小恩爱技术团队成长及服务器架构演变

钟超

小恩爱

全球最受欢迎的情侣应用

免费短信超过200亿条

免费照片存储超过40亿张

免费通话累计超过40万小时

免费下载

小恩爱的用户增长

小恩爱的用户增长

提纲

1. 零到百万：快速迭代
2. 百万到千万：局部优化
3. 千万到亿：架构调整
4. 大于一亿：待续

一、零到百万

零到百万

缺钱

需求变化

岗位欠缺

缺人

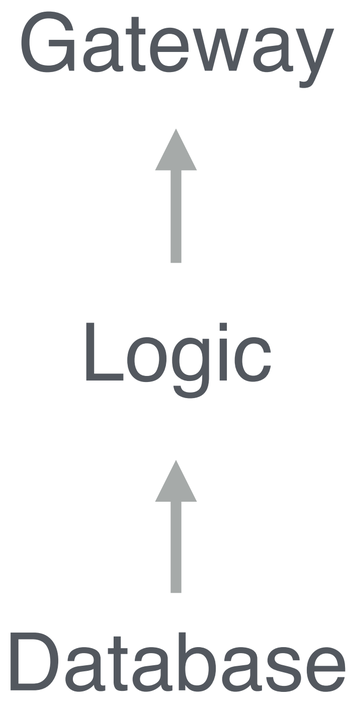
时间不够用

降低人力和时间成本，尽量使用云服务器，第三方 SDK

降低人力和时间成本，使用第三方服务，使用开源产品

重视迭代速度，而不是性能，避免过度设计

避免过度设计，简单的服务器架构



最简单的三层结构

可不可以更简单一点?

把大部分工作交给云

Gateway　　云服务提供的负载均衡

Logic　　只关注业务（踩坑）

Database　　云数据库

静态文件？交给又拍云就可以

运维？使用第三方服务，后端开发兼运维

即使大量使用第三方服务，我们还是必须一人多用

二、百万到千万

百万到千万

用户井喷式增长

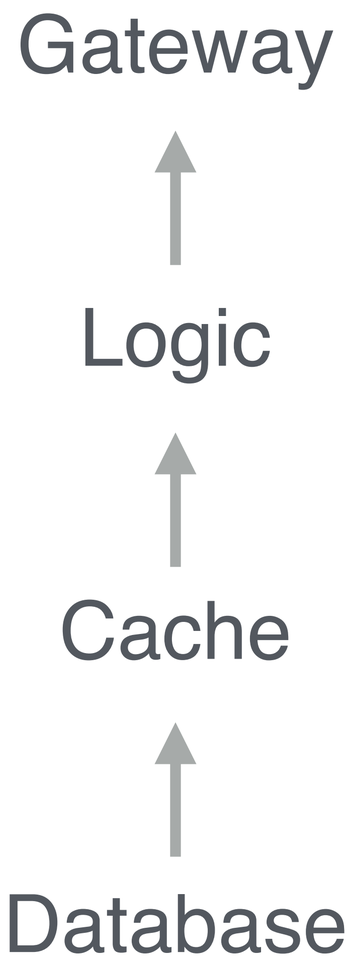
缺人

业务变复杂

时间不够用

快速迭代的同时需要关注性能

没有重新设计，增加缓存



增加了缓存，拆分模块（踩坑）

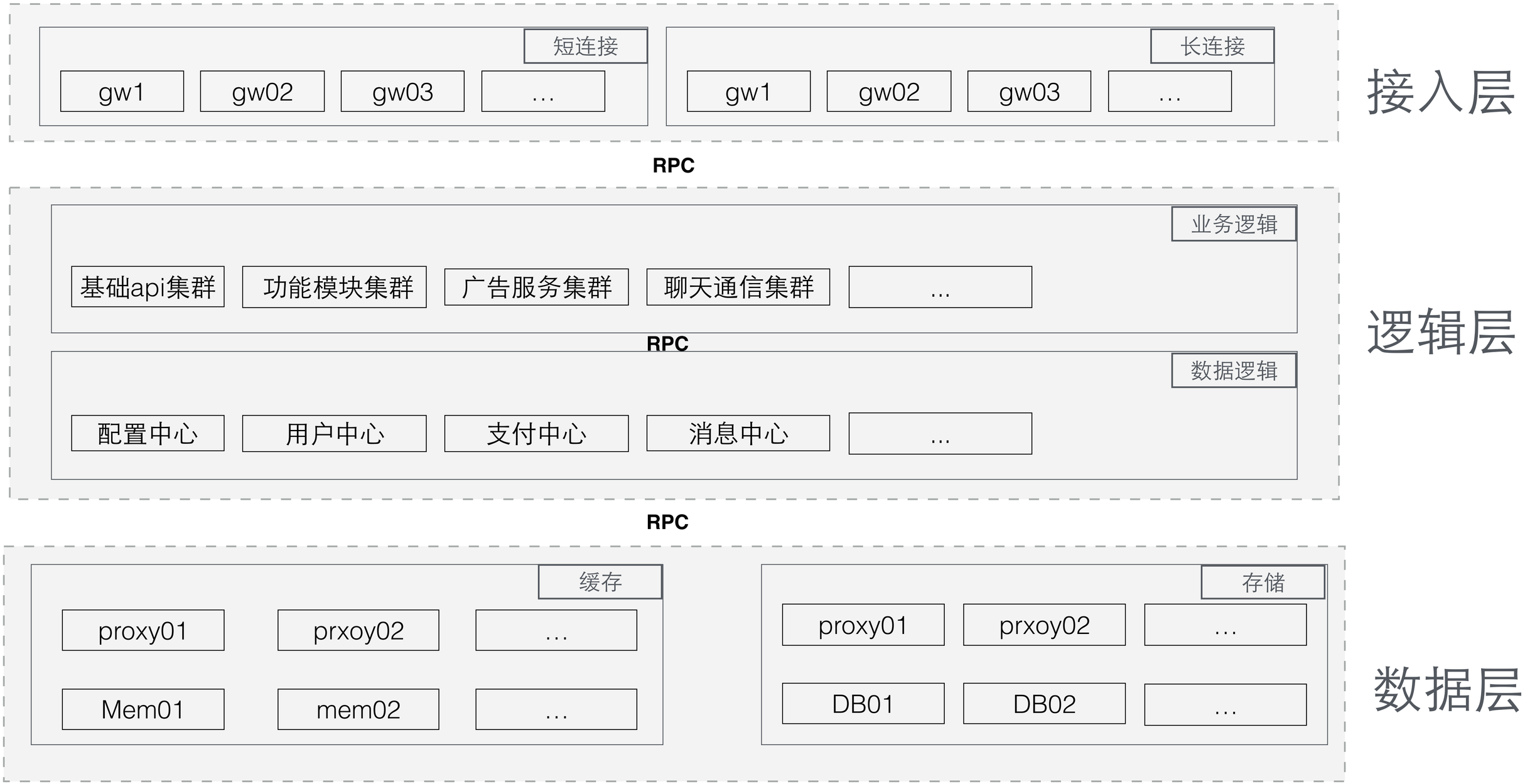
避免不了的宕机，自己搭建业务层监控

三、千万到亿

技术转型

- 之前的坑，Ruby 性能瓶颈
- Java？不够酷，代码繁琐
- C/C++？开发成本高，开发效率低
- 任性的选择 Go

架构调整，大系统小做，去耦合


短连接
gw1
gw02
gw03
…
长连接
gw1
gw02
gw03
…
接入层
RPC
业务逻辑
基础api集群
功能模块集群
广告服务集群
聊天通信集群
…
RPC
数据逻辑
配置中心
用户中心
支付中心
消息中心
…
逻辑层
RPC
缓存
proxy01
prxoy02
…
Mem01
mem02
…
存储
proxy01
prxoy02
…
DB01
DB02
…
数据层

尽可能自动化，移动化

具体哪些工作

- 自动化运维
- 自动化测试
- 移动运维
- 其他能用代码解决的问题，坚决不手动操作

注重技术成长，技术积累沉淀

听起来很虚？如何实施

- 框架代码模板化，尽量用代码解决的问题
- 搭建技术博客
- 搭建内部代码库
- 每周技术分享，内部分享或邀请行业大牛
- 部分项目开源？将来也许可能

总结

成长中的公司永远缺人

Thanks!

小恩爱欢迎更多牛人加入

zc@xiaoenai.com