中生代技术
FRESHMAN TECHNOLOGY

iBOXCHAIN
盒子科技

# 走进 盒子科技

金融科技　分布式缓存　支付　架构

2018年4月21日13:00 – 18:00

深圳市南山区软件产业基地5栋C座503盒子科技

# 微博Cache架构设计实践

新浪微博 陈波@fishermen

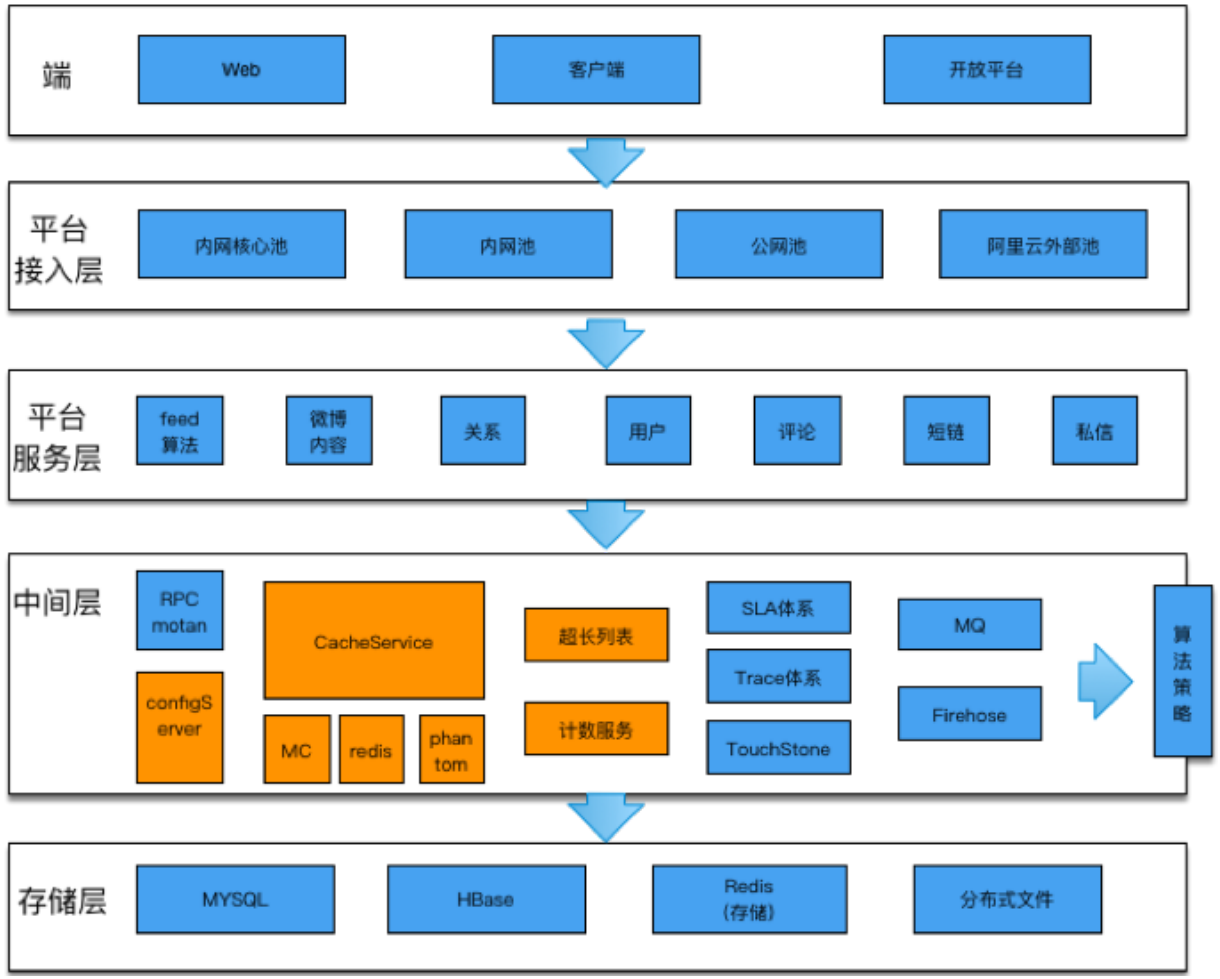
# 大纲

- 数据挑战
- Feed系统架构
- Cache架构及演进
- 总结及展望

# 数据挑战

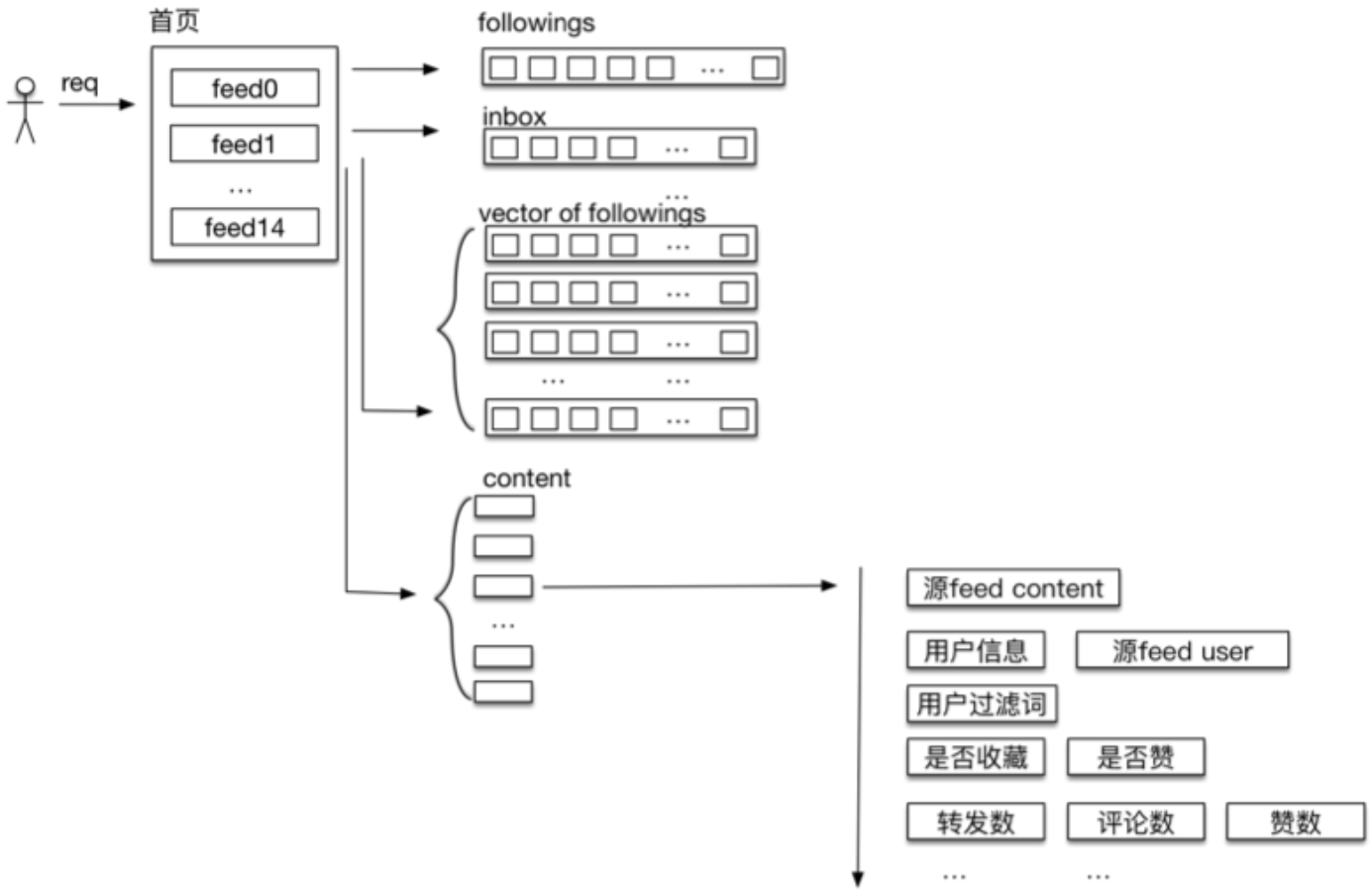
- 日活用户 1.6亿+
- 平台接口
  日访问百亿级
- 核心记录千亿级
- Cache
  内存百T级
- Cache
  日访问万亿级
- 单个核心数据
  Cache QPS百万级

# Feed平台系统架构


端
Web
客户端
开放平台
平台
接入层
内网核心池
内网池
公网池
阿里云外部池
平台
服务层
feed
算法
微博
内容
关系
用户
评论
短链
私信
中间层
RPC
motan
configS
erver
CacheService
MC
redis
phan
tom
超长列表
计数服务
SLA体系
Trace体系
TouchStone
MQ
Firehose
算法策略
存储层
MYSQL
HBase
Redis
(存储)
分布式文件

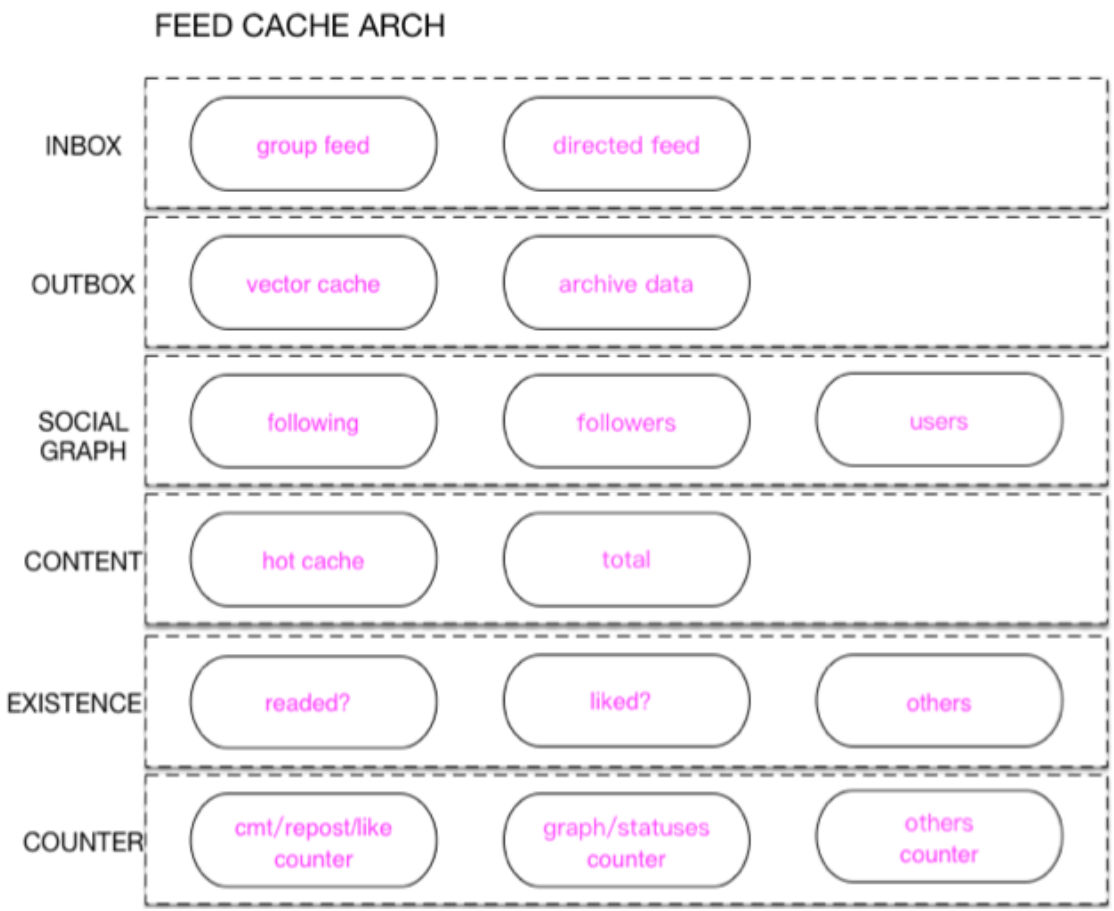
# Feed timeline

- 构建流程



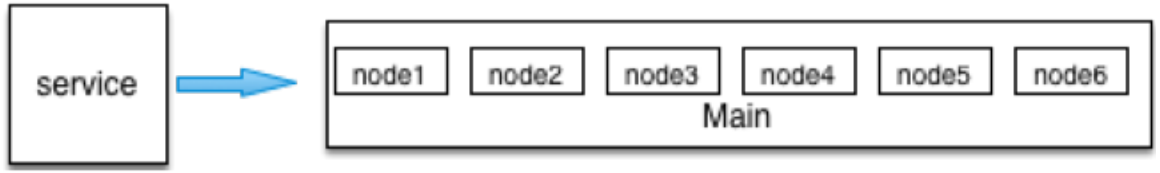
# Feed Cache 架构

- Cache架构

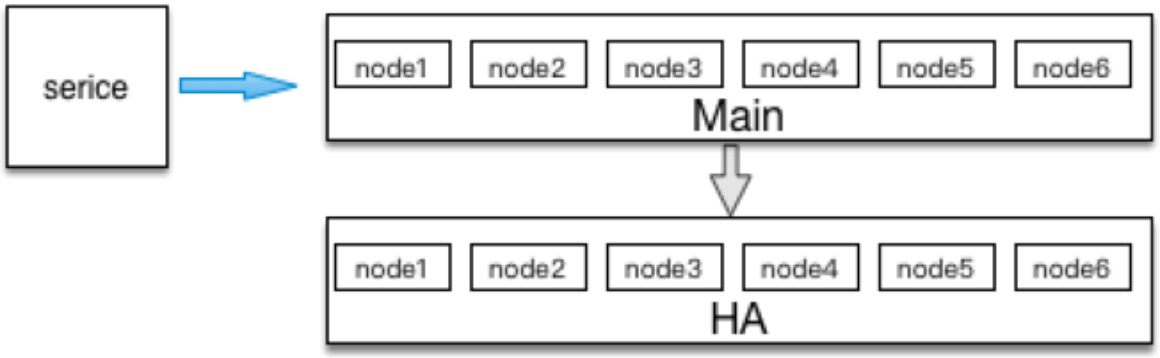


# Cache架构及演进-简单KV数据类型

- 单层Hash

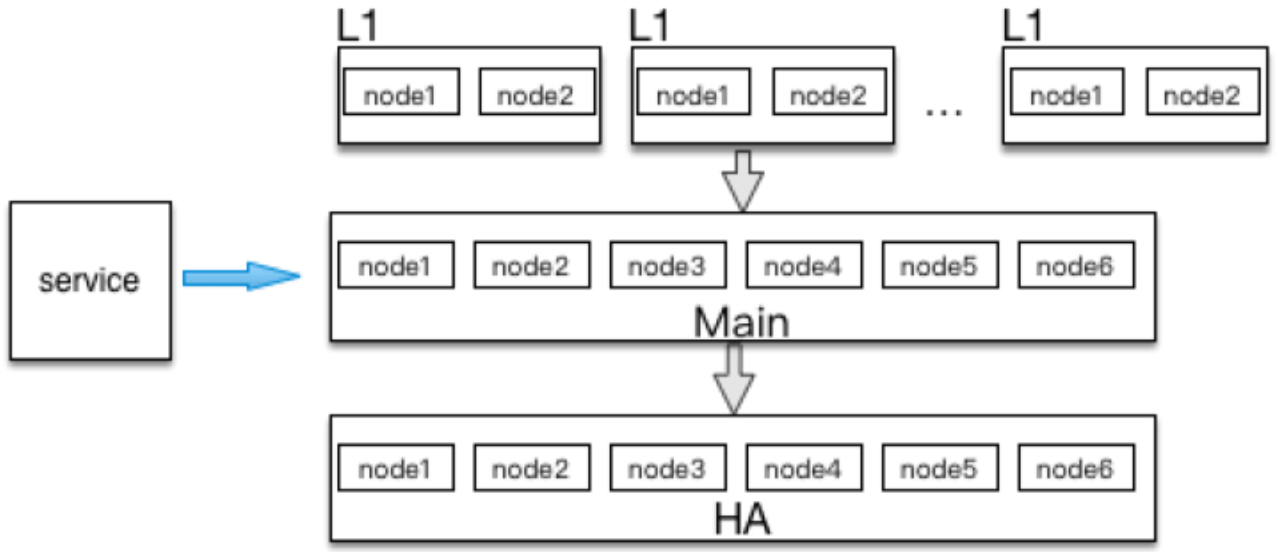



- Main-HA



# Cache架构及演进-简单KV数据类型

- Main-HA-L1



# Cache架构及演进-简单KV数据类型

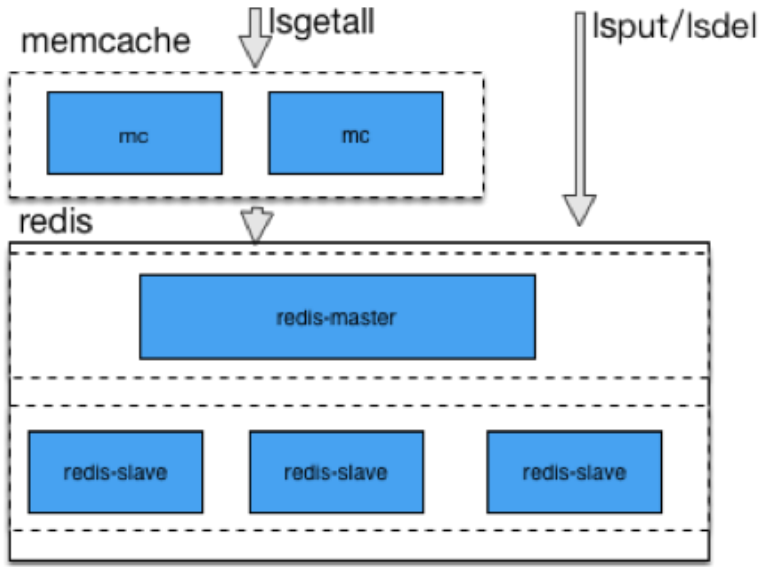
- Key Point
  - Memcached 为主
  - 层内HASH节点不漂移，miss则穿透
  - 多组L1 读取性能升 峰值流量成本降
  - 读写策略
    - Write：多写
    - Read：逐层穿透，miss 回写
  - Json/xml → Protocol Buffer
  - QuickLZ 压缩

# Cache架构及演进- 集合类数据

- 业务特点
  - 部分修改
  - 分页获取
  - 资源计算：联动计算
  - 类型：关注，粉丝，分组，共同关注，XX也关注
- 方案：Redis
  - Hash 分布，MS，cache/storage
  - 30+T 内存，2-3万亿rw/day

# Cache架构及演进-集合类数据

- Redis扩展（Longset）
  - Long型开放数组，Double Hash 寻址
  - Client 构建数据结构，elements 单次写入
  - Lsput：填充率过高，由client重建
  - Lsgetall → Lsdump
  - 少量而超热数据：mc抗读



# Cache架构及演进-集合类数据

- Redis其他扩展
  - 热升级：10+分钟→毫秒级
  - AOF：Rotate
  - RDB：Pos of AOF
  - 全增量复制
  - 落地/同步速控

# Cache架构及演进-其他数据类型-计数

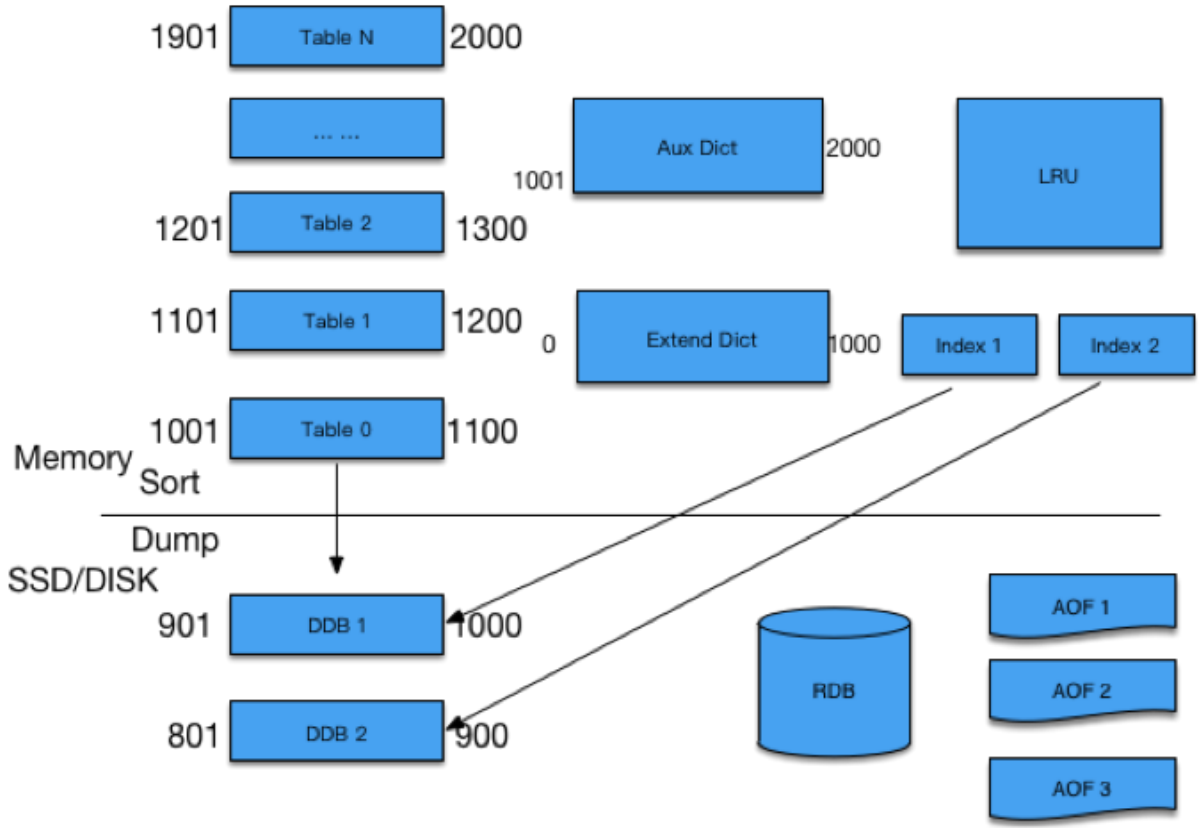
- 业务特点
  - 单key有多计数（微博/用户多种计数）
  - Value size较小（2-8个字节）
  - 每日新增记录近十亿级，总记录千亿级
  - 单次请求多条kv

# Cache架构及演进-其他数据类型-计数

- 选型1: Memcached
  - MC 剔除，重启数据丢失
  - 大量计数为0，如何存
- 选型2: Redis
  - 内存有效负荷低
  - 访问性能
- 最终方案：自研CounterService
  - Shema 支持多列，按bit分配
  - Tables 预分配，double-hash寻址
  - 内存降为1/5-1/15以下
  - 冷热分离，SSD 存放老数据，老热数据入LRU
  - 落地 RDB + AOF
  - 全增量复制
  - 单机：热数据百亿级，冷数据千亿级

# Cache架构及演进-计数器 CounterService

- 存储架构



# Cache架构及演进-其他数据类型-存在性判断

- 业务类型需求
  - 检查是否存在（阅读 赞）
  - 单条记录量小，value 1bit (0/1)
  - 总数据量巨大，大量value为0
  - 每日新增数量大 千亿级

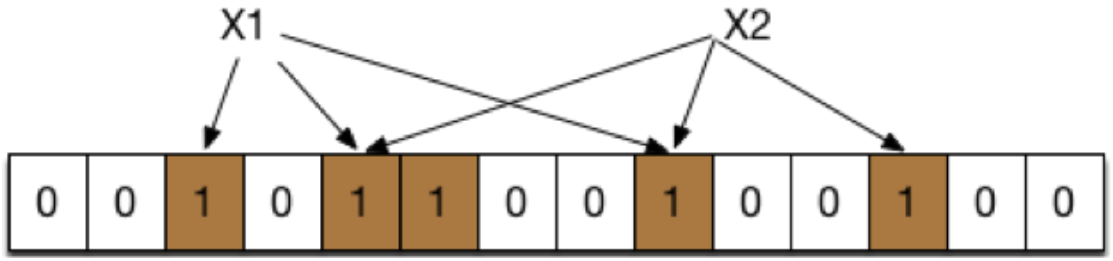
# Cache架构及演进-其他数据类型-存在性判断

- 选型1: Redis
  - 单条kv：65 bytes
  - 每日新增内存 6T （不考虑HA）

- 选型2 CounterService
  - 单条kv：9 bytes
  - 每日新增内存 900G （不考虑HA）

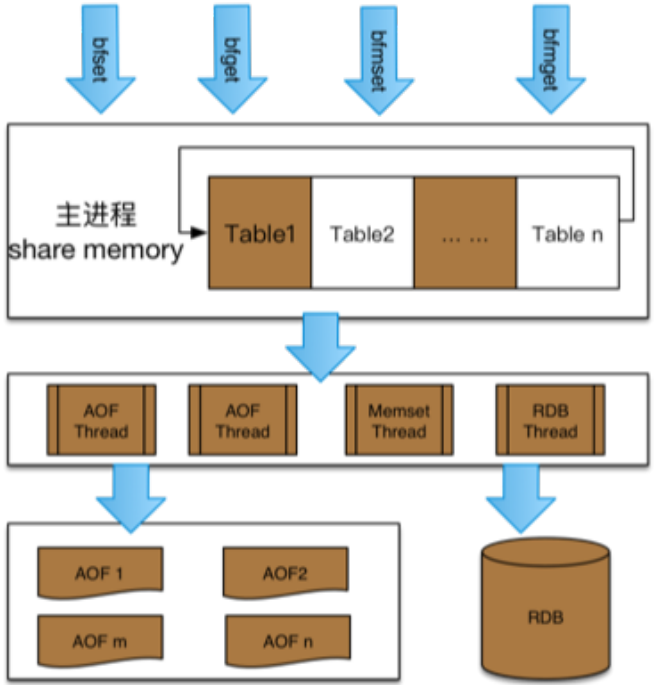
# Cache架构及演进-其他数据类型-存在性判断

- 最终方案：自研Phantom
  - Table 分段预分配，段内 bloomfilter
  - 每条kv：1.2 bytes (1%误判)
  - 每日新增内存：120G < 800G < 6T



# Cache架构及演进-其他数据类型-存在性判断

- Phantom系统架构
  - 数据存放共享内存，重启不丢数据
  - 落地 RDB + AOF
  - 兼容Redis 协议



# Cache架构及演进-小结

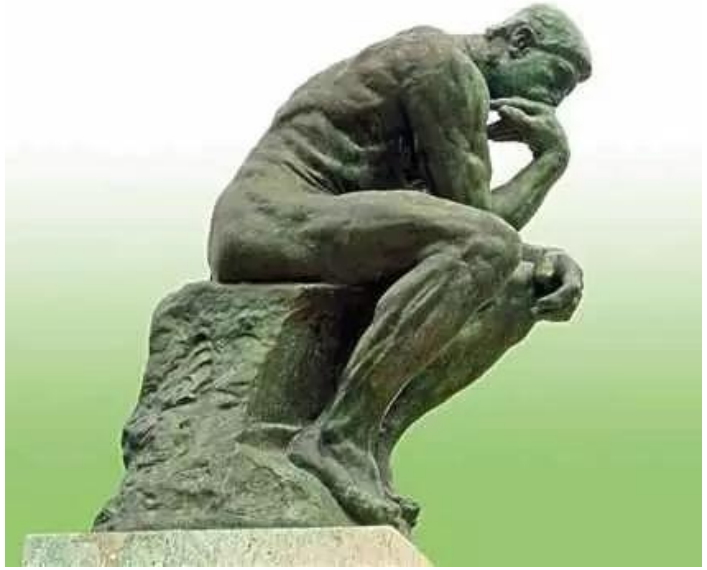
- 关注点
  - 集群内高可用
  - 集群内扩展性
  - 组件高性能
  - 存储成本

# Cache架构及演进-进一步优化

面向资源/组件管理
如何简化运维？

本地配置模式
如何快速变更？

常规峰值、突发流量
如何快捷、低成本应对？



业务数据分类多
如何独立管控SLA？

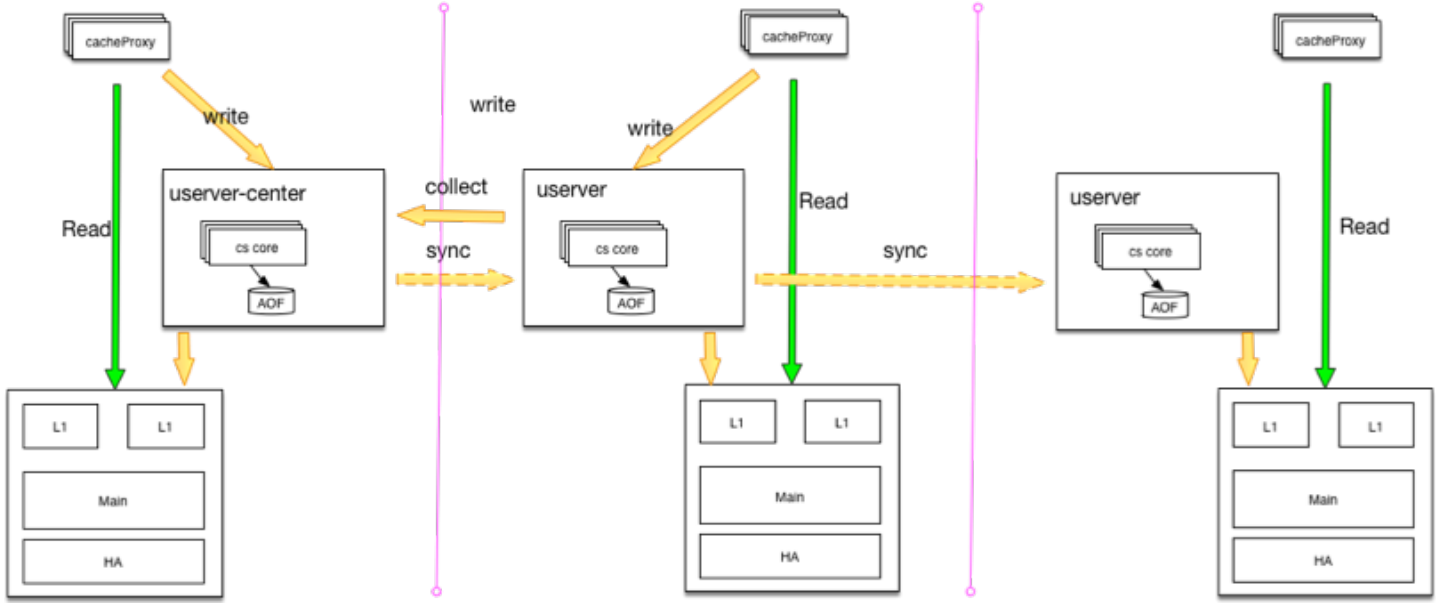
业务关联资源太多
如何简化开发？

# Cache架构及演进-服务化

- 本地Confs → 配置服务化
  - configServer 管理配置/服务，避免频繁重启
  - 资源/服务管理 API 化
  - 变更方式：script 修改， smart client 异步更新

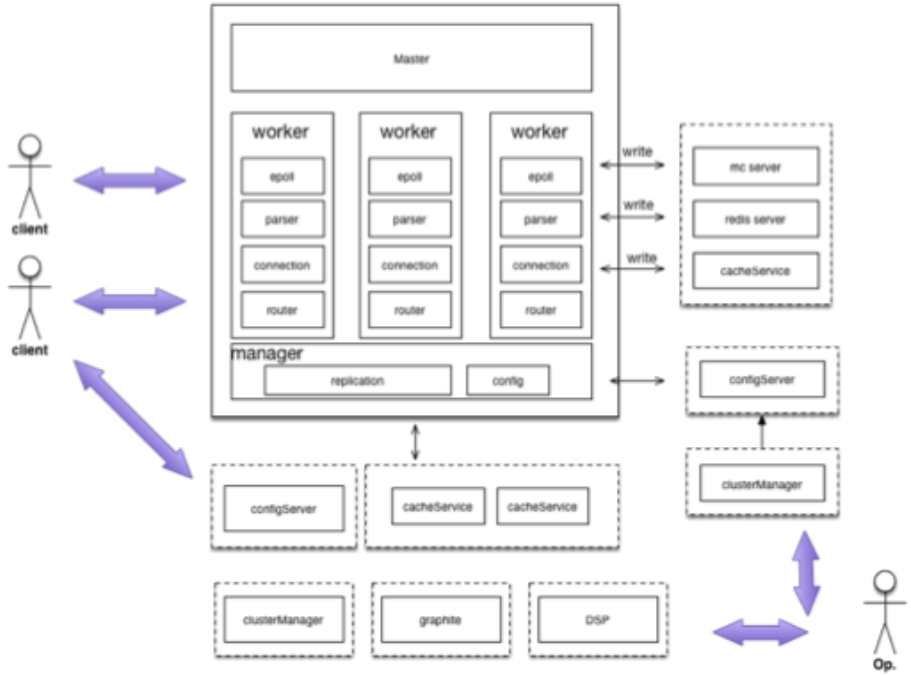
# Cache架构及演进-服务化

- Cache 访问
  - Proxy 化
- IDC数据一致性
  - Collecting/replication



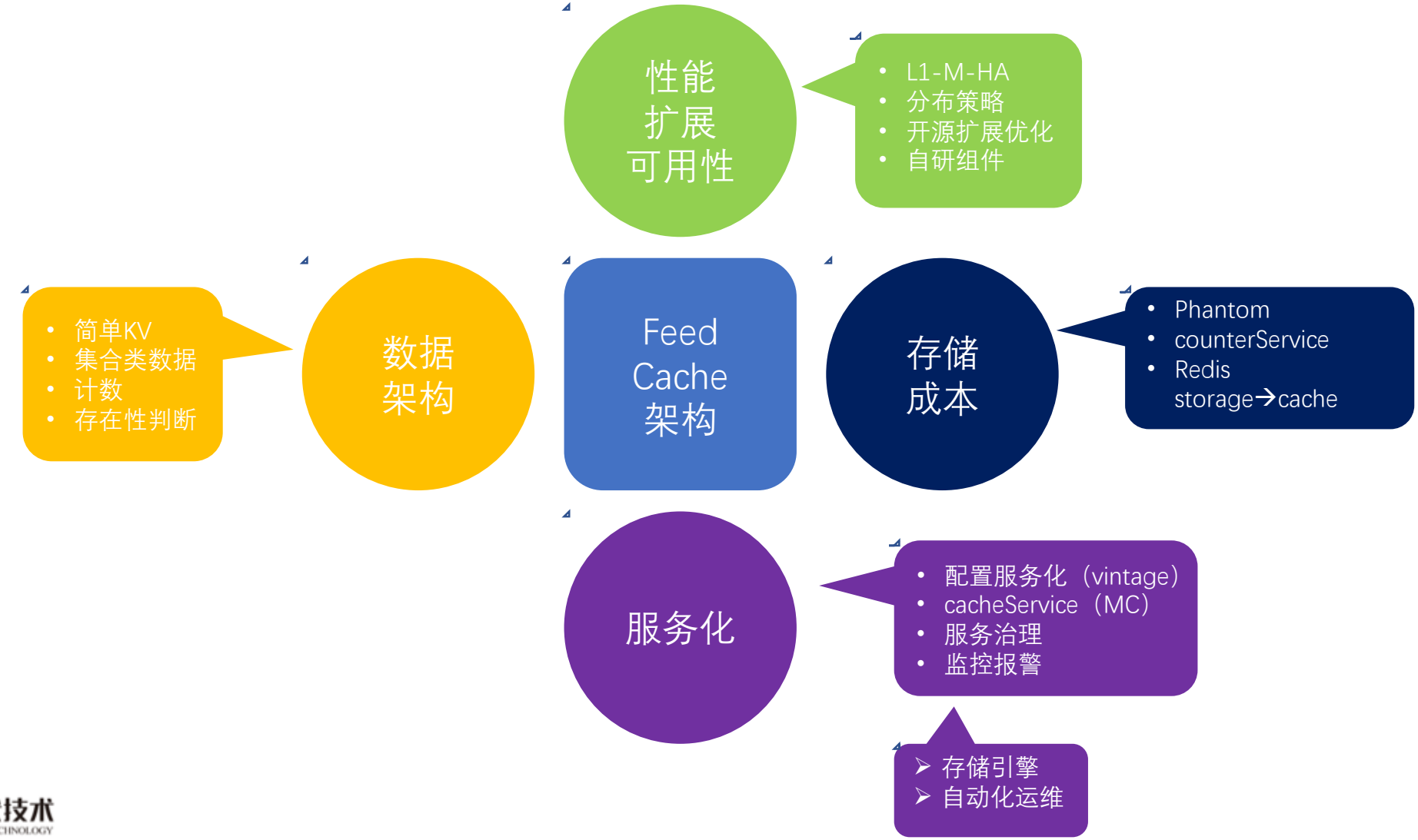
# Cache架构及演进-服务化

- ClusterManager
  - 脚本化→Web 界面化
  - 服务校验 业务SLA
  - 面向服务管控资源
- 服务治理
  - 扩容、缩容
  - SLA 保障
  - 监控报警
  - 故障处理
- 简化开发
  - 屏蔽Cache资源细节
  - 单行配置访问




```
<weibo:cstemplate id="cacheService" namespace="unread-feed"/>
```

# 总结与展望


性能
扩展
可用性
L1-M-HA
分布策略
开源扩展优化
自研组件
数据
架构
简单KV
集合类数据
计数
存在性判断
Feed
Cache
架构
存储
成本
Phantom
counterService
Redis
storage→cache
服务化
配置服务化（vintage）
cacheService（MC）
服务治理
监控报警
存储引擎
自动化运维

# Q&A

谢谢大家的聆听！