

爱奇艺移动服务端缓存系统的演进

爱奇艺移动服务端基础架构组 王程明



移动服务端应用的场景和特点

页面展现服务

大量的只读操作
需要对接很多业务方进行数据获取



数据更新延迟

剧集更新不能太慢
热点事件头部内容实时展现



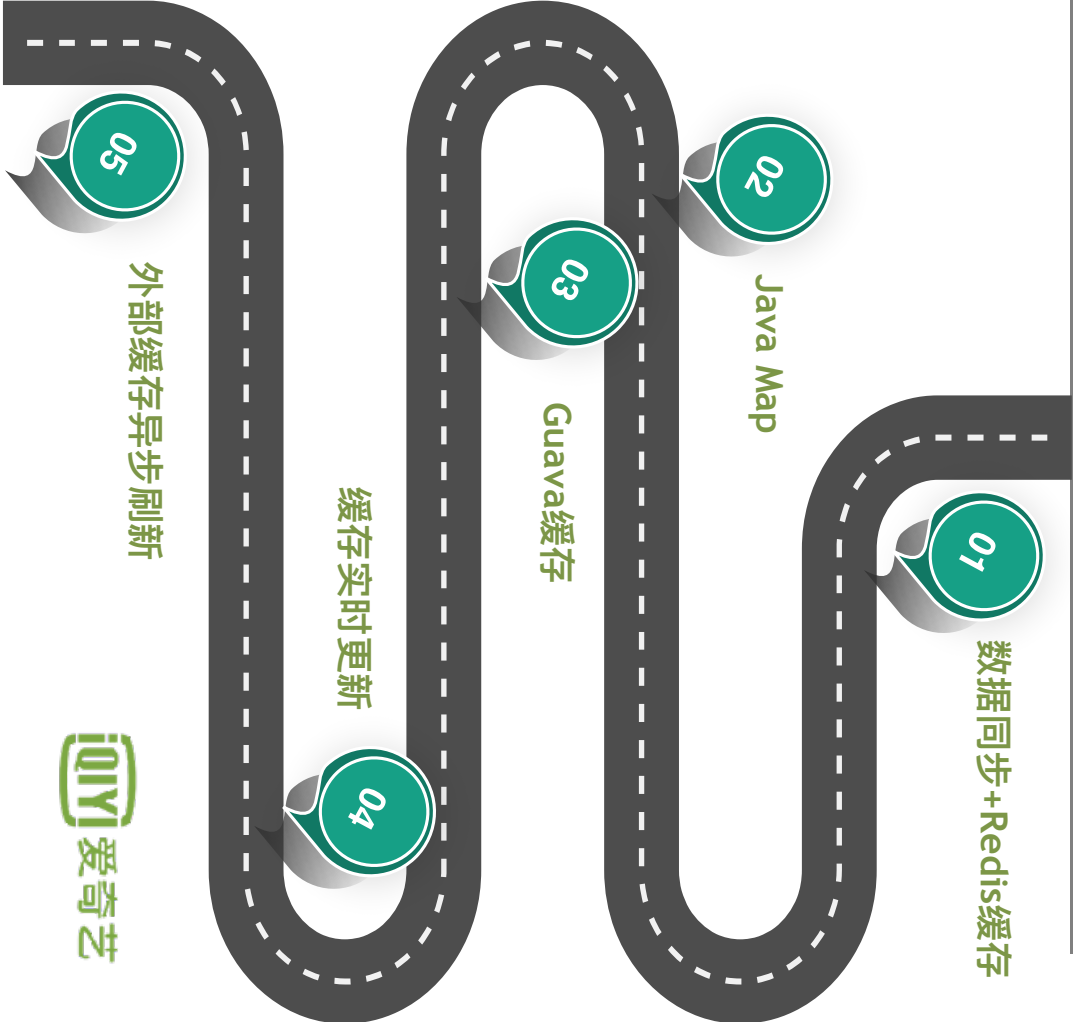
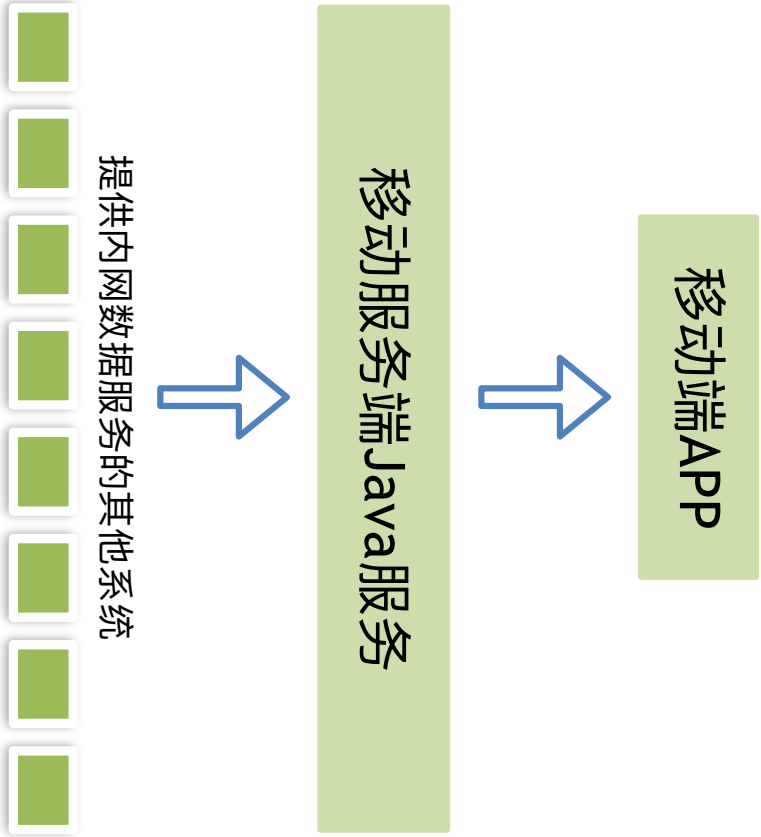
依赖的数据服务容量较小

新兴业务接口服务QPS容量不高

依赖的数据服务不稳定

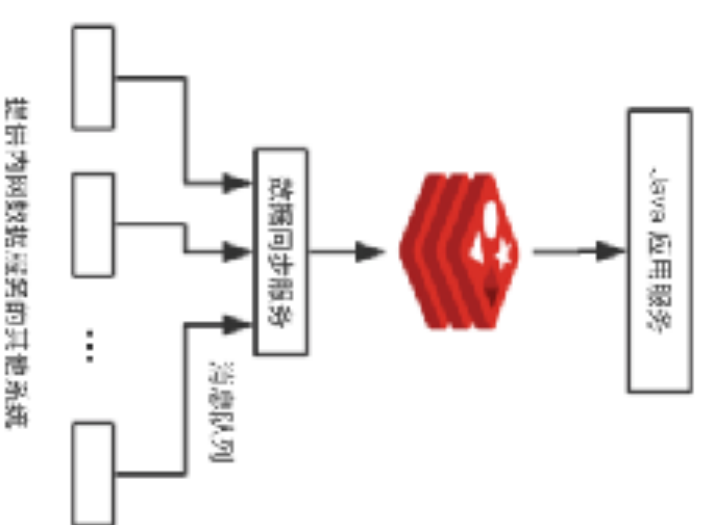
接口服务成功率异常
机房断网，天灾人祸

应用服务缓存优化



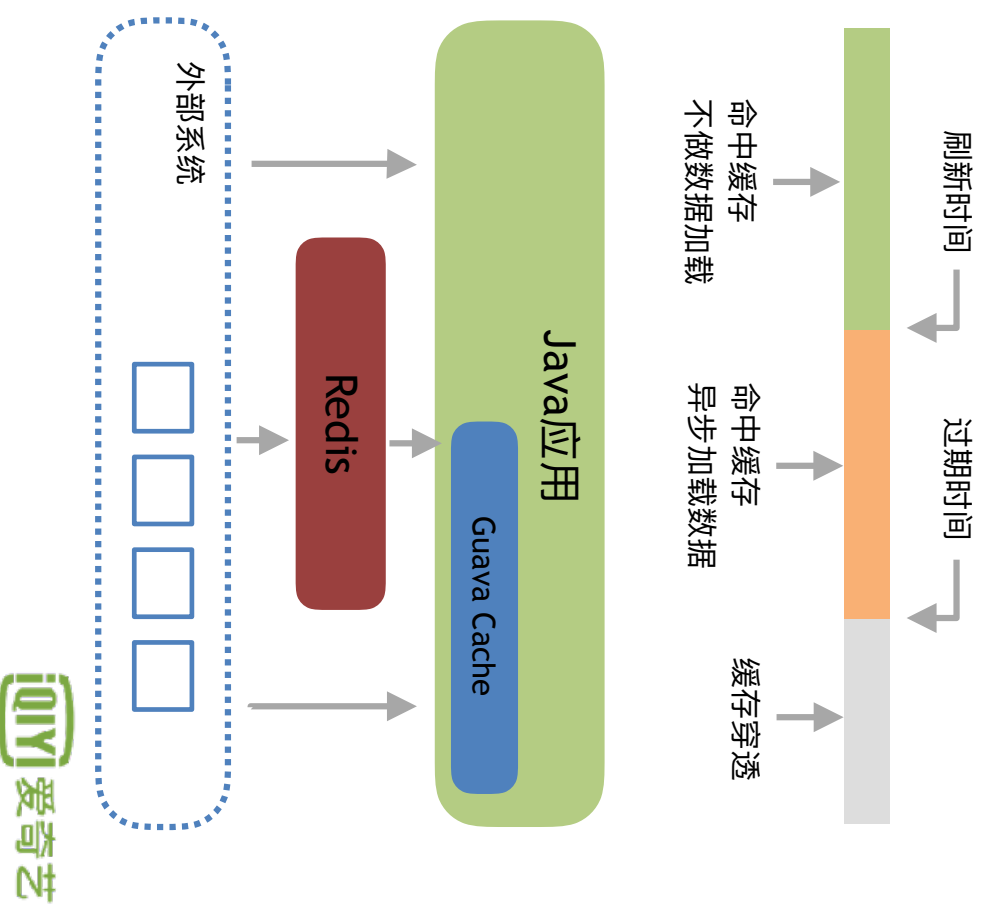
Redis缓存+数据同步

- 实现方式
 - 数据同步服务通过消息队列将全量数据同步到Redis
 - Java应用服务直接从Redis中获取数据
- 优点
 - 数据更新快
 - 不依赖其他系统的接口成功率
- 问题
 - 严重依赖Redis的稳定性和性能
 - 并不是所有的外部系统都提供通知服务
 - 全量数据庞大



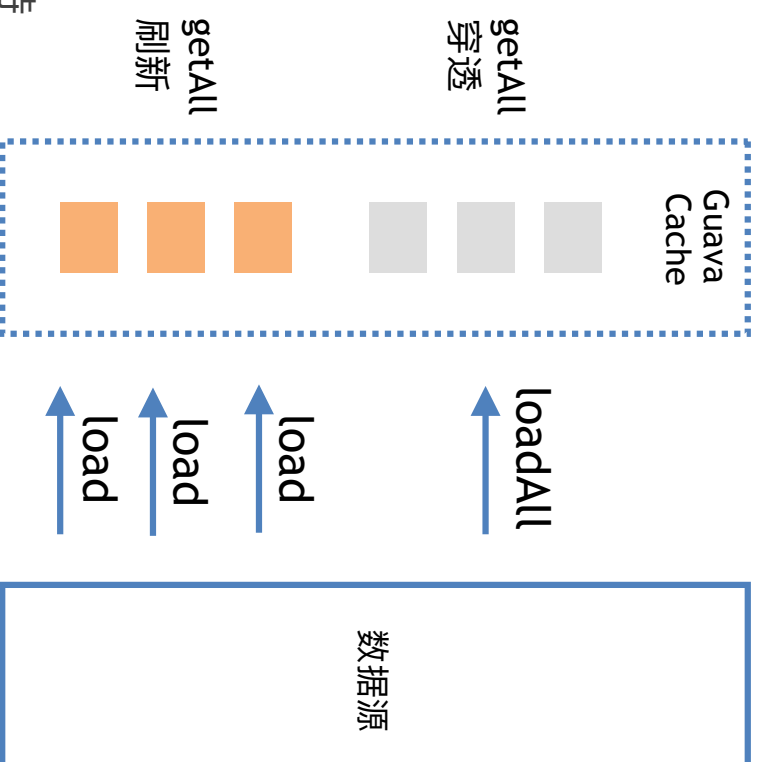
从 Java Map 到 Guava Cache

- 应用内部缓存
 - 阻拦流量
 - 屏蔽故障，服务降级
- Guava Cache
 - 缓存刷新时间&缓存过期时间
 - 支持批量获取
 - 并发访问相同的Key只执行一次数据加载
 - 缓存规模上限
- 问题
 - 批量Key加载退化为单个Key加载
 - 缓存污染
 - 缓存命中率优化
 - GC压力大
 - 缓存穿透



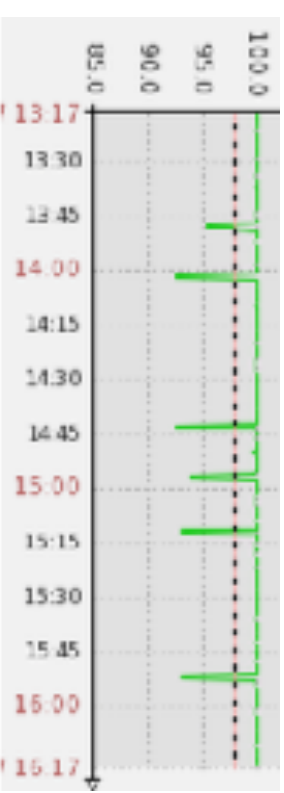
Guava Cache

- 批量Key加载退化为单个Key加载
 - 穿透情况调用一次 loadAll 函数获取
 - 刷新情况每个Key调用一次load
 - load请求入队列，集中处理
- 避免误用：缓存污染
 - 应用逻辑修改了缓存对象数据
 - Clone & Bean映射 & 代码Review
- 避免误用：Cache Absent
 - 请求成功但是数据为空，缓存absent对象
 - 请求失败抛出异常，不缓存
- 避免误用：加载无限等待
 - Guava Load方法无限等待造成应用线程无限等待



Guava Cache - GC调优

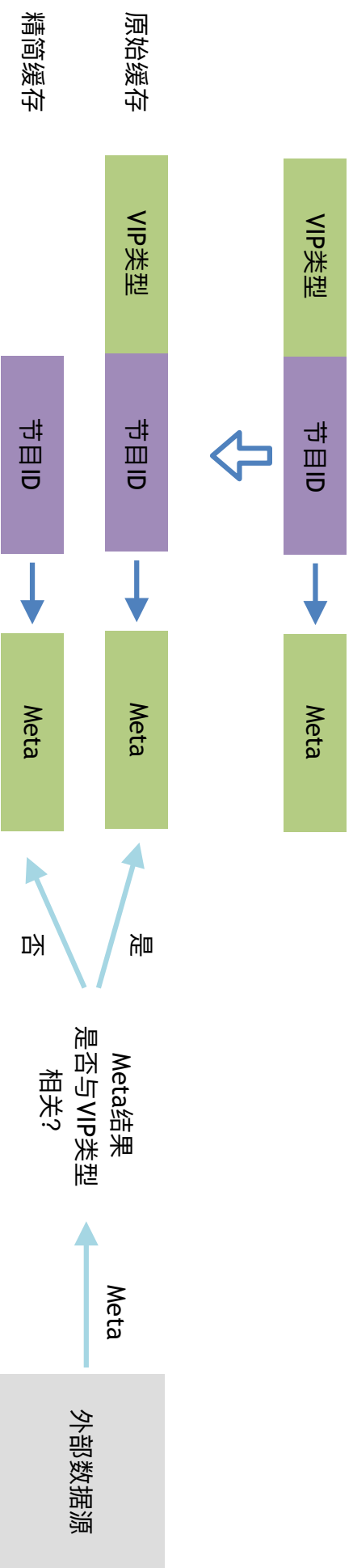
- 大量的缓存导致成功率间歇下降
- JVM参数设置
 - 堆大小 22G
 - GC算法使用CMS (UseConcMarkSweepGC)
 - 开启了降低标记停顿 (CMSParallelRemarkEnabled)
 - 年轻代并行收集 (UseParNewGC)
 - 年轻代和老年代的比例为1:2 (NewRatio=2)
- 客户端超时时间设置为3秒，超过3秒的GC会引发成功率下降
 - 强制在remark阶段和FULL GC阶段之前先在进行一次年轻代GC
 - CMSScavengeBeforeRemark
 - G1 ?



```
[1 CMS-remark: 14242288K(13379136K)] 205208820K(22209776K) 4.4227220 secs]
[1 CMS-remark: 1426578K(13379136K)] 20327164K(22209776K) 4.1522870 secs]
[1 CMS-remark: 14489796K(13379136K)] 15898357K(22209776K) 8.1757770 secs]
```

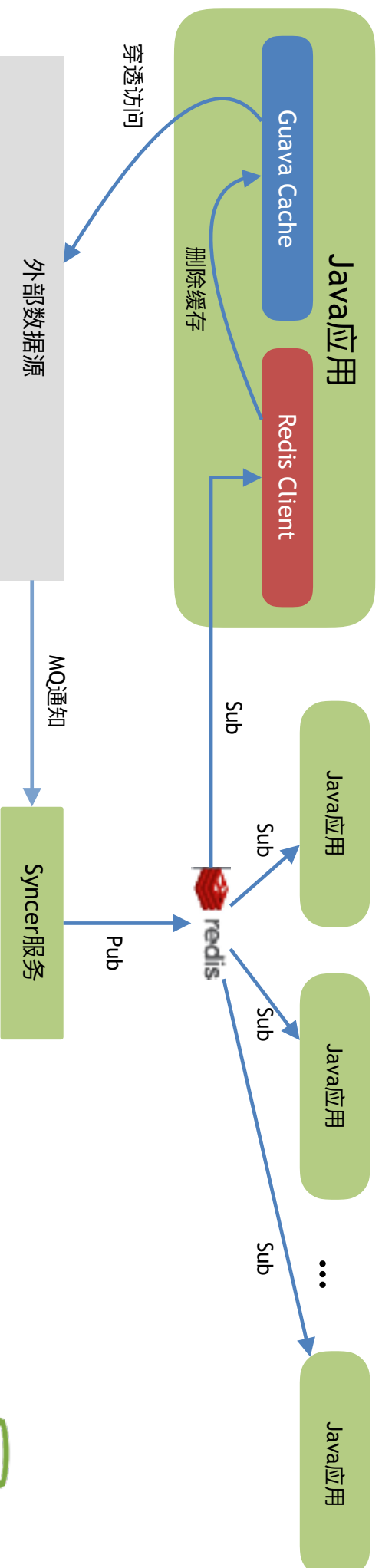
Guava Cache - 缓存命中率优化

- 缓存Key每增加一个维度规模就增加N倍
- 让数据源决定缓存策略



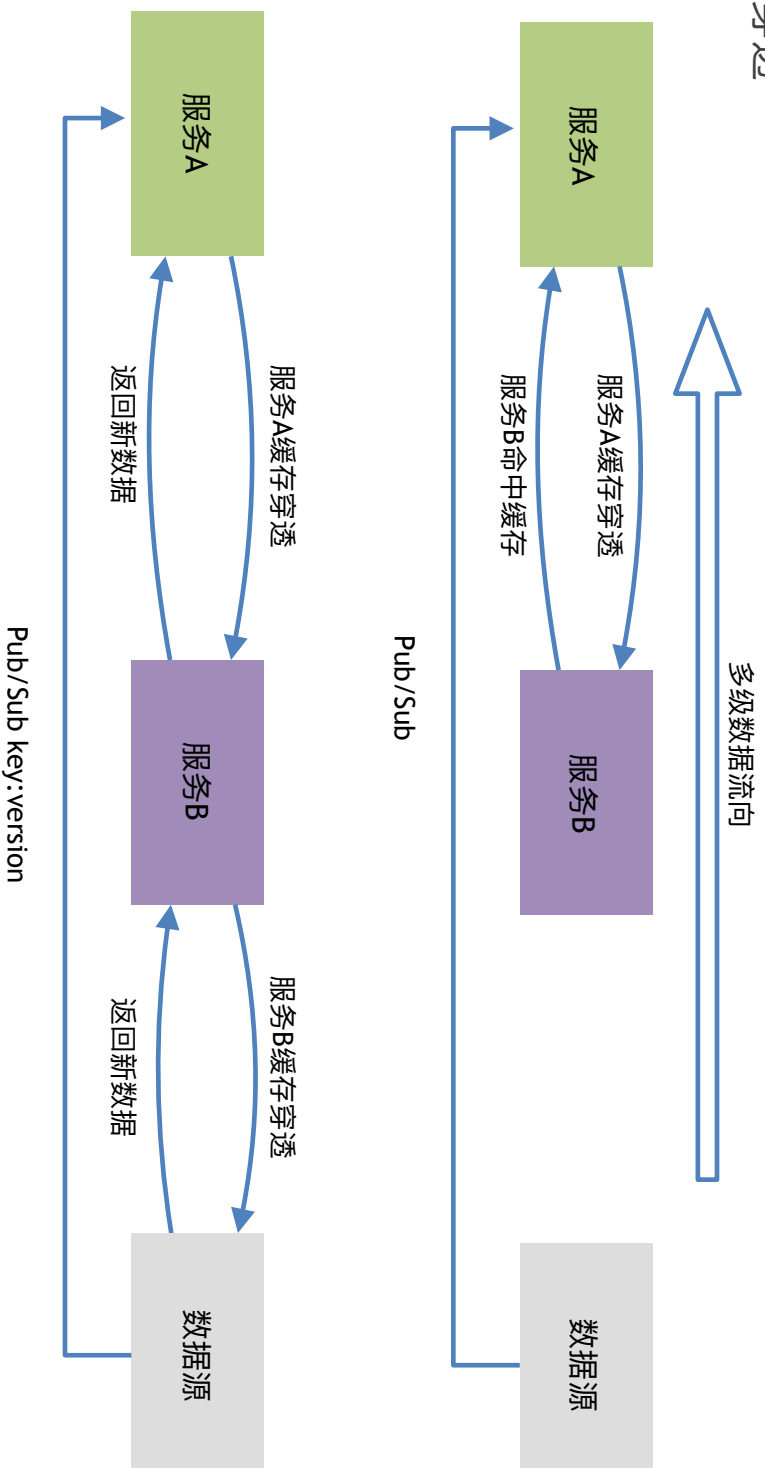
Guava Plus: 缓存实时更新

- 电视剧每天定时更新，不能跳票
 - 缓存刷新时间改小 - 成本过高
 - 缓存刷新时间改大 - 更新时间过长，评论区要炸
- Redis pub/sub
 - 消息不可靠？缓存刷新机制保底



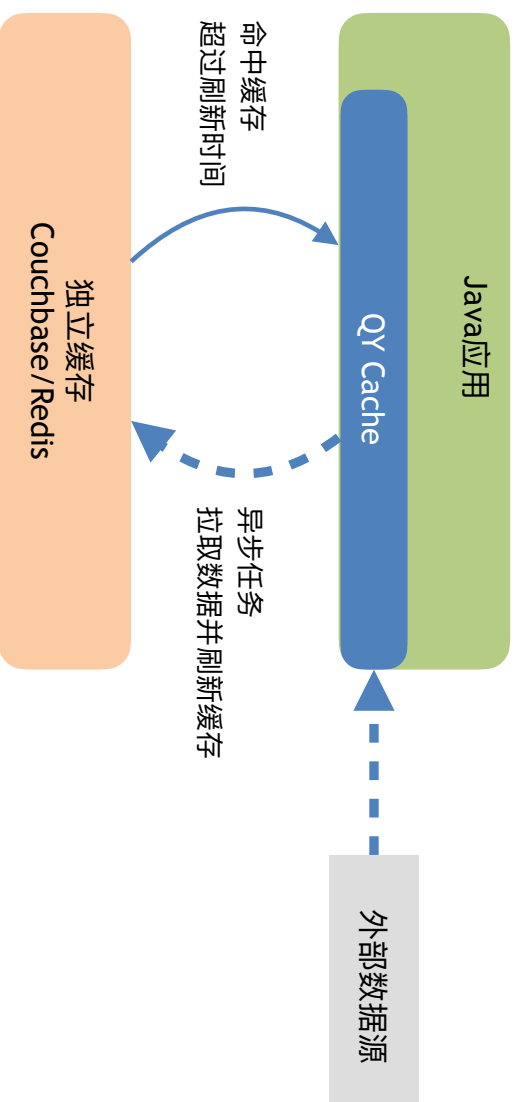
Guava Plus: 缓存实时更新

- 微服务化改造引发不同层级服务之间的缓存不同步
- 版本号机制穿透



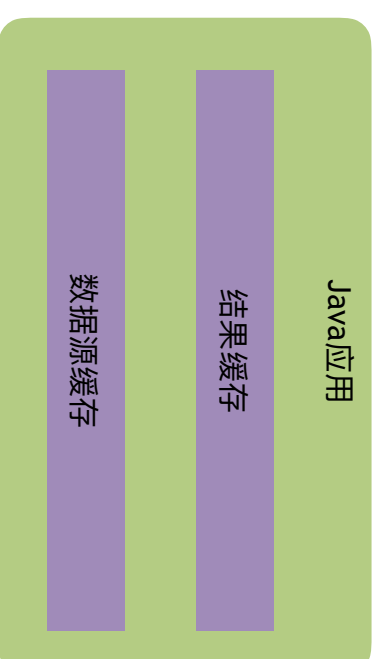
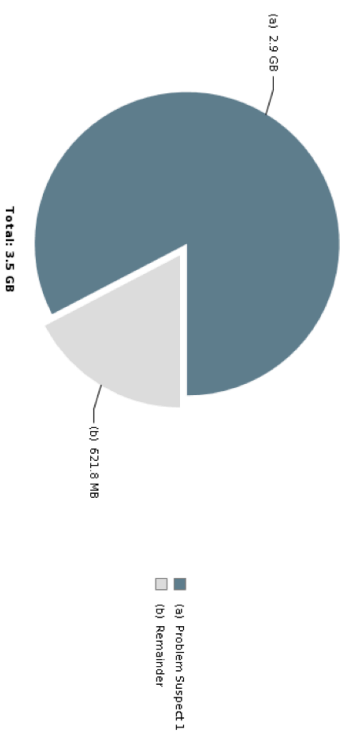
缓存雪崩

- 应用服务上线造成缓存雪崩
- 独立缓存无法提供刷新功能，无法兼顾命中率和实效性
- QYCache 独立缓存版本的GuavaCache
 - 支持Redis&Couchbase
 - 设置缓存过期时间
 - 设置缓存刷新时间



Cache 不是万金油

- 缓存状态不一致
- 不必要的缓存导致性能下降
- 两级缓存：数据源缓存+计算结果缓存
- 缓存单条数据较大
- 计算结果Key取值范围很广
- 服务成功率偏低，GC卡顿



数据源服务

