网易云
163yun.com

高性能、高可用服务化架构漫谈

曾宇星　网易云资深解决方案架构师

目录

典型互联网应用

网站

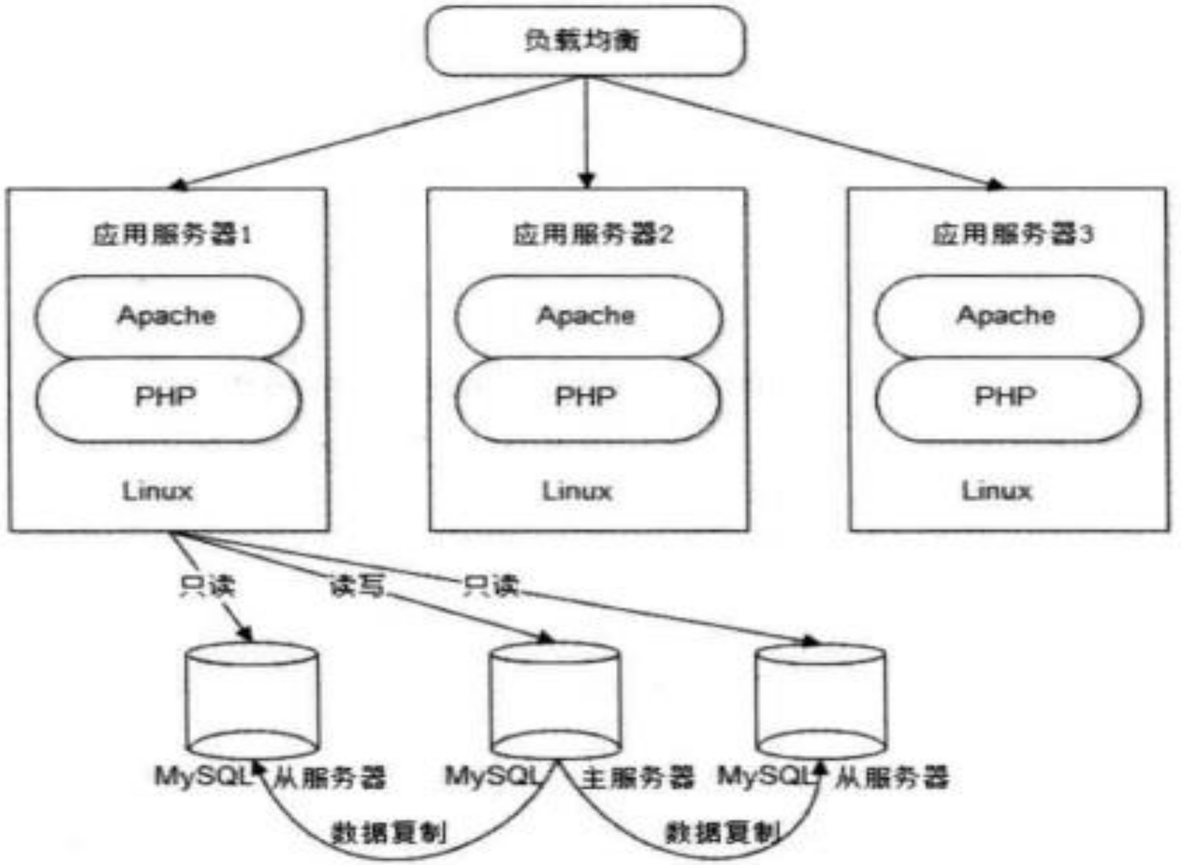
游戏

无状态应用

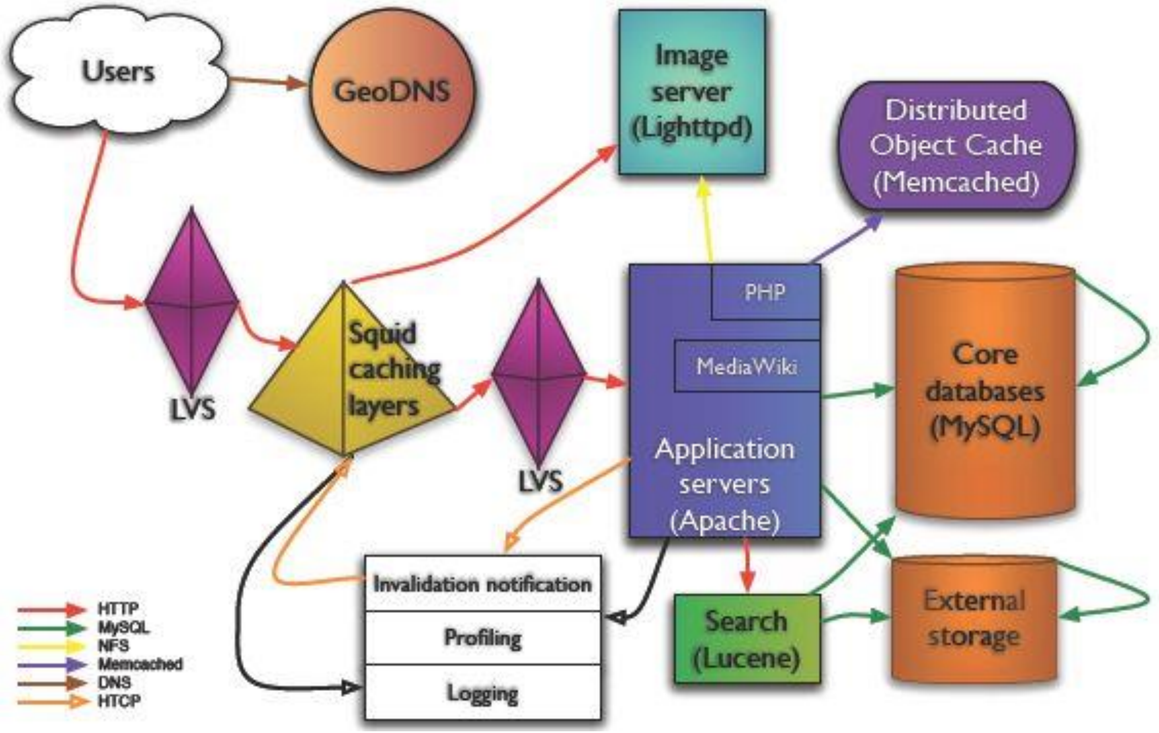
有状态应用

具体应用架构都不一样

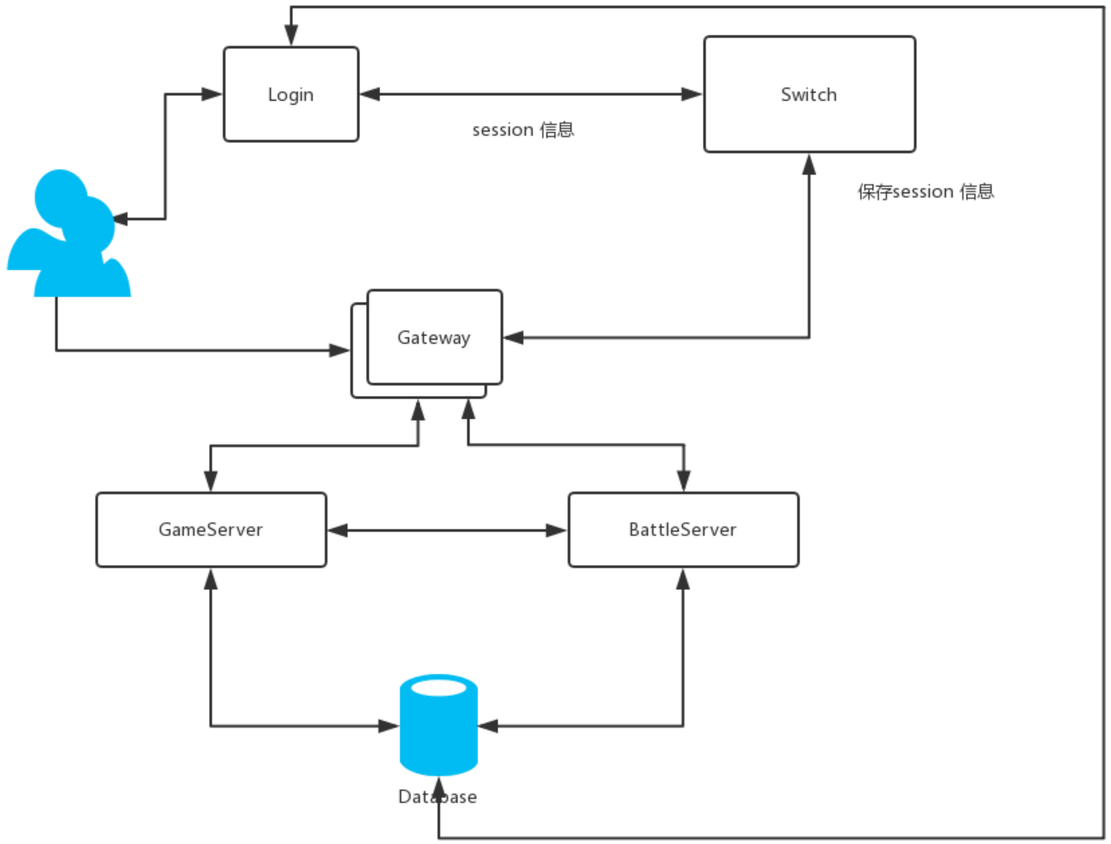
2003 年某宝网架构



Wikipedia 架构

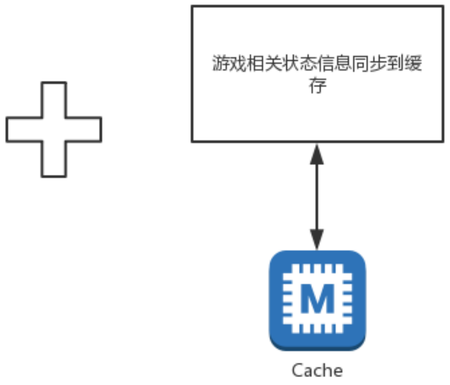


游戏-有状态应用



Tcp 长连接
短连接

Pvp
即时战斗

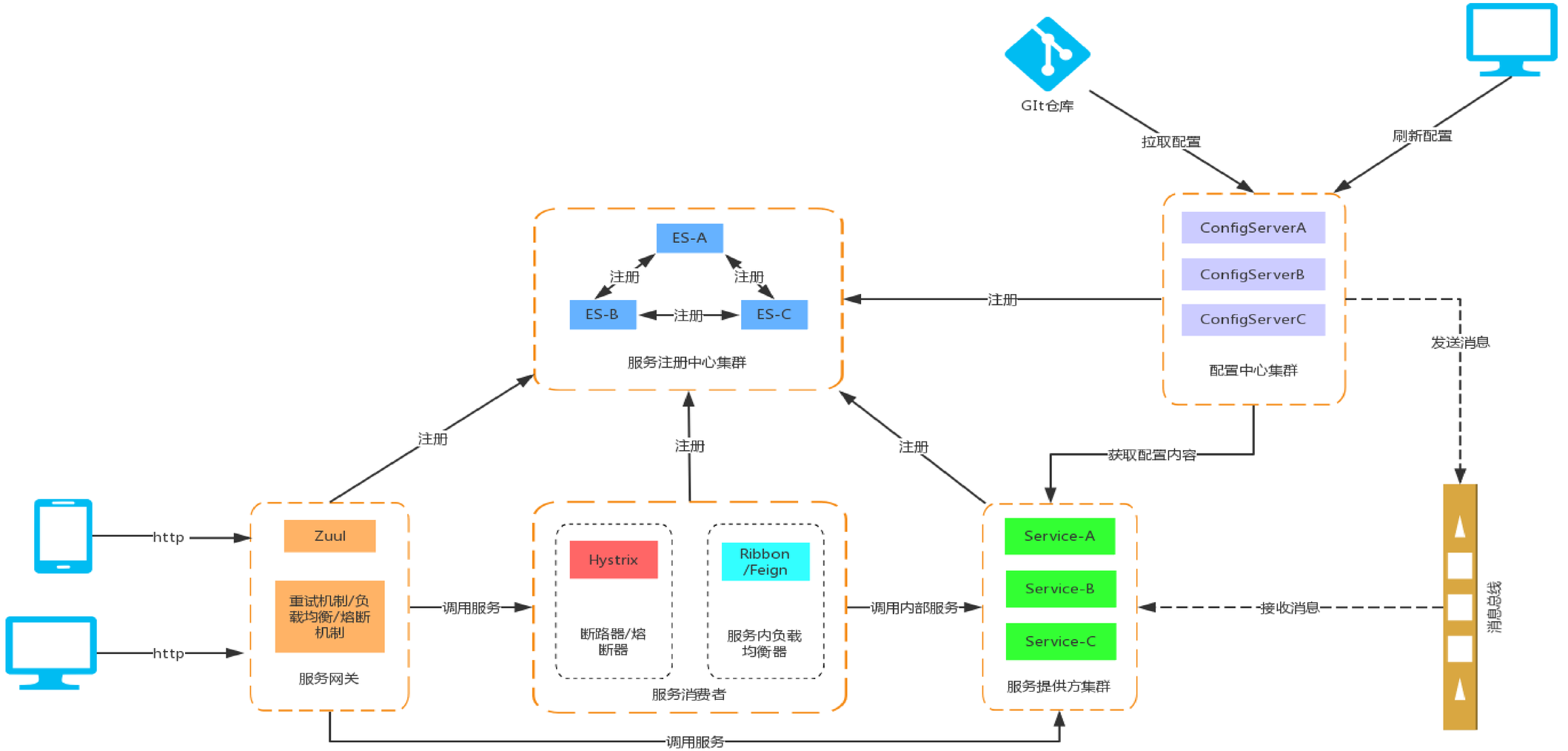


状态和应用剥离：

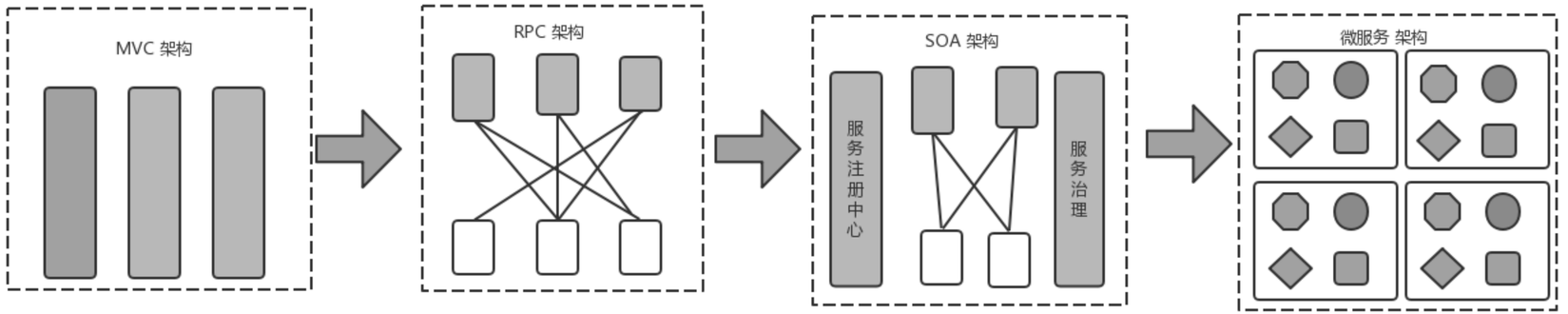
状态信息同步更新到缓存
Redis/MemCache
应用服务宕机，游戏状态信息可以立即从缓存恢复，重建应用服务

客户端应用：
连接断开重连处理，使玩家无感知

基于Spring Cloud 的微服务架构



架构演变



关键：是否满足业务发展需求

业务发展、用户规模不断扩大

传统架构 VS 服务化架构

传统架构

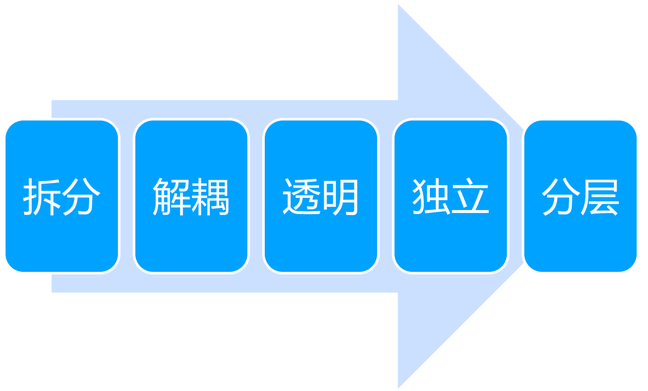
研发成本高

- 代码重复率高（顶多有公共类库，但公共类库维护及版本升级困难）
- 需求变更困难（user数据库变化，各个模块都需要修改)
- 无法满足新业务快速创新和敏捷交付

运维效率低

- 测试、部署成本高：业务运行在一个进程中，因此系统中任何程序的改变，都需要对整个系统重新测试并部署
- 可伸缩性差：水平扩展只能基于整个系统进行扩展，无法针对某一个功能模块按需扩展
- 可靠性差：某个应用BUG，例如死循环、OOM等，会导致整个进程宕机，影响其它合设的应用

解决对策-服务化架构



架构设计相关定律

墨菲定律

二八定律

康威定律

分布式架构要素

CAP

性能

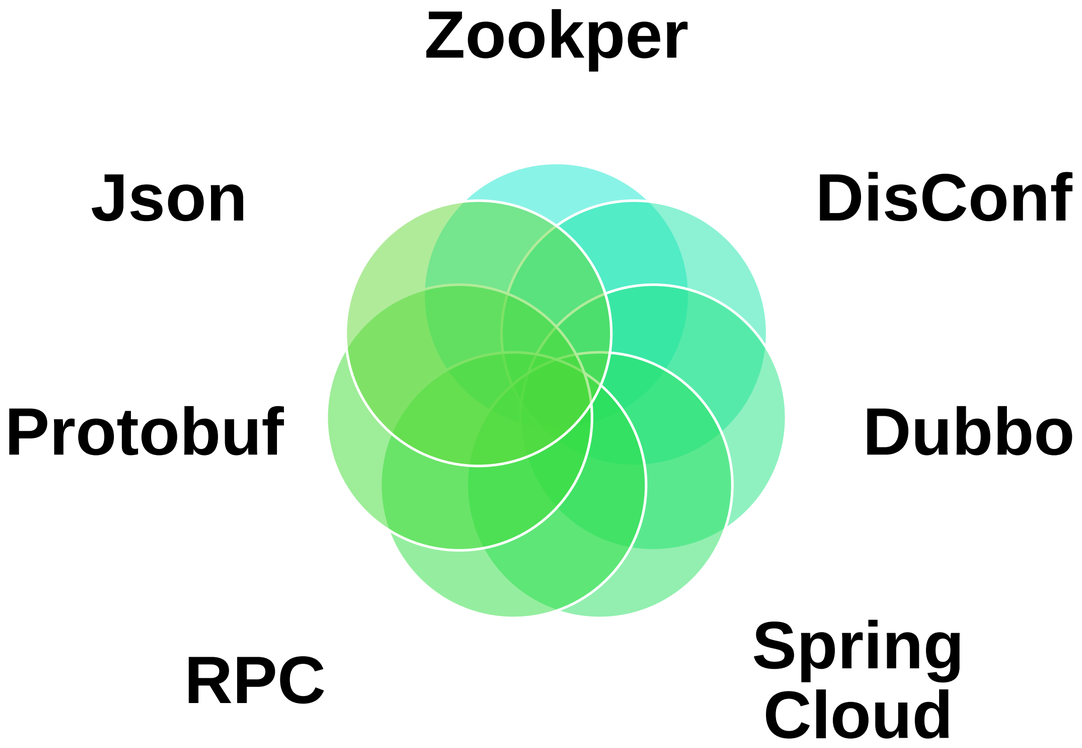
伸缩性

扩展性

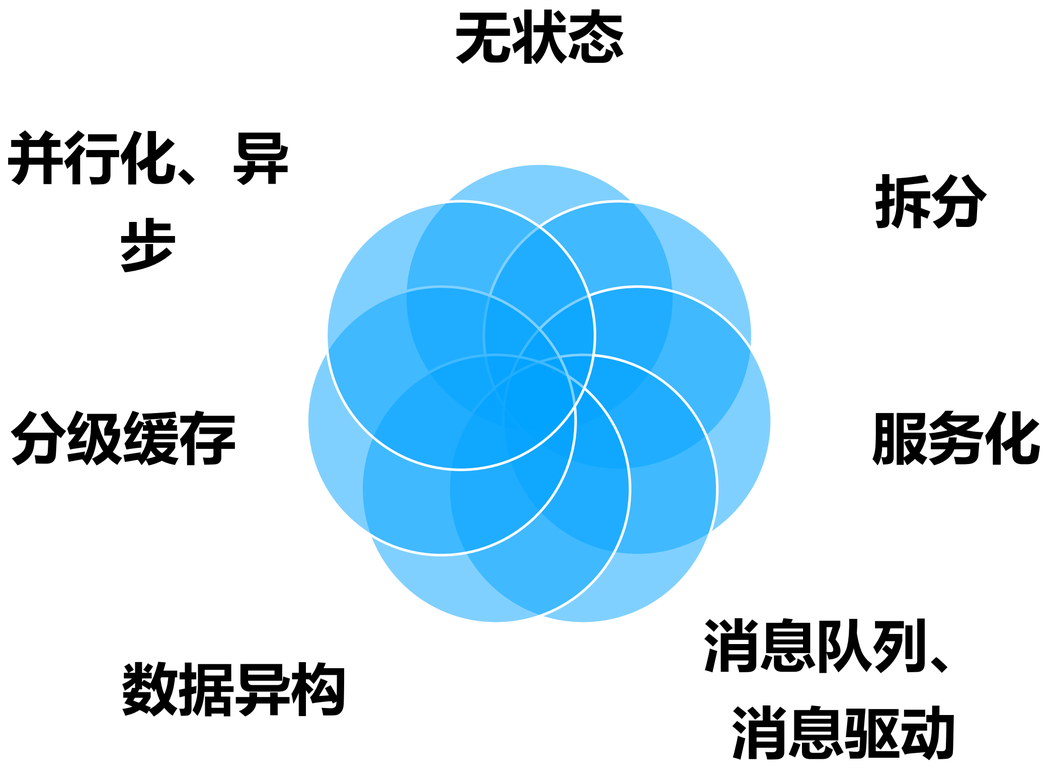
安全性

性能、稳定性、用户体验之间做平衡

服务化架构-关键词



高性能/并发架构 原则



高性能服务器架构（RPC）

第一，服务端尽可能多的处理并发请求
第二，同时尽可能短的处理完毕。

高性能服务器架构-同步&异步

IO模型：

1. 传统的阻塞 I/O（Blocking I/O）

2. 非阻塞 I/O（Non-blocking I/O）

3. **I/O 多路复用，两种I/O复用设计模式，Reactor 、Proactor**

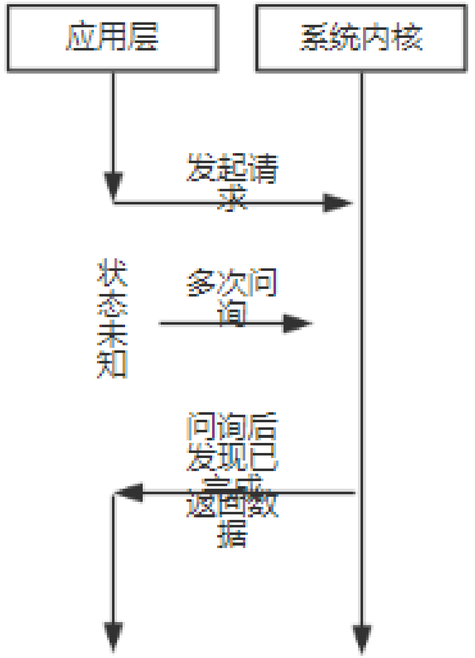
4. 异步 I/O（Asynchronous I/O）

高性能服务器架构-同步&异步

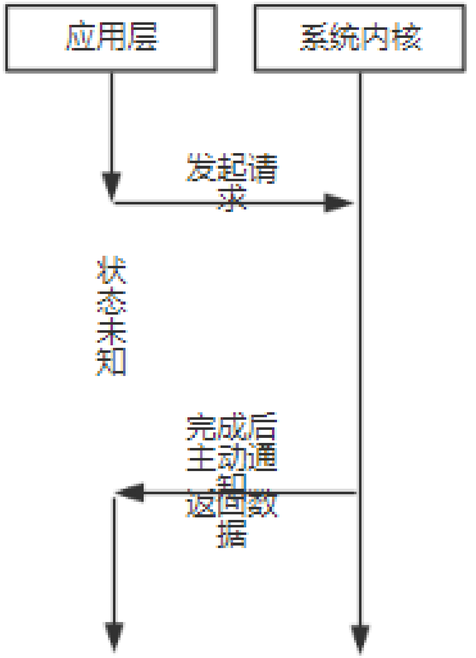
易混淆概念：

- **同步& 异步**
- **阻塞& 非阻塞**

关注点不同，同步和异步关注的是多件事情是否可以并发，而阻塞和非阻塞关注的是程序等待调用结果时的状态。

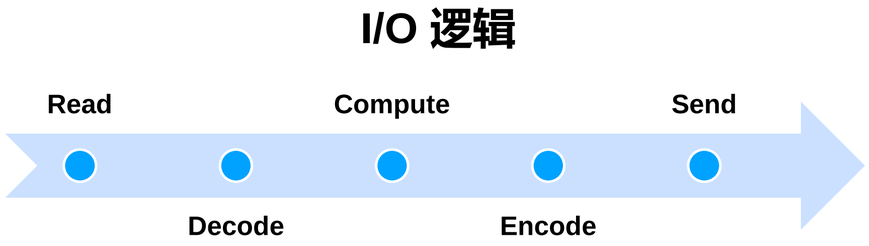


同步型调用



异步型调用

高性能服务器架构-并发



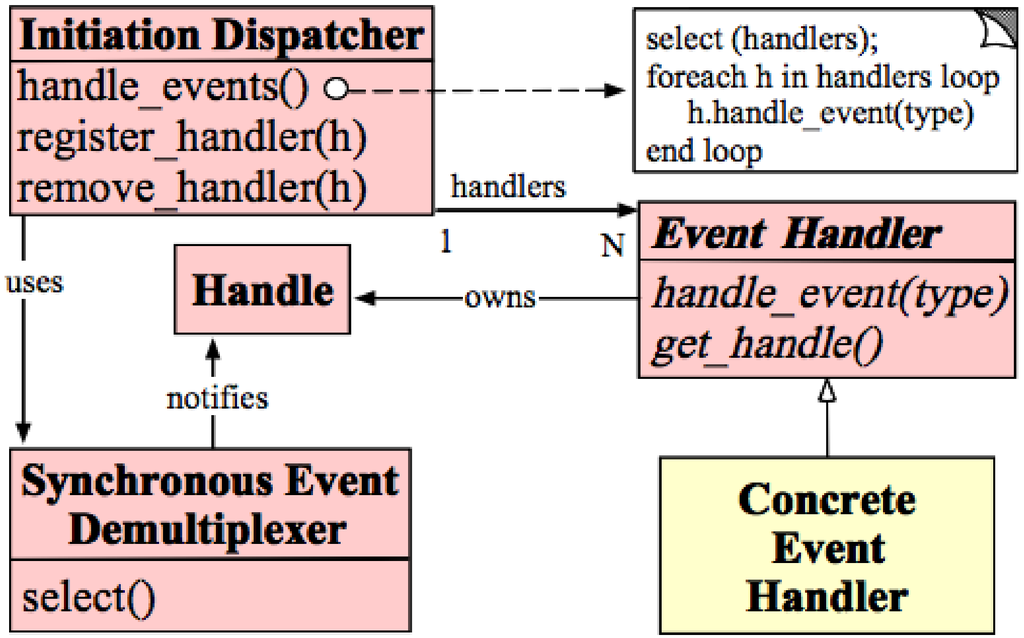
进程 / 线程模型

传统的阻塞 I/O + 线程池 -> C10K 问题

I/O 多路复用 -> Reactor 模式 (Redis、Nginx、Node.js、Netty)

异步I/O -> Proactor (Windows IOCP)

Reactor 模式



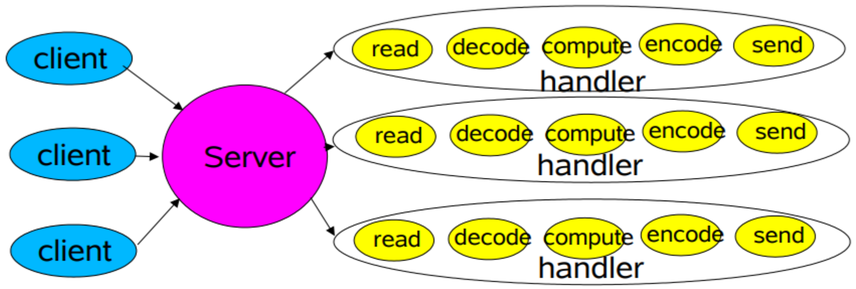
I/O 复用

- **Select**
- **Poll**
- **Epoll**
- **kqueue**

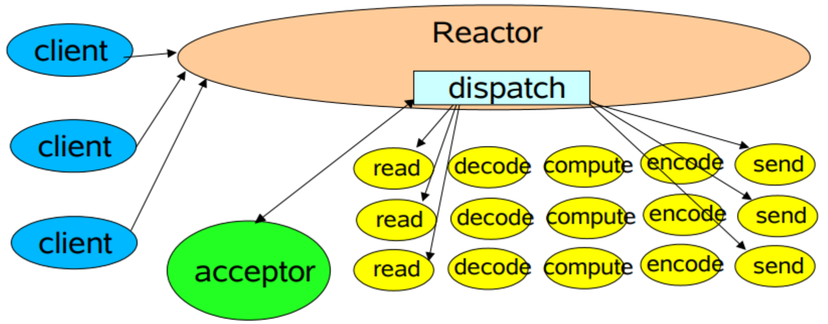
框架

- **Libevent**
- **Netty**

Reactor 模式-单线程

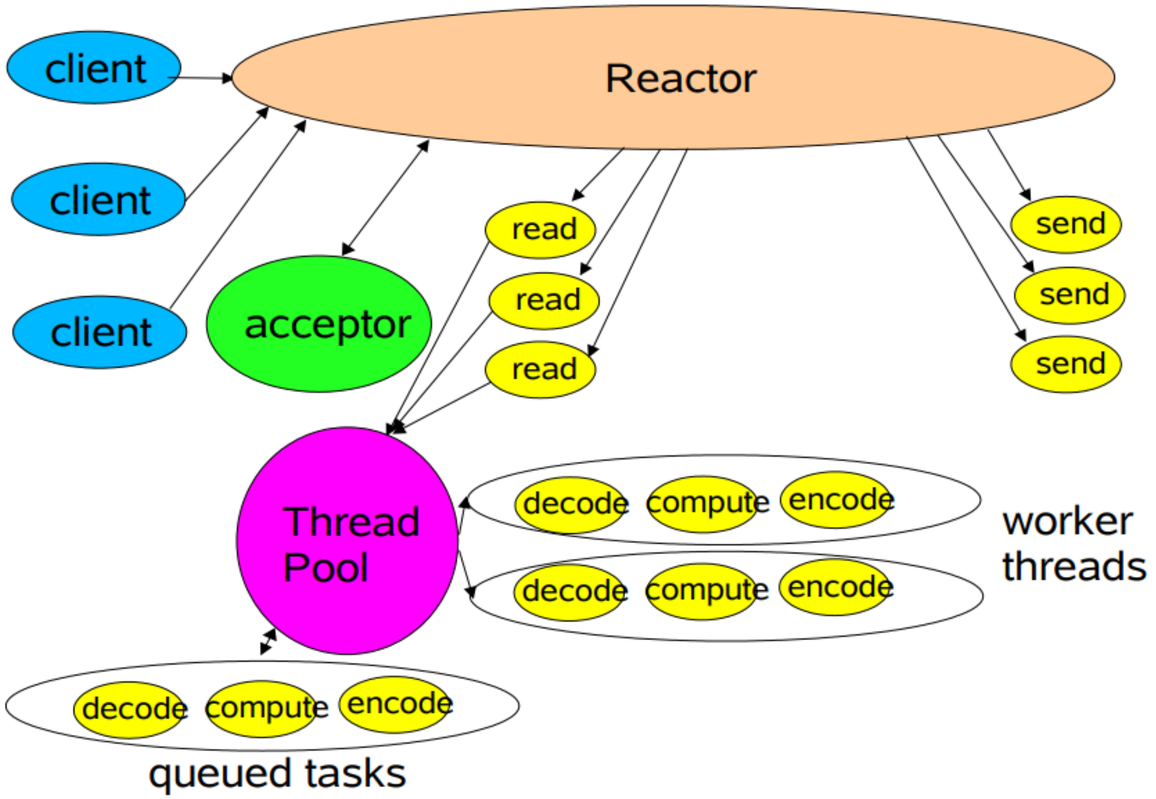


Each handler may be started in its own thread

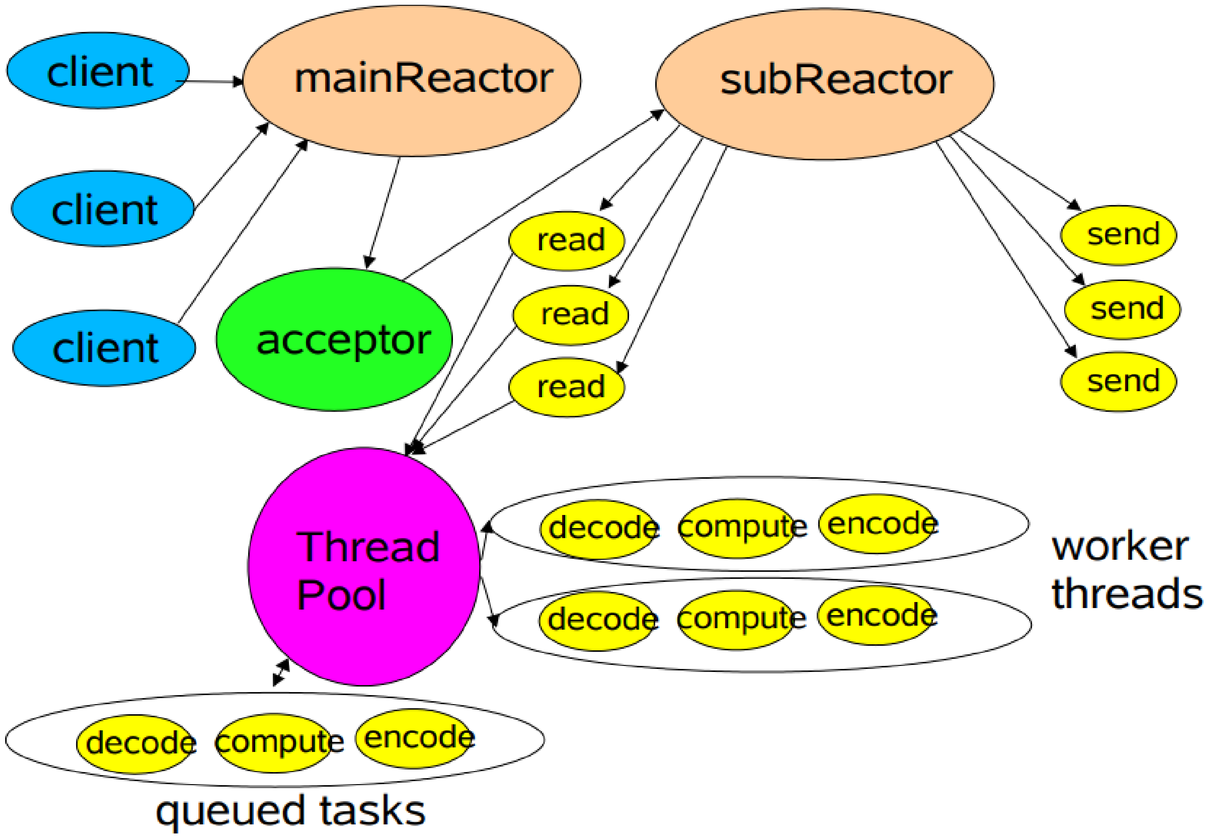


Single threaded version

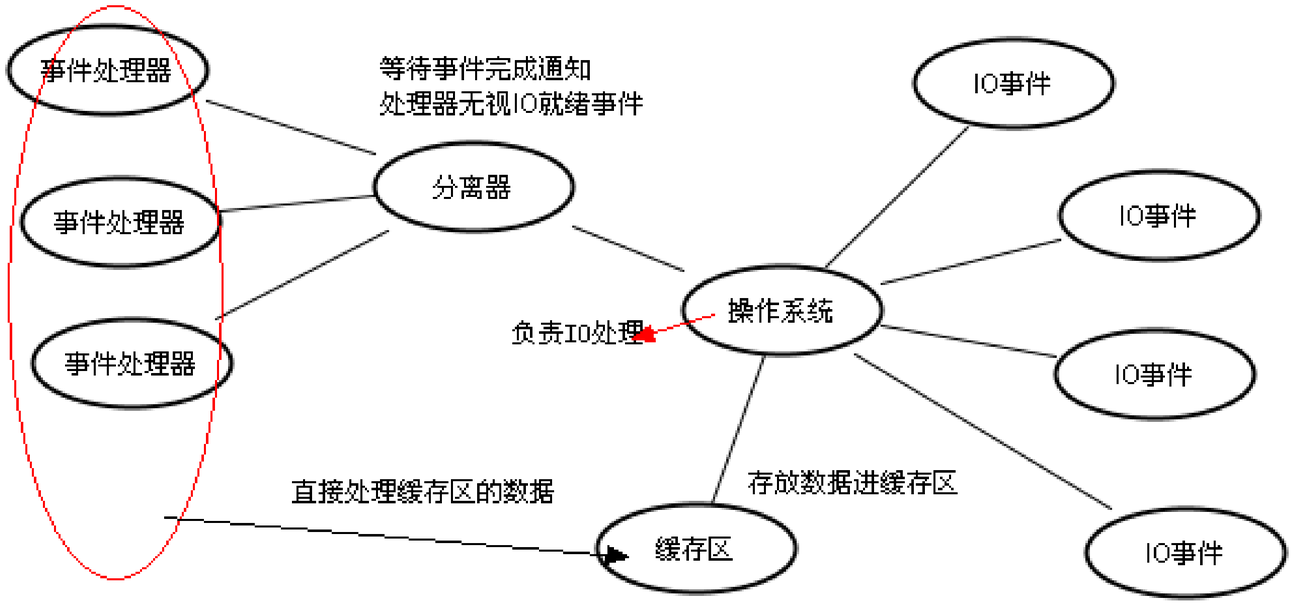
Reactor 模式-多线程



主从Reactor/ Multi-Reactor



Proactor 模式



Proactor模式：
操作系统必须支持异步IO

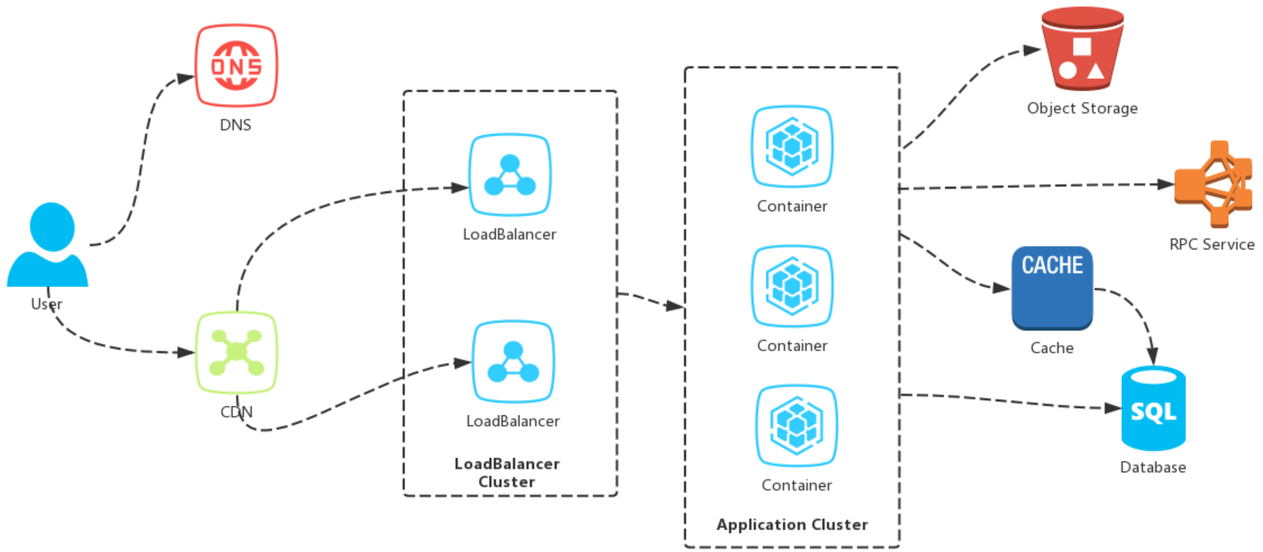
高性能服务化架构-分级缓存

按端划分：

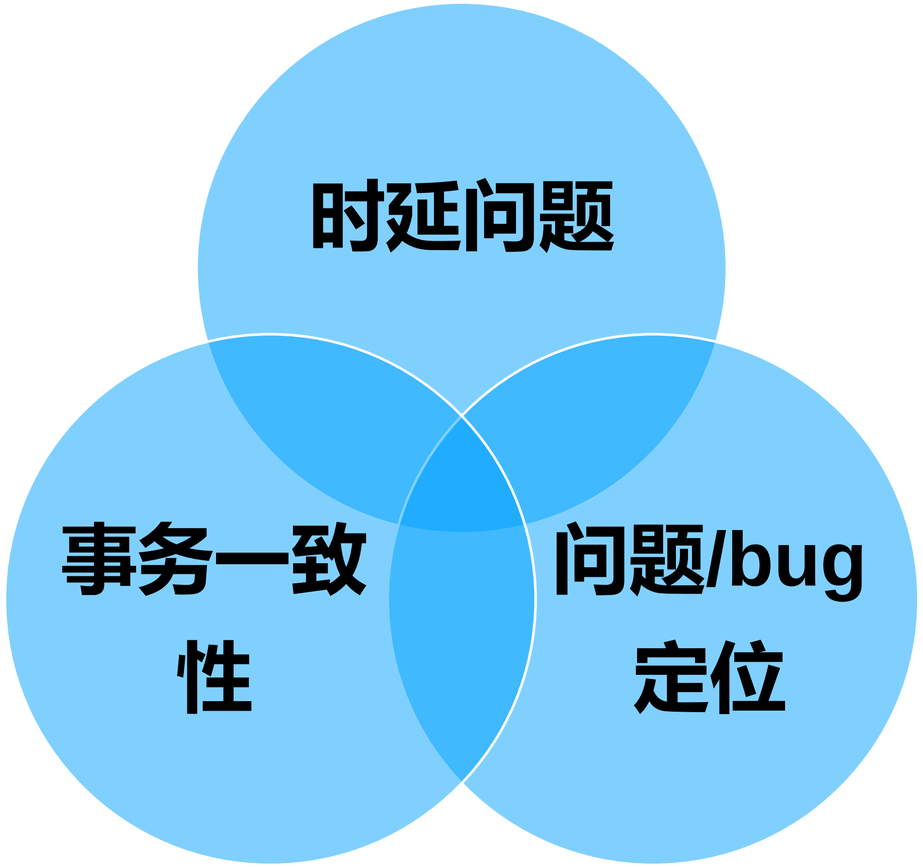
- **客户端缓存 （ 移动app、浏览器 ）**
- **服务端缓存**

常见缓存：

- **浏览器缓存 （ http 缓存 ）**
- **CDN**
- **反向代理 （ Nginx 缓存 ）**
- **分布式缓存 redis、memcache等**



服务化不是银弹



衡量系统可用性指标SLA

SLA： 俗称几个9

不同公司的计算标准不一致

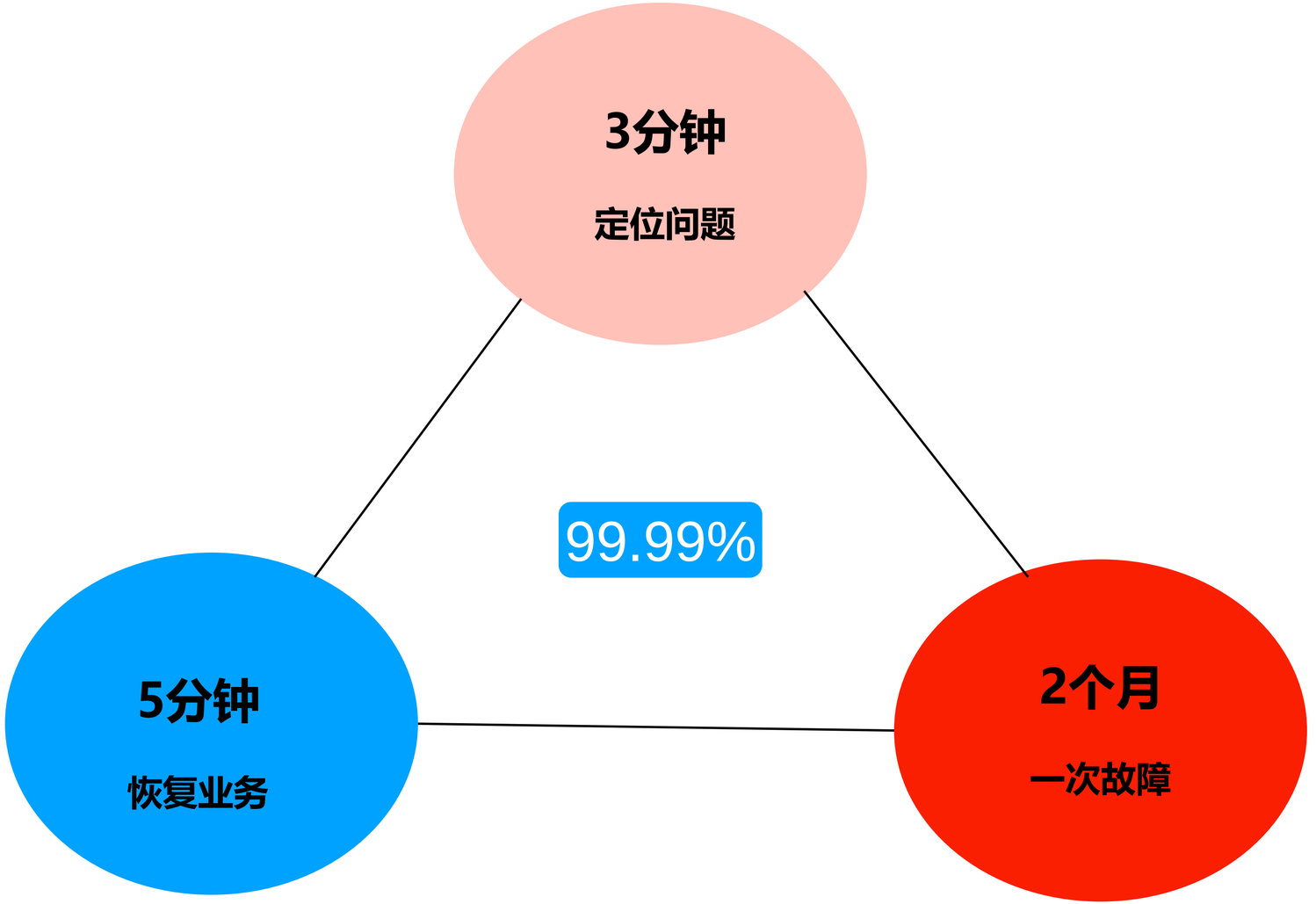
大体是：

服务周期内所有可用时间 / 服务周期总时间。

Availability %	Downtime per year	Downtime per month	Downtime per week	Downtime per day
90% ("one nine")	**36.5 days**	**72 hours**	**16.8 hours**	**2.4 hours**
95% ("one and a half nines")	18.25 days	36 hours	8.4 hours	1.2 hours
97%	10.96 days	21.6 hours	5.04 hours	43.2 minutes
98%	7.30 days	14.4 hours	3.36 hours	28.8 minutes
99% ("two nines")	**3.65 days**	**7.20 hours**	**1.68 hours**	**14.4 minutes**
99.5% ("two and a half nines")	1.83 days	3.60 hours	50.4 minutes	7.2 minutes
99.8%	17.52 hours	86.23 minutes	20.16 minutes	2.88 minutes
99.9% ("three nines")	**8.76 hours**	**43.8 minutes**	**10.1 minutes**	**1.44 minutes**
99.95% ("three and a half nines")	4.38 hours	21.56 minutes	5.04 minutes	43.2 seconds
99.99% ("four nines")	**52.56 minutes**	**4.38 minutes**	**1.01 minutes**	**8.64 seconds**
99.995% ("four and a half nines")	26.28 minutes	2.16 minutes	30.24 seconds	4.32 seconds
99.999% ("five nines")	**5.26 minutes**	**25.9 seconds**	**6.05 seconds**	**864.3 milliseconds**
99.9999% ("six nines")	**31.5 seconds**	**2.59 seconds**	**604.8 milliseconds**	**86.4 milliseconds**
99.99999% ("seven nines")	**3.15 seconds**	**262.97 milliseconds**	**60.48 milliseconds**	**8.64 milliseconds**
99.999999% ("eight nines")	**315.569 milliseconds**	**26.297 milliseconds**	**6.048 milliseconds**	**0.864 milliseconds**
99.9999999% ("nine nines")	**31.5569 milliseconds**	**2.6297 milliseconds**	**0.6048 milliseconds**	**0.0864 milliseconds**

比如我们说月度99.95%的SLA，意味着每个月服务出现故障的时间只能占总时间的0.05%,如果这个月是30天，那么就是21.6分钟。

更直观的SLA



聚焦于业务　容易分解　衡量

解决系统可用性的手段

◆减少故障出现次数

—— 不出问题

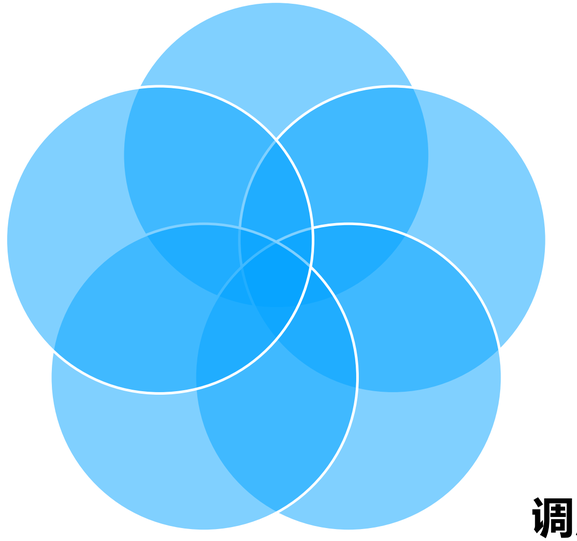
◆缩短故障处理时间

—— 尽快解决

服务化实践-服务治理

流量调配-权重调节

实时日志查询
ELK

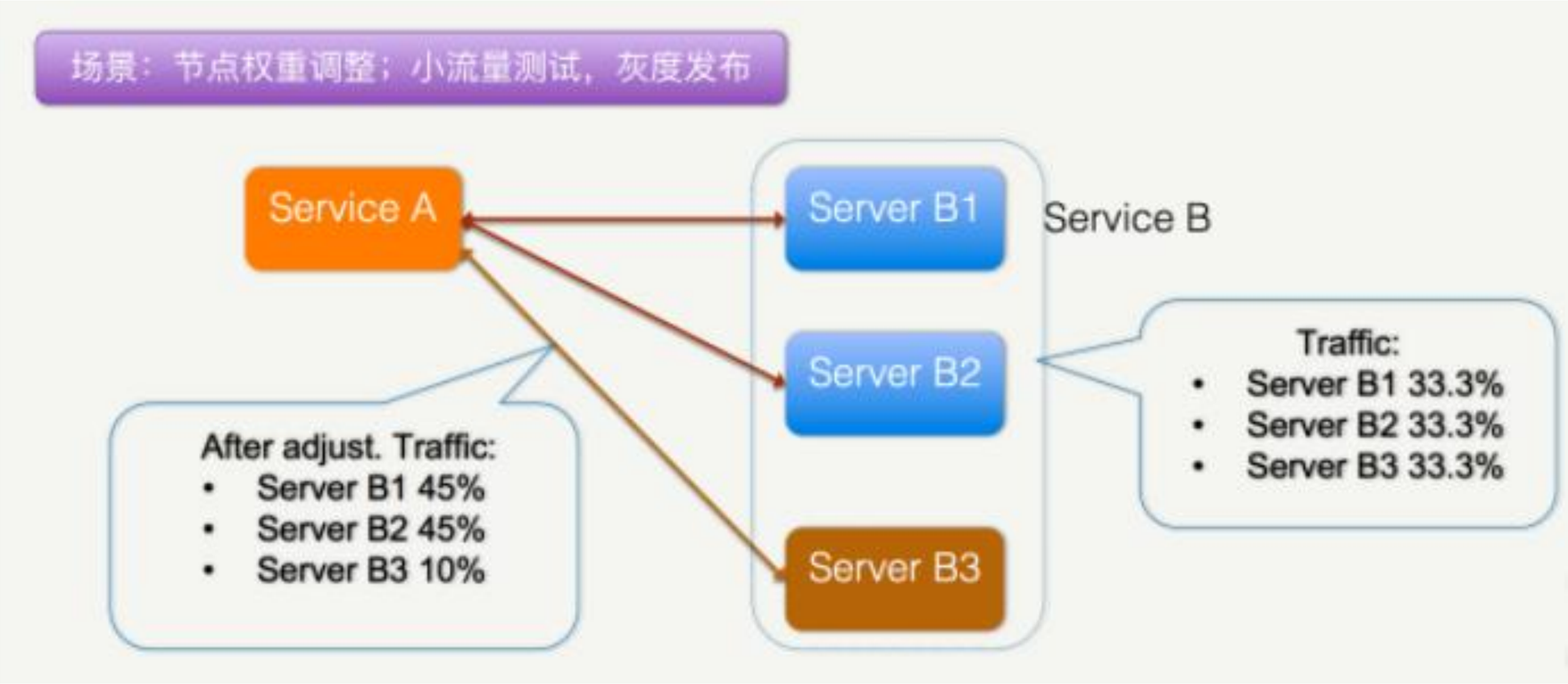


服务路由：机房感知、服务自定义分组、动态路由

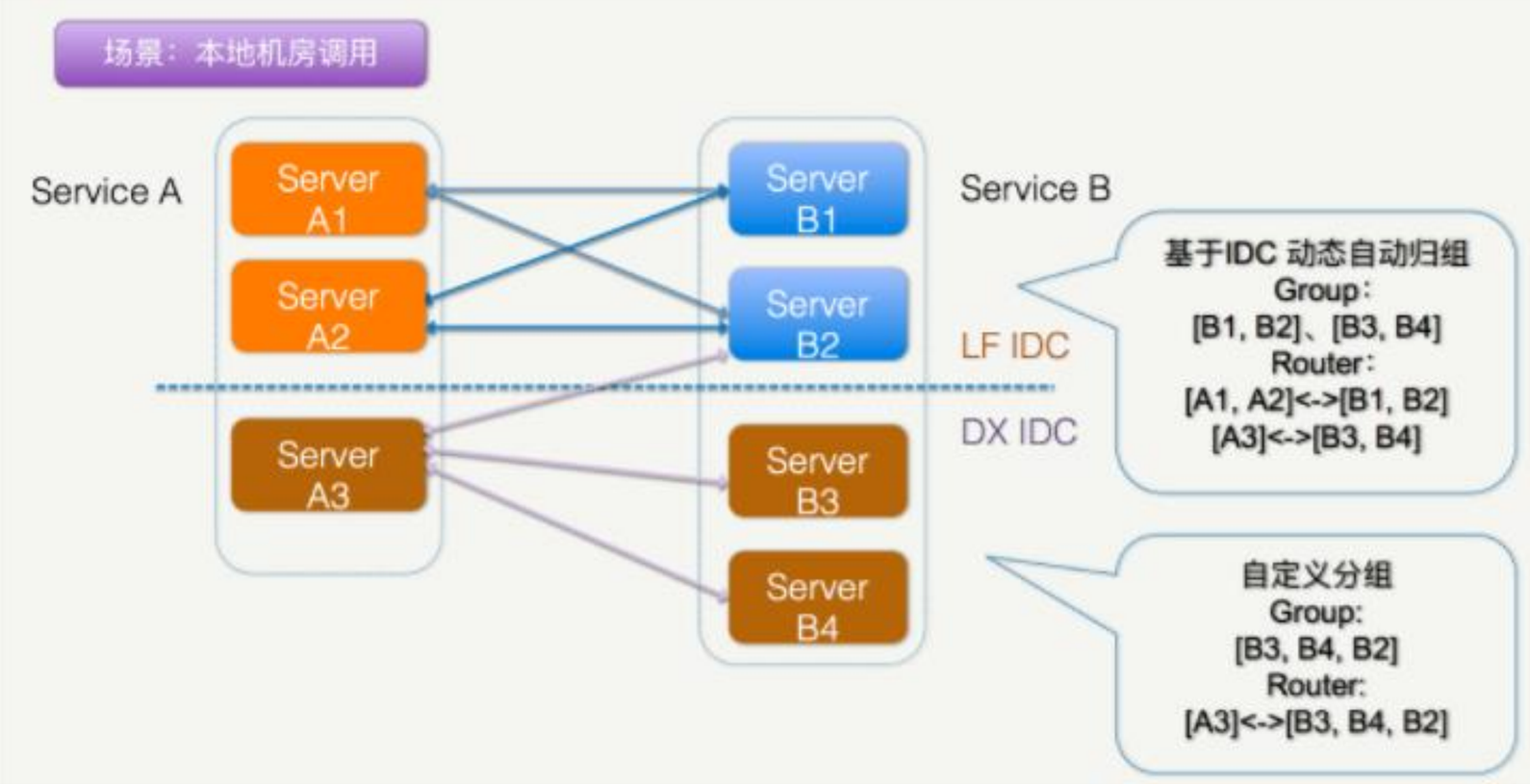
监控报警

调用链数据分析

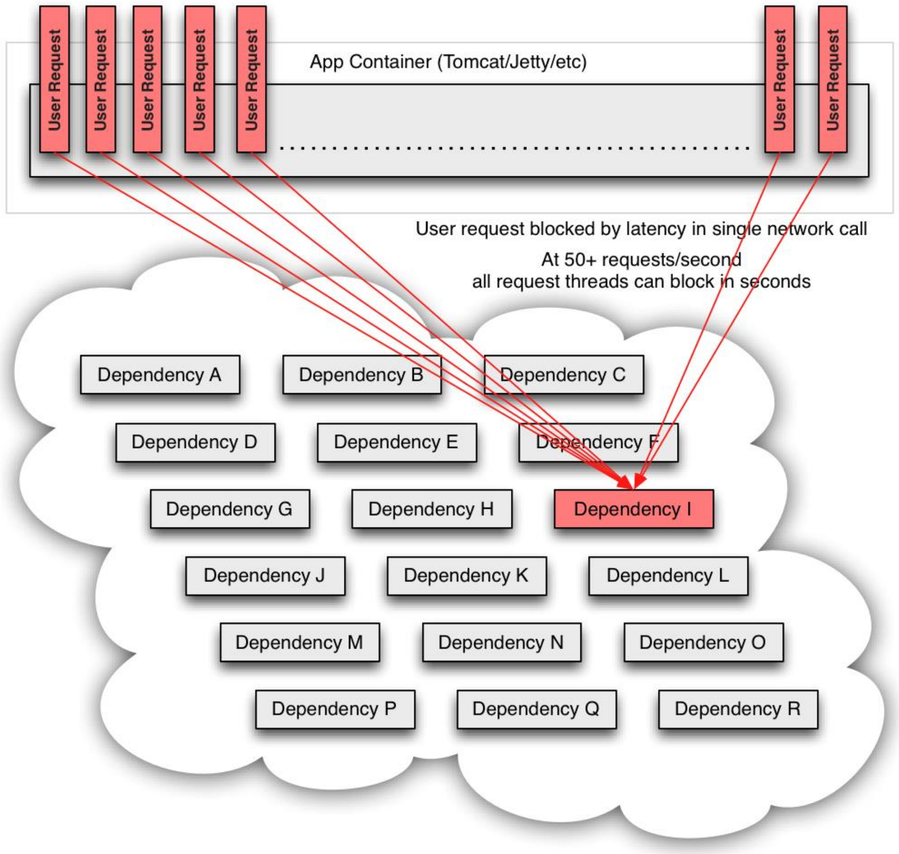
服务化实践-服务治理-权重调节

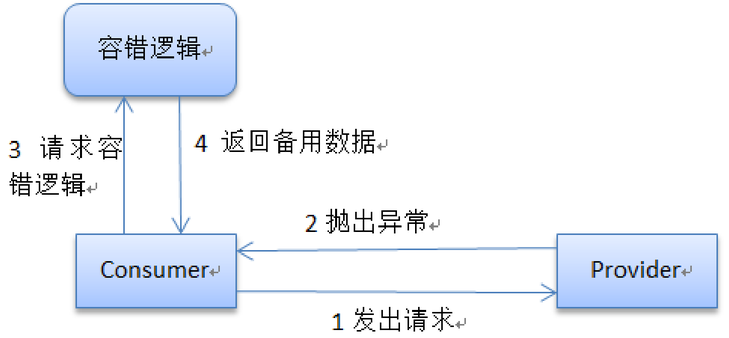


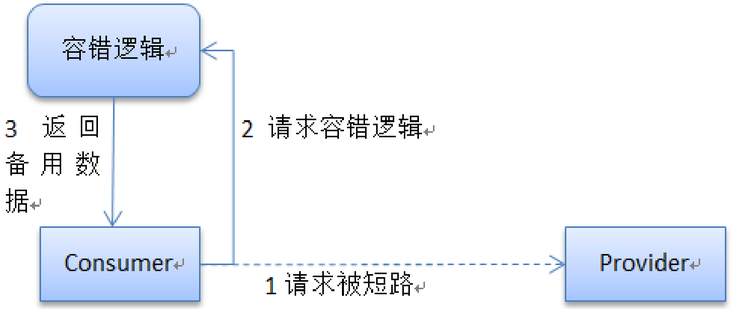
服务治理-本地机房调用



服务化治理-故障隔离



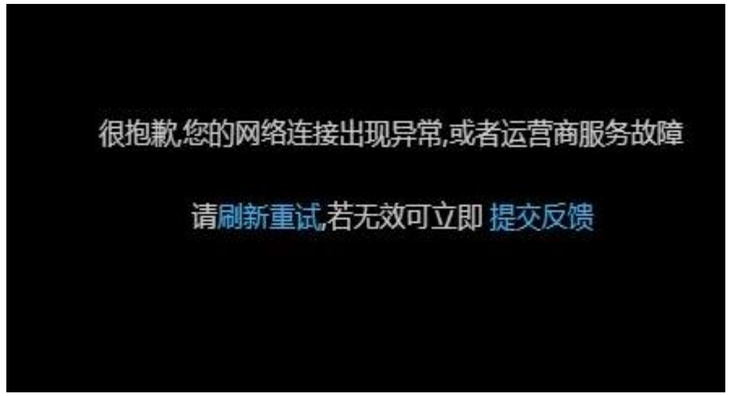


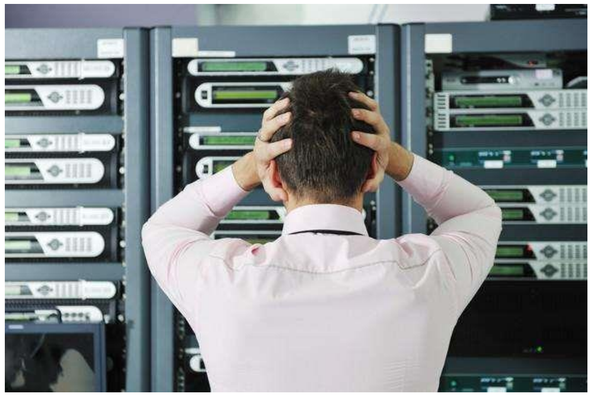


单实例故障

原因：

网络中断
机器故障
程序崩溃

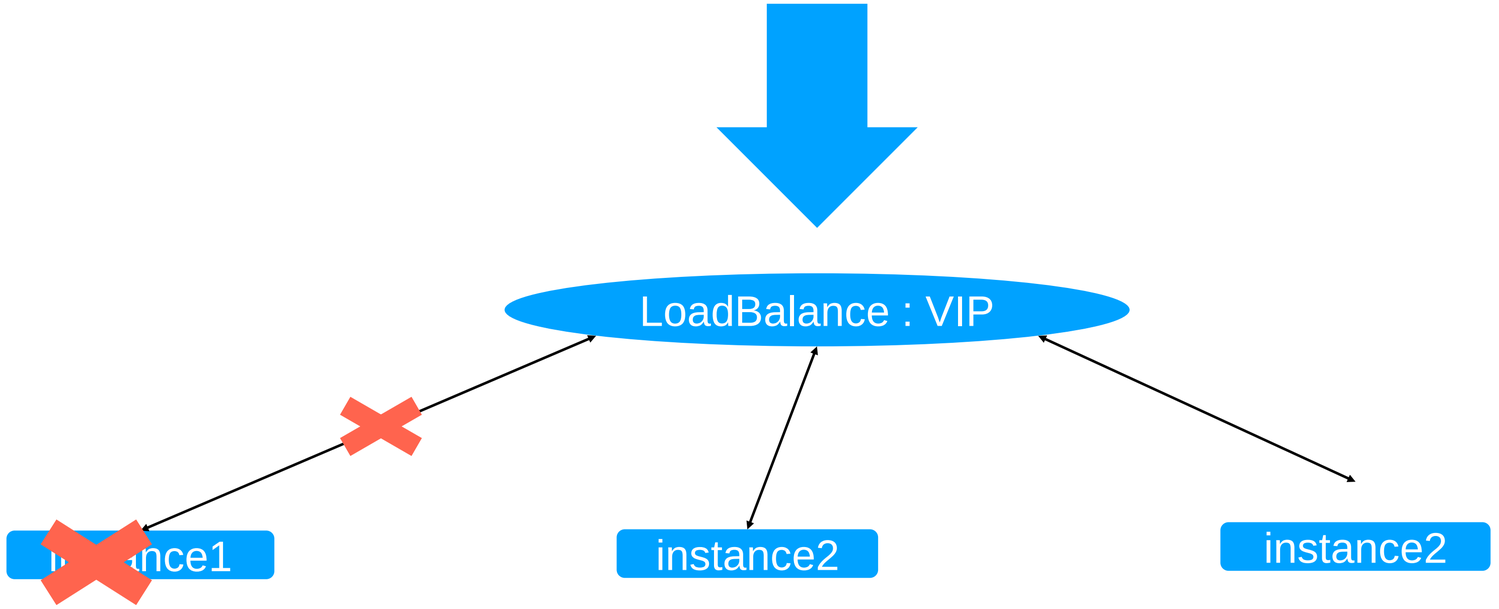




```
Core was generated by `./larbin'.
Program terminated with signal 11, Segmentation fault.
#0  newId (this=0xb7adae9c) at site.cc:366
366         strcpy(name, u->getHost());
(gdb) where
#0  newId (this=0xb7adae9c) at site.cc:366
#1  NamedSite::newQuery (this=0xb7adae9c) at site.cc:182
#2  0x08053bbd in fetchDns () at fetchOpen.cc:55
#3  0x08056b1b in main (argc=Cannot access memory at address 0x19
) at main.cc:101
(gdb)
```

无状态单实例故障

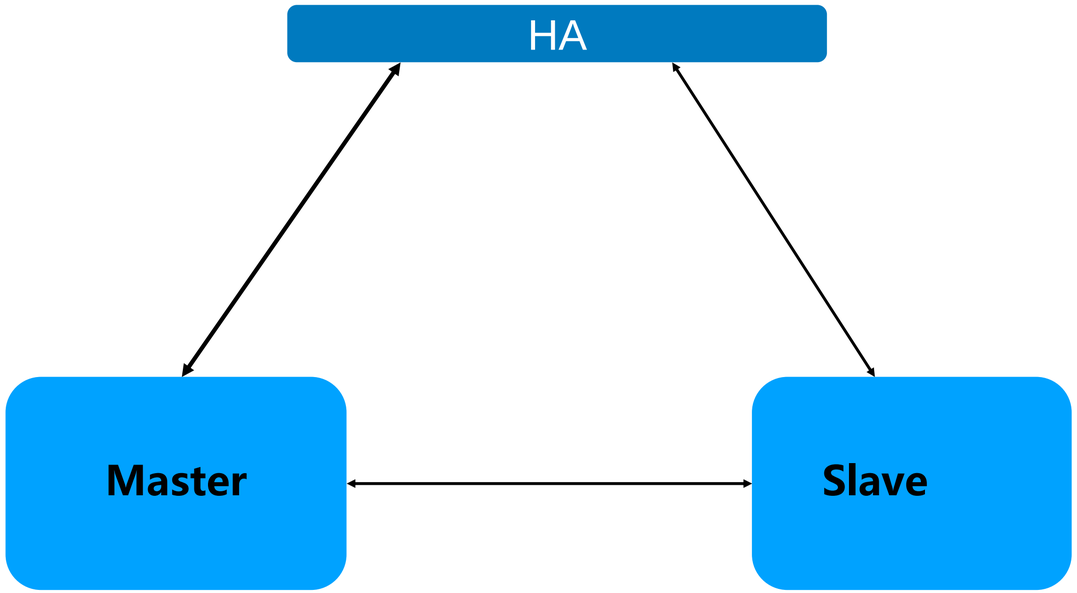
健康心跳
负载均衡
快速失败
超时时间
控制重试
自动恢复



有状态单实例故障

◆第三方仲裁者HA
数据库主备切换

◆自动选举
Paxos
Raft



服务集群故障

原因：

容量不足

变更引起

网络故障

程序BUG

…

服务集群故障-自我保护

限流：

按系统负载限流
按业务优先限流

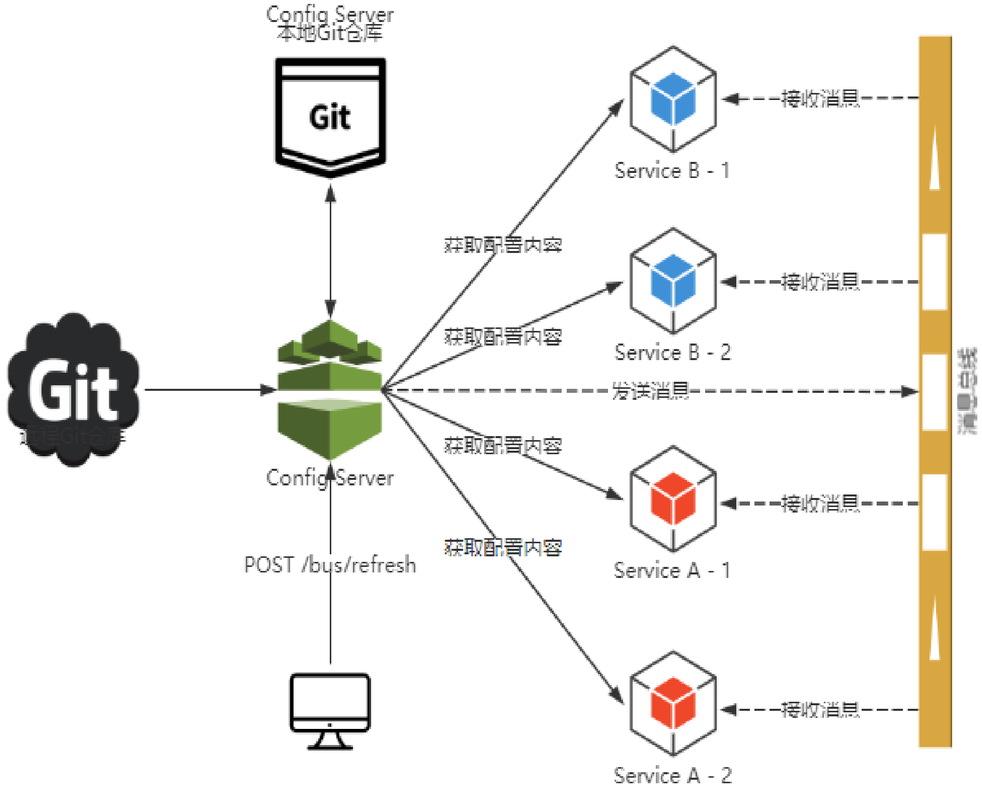
降级：

依赖模块的降级
业务功能的降级

服务集群故障-变更

快速变更

- **容量伸缩**
  - **应用规范**
  - **自动部署**
- **变更管理**
  - **变更记录-版本化**
  - **快速回滚**

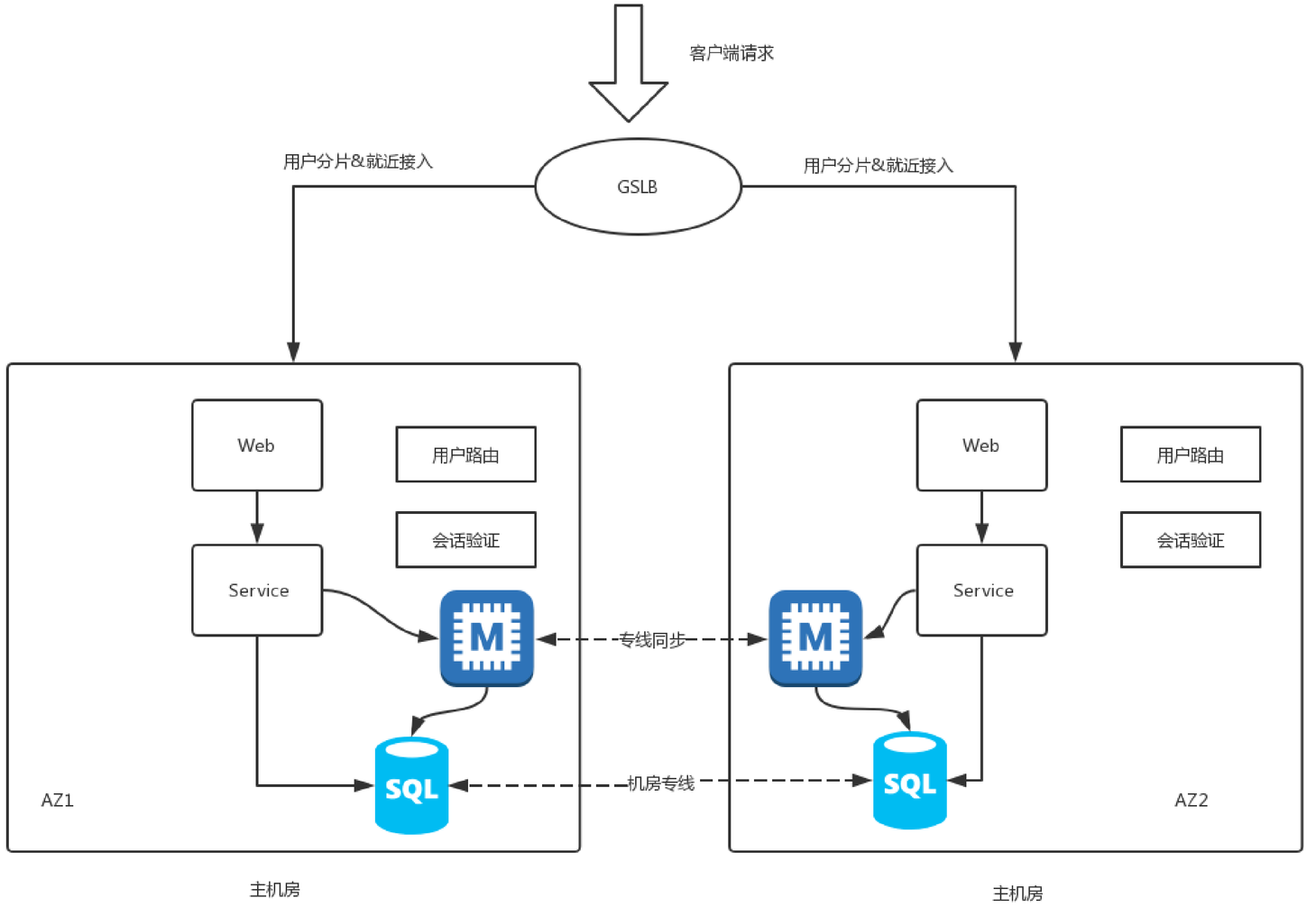


机房层面故障

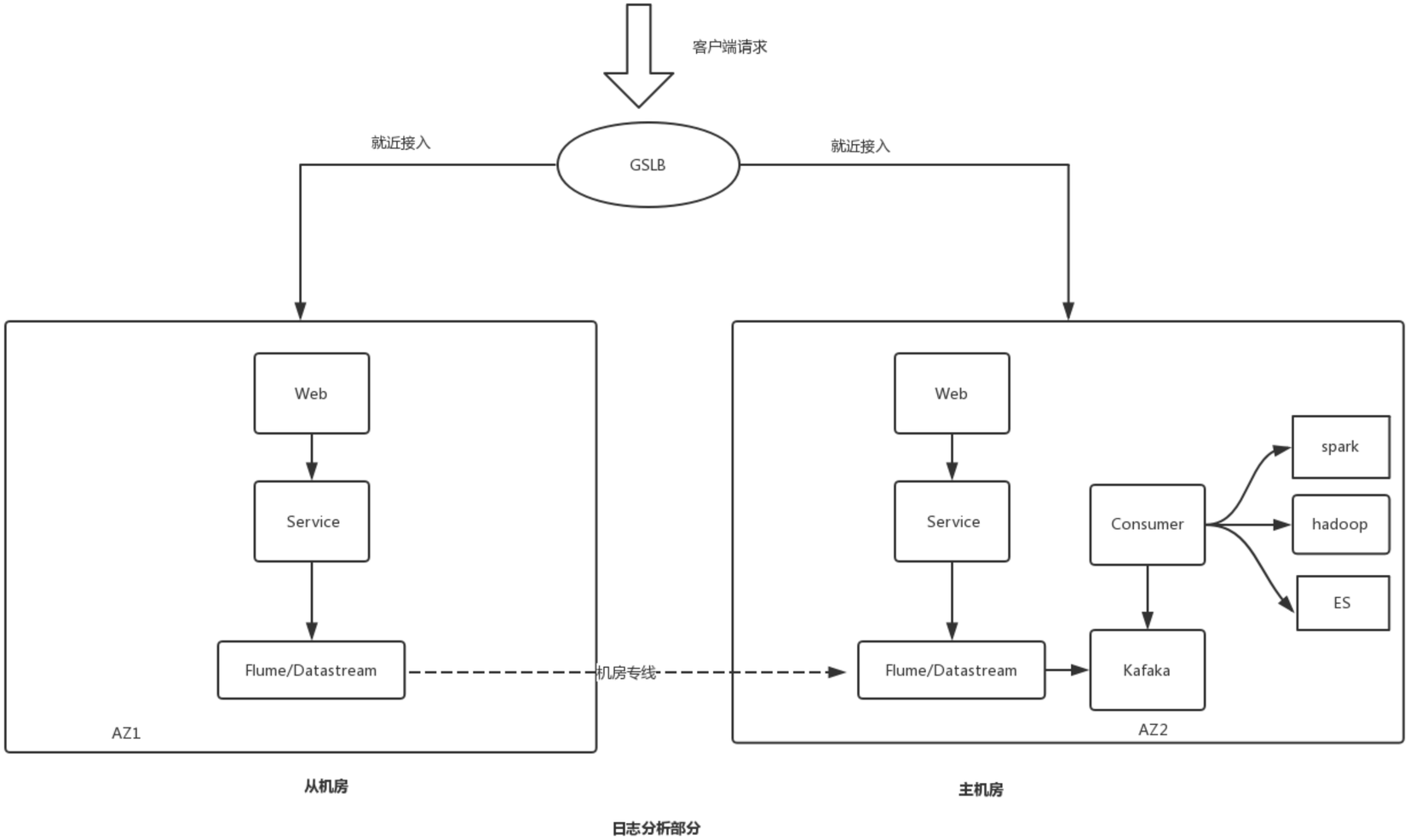
网络核心故障
接入层故障
机房断电
。。。

以不变应万变------多机房/可用区容灾

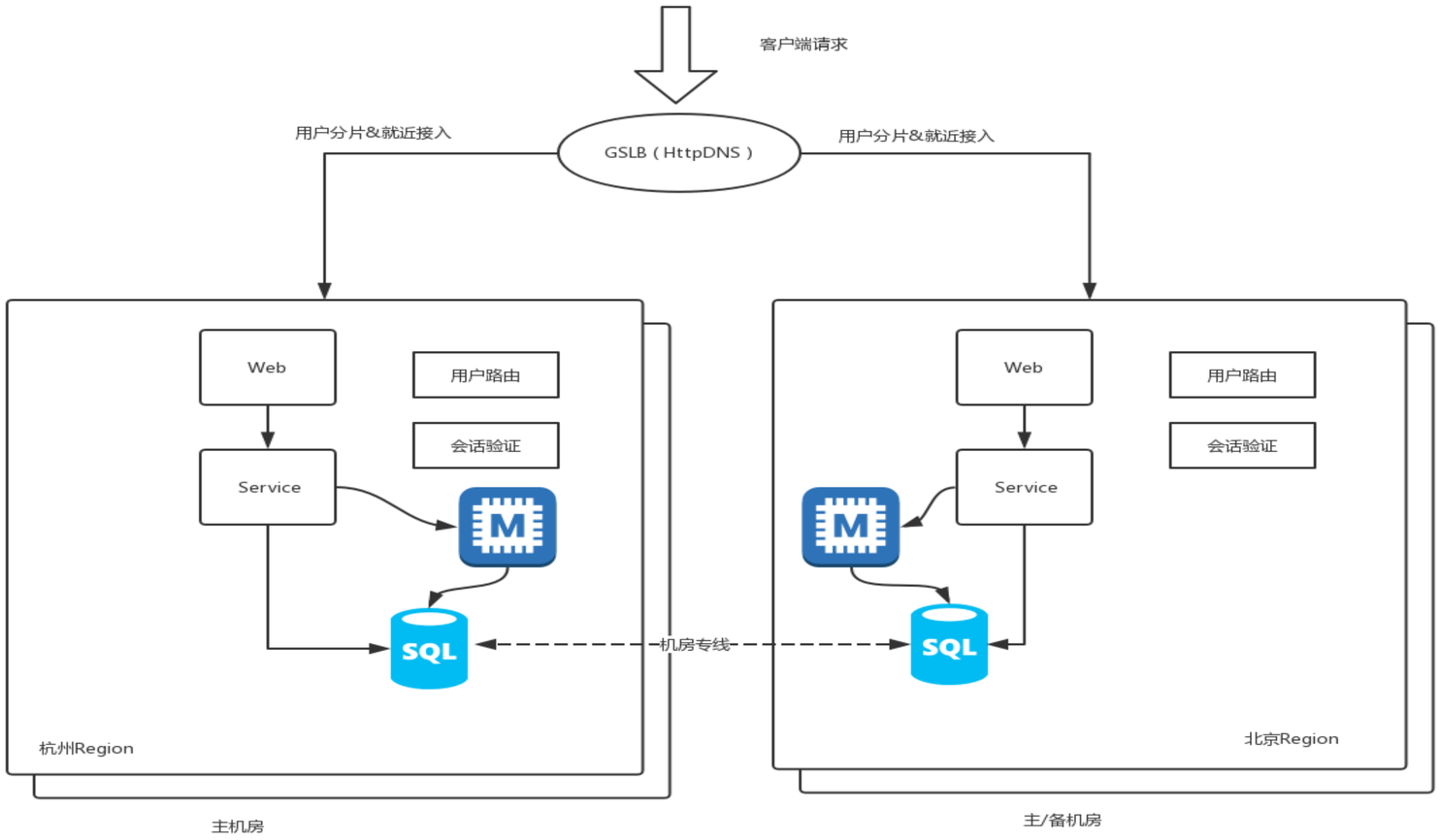
同城-多可用区部署架构-单元化、多活



单元化-主从模式

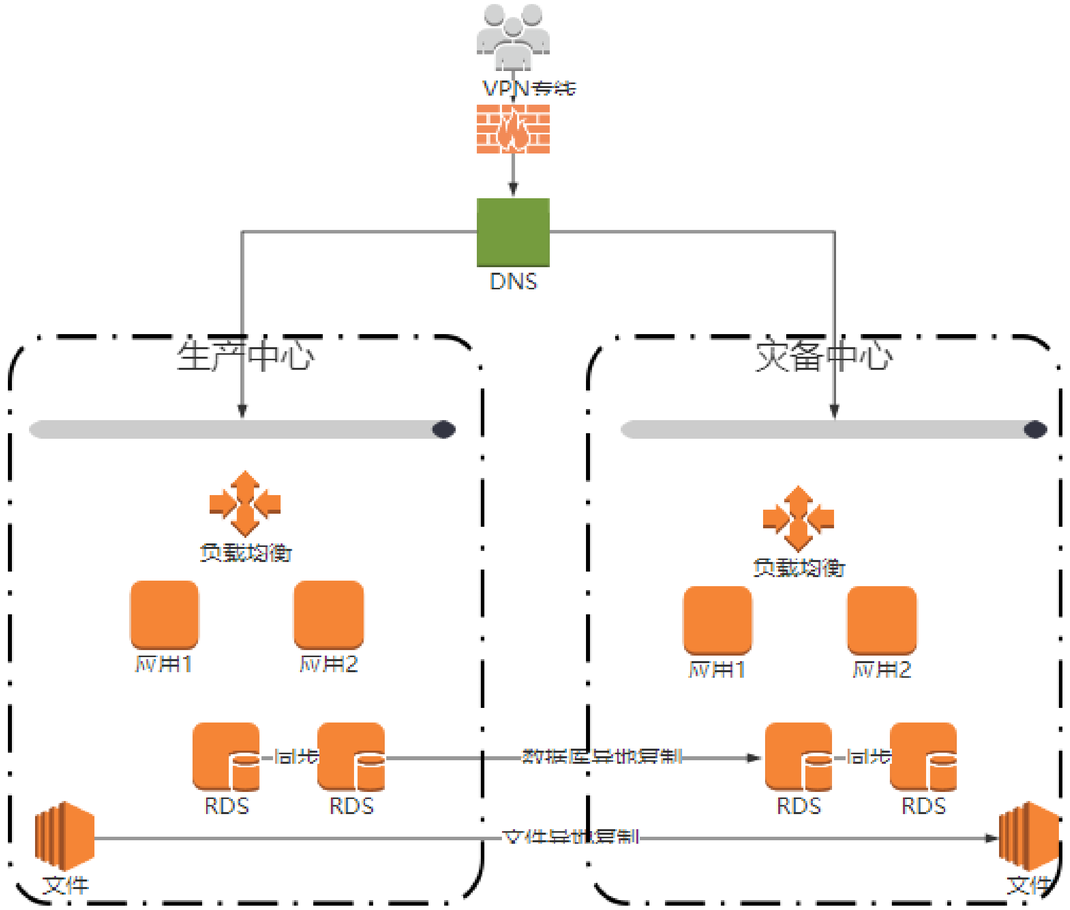


面向跨区域的设计-多活

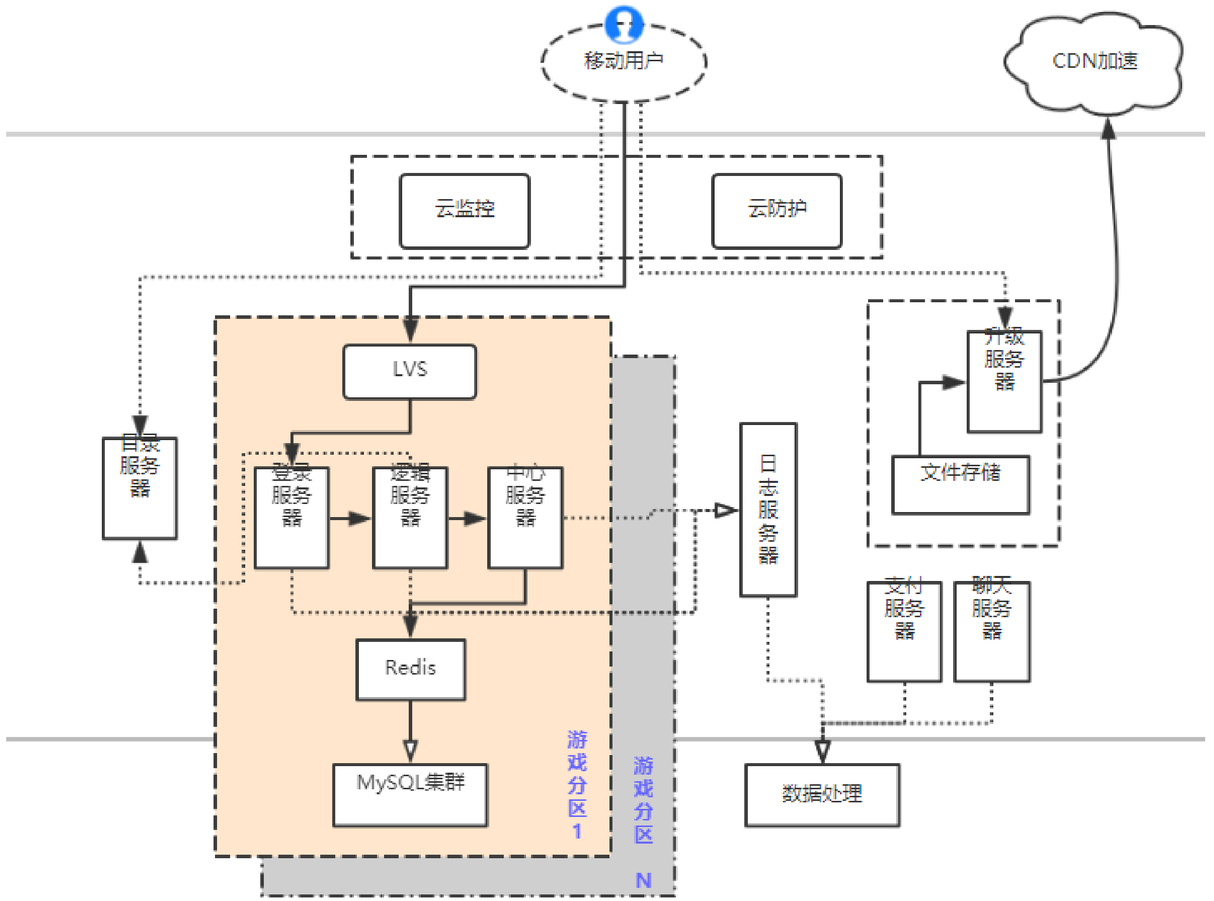


跨Region应用是否多活-处决于数据延迟对业务的影响

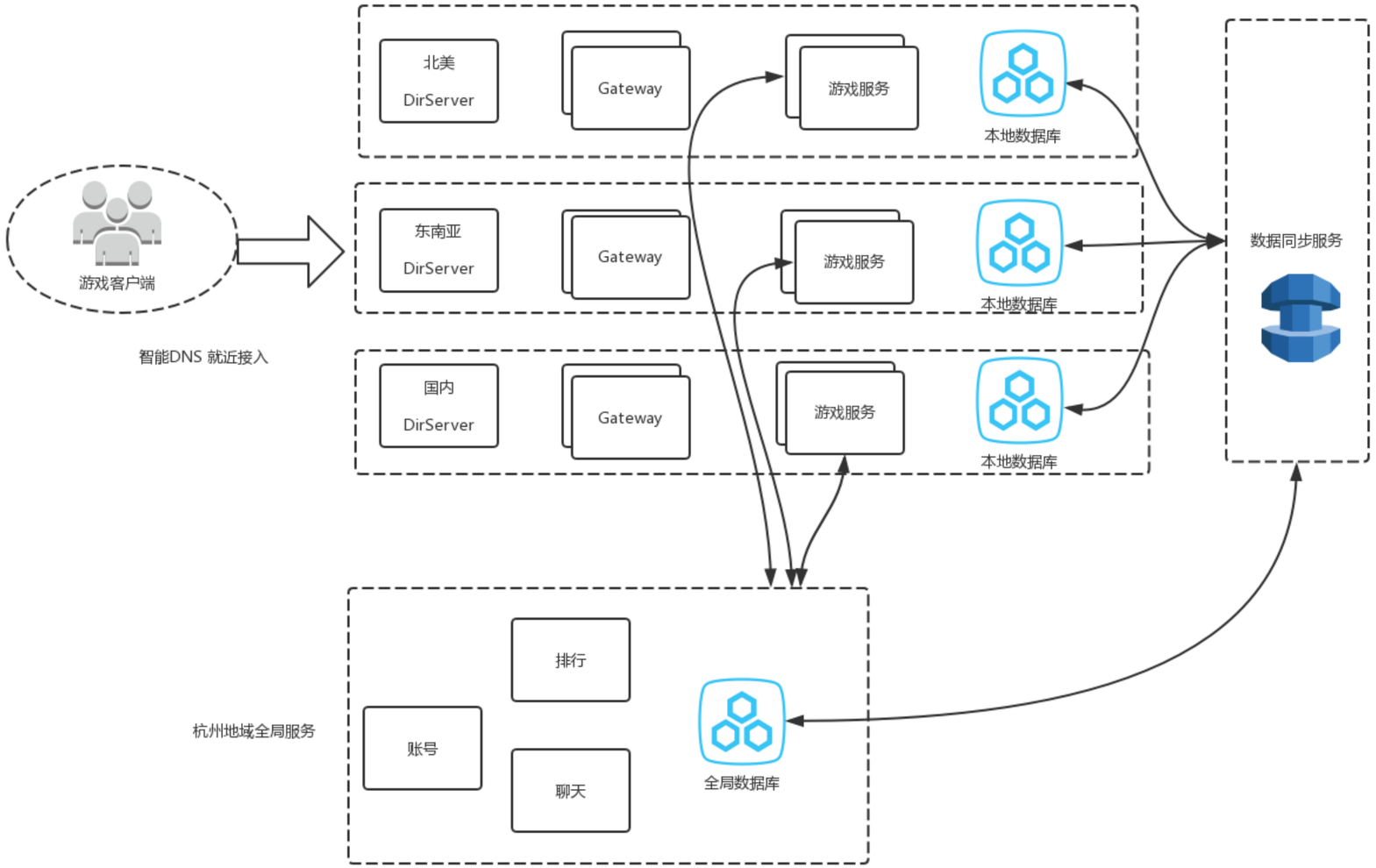
跨Region-异地容灾



云端游戏高可用架构



游戏全球同服架构

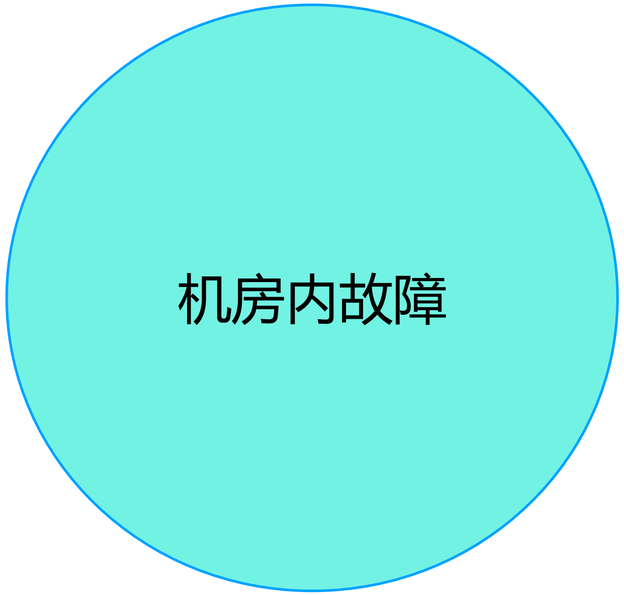


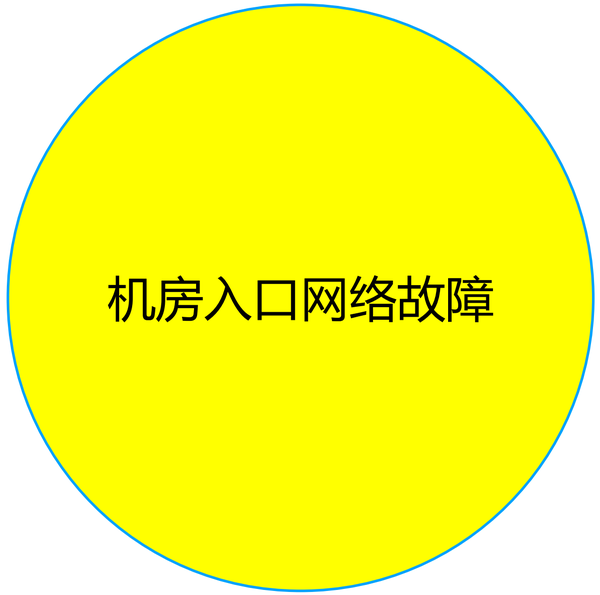
可用性

- Vm挂掉应被当做是 【常态】

- **针对不同的云服务，不同的容灾策略**
  大原则：切、切、切
  VM挂掉的处理
  LB 挂掉的处理
  机房核心网络异常的处理

可用性



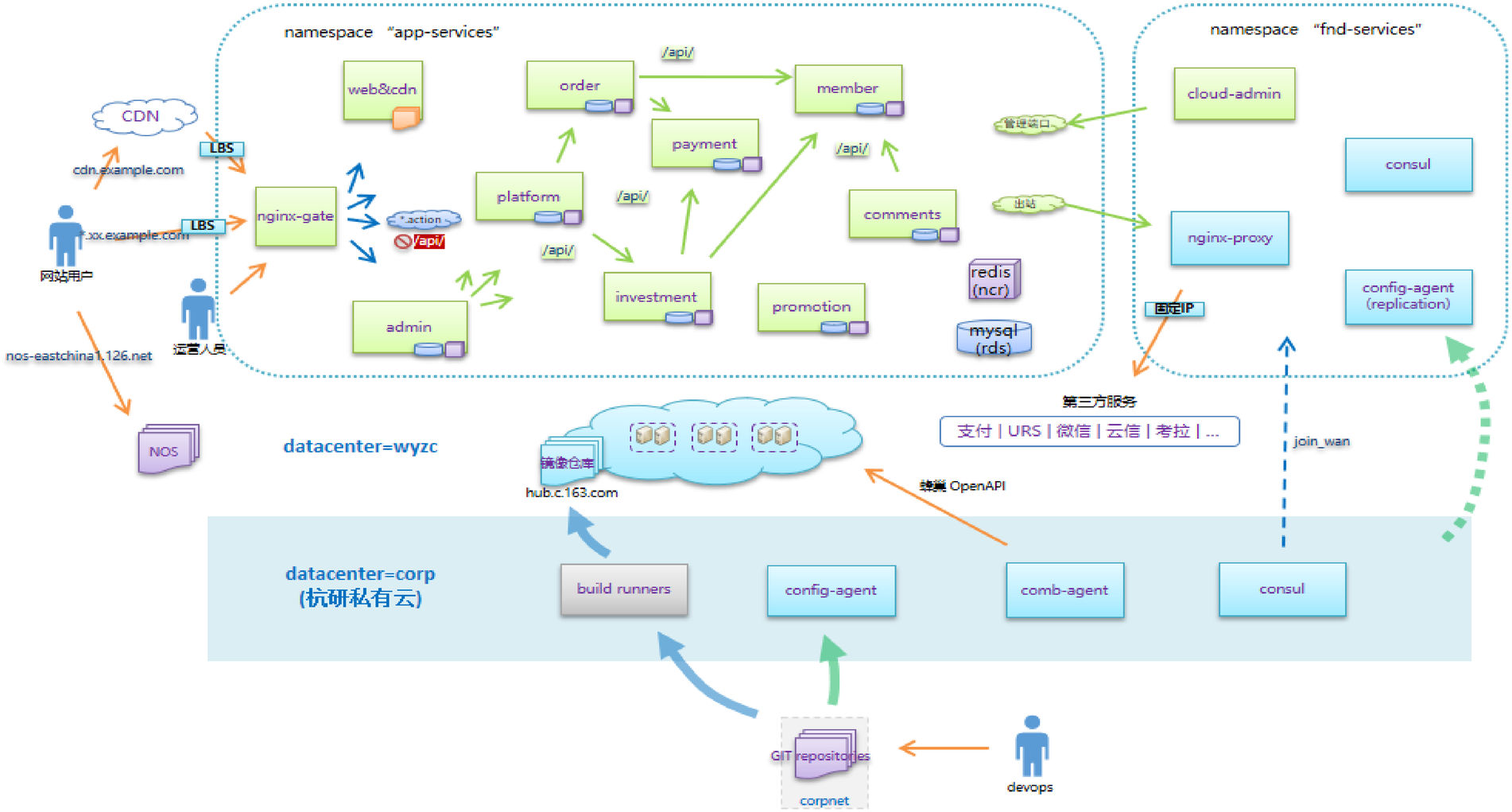


自动流量切换，业务几乎无影响

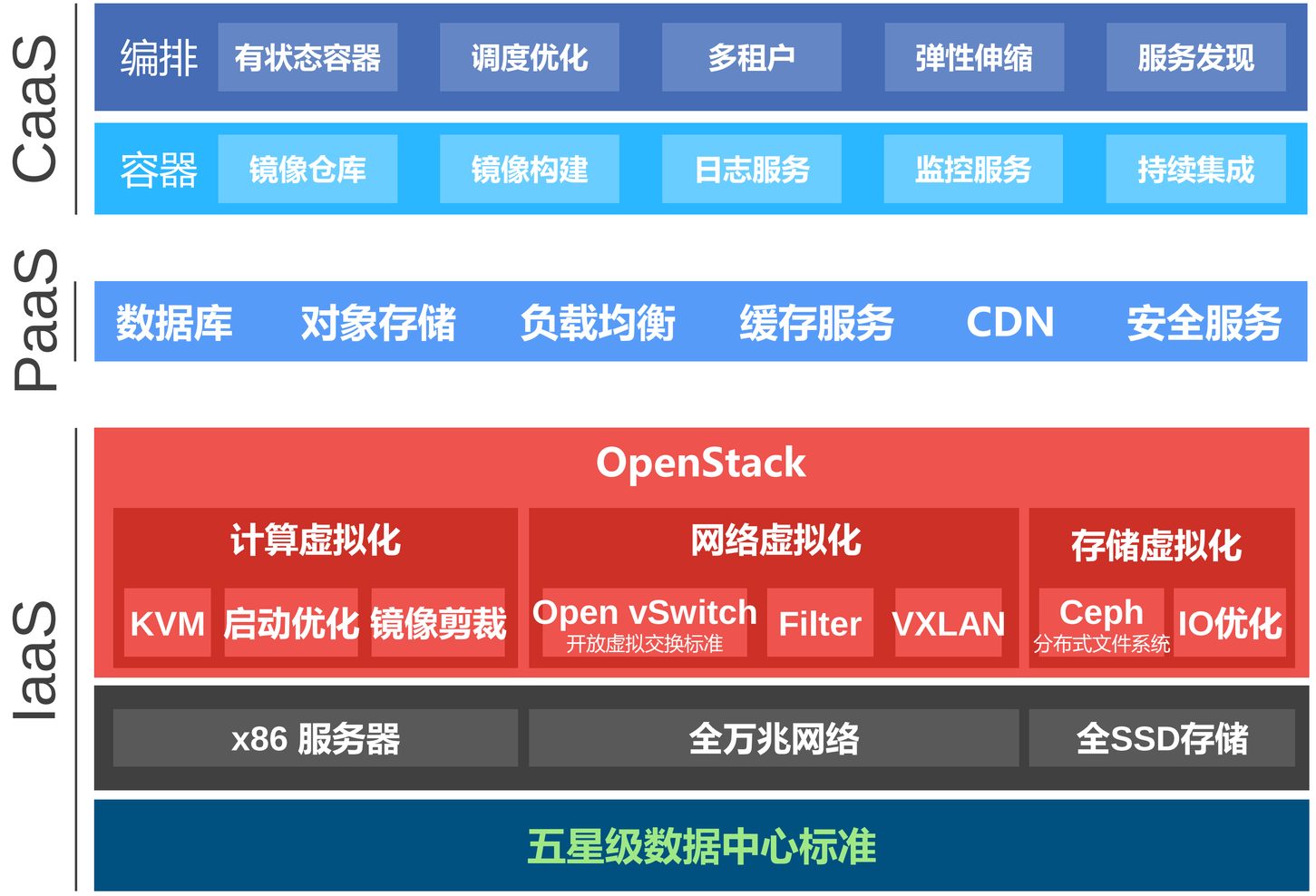
系统扩容

系统能够任意水平扩容

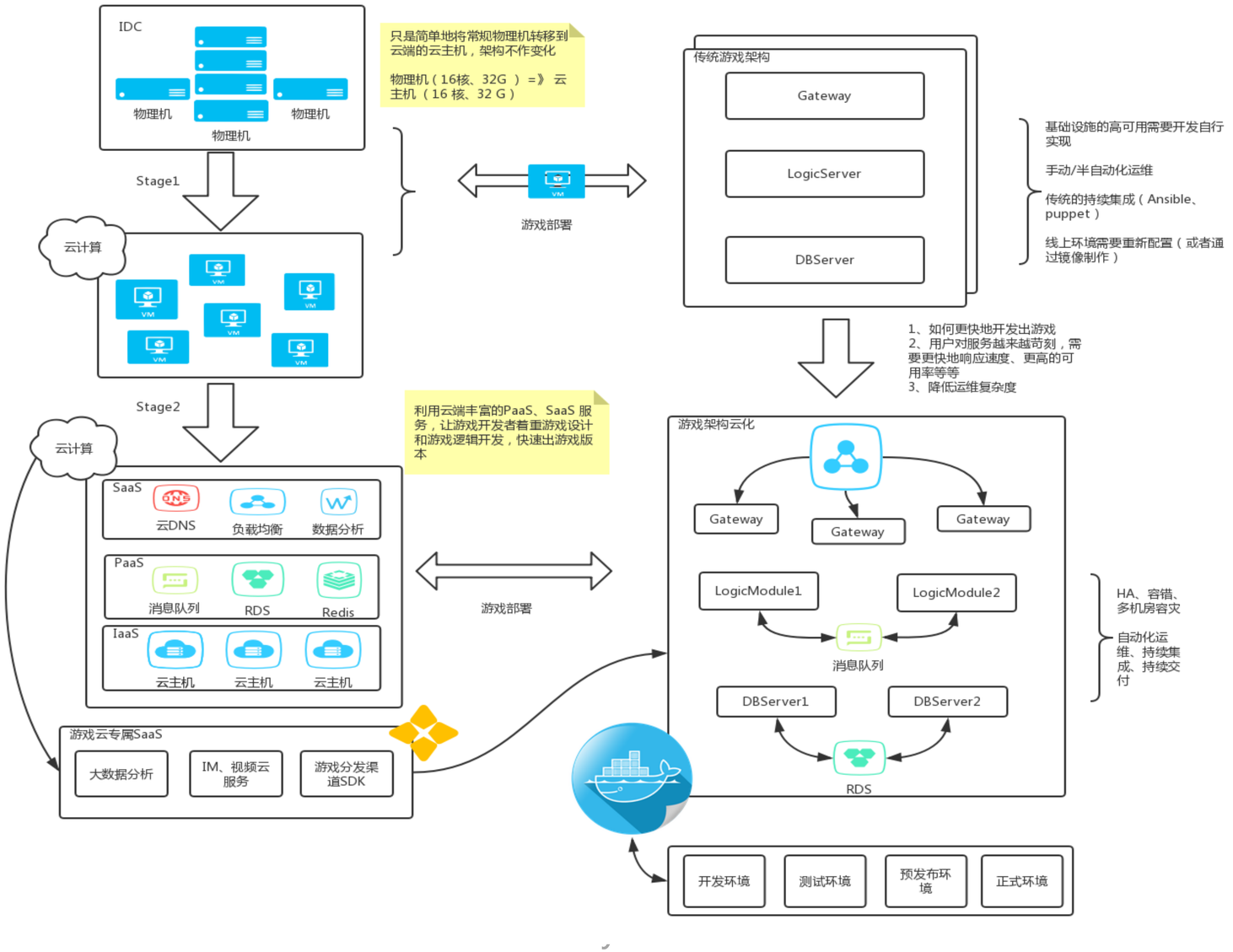
架构演进方向-云端微服务



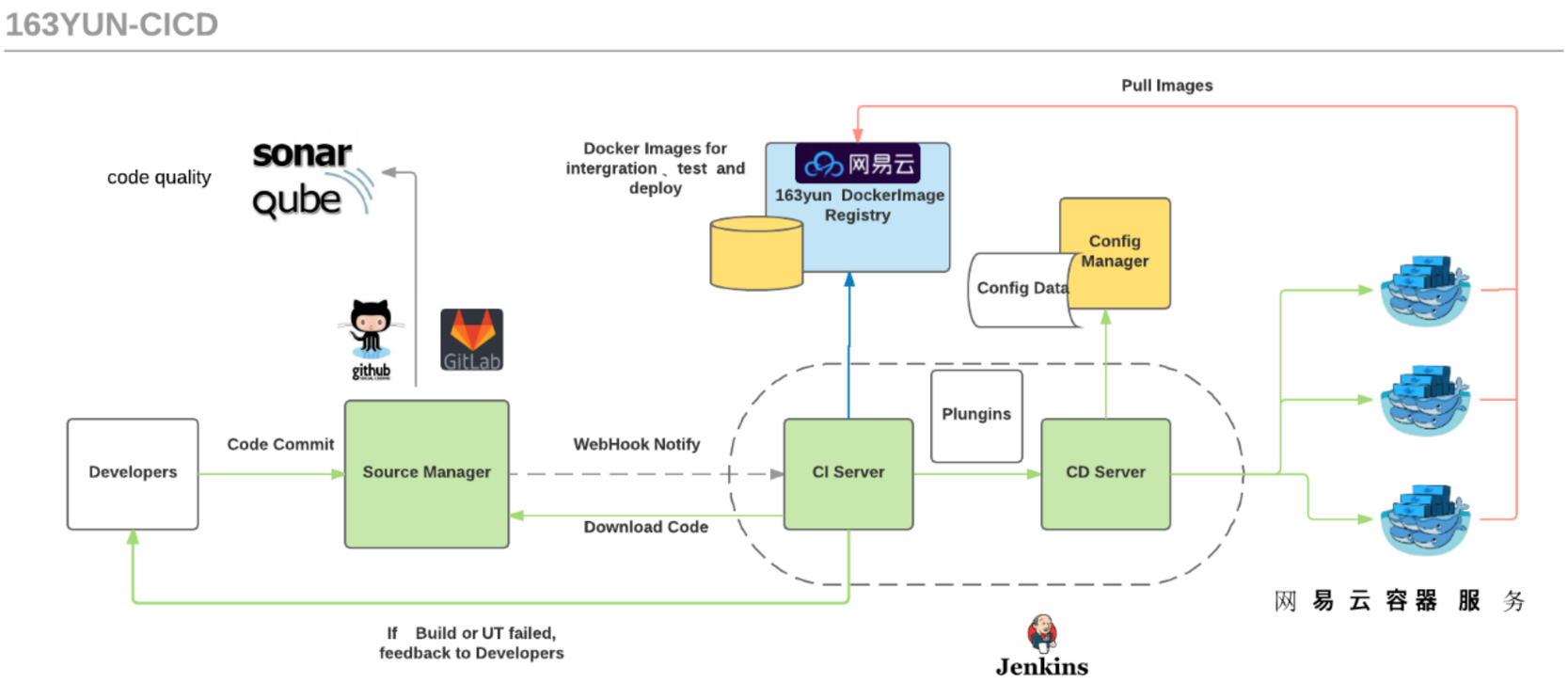
网易云基础服务--总体架构全部基于开源平台



游戏案例


IDC
物理机
物理机
物理机
只是简单地将常规物理机转移到云端的云主机，架构不作变化
物理机（16核、32G ）=》 云主机（16 核、32 G ）
Stage1
云计算
游戏部署
传统游戏架构
Gateway
LogicServer
DBServer
基础设施的高可用需要开发自行实现
手动/半自动化运维
传统的持续集成（Ansible、puppet ）
线上环境需要重新配置（或者通过镜像制作）
1、如何更快地开发出游戏
2、用户对服务越来越苛刻，需要更快地响应速度、更高的可用率等等
3、降低运维复杂度
Stage2
云计算
利用云端丰富的PaaS、SaaS 服务，让游戏开发者着重游戏设计和游戏逻辑开发，快速出游戏版本
SaaS
云DNS
负载均衡
数据分析
PaaS
消息队列
RDS
Redis
IaaS
云主机
云主机
云主机
游戏部署
游戏架构云化
Gateway
Gateway
Gateway
LogicModule1
LogicModule2
消息队列
DBServer1
DBServer2
RDS
HA、容错、多机房容灾
自动化运维、持续集成、持续交付
游戏云专属SaaS
大数据分析
IM、视频云服务
游戏分发渠道SDK
开发环境
测试环境
预发布环境
正式环境

持续交付-网易云最佳实践



架构怎么快速落地？

如果你是网易云客户？

云计算不仅是提供计算、存储、网络

还有Paas 和 SaaS

但光有这些还不够，还需要好的架构-> 架构师

网易云

共创云上精彩世界