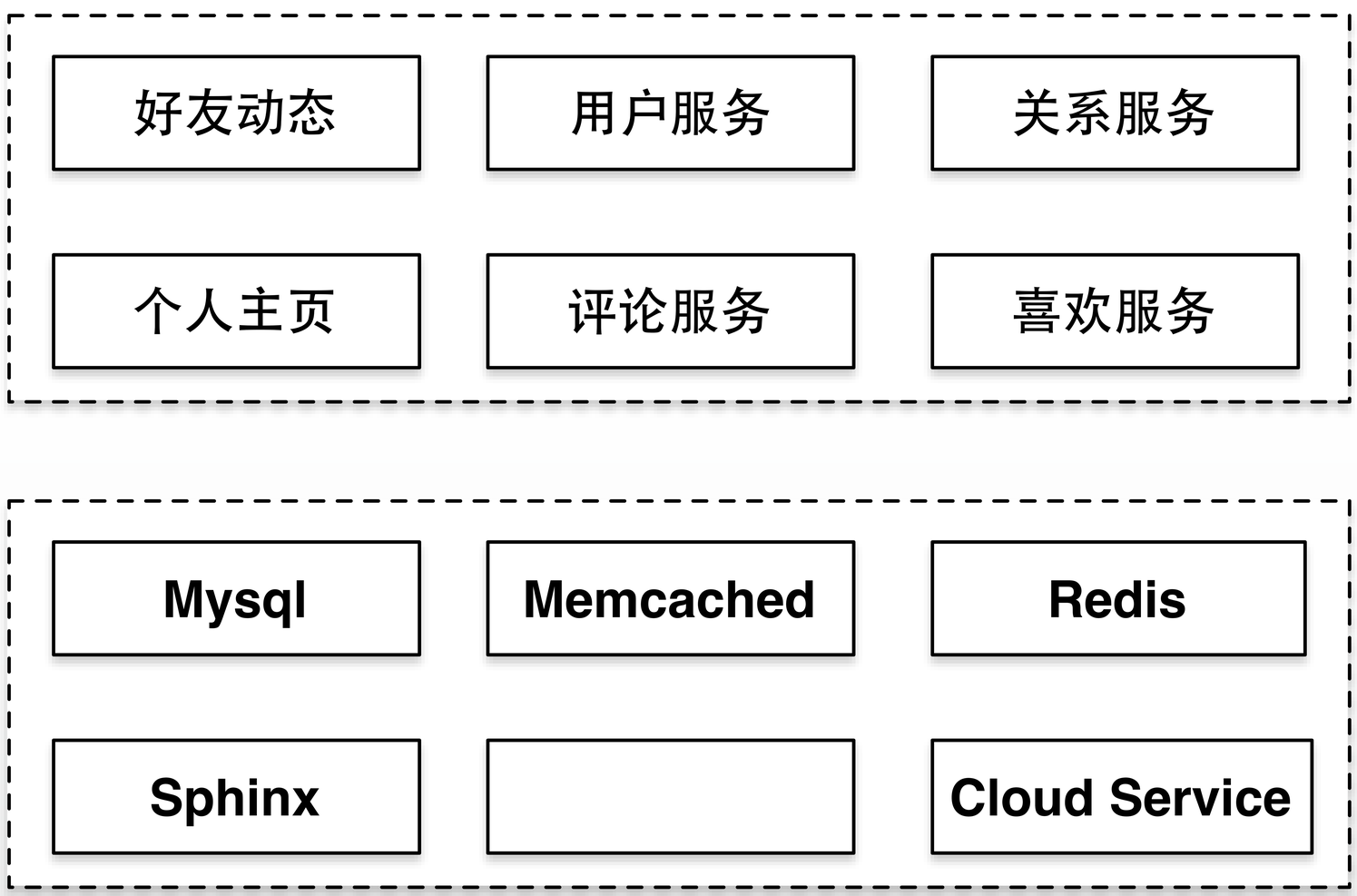
极简化设计阶段

- 产品功能



极简化设计阶段

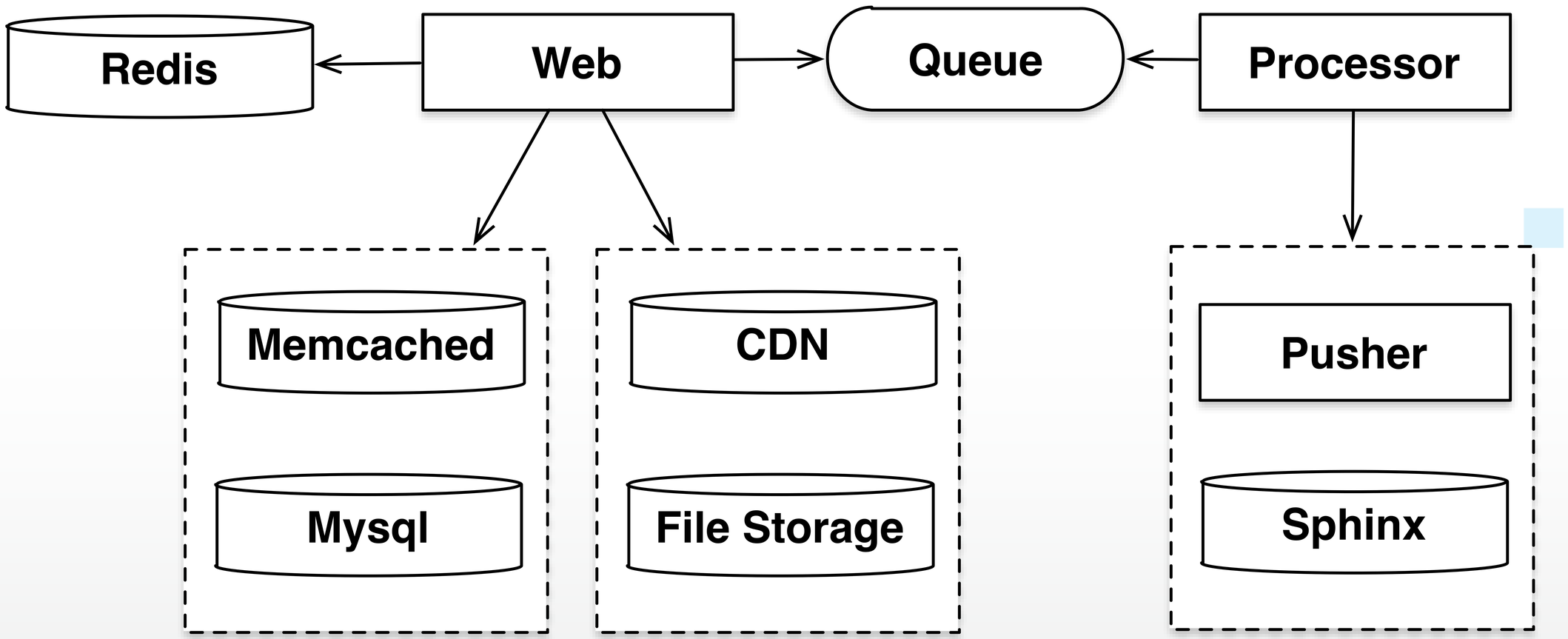
- 系统概况（单节点部署简单化设计）



- 缓存：Memcached
- 存储：Mysql
- 计数器：Redis
- 队列：Redis
- 搜索：Sphinx
- 图片/视频：云服务

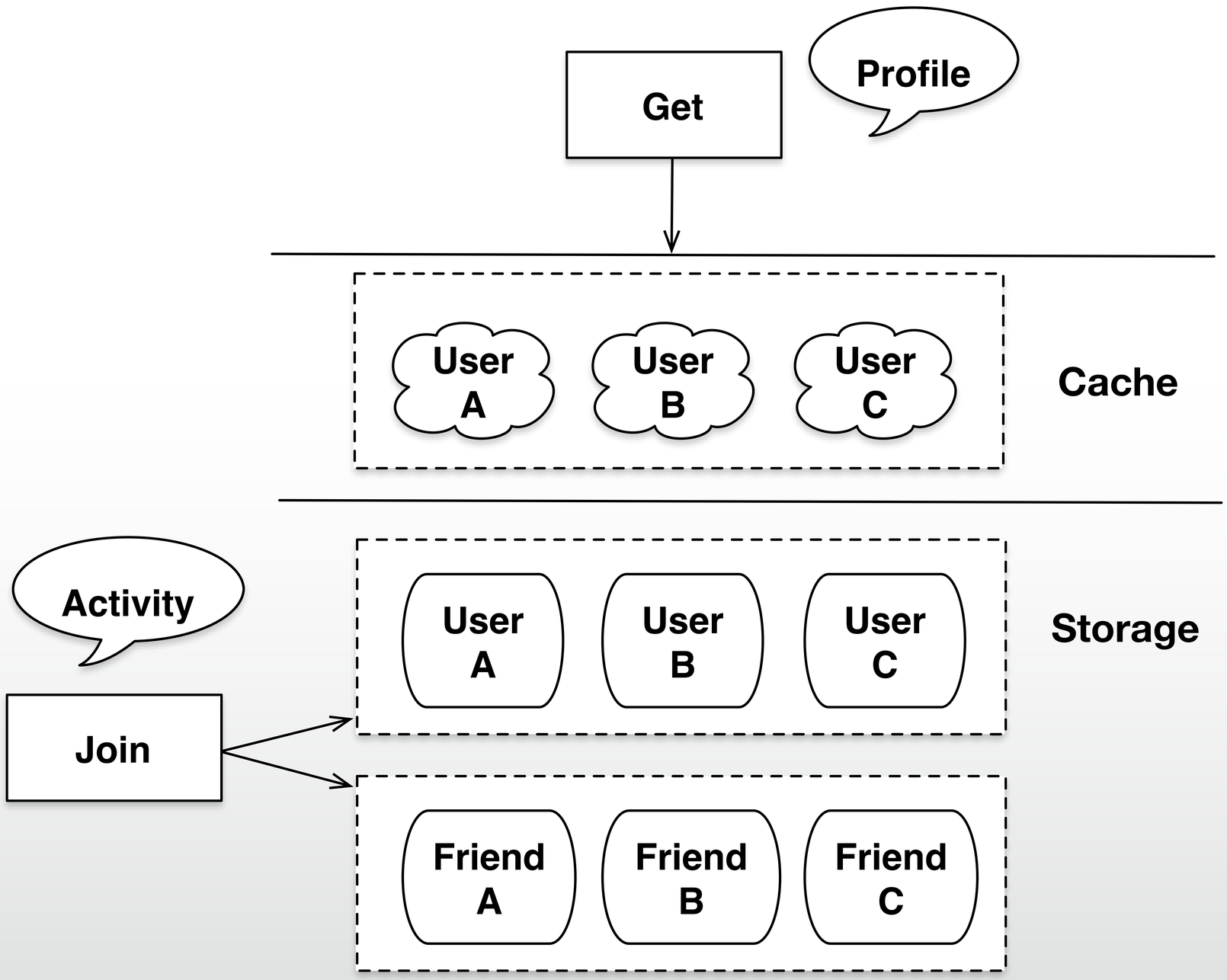
极简化设计阶段

- 发布美拍



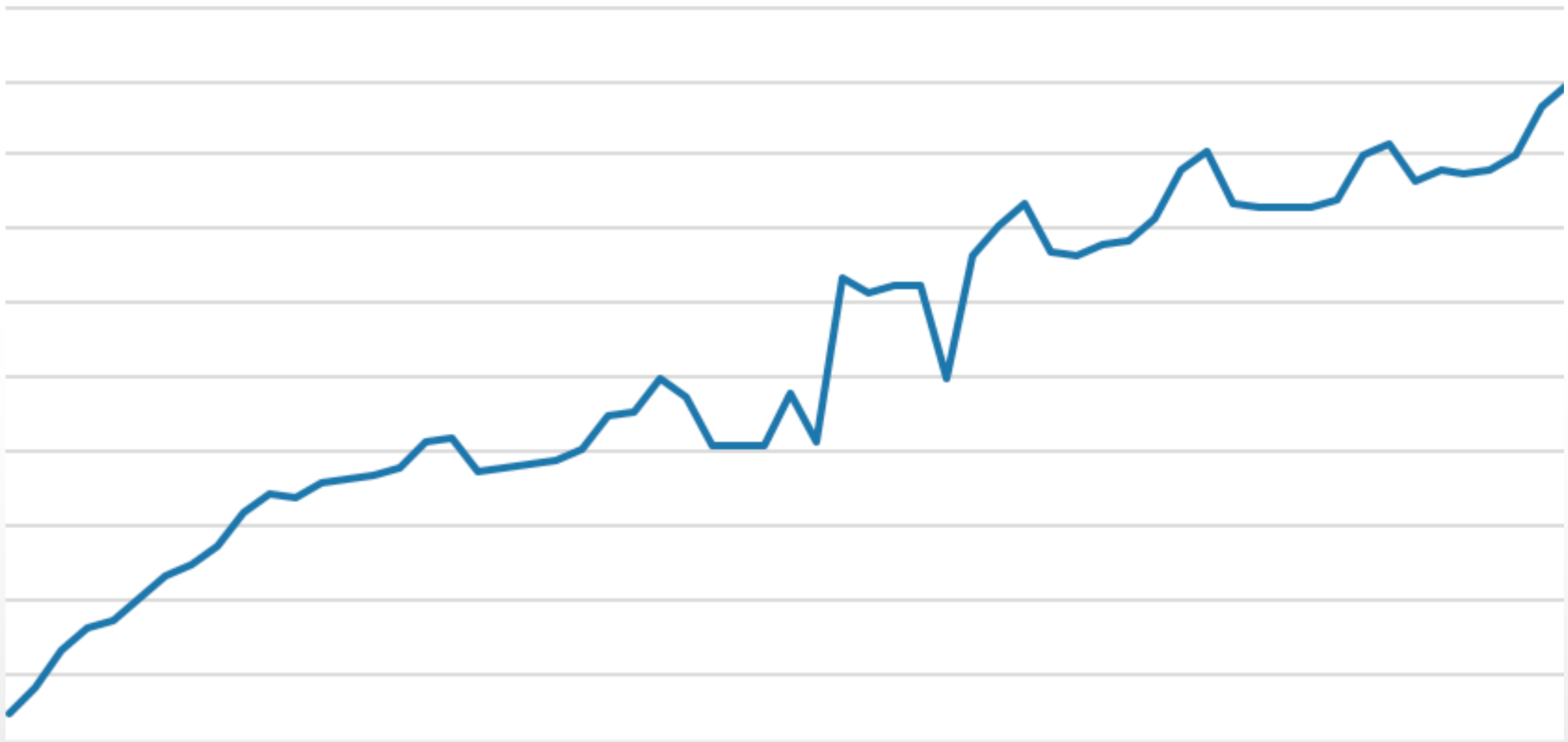
极简化设计阶段

- 个人主页和好友动态



极简化设计阶段

- 业务飞速增长 - 用户活跃度增长趋势



时间点： 2014.05.08 – 2014.07.06

极简化设计阶段

- 开始出现一些瓶颈 - 接口响应时间变长
  - Memcached大量的连接超时
    - 连接数瓶颈（配置2048）
    - 调高连接数至10240
  - 缓存命中率降低
    - 容量瓶颈
    - 扩容至8个节点

极简化设计阶段

- 开始出现一些瓶颈 – 接口响应时间变长

  – Mysql请求出现慢查询情况
    - 读瓶颈
    - 读写分离、增加从库

  – Mysql请求出现写入慢情况
    - 写入瓶颈
    - 升级服务器：SATA -> SAS -> SSD

极简化设计阶段

- 开始出现一些瓶颈 - 计数器服务延迟
  - Redis读大量超时
    - 单核CPU接近100%
    - 扩容使用多从
  - Rdb Dump期间影响用户请求
    - 凌晨期间做Rdb Dump
    - 使用独立的不对外服务的机器来做Rdb Dump

极简化设计阶段

- 实践经验
  - 系统能跑起来比任何架构先进性都重要
  - 优先考虑Scale Up方案（如硬件升级）
  - 优先考虑低成本的Scale Out方案（如多从）
  - 使用云服务和成熟技术来加速研发

极简化设计阶段

- 小结

Scale Up

Scale Out(低成本)

保持简单性设计阶段

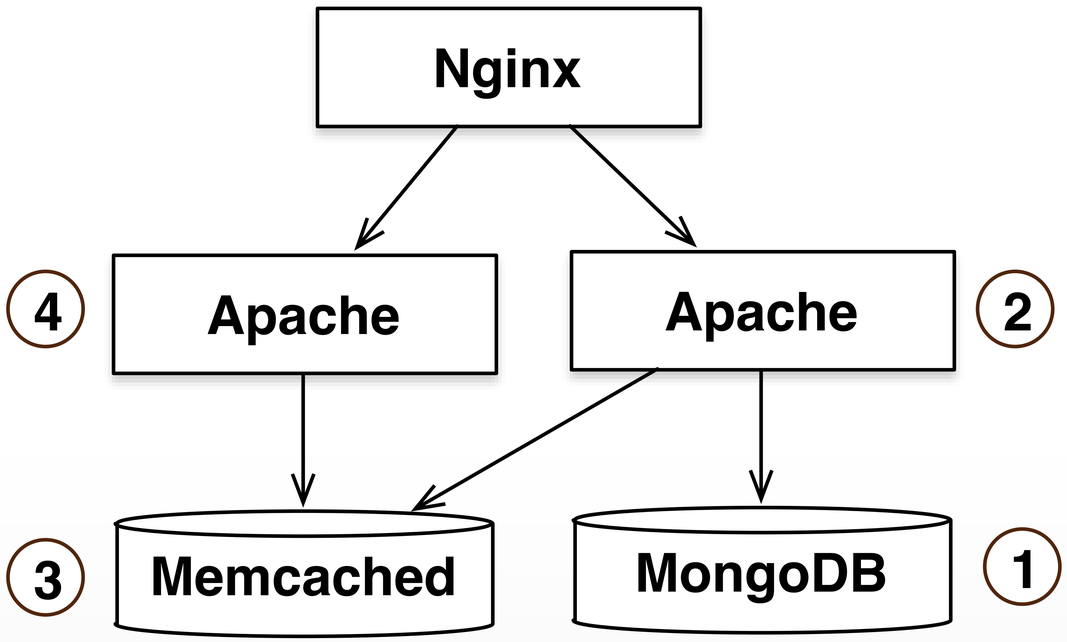
- 满足快速迭代的产品需求
- 支撑快速发展的业务系统
- 技术选型
  - 尝试使用部分新技术，比如MongoDB
  - 使用更多同类的云服务

保持简单性设计阶段

- 新一波问题出现 – 接口响应时间变长
  - Memcached命中率降低
    - 整体业务命中率降低
    - 个别新业务命中率奇低
  - 问题根源
    - 缓存容量成为瓶颈
    - slab calcification problem 【方案：核心业务隔离性部署】

保持简单性设计阶段

- 新一波问题出现 – 依赖影响引发的血案
  - 线上多个业务同时出问题
    - 响应时间很长

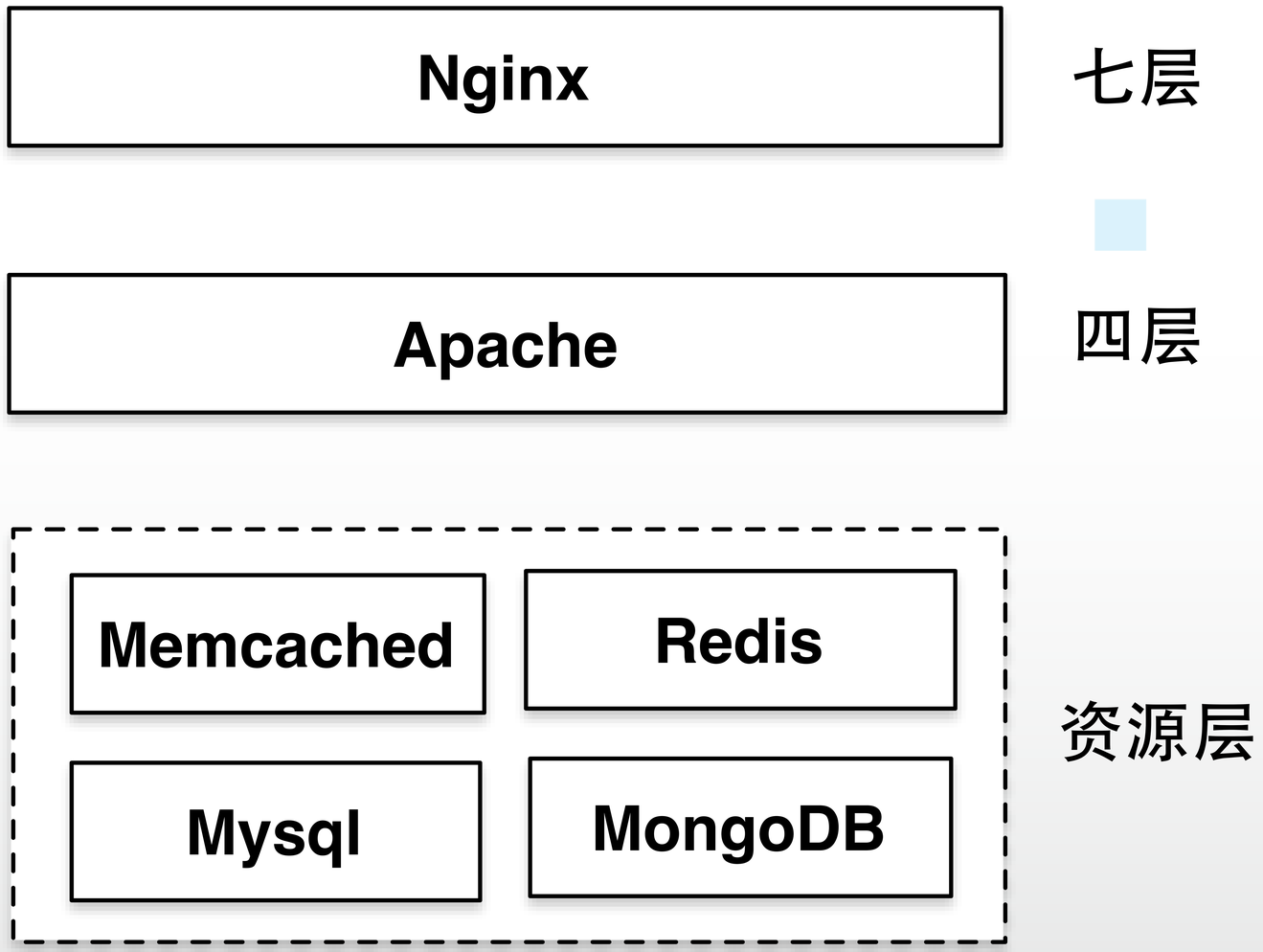


  - 问题根源
    - 1、MongoDB升级忘加索引
    - 2、大量MongoDB请求超时，Apache进程数飙高
    - 3、Memcached连接数飙高
    - 4、Apache接口大量超时，进程数飙高

【方案：核心业务隔离性部署】

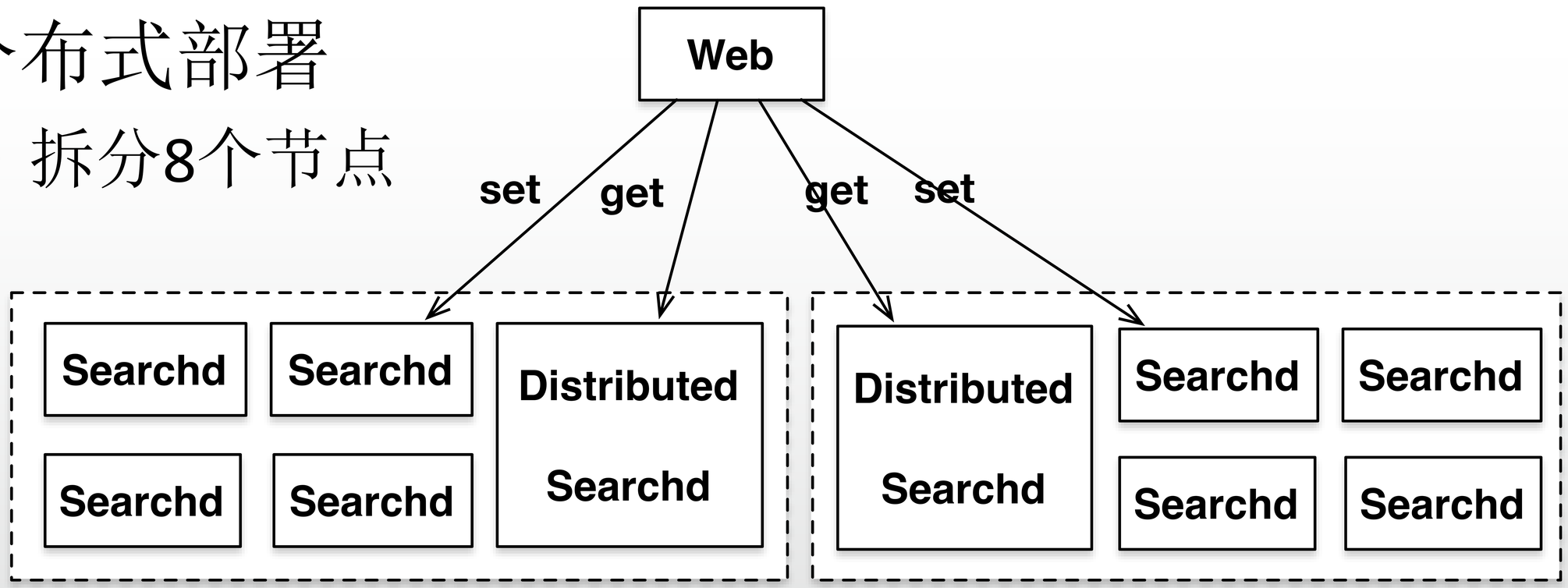
保持简单性设计阶段

- 核心业务隔离性部署
  - 避免依赖影响
  - 很合算的Scale Out



保持简单性设计阶段

- 新一波问题出现 – Sphinx昵称搜索延迟
  - 搜索延迟
    - 搜索请求响应时间到秒级甚至十秒级
    - 单个searchd节点数据量超过4kw出现瓶颈
  - 分布式部署
    - 拆分8个节点



保持简单性设计阶段

- 关于搜索可以在业务层做的事情
  - 搜索请求分布

	Top50关键词	top1000关键词	top10000关键词
内容搜索	18%	52%	77%
话题搜索	16%	40%	57%

  - 优化方式
    - 业务层加缓存能抗大量热请求
    - 需要评估数据即时性问题及其访问量

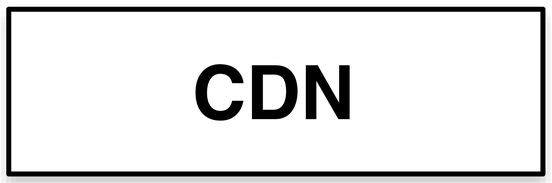
保持简单性设计阶段

- 新一波问题出现
  - 多个服务出现Mysql写入慢
    - 好友关系：分128张表
    - 喜欢：分128张表
  - 监控报警
    - 了解信息有限 -> 添加细粒度监控项
    - 运行性能瓶颈 -> 使用更好配置机器

保持简单性设计阶段

- 云服务保障 – 美拍视频和图片

– CDN



– 云处理

– 云存储

保持简单性设计阶段

- 云服务保障 – CDN

  – 踩过的坑
    - 受到DNS系统被攻击、被劫持影响
    - 受到灰度发布、机房网络改造影响
    - 弱网络情况视频播放问题

  – 应对措施
    - 多家CDN厂商分流和灾备
    - 与运营商更深入配合
    - 客户端智能适配接入网络、自动探测切换备选节点等

保持简单性设计阶段

- 云服务保障 – 处理

  – 本地处理
    - 现有移动设备计算能力已经很强大
    - 涉及本地效果处理需要评估发布包大小

  – 云端处理
    - 基于开源的FFmpeg、ImageMagick或GraphicsMagick
    - 需要强大的计算集群，成本不低

  – 尽可能利用本地计算资源
    - 上传处理过效果的视频及其视频缩略图到云服务
    - 云服务上尽可能轻量的处理（视频转码、图片缩略等）

保持简单性设计阶段

- 云服务保障 – 存储

  – 云服务现状
    - 现有云服务已有较高可用性保证
    - 但偶发网络、服务等问题比较被动

  – 保障措施
    - 业务期望达到永远可写
    - 使用多家云服务
    - 服务端开关控制写节点

保持简单性设计阶段

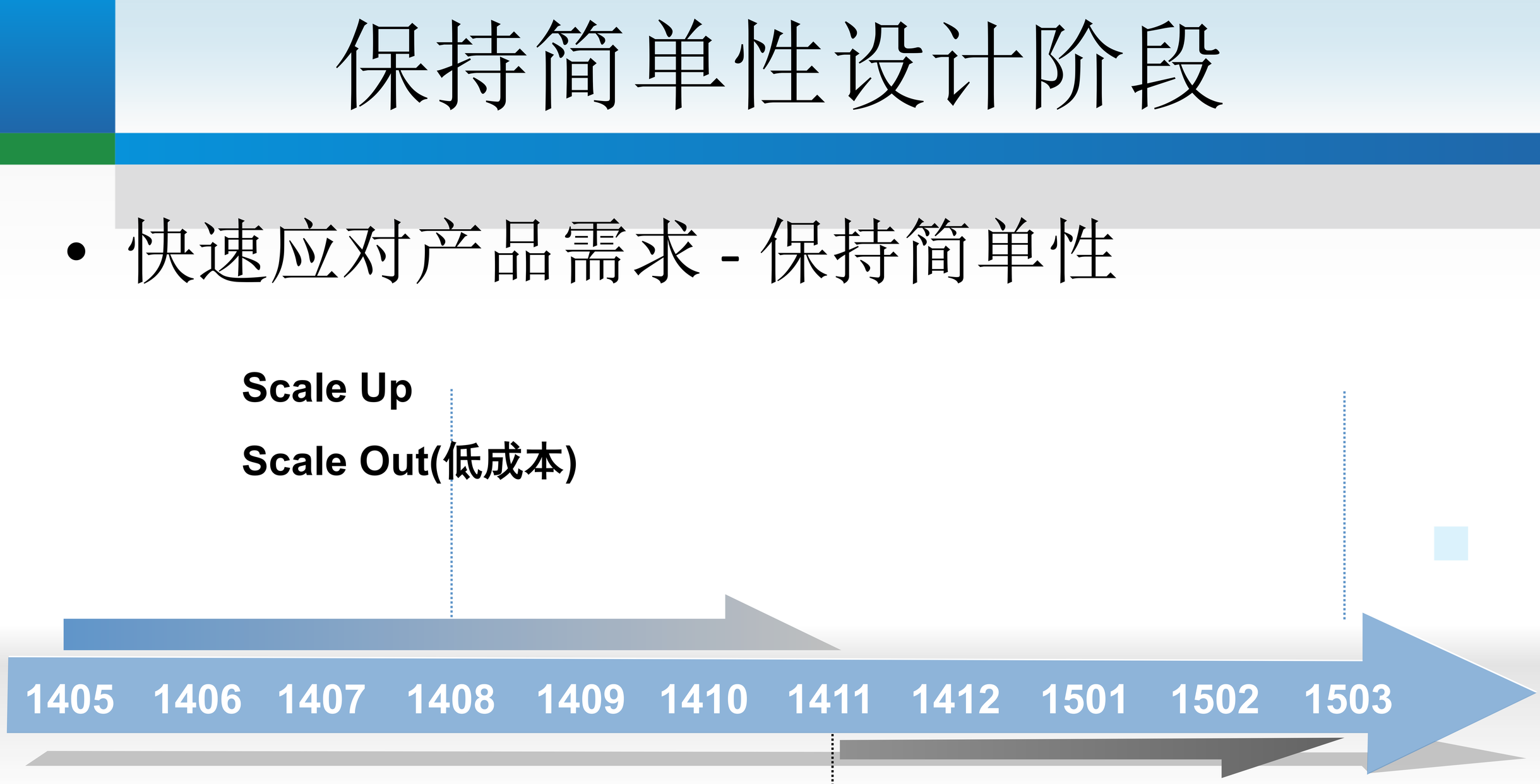
- 实践经验
  - 保持架构简单性，便于更快的产品迭代
  - 业务快速发展过程随时做好资源扩容的准备
  - 核心系统的隔离性部署能避免踩很多坑
  - 云服务的保障也很重要

保持简单性设计阶段

- 快速应对产品需求 - 保持简单性

Scale Up

Scale Out(低成本)



Scale Out（资源扩容）

可用性保证（云服务）

可扩展和可用性保证阶段

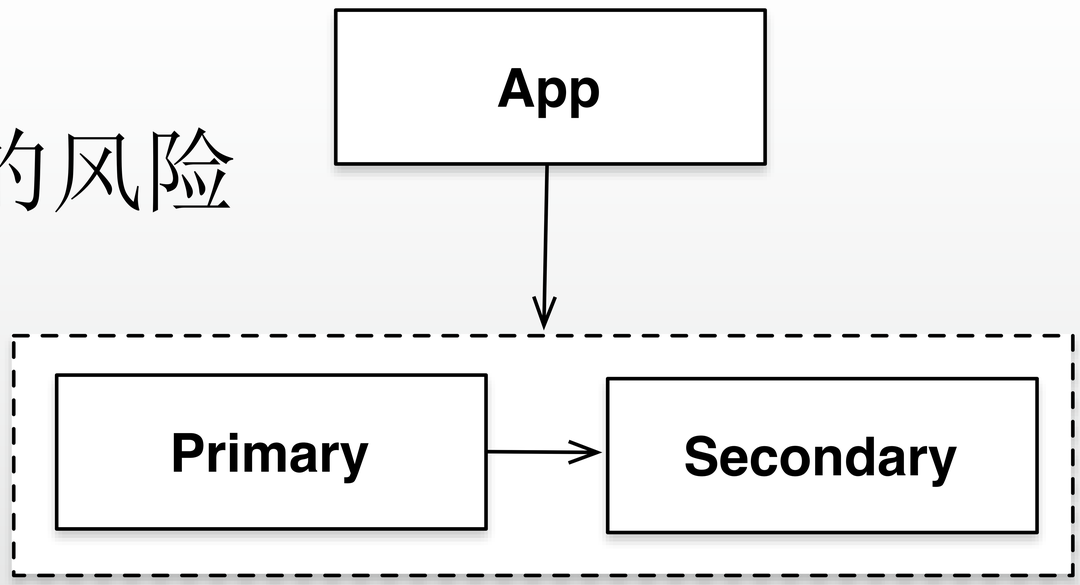
- 新功能研发：新增LBS、私信、表情文等
- 现有系统优化：支撑快速发展的业务
- 支撑春节期间大量级访问及其保证稳定性
- 技术体系：
  - 部分场景引入Java和C技术体系

可扩展和可用性保证阶段

- 新产品功能

可扩展和可用性保证阶段

- 新业务研发 – LBS服务（附近的美拍）
  - 快速研发上线
  - 基于MongoDB实现
    - 使用2dsphere索引，基于geo nearSphere
    - 使用Replica-Set方式，不用Auto-Sharding
  - 通过容量规划先规避一定的风险
    - 独立部署
    - 全内存存储

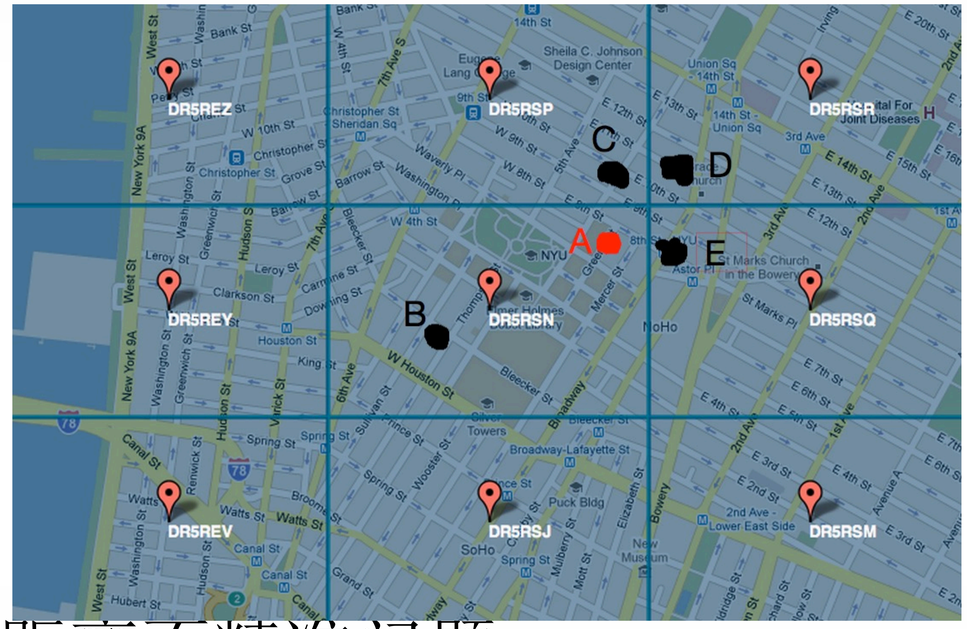


可扩展和可用性保证阶段

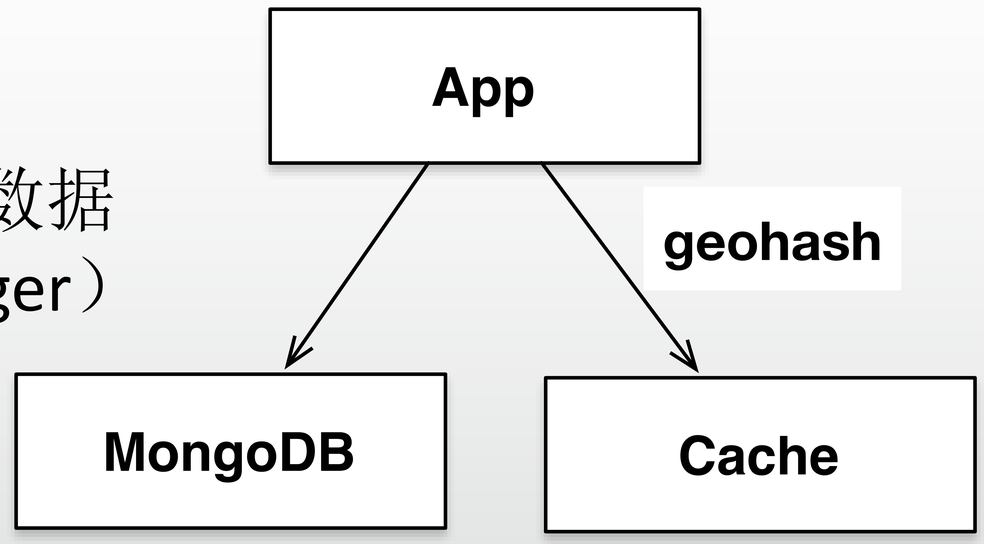
- 新业务研发 – LBS服务（附近的美拍）
  - 业务期望能保证整体可控 - MongoDB是否能胜任？
  - 选择1：运维层面做好保障，也包括容量规划层面
    - 可作为很好的辅助措施，但是不能完全依赖
  - 选择2：对MongoDB深入研究，具备二次研发能力
    - 需要投入较多研发力量和时间
  - 选择3：对MongoDB弱依赖，控制出问题时的影响面
    - 整体相对可控的方案

可扩展和可用性保证阶段

- 新业务研发 – LBS服务（附近的美拍）
  - 可用性保证
  - 基于Geohash实现缓存策略
    - 使用Google s2-geometry-library
    - 缓存热点区域数据
    - 合并计算附近9个格子数据解决距离不精准问题



  - 实现对MongoDB的弱依赖
    - 出故障仅影响非热点区域和长尾数据
    - 便于尝试新技术（如使用Wiredtiger）

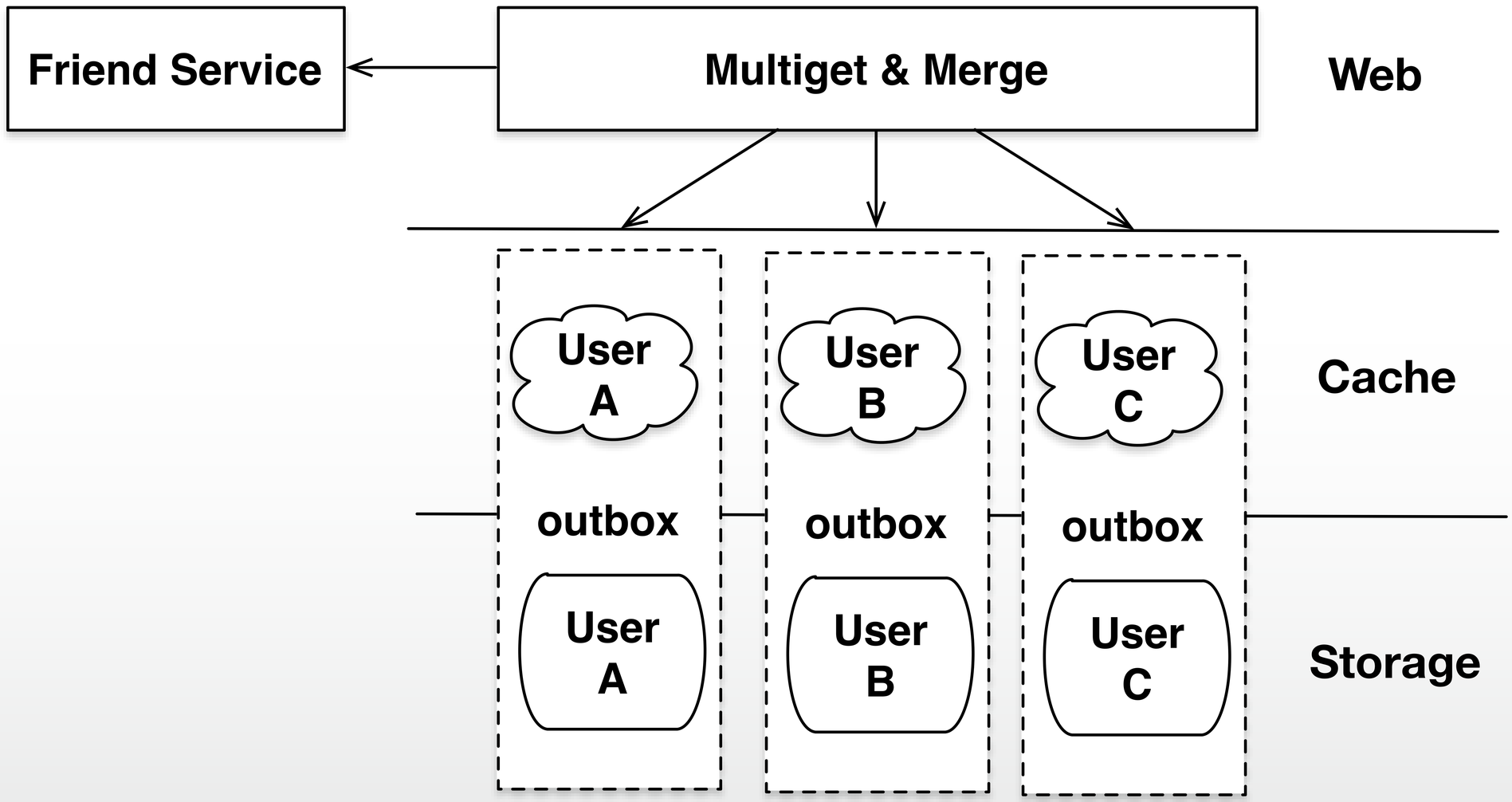


可扩展和可用性保证阶段

- 好友动态架构优化
  - 美拍的社交属性增强，成为APP重要的入口
  - 现有方案添加字段成本偏高
  - 需要应对春节期间峰值的写入和读取
  - 需要可支撑数倍量级并保证稳定性的方案

可扩展和可用性保证阶段

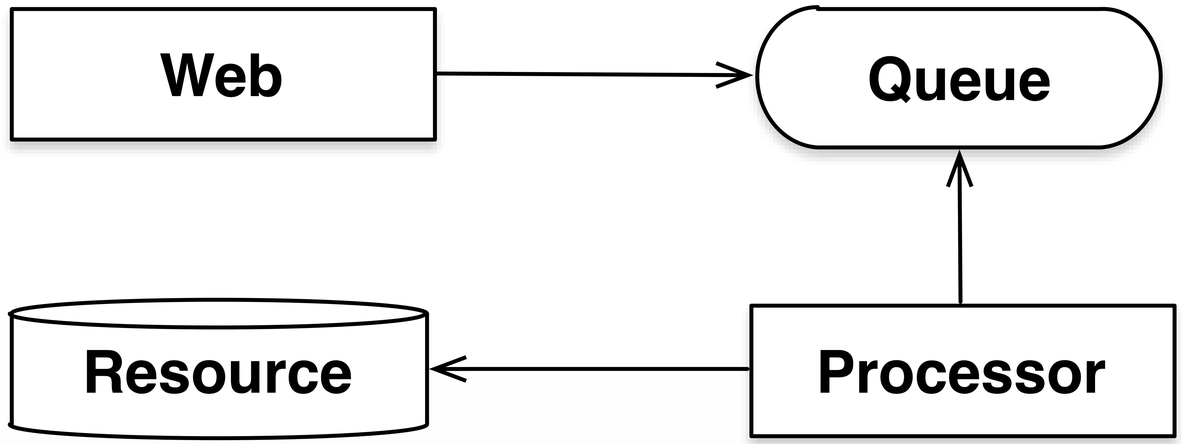
- 好友动态架构优化 - 拉模式



可扩展和可用性保证阶段

- 好友动态架构优化 – 异步写入

– 前端永远可写



– 前后端分离

1、消峰作用

2、利于降级控制

3、多机房部署基础

可扩展和可用性保证阶段

- 好友动态架构优化 - 数据库

  – 索引和内容分离

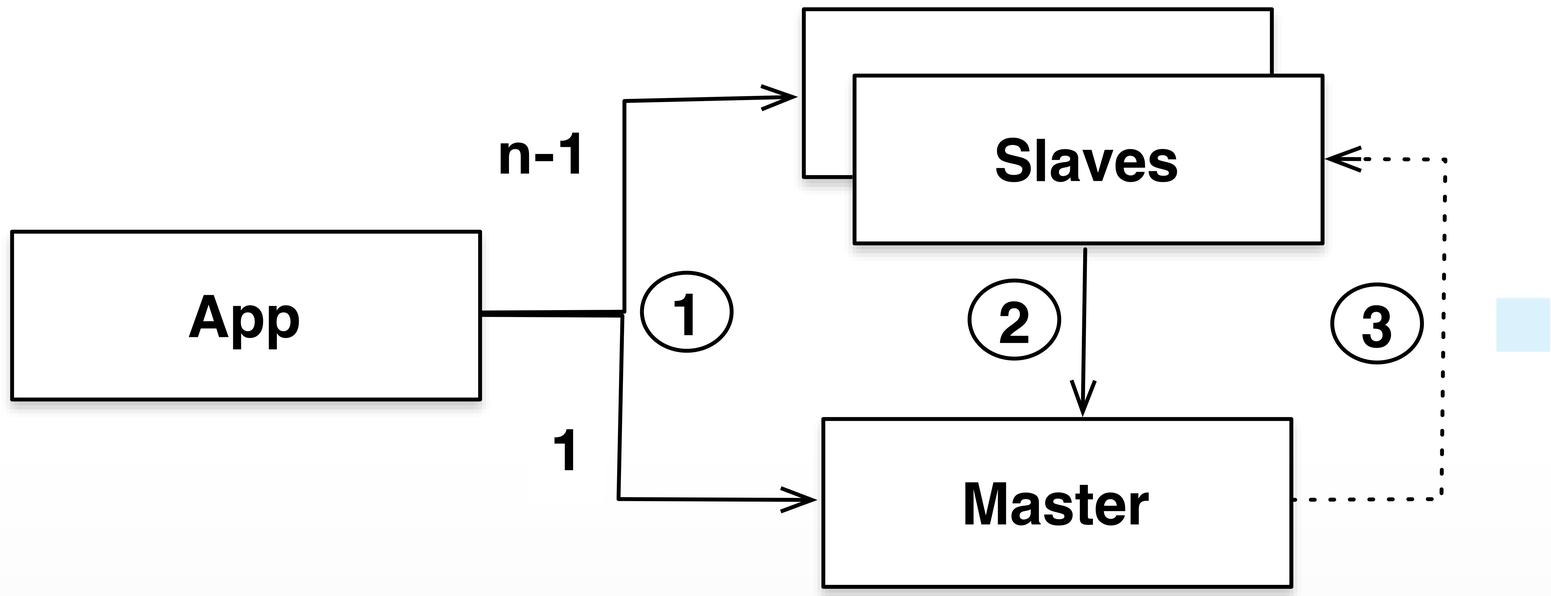
    - 索引：仅存放需要索引的字段，需要保证高性能查找
    - 内容：Key-Value结构，value为pb二进制数据

  – 分库分表策略 – 便于后续冷热数据分离

    - 2个数据库实例，共32个数据库
    - 每个库按月拆表

可扩展和可用性保证阶段

- 好友动态架构优化 – 缓存
  - – 防单点
  - – 线性扩容



1、Master也承担读取：缓存热度的保证

2、Slave穿透到Master：防单点策略

3、Master穿透到Slave：防单点策略（直接请求Master场景）

可扩展和可用性保证阶段

- 好友动态架构优化 – 可运维性

  – 监控报警机制
    - 成功率、响应时间、资源问题、队列写入异常

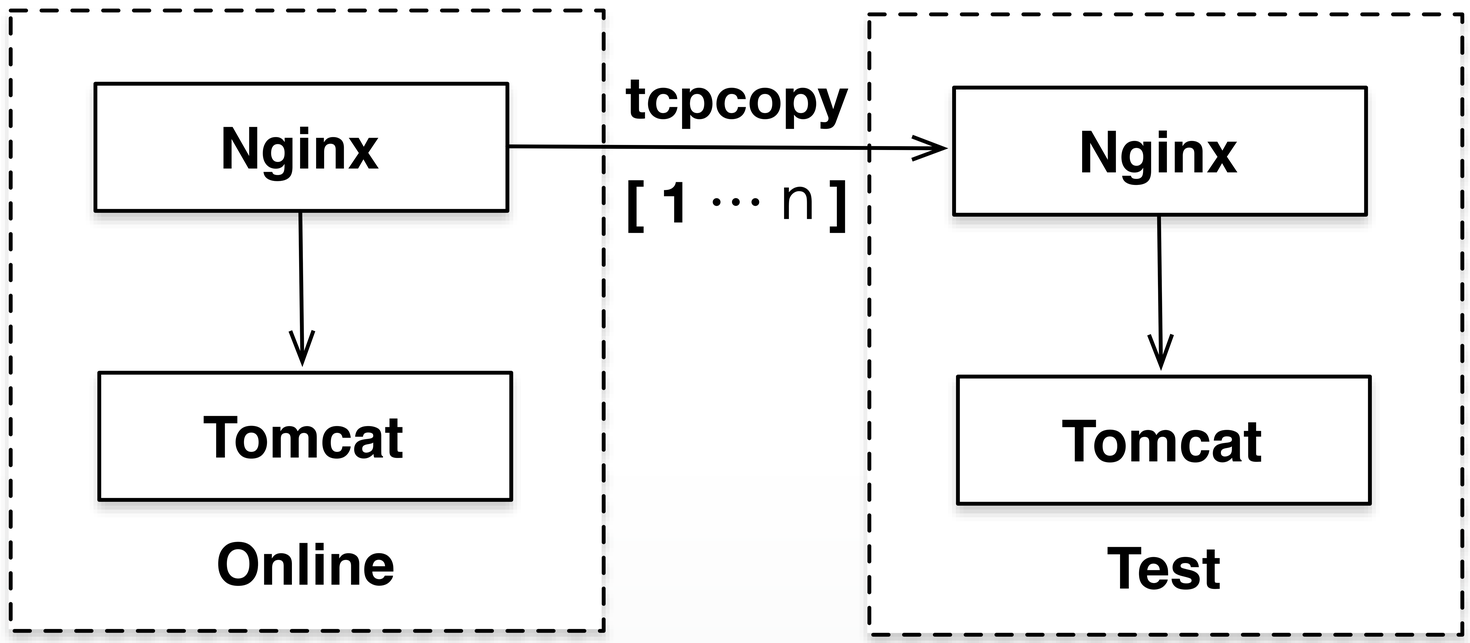
  – 问题定位排查
    - 关键流程处理时间日志、资源性能日志、异常日志

  – 降级策略
    - 开关服务（依赖服务、资源，及其各种策略）
    - 极端异常情况保核心路径可控

注：分布取值为99.9%

可扩展和可用性保证阶段

- 好友动态架构优化 – 在线压测
  - – tcpcopy拷贝流量
  - – 观察接口和资源



接口列表	成功率	响应时间（平均）	响应时间（最大）	响应时间（分布）
接口1	100%	5	50	10
接口2	100%	8	100	20

注：时间单位为ms，分布取值为99.9%

可扩展和可用性保证阶段

- 春节保障
  - 扩容服务器、预留备用服务器
  - 评估核心服务支撑能力：在线压测
  - 完善监控项和报警机制，加入更多业务维度监控
  - 保证开发人员放假期间随时可上网：分发上网卡

可扩展和可用性保证阶段

- 评论、喜欢等架构优化
  - 索引和内容分离
  - 进一步拆表
  - 独立的缓存和数据库

可扩展和可用性保证阶段

- 实践经验
  - 核心服务：投入最多资源全方位的保障
  - 不完全可控服务：尽可能对其弱依赖
  - 可扩展性和可用性逐步成为主角

可扩展和可用性保证阶段

- 更多关注可扩展性和可用性

Scale Up

Scale Out(低成本)

Scale Out(架构优化）

可用性保证

Scale Out（资源扩容）

可用性保证（云服务）

现阶段

- 多机房部署
- 高可扩展和可用性保证
- 研发和上线流程优化
- 团队建设

Q & A

快速发展的美图期待您加入

hxj@meitu.com

Brought by InfoQ

www.infoq.com/cn

InfoQueue

@InfoQ

infoqchina

软件
正在改变世界！