荔枝

荔枝架构实践与演进

荔枝APP架构师 - 黄全

huangquan@lizhi.fm

荔枝

演讲目录

- 品牌介绍
- 架构演进
- 问题与方案
- 未来展望

荔枝

品牌介绍

平台 Platform

海量主播与声音互动功能承载平台

工具 Instrument

人人都能展现自己的声音才华

社群 Community

众多声音自媒体组成的高活跃度

官网：www.lizhi.fm

300w+
月均活跃主播

1亿+
音频节目数量

4000w+
月均活跃用户

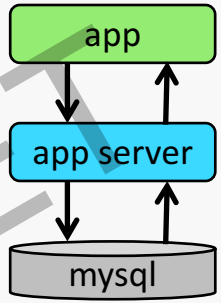
2亿+
全球注册用户

架构演进

v1.0版架构

特点

- APP 直连服务器
- 服务端是单体架构



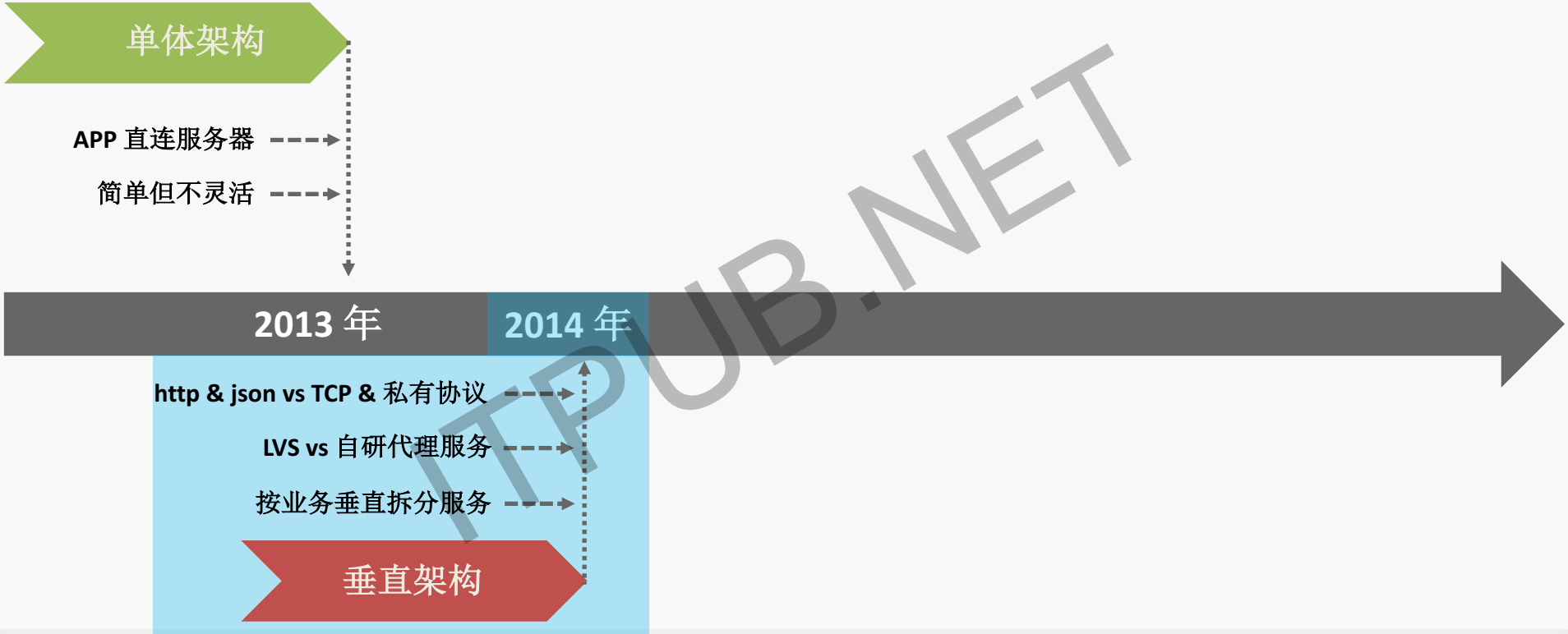
面临问题

- APP 直连服务器不灵活
- 上线后3个月，用户数突破 **100万**，访问量上涨，服务器压力增大

解决方案（见v2.0版架构）

- 分压：APP 与 app server 之间加入代理服务 app proxy，分发请求给多台 app server，分摊压力。
- 拆分：对 app server 按功能进行拆分，数据操作及业务逻辑由 dc server 负责

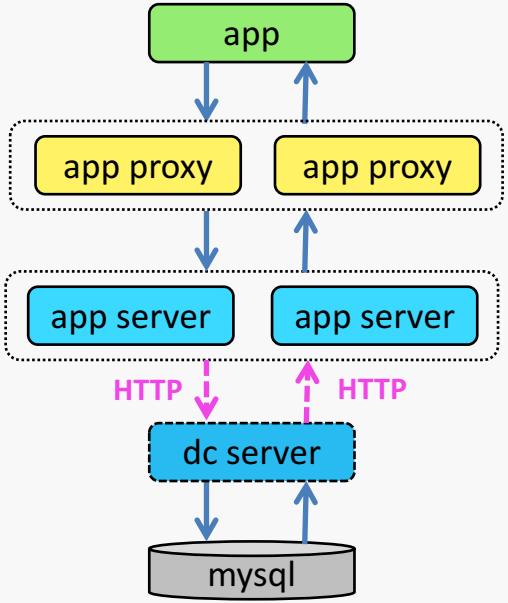
架构演进



v2.0版架构

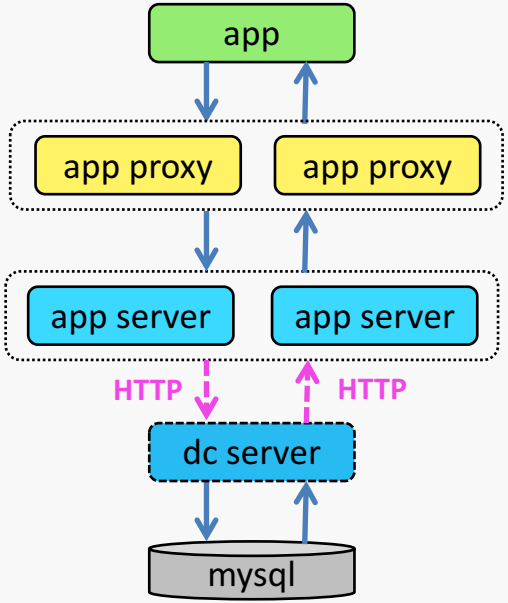
- 特点
  - app server 与 dc server 直连，硬编码 IP 和端口
  - 采用 netty(http) + json 方式交互

- 面临问题
  - dc server 水平扩展时，不能动态化
  - 系统间采用 http 交互时，通讯效率较低，并且数据包较大
  - json 解析速度较慢、体积较大



v2.0版架构

- 解决方案（见v3.0版架构）
  - 引入Linux虚拟服务器集群系统 LVS
  - 采用 TCP 取代 HTTP， 定制私有协议取代 json

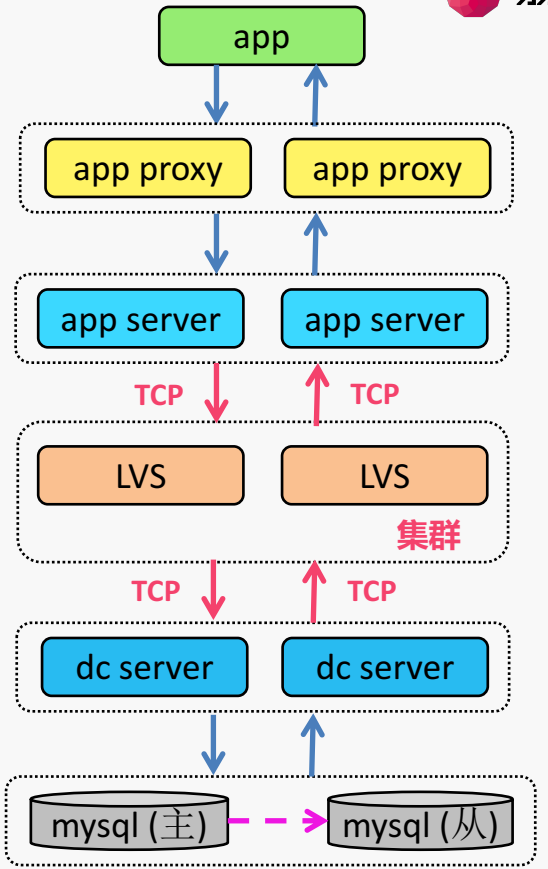


v3.0版架构

- 特点
  - 使用 LVS 集群分发请求

- 面临问题
  - LVS 没有服务注册功能，业务发展快速，人手缺乏，运维起来比较困难

- 解决方案（见v4.0版架构）
  - 采用自研代理服务 dc proxy 取代 LVS



v4.0版架构

特点

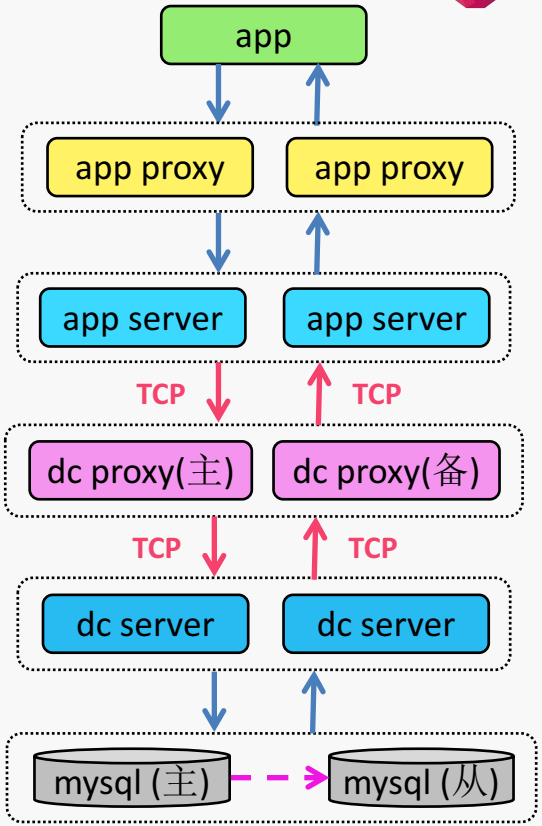
- dc proxy 使用 VIP 与其他服务连接。
- app server 与 dc server 依然是单体架构。
- 支持水平扩展后端服务，但需要重启 app proxy、dc proxy。

面临问题

- 用户量、访问量持续上涨，系统访问压力变大

解决方案（见v5.0版架构）

- 按业务垂直拆分 app server 和 dc server



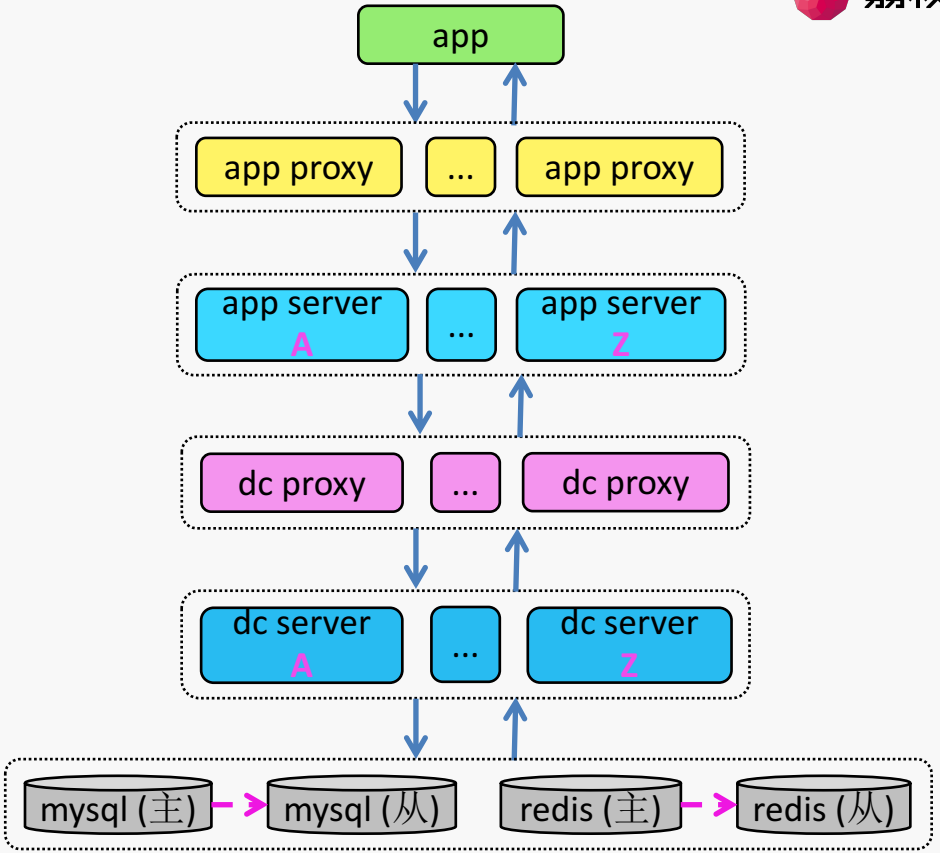
v5.0版架构

- 特点
  - 服务按业务拆分、支持水平扩展
  - 整体架构能抗得住一定的访问压力

  （2014年10月用户量突破 **1千万**）

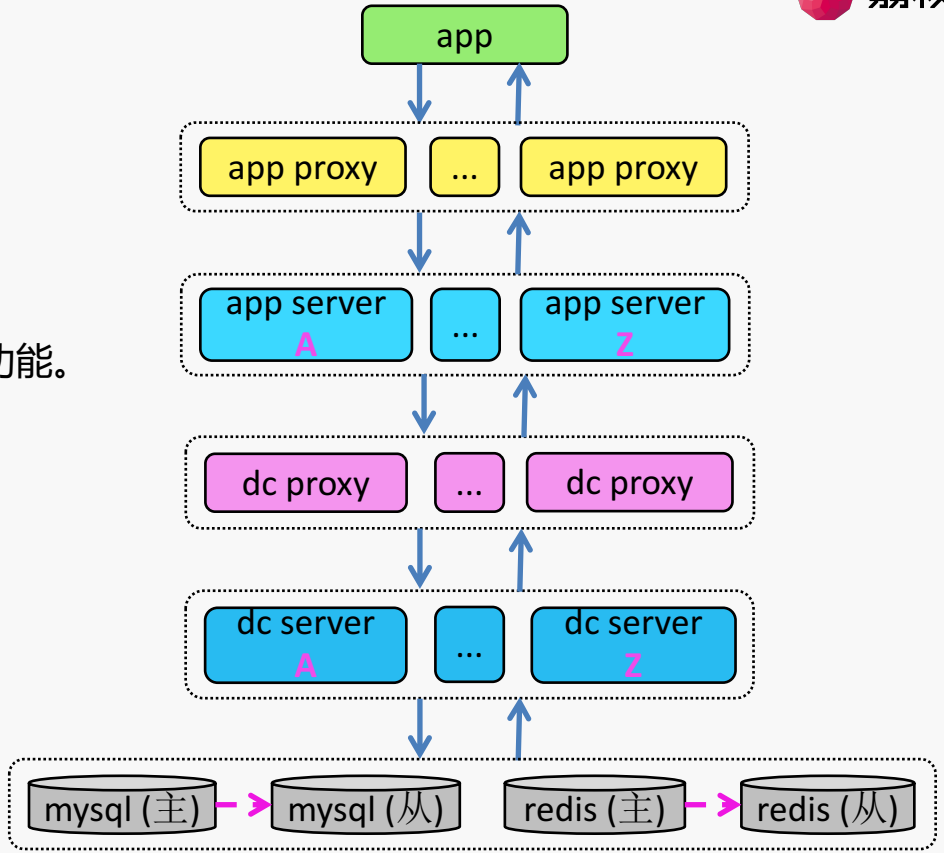
- 面临问题
  - 服务的配置不能做到热更
  - 不同业务的 dc server 之间需要交互，

    需要考虑向微服务化方向发展

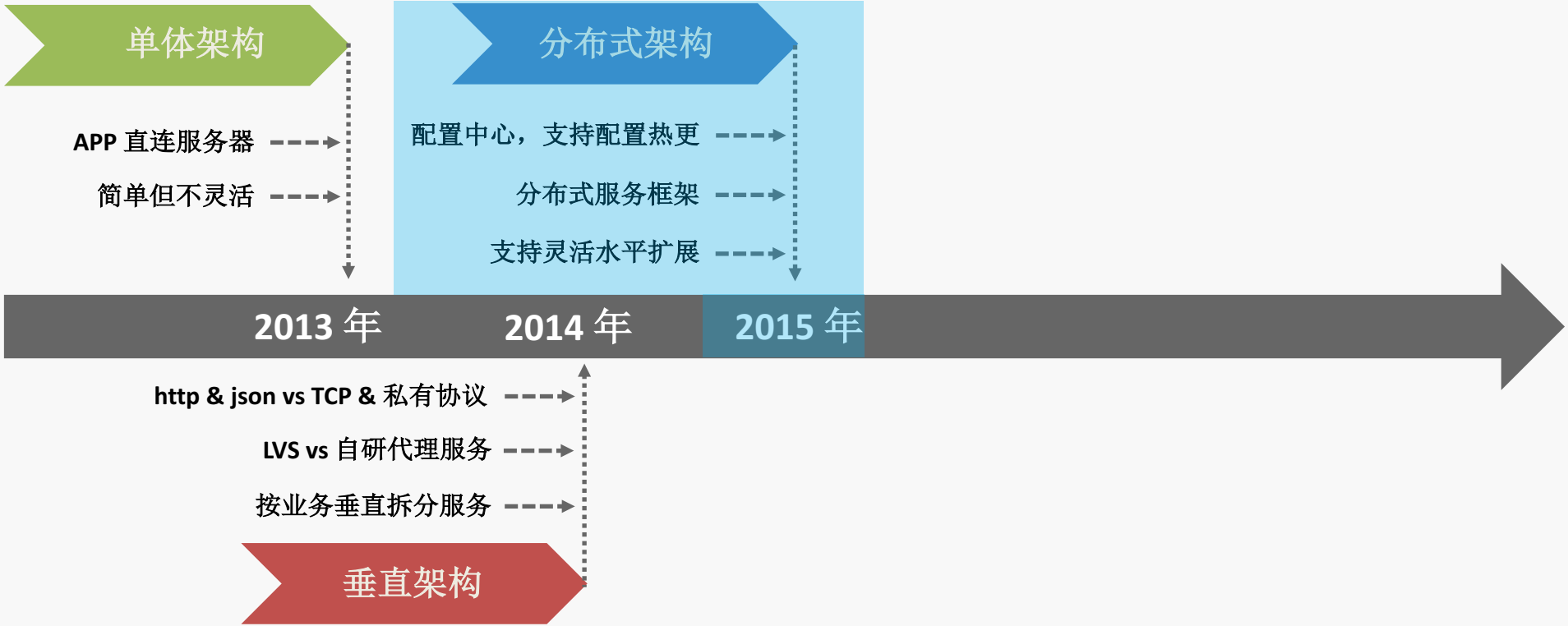


v5.0版架构

- 解决方案（见v6.0版架构）
  - 开发配置中心 config server，实现配置热更。
  - 开发分布式服务框架 lz-RPC，封装远程调用功能。



架构演进


单体架构
APP 直连服务器
简单但不灵活
分布式架构
配置中心，支持配置热更
分布式服务框架
支持灵活水平扩展
2013 年
2014 年
2015 年
http & json vs TCP & 私有协议
LVS vs 自研代理服务
按业务垂直拆分服务
垂直架构

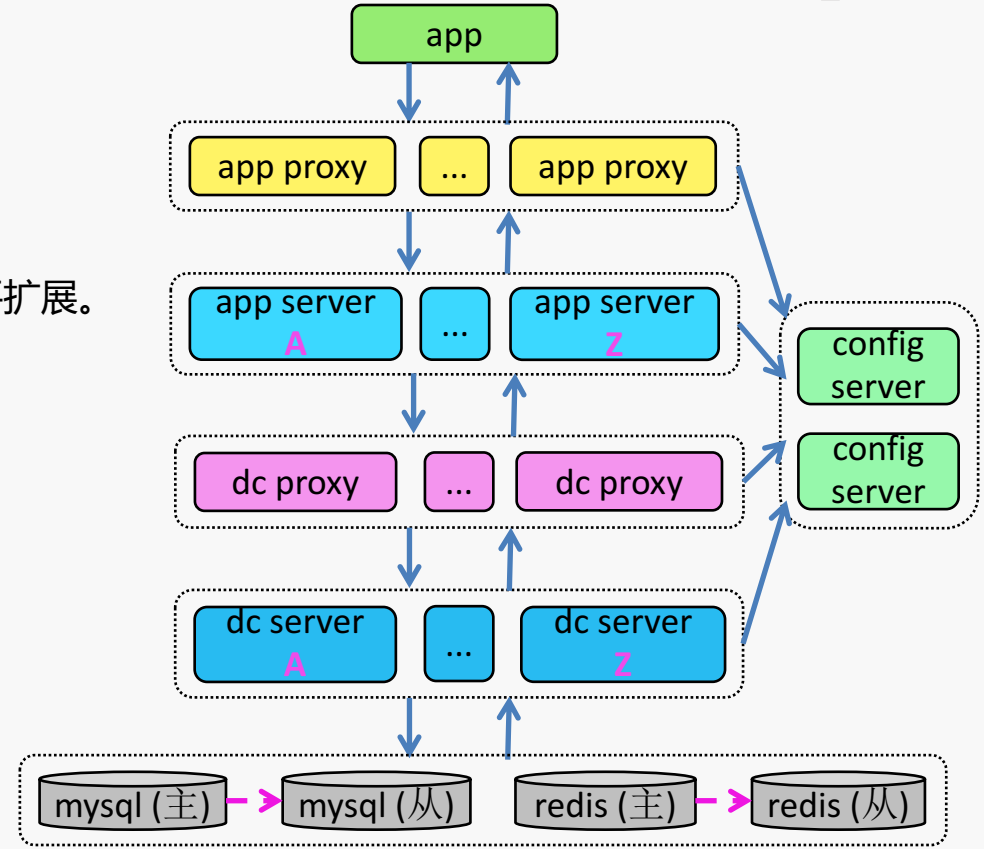
v6.0版架构

- 特点
  - app proxy, app server, dc proxy,

    dc server 配置实现了热更，能灵活水平扩展。

    （2015年9月用户量突破 **5千万** ）

- 面临问题
  - mysql、redis 操作的重复代码多
  - mysql、redis 慢操作不能及时报警
  - 各个服务的数据源配置分散，难以管理
  - 考虑跨机房数据操作和数据同步

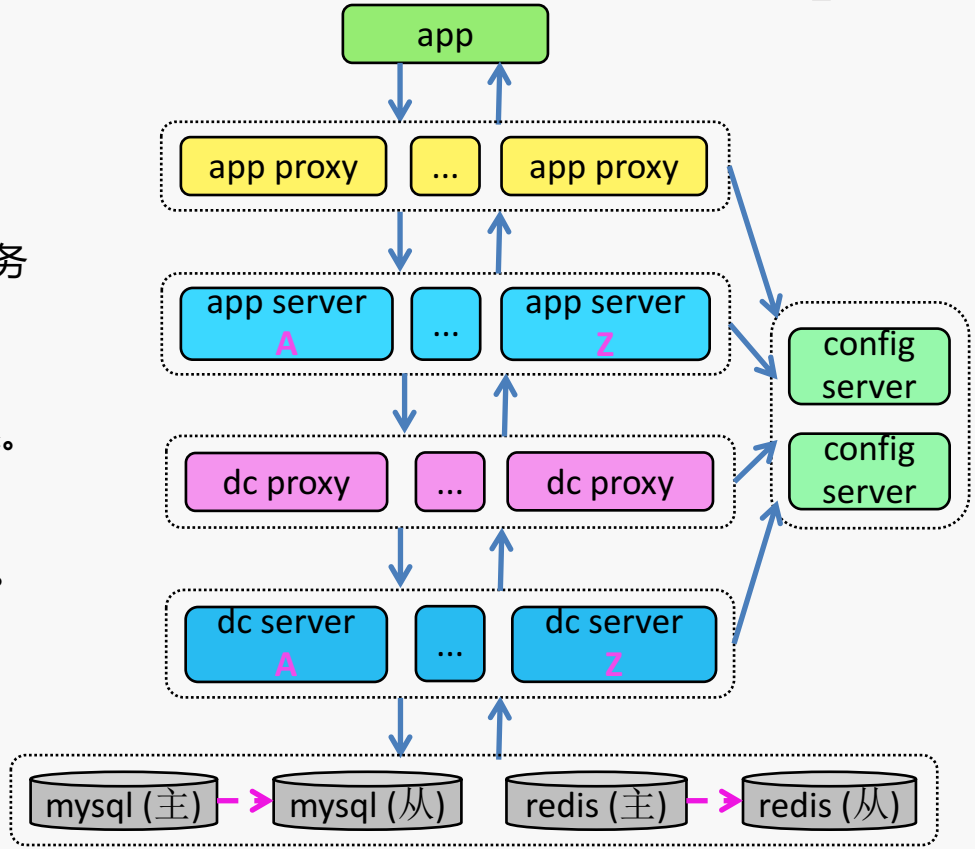


v6.0版架构

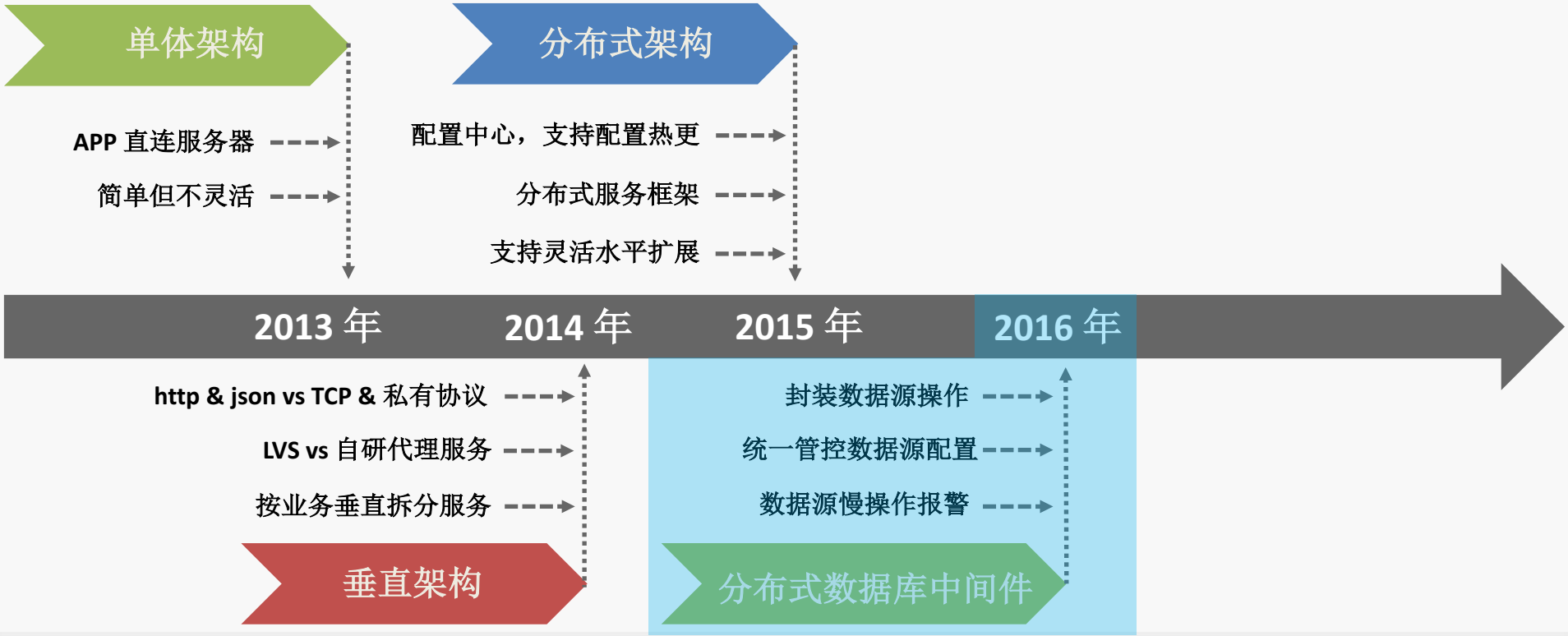
- 解决方案（见v7.0版架构）
  - 开发分布式数据库中间件 data store 服务

特点：

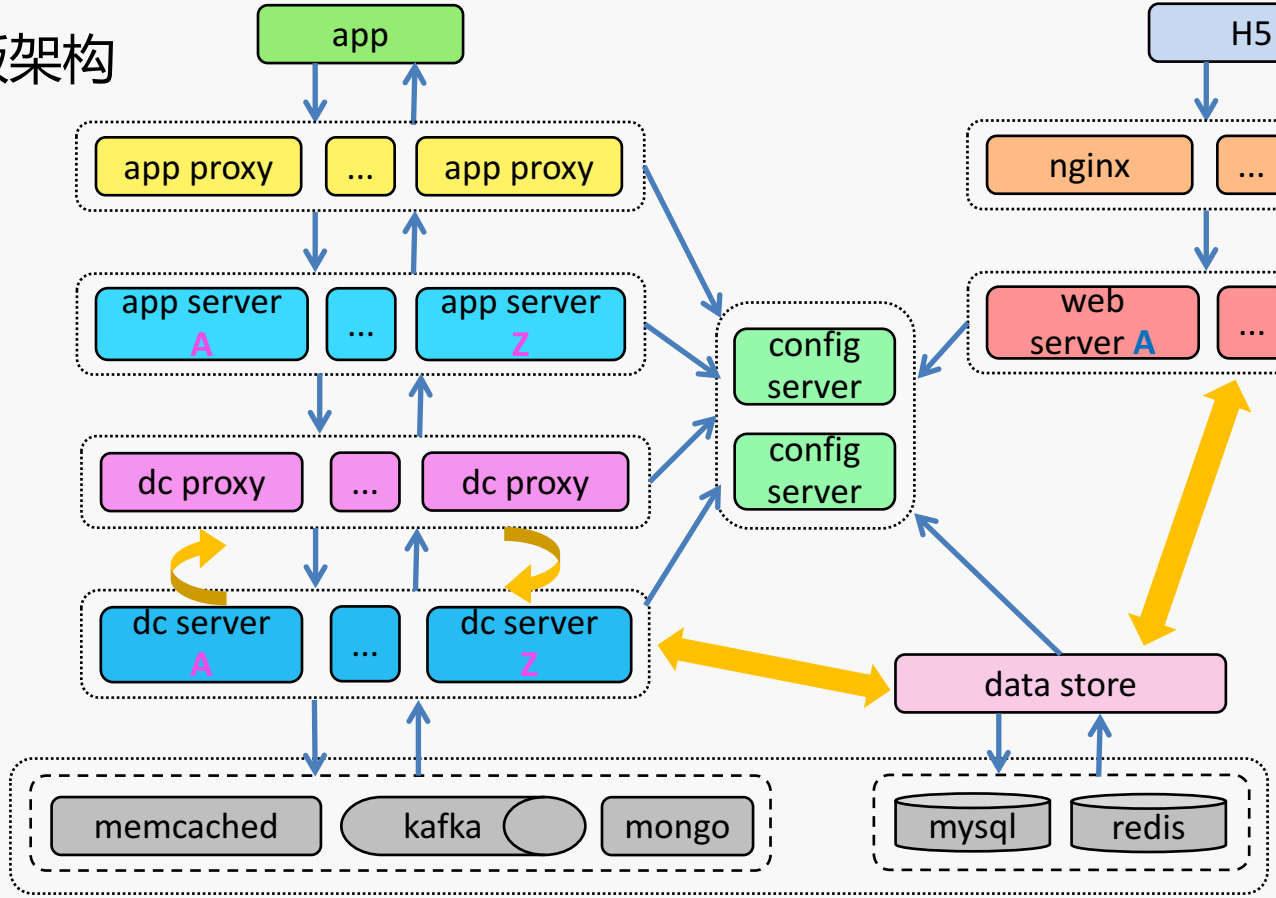
- ✓ 自动维护缓存和数据库表的 **对应关系**。
- ✓ 自动维护缓存与数据库 **数据一致性**。
- ✓ **简单易用**，在类中增加一个注解即可。
- ✓ **减少重复代码**，减少开发工作量。
- ✓ 屏蔽了服务对数据源的管理，便于数据库的 **迁移和扩容** 等操作。



架构演进


单体架构
APP 直连服务器
简单但不灵活
分布式架构
配置中心，支持配置热更
分布式服务框架
支持灵活水平扩展
2013 年
2014 年
2015 年
2016 年
http & json vs TCP & 私有协议
LVS vs 自研代理服务
按业务垂直拆分服务
垂直架构
封装数据源操作
统一管控数据源配置
数据源慢操作报警
分布式数据库中间件

v7.0版架构


app
H5
app proxy
...
app proxy
nginx
...
nginx
app server A
...
app server Z
web server A
...
web server Z
config server
config server
dc proxy
...
dc proxy
dc server A
...
dc server Z
data store
memcached
kafka
mongo
mysql
redis

v7.0版架构

特点

- 一个比较完整的分布式架构。
- data store 封装了常见的 mysql、redis 操作及提供慢操作监控。
- 根据业务发展，引入了 kafka, mongo, zookeeper、hbase 等多种第三方产品。

面临问题

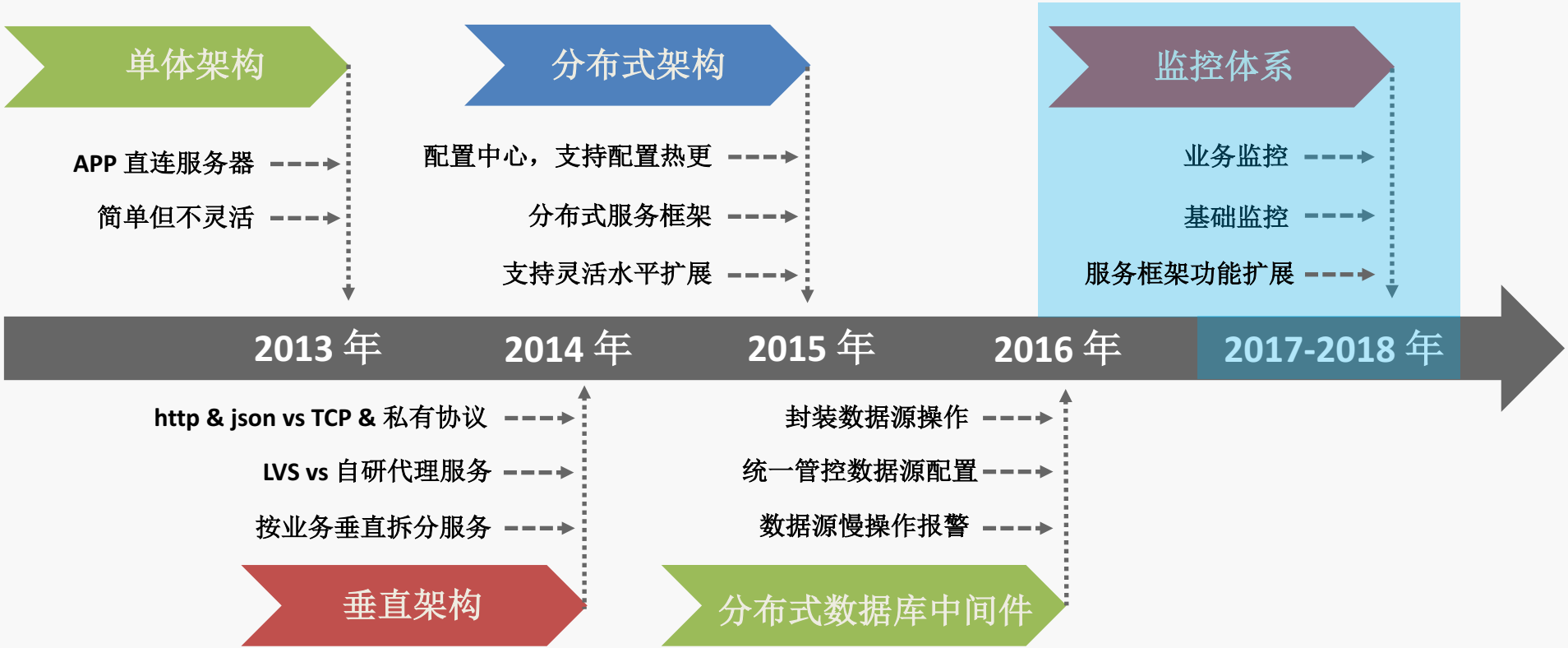
- 资源监控、业务监控、分布式跟踪链 等功能不完善
- 随着访问量上涨，需要扩展分布式服务框架的功能

解决方案（见v8.0版架构）

- 引入第三方产品，完善对服务器资源、业务、跟踪链路 等的监控，扩展服务框架功能

荔枝

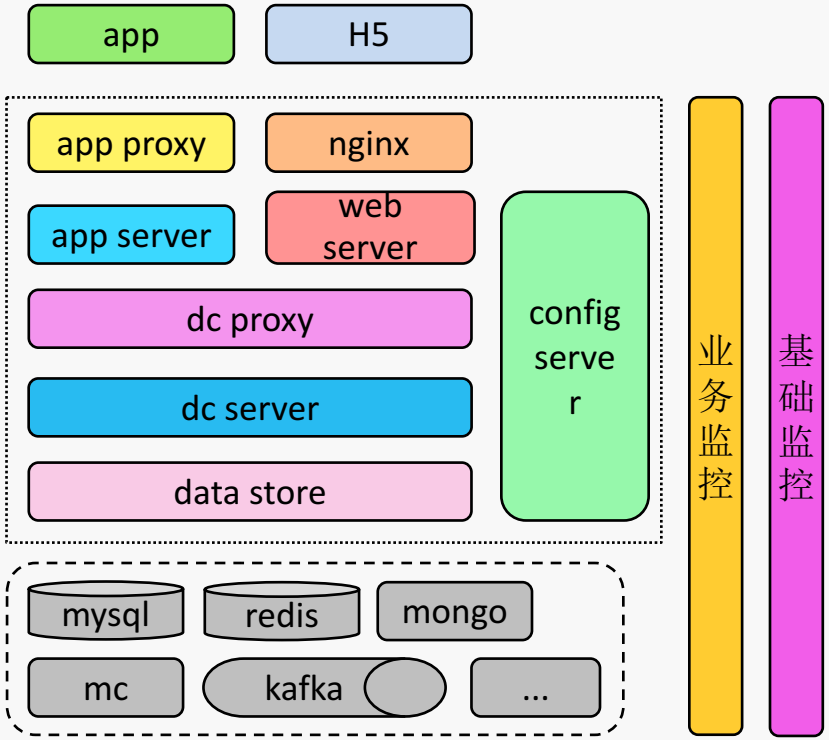
架构演进



v8.0版架构

特点

- **业务监控** 及 **基础监控** 功能比较完善。
- 分布式服务框架扩展了 **接口缓存**、**熔断**、**降级**、**过载保护** 等功能。



问题及方案

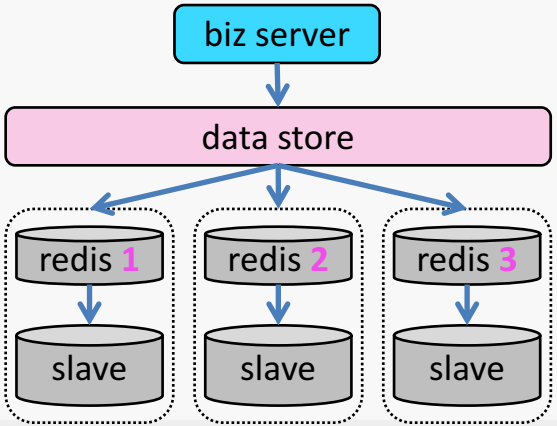
最近两年，踩过的 **坑** 有哪些？

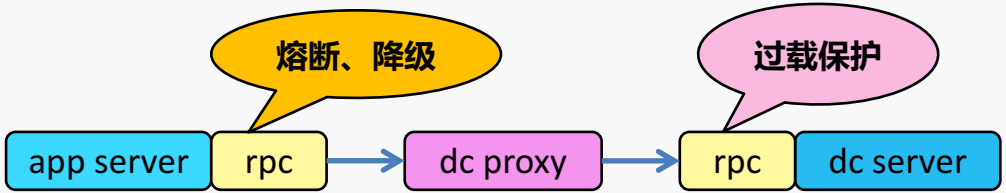
问题及方案

1、大主播开直播（如李易峰等），访问量爆涨，影响其他直播间，卡顿、进入不了直播间、接口超时。

方案：（1）在 data store 中，对 redis 存储开发按前缀 **分片** 的功能，对大主播的直播数据进行 **隔离**，避免影响其他主播的直播效果。

（2）分布式服务框架开发 **熔断、降级、过载保护** 等功能，避免服务器因访问量过高发生雪崩。

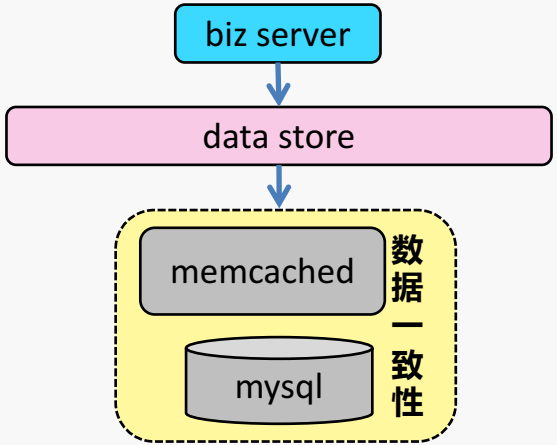




问题及方案

2、在高并发环境下，mysql 查询性能成为瓶颈。

方案：data store 在操作 mysql 时，在数据库上层加入缓存 **memcached** ，提高 **查询性能**，并自动维护缓存和数据库数据的 **一致性。**



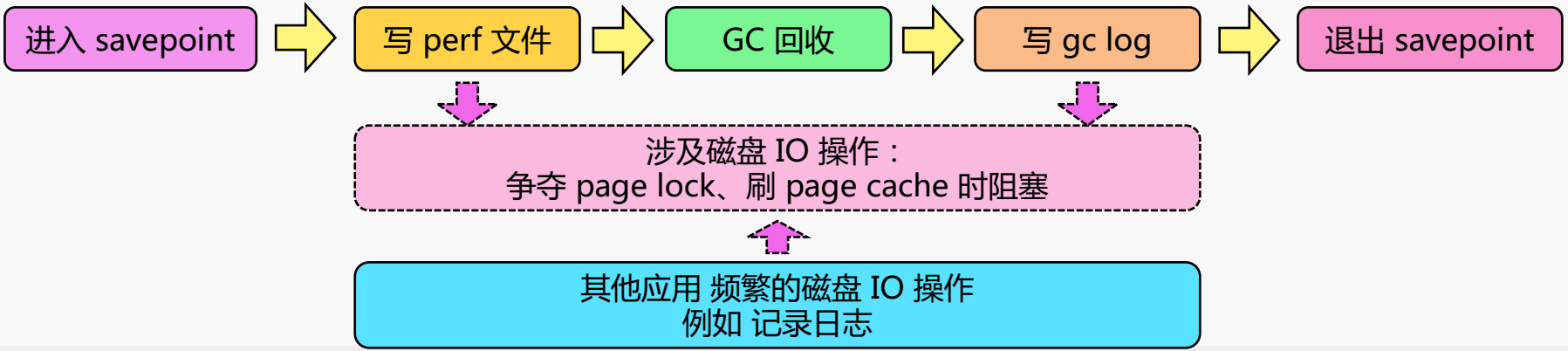
问题及方案

3、访问量上涨，受日志文件的IO影响，服务出现长GC（stop the world，阻塞业务线程）。

方案：

（1）**不生成 perf** 文件：**-XX:+PerfDisableSharedMem**

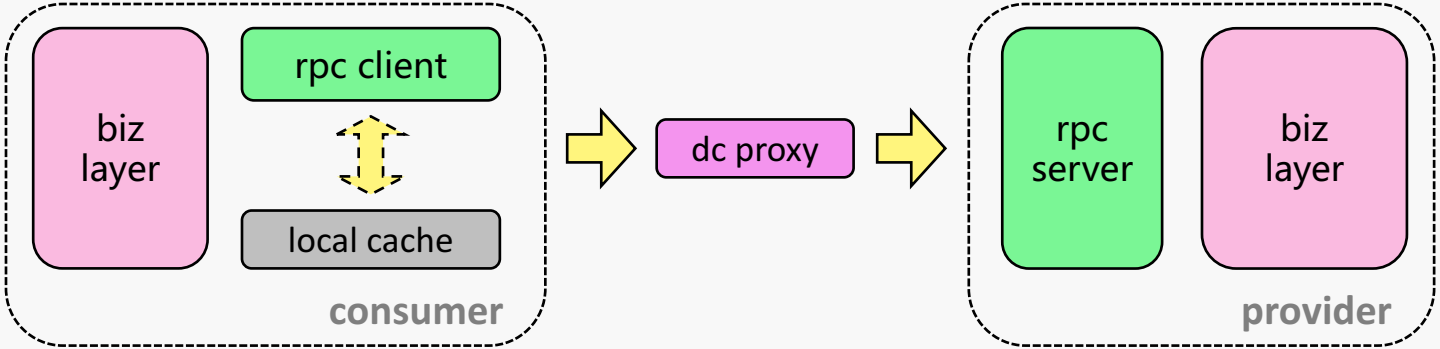
（2）将 **GC日志** 保存到 **内存盘** 中（ tmpfs 或 ramfs ）：**-Xloggc:/dev/shm/lz-app-gc.log**



问题及方案

4、随着业务的发展，系统的整体访问量越来越大，后端服务接口调用耗时越来越长，导致经常超时。

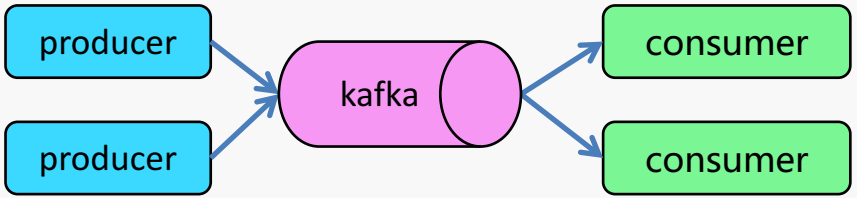
方案：经过分析，实际中以 **读多写少** 的场景居多，在分布式服务框架开发 **缓存接口** 功能。



问题及方案

5、系统间异步消息通知功能不完善。

方案：使用 **kafka** 解决 **系统间消息通知**、**大数据量传输**、**多个消费者** 消费相同消息的场景。

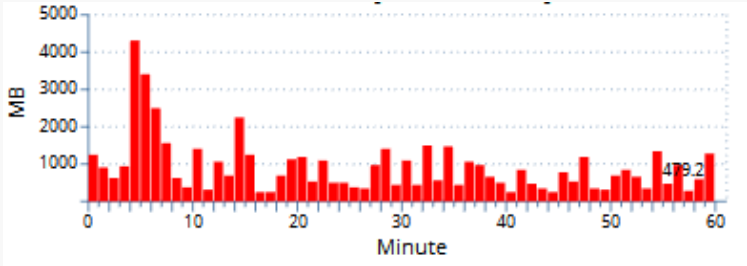


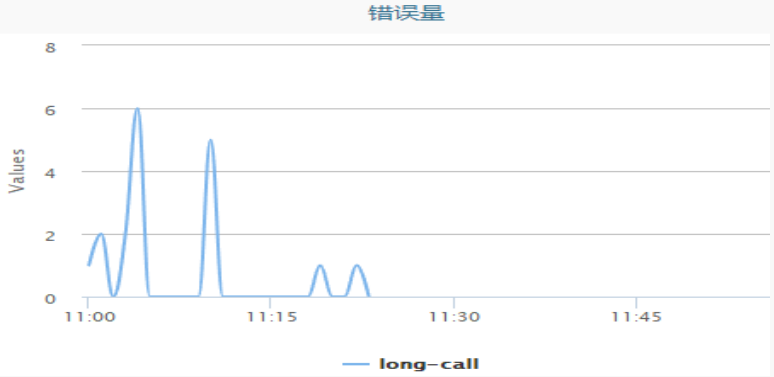
问题及方案

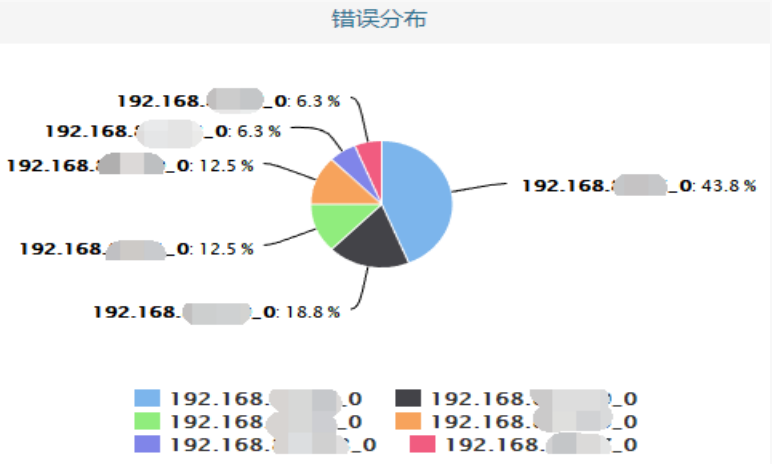
6、服务框架请求失败/超时/异常等，没有报警机制。

方案：使用 **CAT** 监控业务异常和服务框架的请求异常。

- 服务器负载、物理内存、swap、磁盘
- GC 信息（回收时间、次数）
- JVM堆信息
- 请求异常统计等



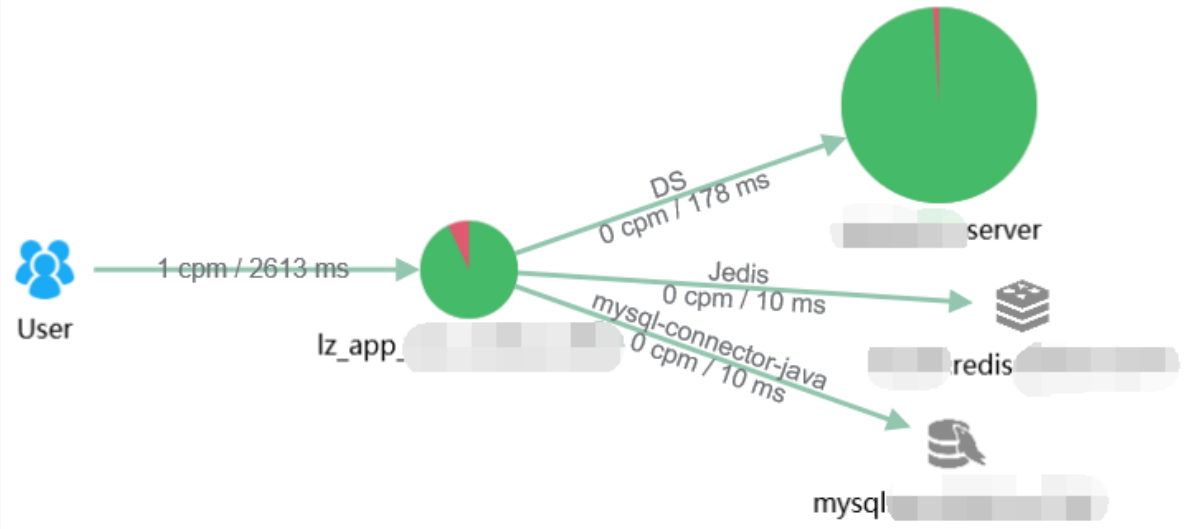




问题及方案

7、整个架构的调用链路不清晰，也不能预知整体架构存在的瓶颈。

方案：使用 **skywalking** 实现调用链跟踪。



问题及方案

8、上线/重启服务操作很原始：人工在本地打包，再上传到服务器。

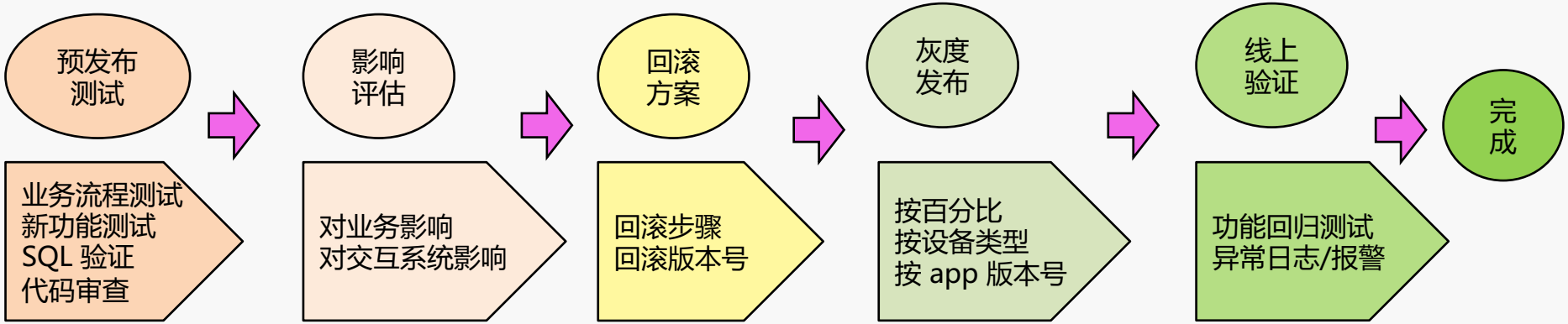
方案：（1）开发 **自动发布平台**，一键式操作。

（2）采用 **jenkins + gitlab**，并接入 **自动发布平台**，实现 **自动打包**、**一键发布**。

问题及方案

9、服务发布流程不够规范。

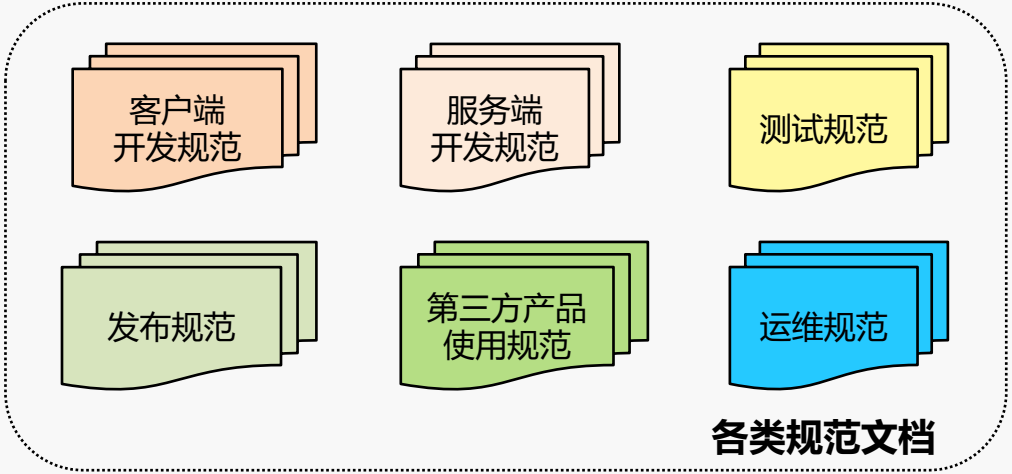
方案：服务发布流程 **规范化。**



问题及方案

10、研发规范不够标准。

方案：制定各种标准的 **开发/操作规范**，包括 **客户端开发规范、服务端开发规范、测试规范、运维规范、**mysql、redis、kafka、mongo 等的**使用规范。**



未来展望

- **微服务+容器化**，实现服务实例的 **动态扩容** 与 **缩容**。
- **service mesh** 探索。
- **业务级别** 的调用链跟踪（根据业务参数 例如 uid 或 设备id 等，可查询调用轨迹）。
- 分布式 **定时任务** 系统。
- data store 支持**分布式事务**。

Q & A

（广州）荔枝APP 团队气氛 **活跃**、**人性化**、**扁平化**、**成长空间大**，欢迎各路贤才加入。

请赐简历到 **huangquan@lizhi.fm**
非常期望您的加入！

官网： **www.lizhi.fm**

THANKS