

SACC 2014中国系统架构师大会
SYSTEM ARCHITECT CONFERENCE CHINA 2014

发现架构之美

新浪微博redis优化历程

陈波 @fishermen

大纲

- 业务场景
- Redis存储架构演进
- 一些经验
- Q&A

业务场景-业务

- Redis在新浪微博的应用

通知提醒

The screenshot shows a Weibo profile for 'fishermen' (http://weibo.com/277556300). The profile includes statistics: 1108 关注 (Followers), 1243 粉丝 (Fans), and 2061 微博 (Weibos). The '我的主页' (My Home) tab is selected. The profile bio mentions '关注高并发、大容量、分布式的开发及技术' (Focus on high concurrency, large capacity, and distributed development and technology). The '我的关注' (My Follows) section shows 1108 follows, including users like '清华南都', '严锋', '洋专家', and '华为荣耀...'. The '我的粉丝' (My Fans) section shows 1243 fans, including users like '妍小妞的...', 'ladyWEI...', '琨子kare...', and '李岩lostk...'. The '我的微博' (My Weibos) section shows a tweet from '华景新城' (Huajing New City) dated '9月13日 08:20' from an iPhone client, with 415 reads, 5 likes, 1 retweet, and 1 comment. Annotations with red circles and lines highlight specific features: '通知提醒' (Notification Reminder) points to the top right notification bar; '计数 (counter)' (Count (counter)) points to the follower/follow counts; '关系 (graph)' (Relationship (graph)) points to the '我的关注' and '我的粉丝' sections.

业务场景-数据

- 一些数据

6 IDC

500+servers

3700+ instances

千亿条记录

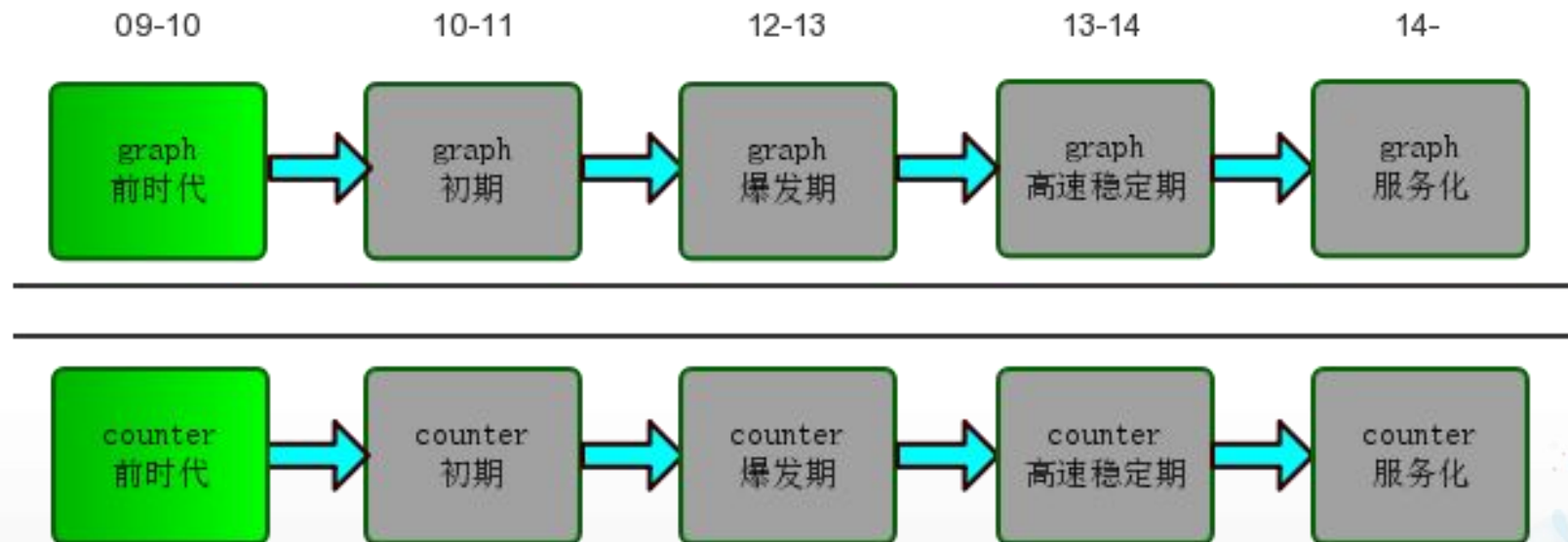
24T+内存

7千亿cmds/day

1.2万亿read/day

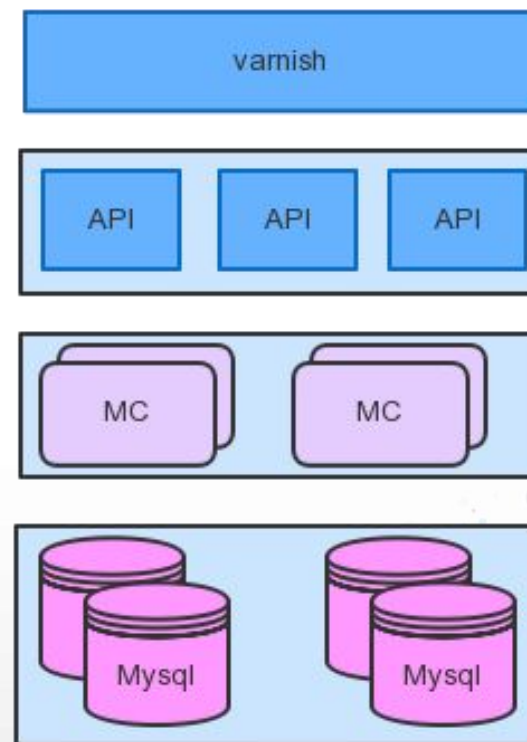
2千亿write/day

Redis 存储前时代



Redis前时代

- 热数据mc
- 全量落地mysql
- 数据量不大: Graph mc 10G, 计数器 mc 2G
- 开发速度



问题出现

- 2010年, Graph mc 30G+, 峰值 10wTPS
- Mysql成为瓶颈
 - 线程阻塞, 访问卡顿
 - List类型业务不适合mysql
- 新的关系计算需求实现困难
 - 大量关系计算: 从MC取全量+本地计算->超时

解决方案

- 初期方案
 - 增大mc容量到40G, Graph db 增至一主六从
 - 监控并及时清理僵死线程
 - 关系计算性能问题暂时无解
- 最终方案
 - 引入Redis做storage (graph/counter)
 - 关系计算 在redis实现 $O(1)$
 - 促进更多复杂需求
 - Graph db恢复一主三从

小结

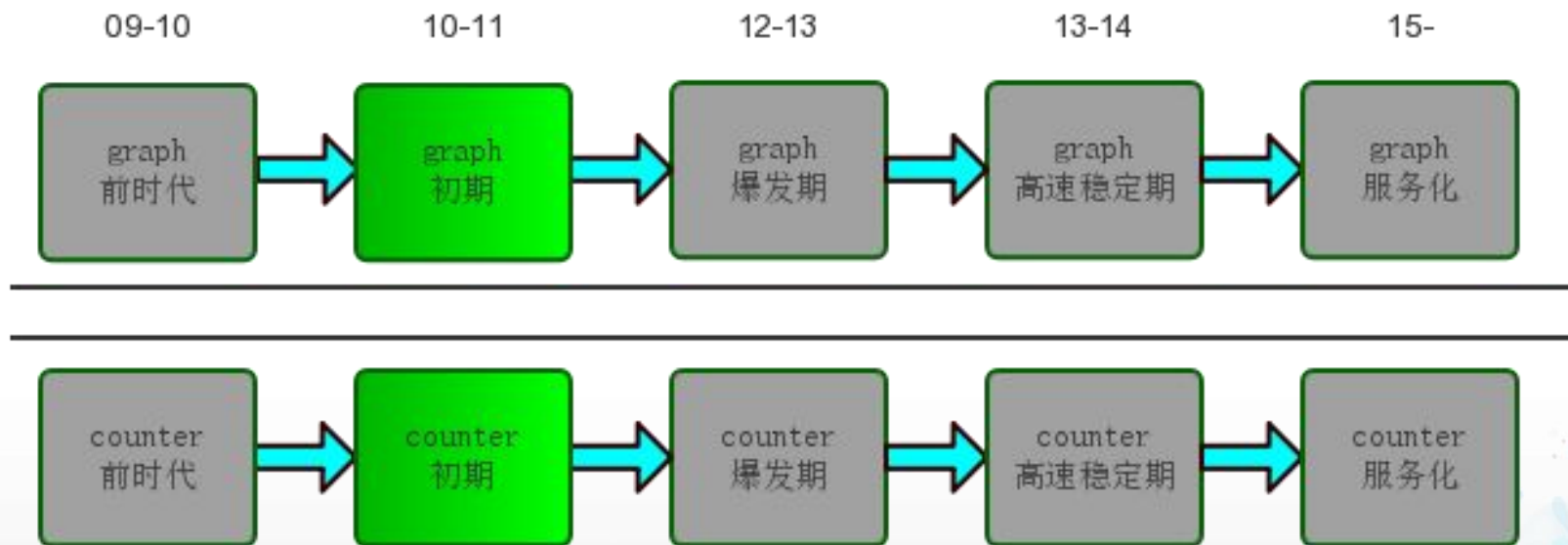
- 项目初期

- 30G- 日PV5kw-
- 技术选型 熟悉度
- 拼的是开发速度

- 产品需求与新技术相互促进



Redis 存储初期

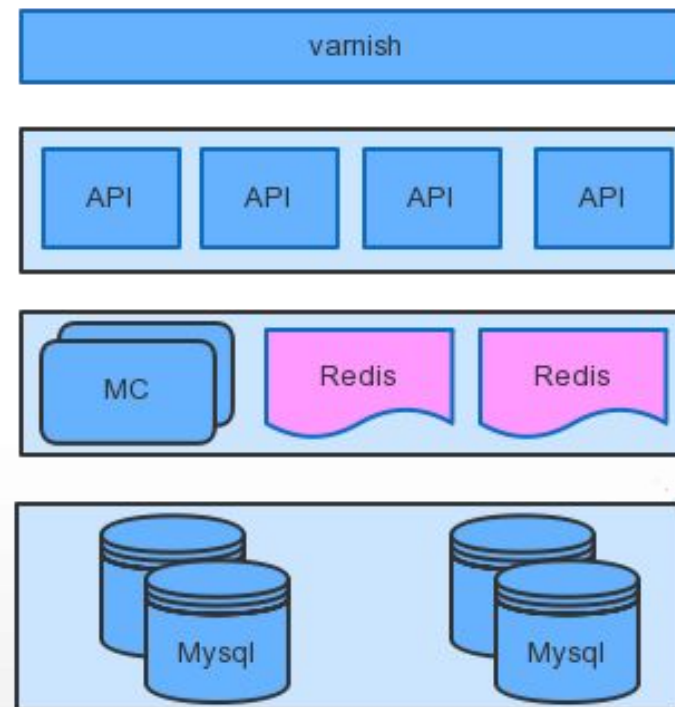


Redis初期

- Redis 2.0

- Graph存hash, 40G 10w TPS, 4 Server

- Counter: 20G 2w TPS, 2 Server



问题出现

- 2011年，初期使用经验不足
 - 数据分片过少，扩容困难
 - 部分数据类型使用不当，内存超预期
 - 多业务混放，拆分不便
- 可用性不够
 - 小业务初期没有slave，server故障→服务异常
 - 大业务挂载3-4个slave，高峰期write超时，请求失败
- 重启耗时，10-20分钟服务异常

解决方案

•容量规划

- 提前预估容量，上线前预拆足够的数据分片
- 选择合适的数据类型，慎用zset
- 业务独立存储，拒绝混放

| Redis 容量评估表 | |
|-------------|---------------------------|
| 需求 | |
| 1 | |
| 2 | |
| 3 | 业务名称 |
| 4 | 用途 |
| 5 | 数据类型 (List, String, Hash) |
| 6 | 单位 (user) |
| 7 | 当前单位容量 |
| 8 | 预估最大单位容量 |
| 9 | 当前数量 |
| 10 | 预估最大数量 |
| 11 | 当前峰值 (rps 读) |
| 12 | 当前峰值 (rps 写) |
| 13 | 预估最大峰值 (rps 读) |

解决方案

- 提高可用性

- 所有Redis全部增加Slave
- Master挂载slave不超过2个，采用M-S-S方式挂载
- 多IDC 单Master，复制同步

- 凌晨低峰升级，访问 IP→域名

- 不完美，但基本可work

问题升级

- 2011年底，Graph 100G+ 灵异事件
 - 凌晨3点低峰期，redis无征兆崩溃
 - 批量升级、扩容拆分，引发其他业务异常报警
- 多个slave严重负载不均，请求数最大差1-2个数量级，峰值 响应从 不足1ms→3ms
- 在线版本增多
 - 最多6个版本
 - BUG重复修复，运维困难

问题分析

- 崩溃：读写会用pageCache，导致redis进swap而崩溃
- 其他服务报警：复制 全量推送导致网络阻塞
- 负载不均：client通过域名访问，域名解析返回随机ip，结果连接不均衡，最终导致负载不均衡

问题解决

- 紧急方案

- 超过物理内存3/5 → 迁移端口
- 错峰升级/扩容 对网络仍然有一定冲击
- 开发ClientBalancer组件，保持域名下IP连接均衡，负载均衡

- 进一步优化方案：

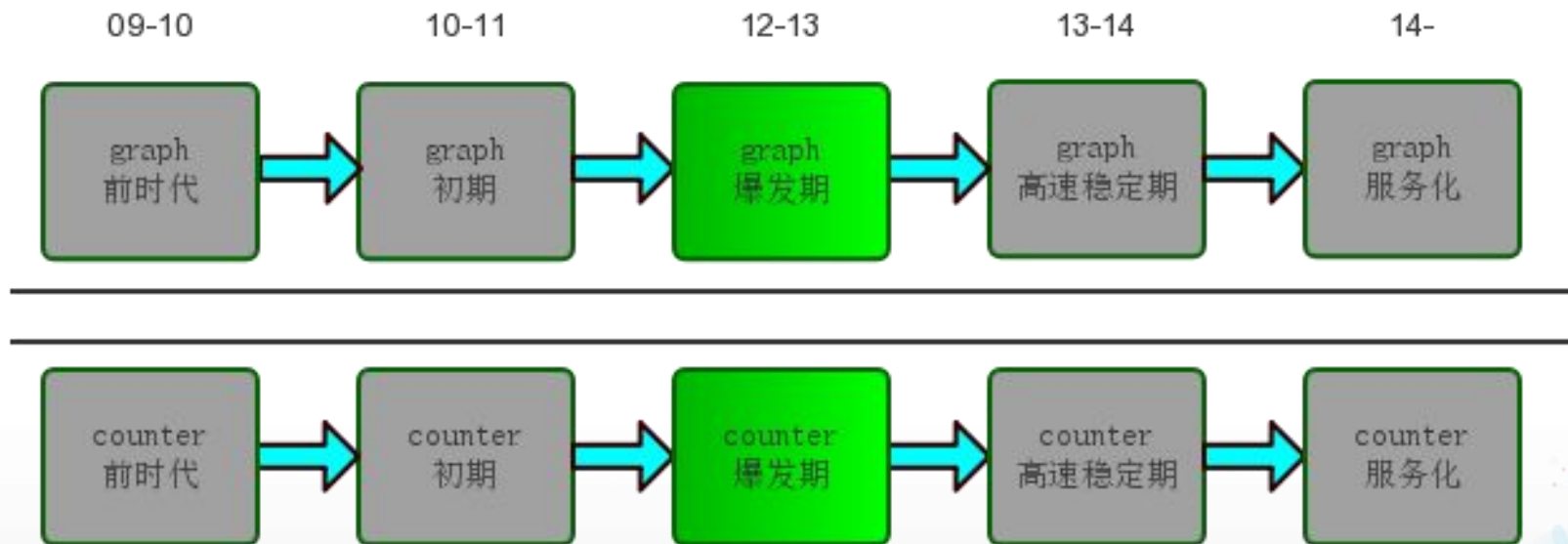
- 及时清理pagecache，减少对正常业务影响
- Aof去掉rewrite，改用rotate
- 类似mysql，独立IO线程对rdb、aof转发复制（社区版psync，repl-backlog-size，repl-backlog-ttl）
- 支持热升级，避免重启，提高可运维性
- Others...

小结

- 小规模 50G 1-2个集群
 - 人肉运维
- 中规模 100G+, 3+集群
 - 可运维性->重要
 - 开源组件->熟悉架构实现

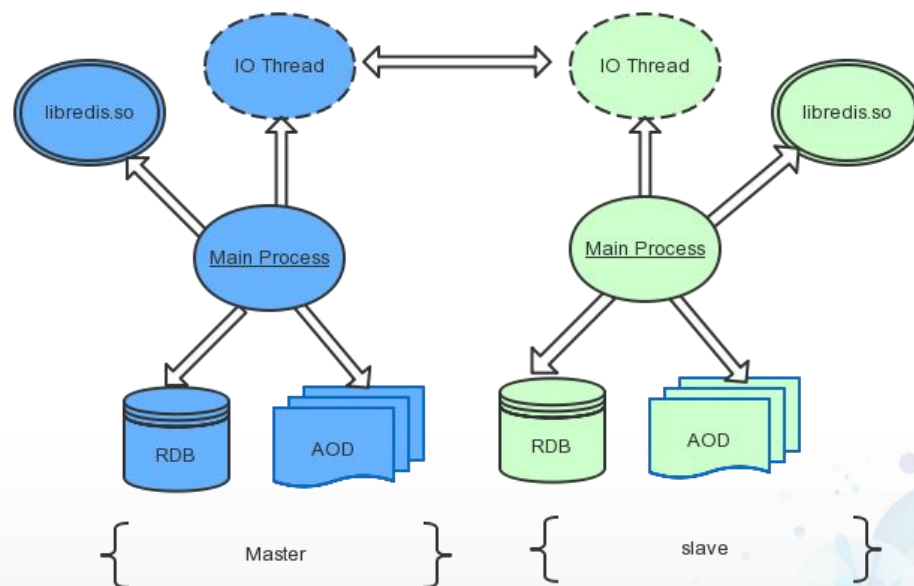


Redis 存储爆发期



Redis存储爆发期

- 完全增量复制
- 在线热升级
- SLAVE均衡访问
- 大量子业务切入
- 单业务数百G稳定



问题出现

- 2013年，Graph海量规模
 - 数据T级，MS 十T级
 - 数百台server，而且还在快速增加
 - Graph用Hash结构，存储效率不高

问题出现

- Counter 业务增加，增长迅猛
 - 日增：计数亿条 内存5G+
 - 总数据百G级， MS T级
 - Feed请求 计数近百倍读放大， 高峰超时报警
 - 存储效率低 <30%

2013000001.rep
800

2013000001.cmt
360

2013000001.like
1000

2013000001.read
10000

问题出现

- 占用机器增加迅猛，成本合理性需要考虑
- 部分机房机架饱和

解决方案

- Graph

- 定位: storage→cache
- 定制: hash→Longset

- Counter

- 定制cdb, 通过table分段存储计数
- 一个KV存多个计数

20130000001.rep
800

20130000001.cmt
360

20130000001.like
1000

20130000001.read
10000

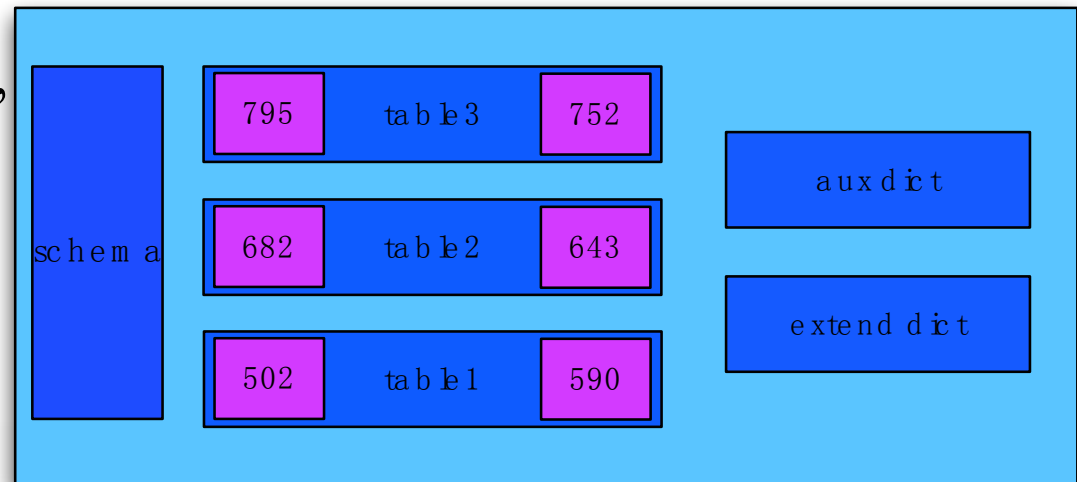


20130000001
800|360|1000|10000

解决方案

- Counter存储结构

- $cdb = schema + tables$
- 计数double-hash寻址, 消除冗余robject存储结构
- 冲突过多aux-dict
- 数值过大extend-dict



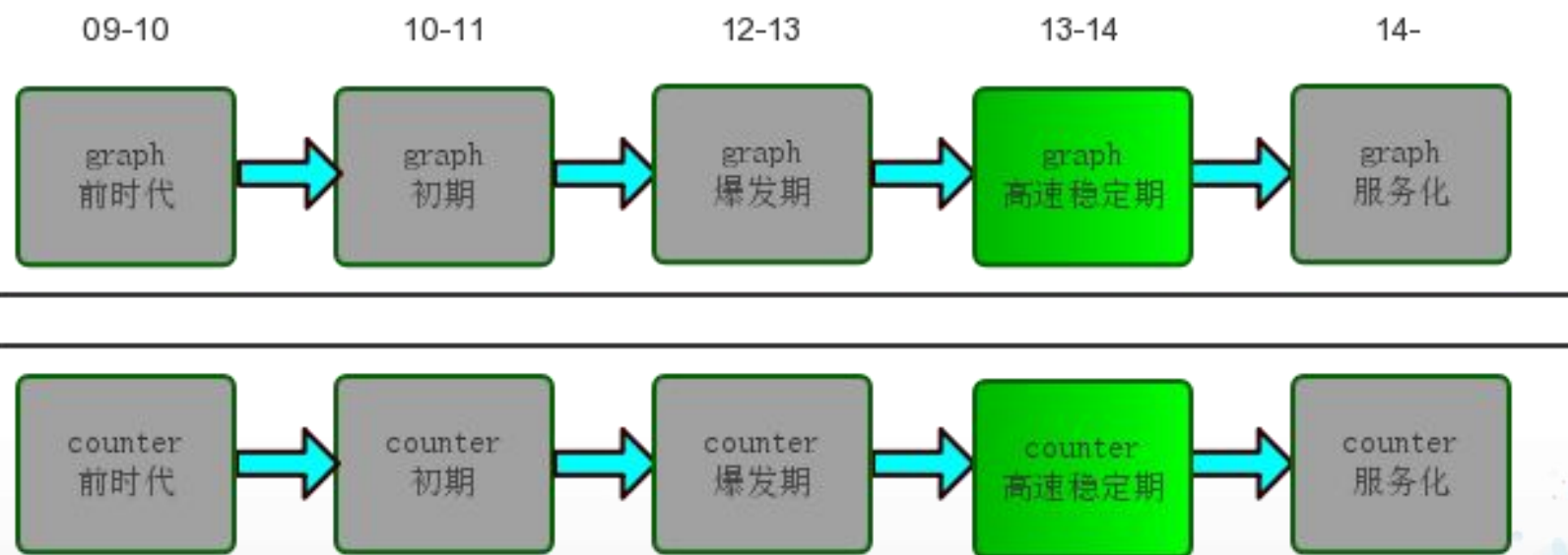
- 多管齐下, 节省数百台机器
- 下线低配server, 寻找廉价新机房

小结

- 量变->质变，极端业务定制
- 大规模集群 T级 3+idc 成本
 - 单个请求成本
 - 总拥有成本

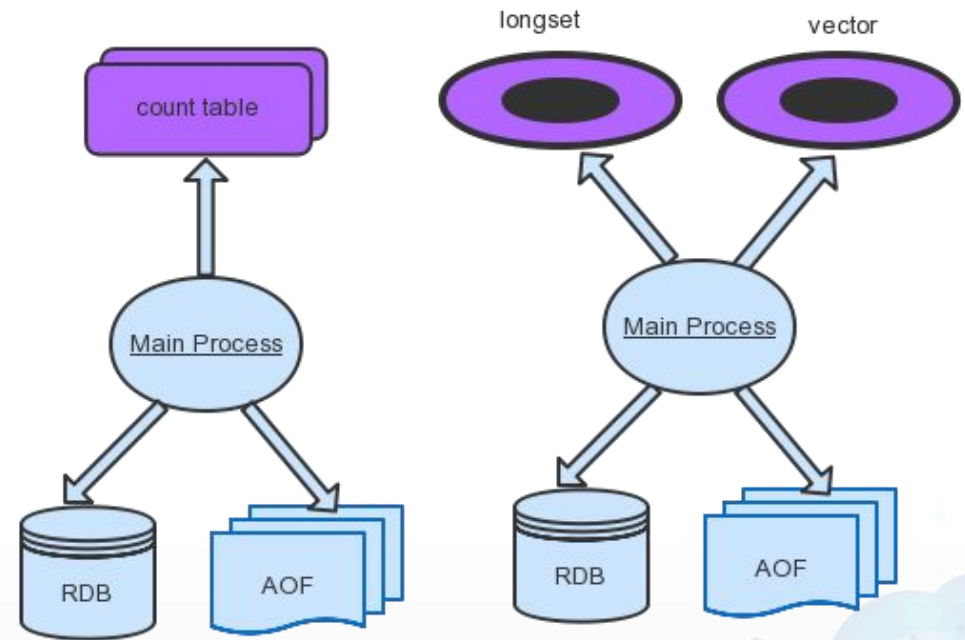


Redis 存储高速稳定期



Redis存储高速稳定期

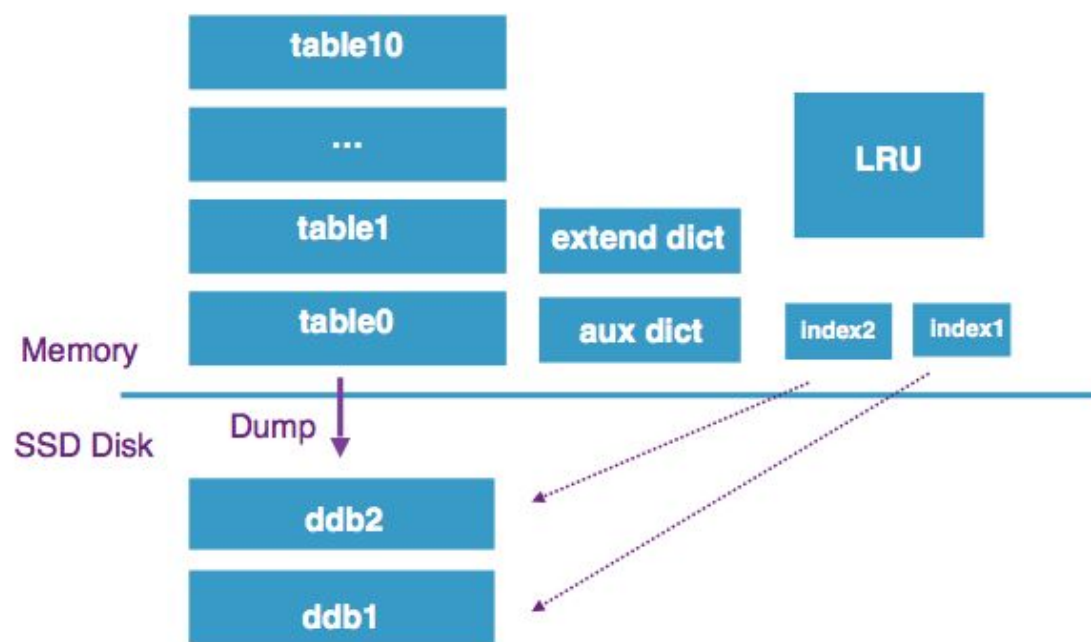
- Graph 定位cache
定制longset
 - 内存降为 1/10
 - 性能接近
- Counter 定制cdb
 - 内存降为 1/5
 - 性能增3-5倍



Redis存储高速稳定期

- 继续定制

- Counter 落地SSD，容量提升20倍，8个月→10年
- Vector
- Others...



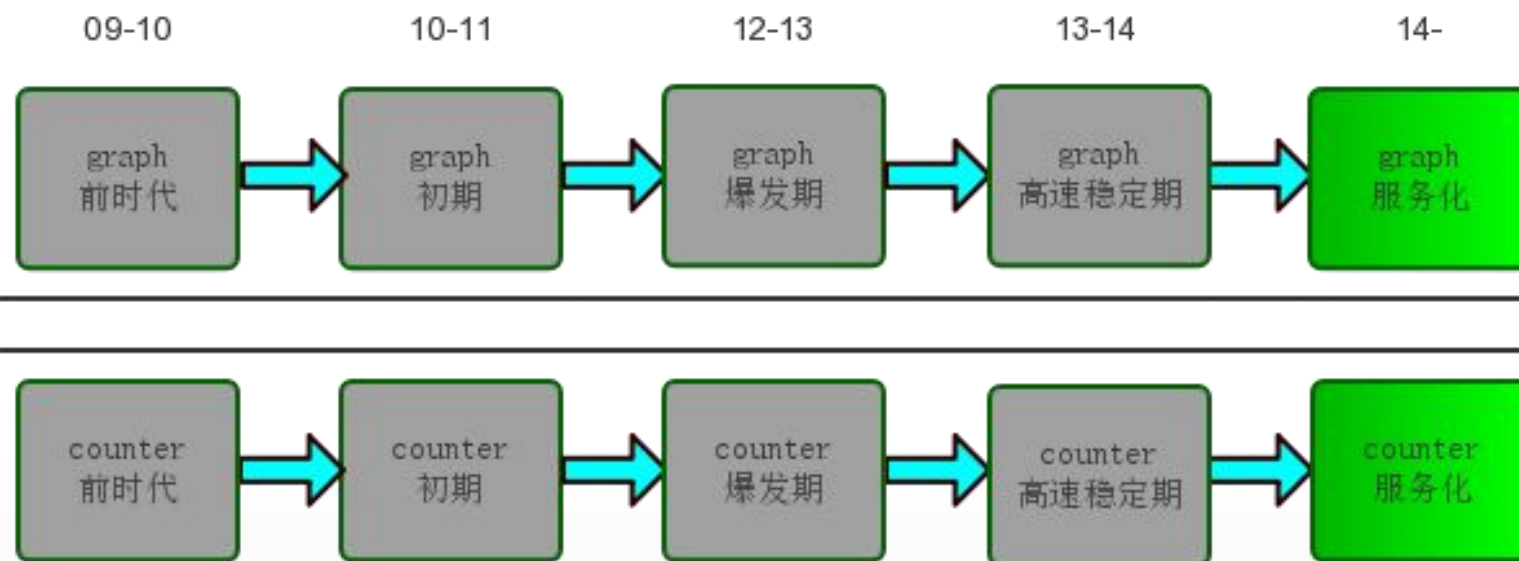
问题出现

- 2014, SLA 目标6个9
 - 数千关联Server 6+IDC 跨地域分布
 - 海量数据 24T+
 - 峰值 5000w+ TPS, 响应毫秒级
 - 硬件/网络故障时有发生, 如何实现?

问题解决

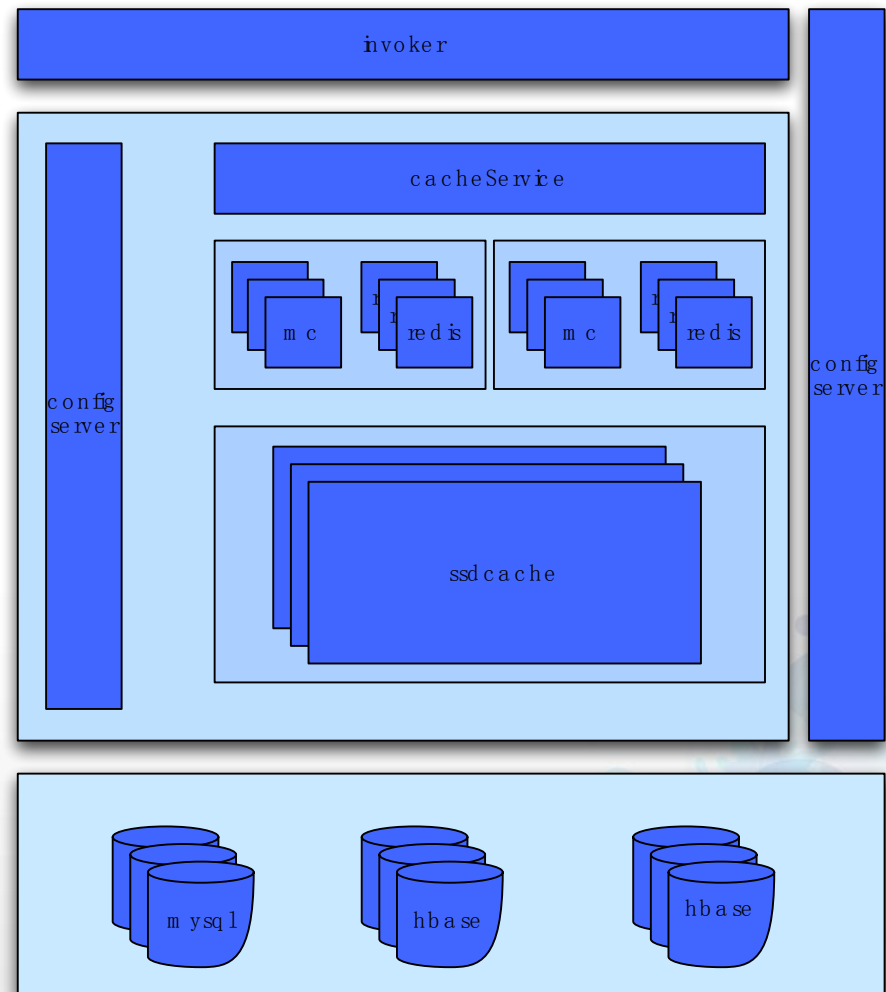
- 资源服务化 →
 - Configserver 用于服务的发布与订阅
 - CacheService 用于集群管理
 - ✦ 数据路由
 - ✦ 负载均衡
 - ✦ 数据在线迁移
 - ✦ 服务治理（生命周期 故障转移etc.）
 - 运维标准化、自动化（扩/缩容etc.）

服务化



服务化

- 服务化 →
 - 业务服务化 motan ✓
 - 资源服务化 →



小结

- 避免过早优化，小步快跑
- 架构没有最好，只有更适合



一些经验

- 结合发展阶段选择最合适的技术和架构，避免过早优化
- 拥抱需求，需求、技术相互促进
- 解决问题的root cause

Q&A

THANKS

SequeMedia
盛拓传媒

IT168.com
www.it168.com

ChinaUnix

ITPUB