ArchSummit

ArchSummit全球架构师峰会 深圳站2016

微博消息系统架构演进

袁武林 @V武林

InfoQ

内容大纲

- **新浪微博消息系统介绍**
- **伴随着业务发展，消息系统的演进过程**
  - ➢ 起步阶段
  - ➢ 快速发展阶段
  - ➢ 稳定阶段
  - ➢ 高可用阶段
- **实践经验**

微博消息系统介绍

- 业务范围
  - ➢ 点对点私信
  - ➢ 群组聊天
  - ➢ 消息群发

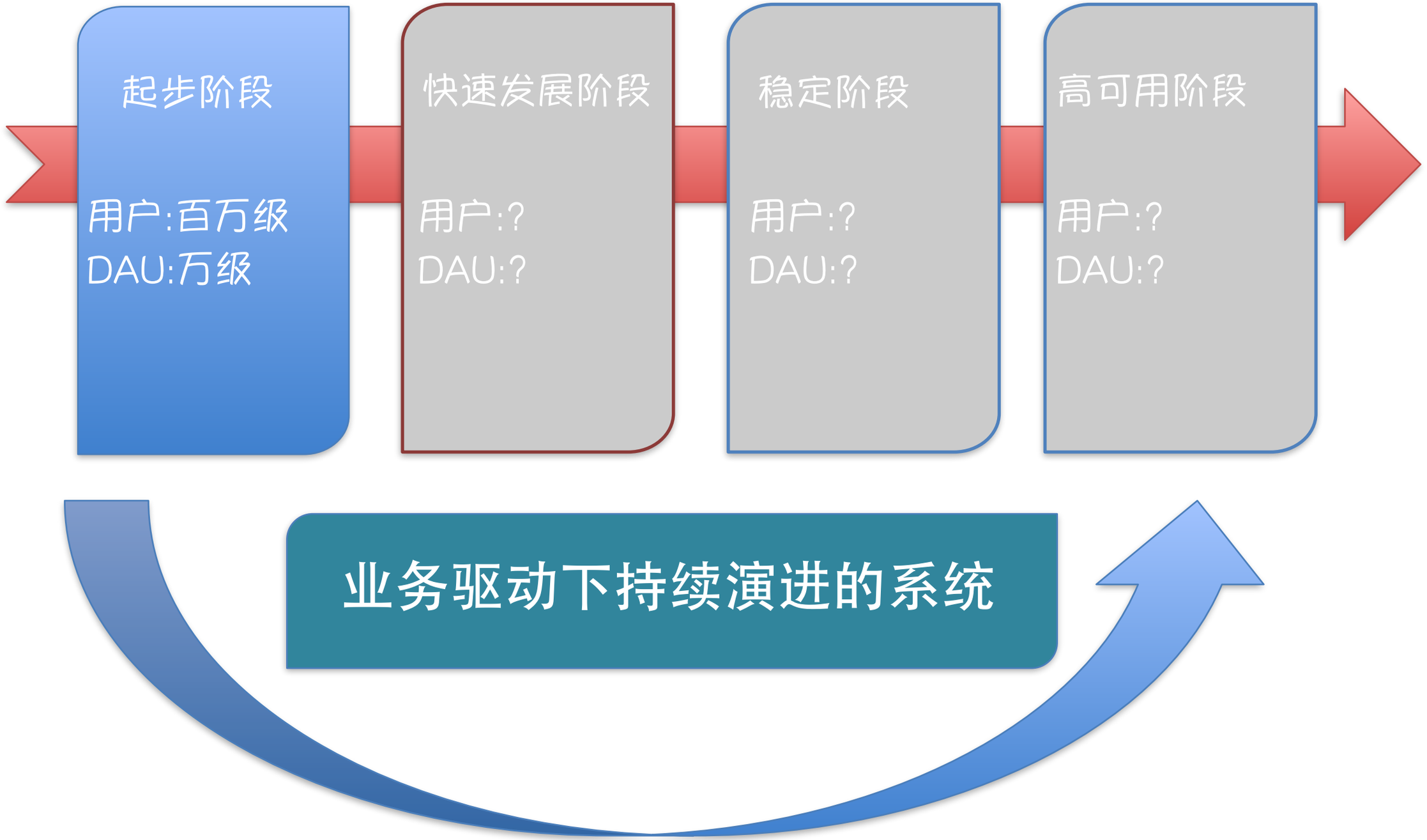
业务数据和系统能力

用户数	亿级
同时在线连接数	千万级
点对点消息日发送量	千万级
群发消息日发送量	十亿级
消息发送能力	百万/s

大致经历的几个阶段

- 起步阶段－从无到有。**快速实现功能**
- 快速发展阶段－用户快速增长。**提升吞吐量和性能**
- 稳定阶段－ 移动用户占比高。**针对移动端优化**
- 高可用阶段－用户注重服务质量。**提高服务可用性**

消息系统演进

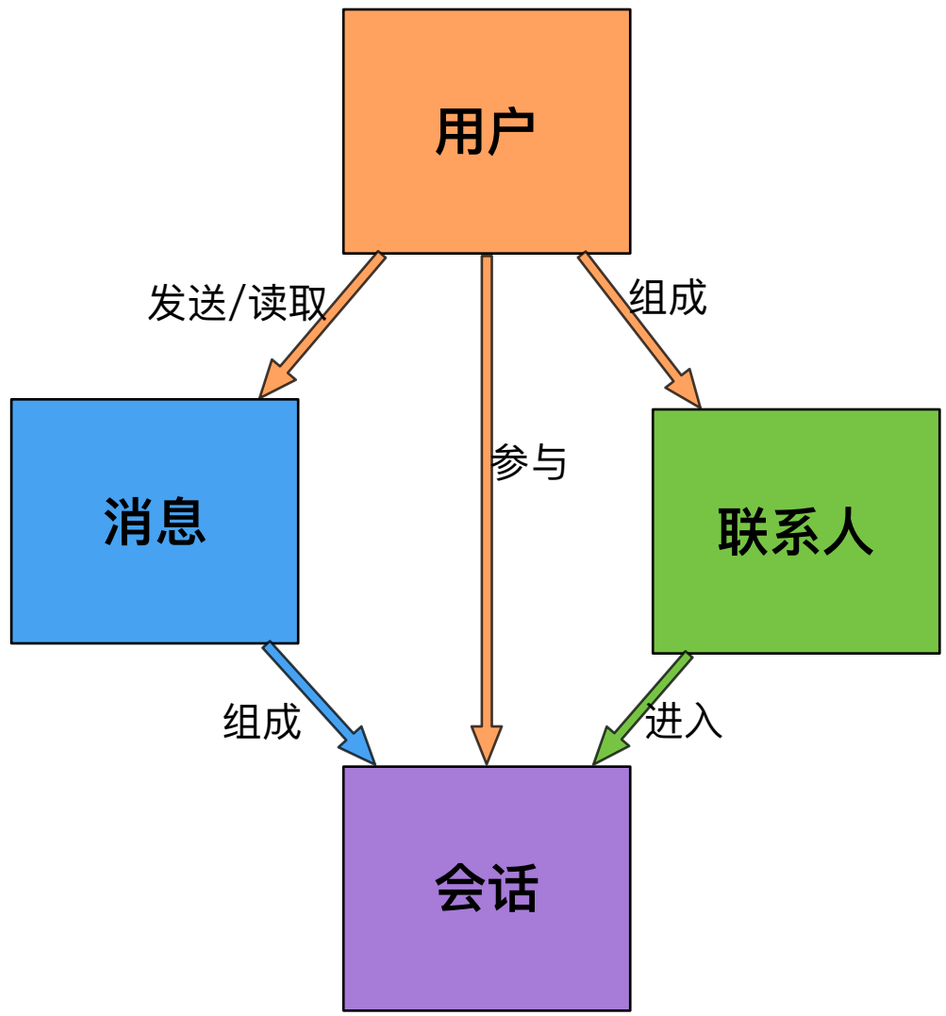

起步阶段
用户:百万级
DAU:万级
快速发展阶段
用户:?
DAU:?
稳定阶段
用户:?
DAU:?
高可用阶段
用户:?
DAU:?
业务驱动下持续演进的系统

起步阶段-快速实现

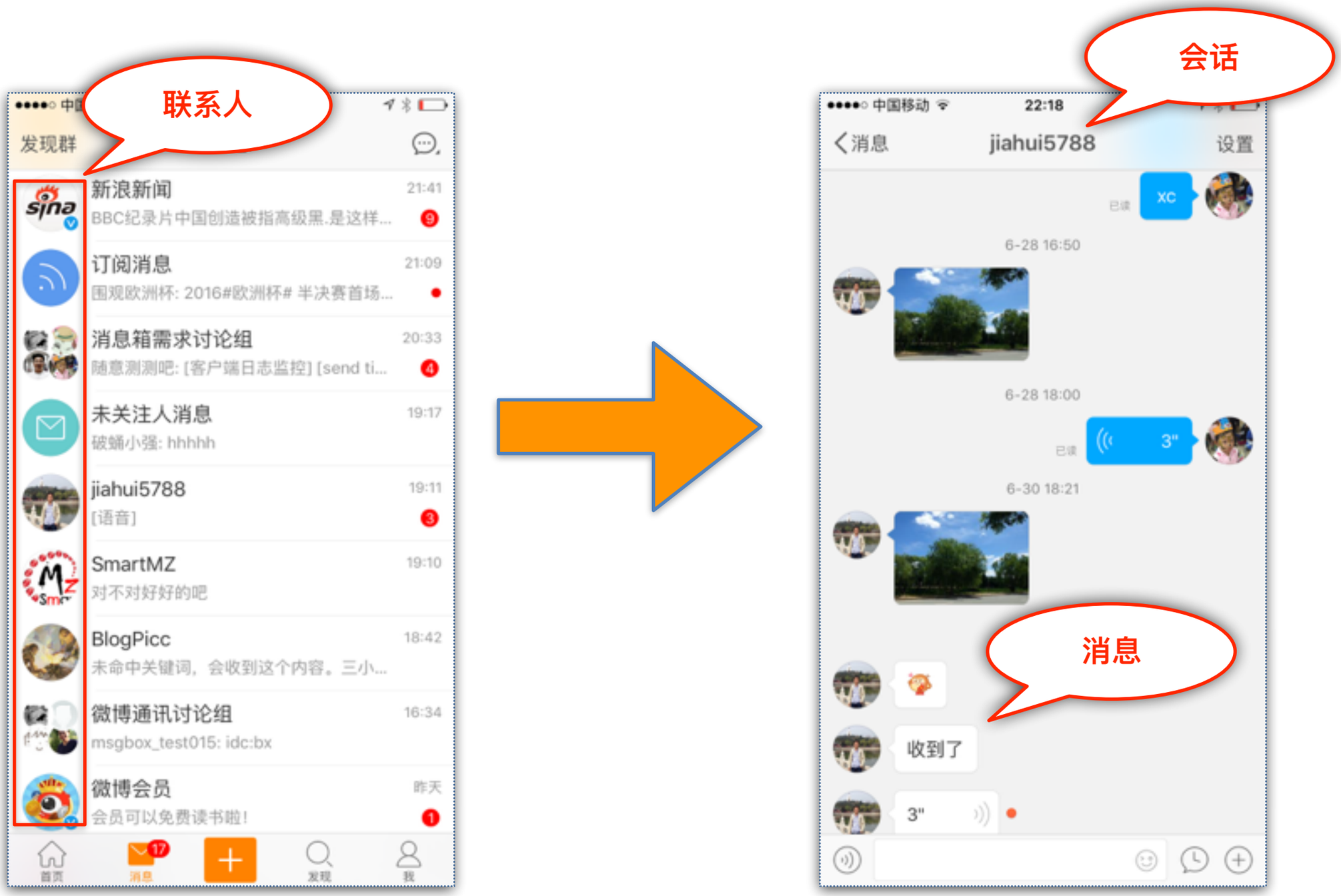
消息系统的构成

- **消息系统的基本要素**
  - ➢ 用户、联系人、消息、会话

- **消息系统的基本功能**
  - ➢ 消息发送和读取。
  - ➢ 关系维护。
  - ➢ 会话管理。



起步阶段-快速实现


联系人
发现群
新浪新闻
21:41
BBC纪录片中国创造被指高级黑.是这样...
订阅消息
21:09
围观欧洲杯: 2016#欧洲杯# 半决赛首场...
消息箱需求讨论组
20:33
随意测测吧: [客户端日志监控] [send ti...
未关注人消息
19:17
破蛹小强: hhhhh
jiahui5788
19:11
[语音]
SmartMZ
19:10
对不对好好的吧
BlogPicc
18:42
未命中关键词，会收到这个内容。三小...
微博通讯讨论组
16:34
msgbox_test015: idc:bx
微博会员
昨天
会员可以免费读书啦!
首页
消息
发现
我
会话
中国移动
22:18
消息
jiahui5788
设置
XC
6-28 16:50
6-28 18:00
3"
6-30 18:21
消息
收到了
3"

起步阶段-面临的挑战

业务层面

- ➢ 从无到有
- ➢ 需求迭代周期短，变更频繁

架构层面

- ➢ 技术选型
- ➢ 快速开发

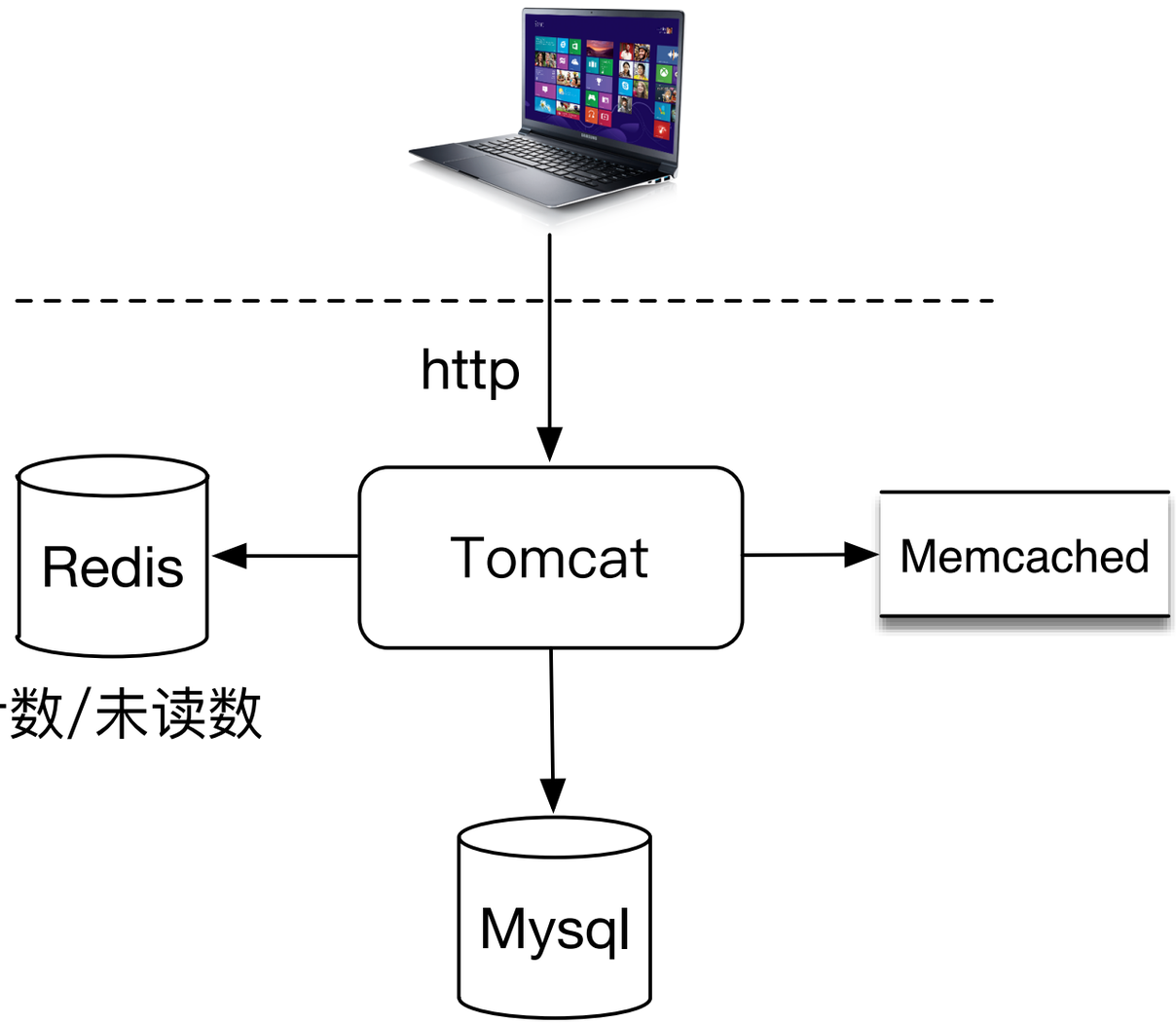
起步阶段-快速实现

如何解决?

- **技术选型**
  - ➢ 成熟可控的技术栈
  - ➢ 可复用的经验
- **快速开发**
  - ➢ 快速迭代（回归测试，上线回滚自动化）

起步阶段-快速实现

- **起步阶段的架构**
  - ➢ 功能简单，站内信模式。
  - ➢ 通过http短连实现消息收发。



起步阶段-快速实现

- **实践经验**

➢ “实用主义”

——选择成熟可控的技术栈。

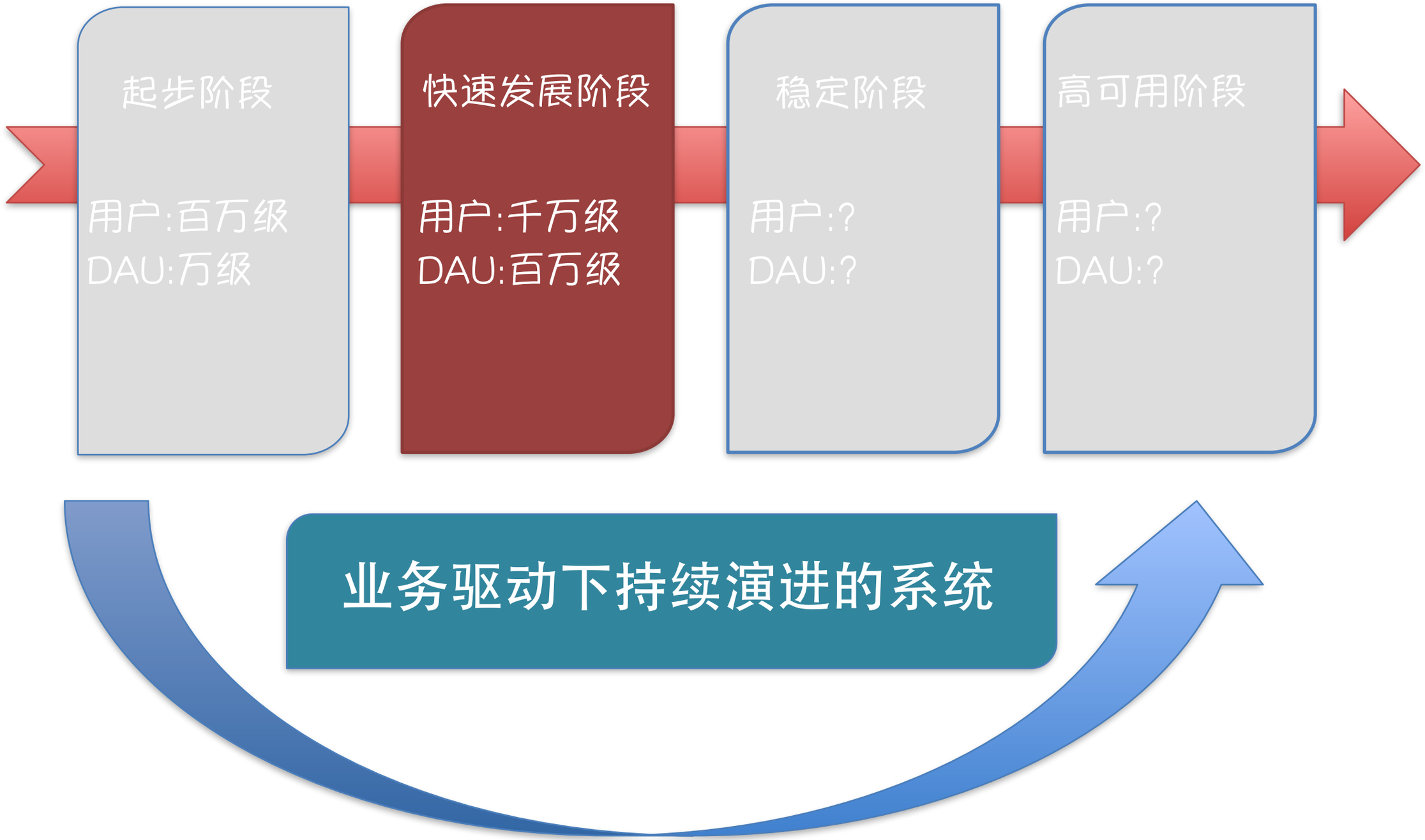
➢ “拿来主义”

——借鉴已有的验证过的经验和成果。

➢ 通过工具提高开发上线效率

——回归测试、自动化上线回滚

消息系统演进


起步阶段
用户:百万级
DAU:万级
快速发展阶段
用户:千万级
DAU:百万级
稳定阶段
用户:?
DAU:?
高可用阶段
用户:?
DAU:?
业务驱动下持续演进的系统

快速发展阶段-面临的问题

业务的变化

- ➢ 整体用户量和访问量快速增长。
- ➢ 新业务不断接入：消息群发、群组聊天。

架构的挑战

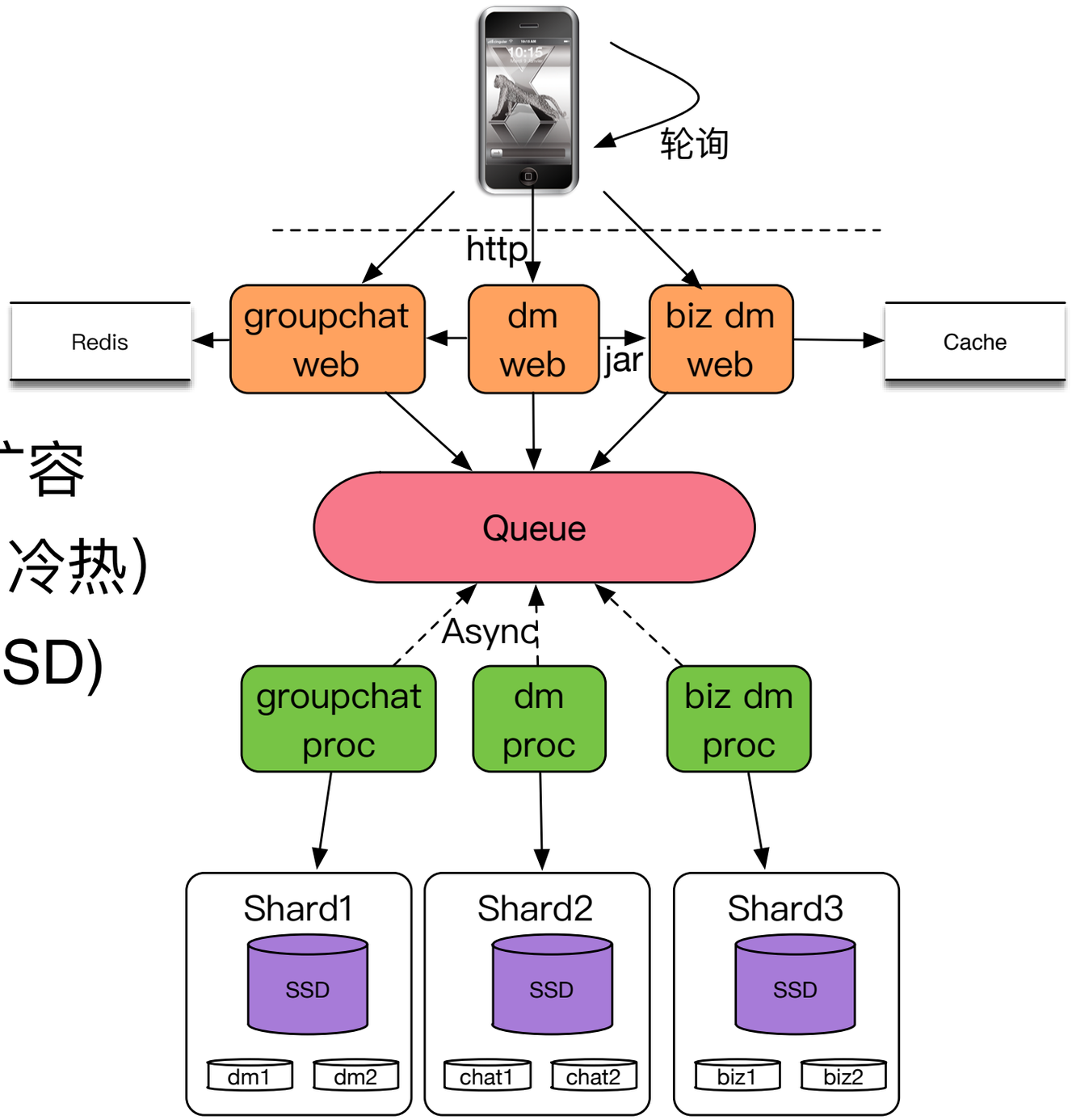
- ➢ 业务复杂度增加，系统出现瓶颈。
- ➢ 突发流量峰值应对能力差。

快速发展阶段-吞吐量问题

如何解决?

系统吞吐量瓶颈

- 异步写
- 业务拆分，热点业务可独立扩容
- 分库分表（eg. 业务，用户，冷热）
- 提高设备硬件性能（SAS->SSD)



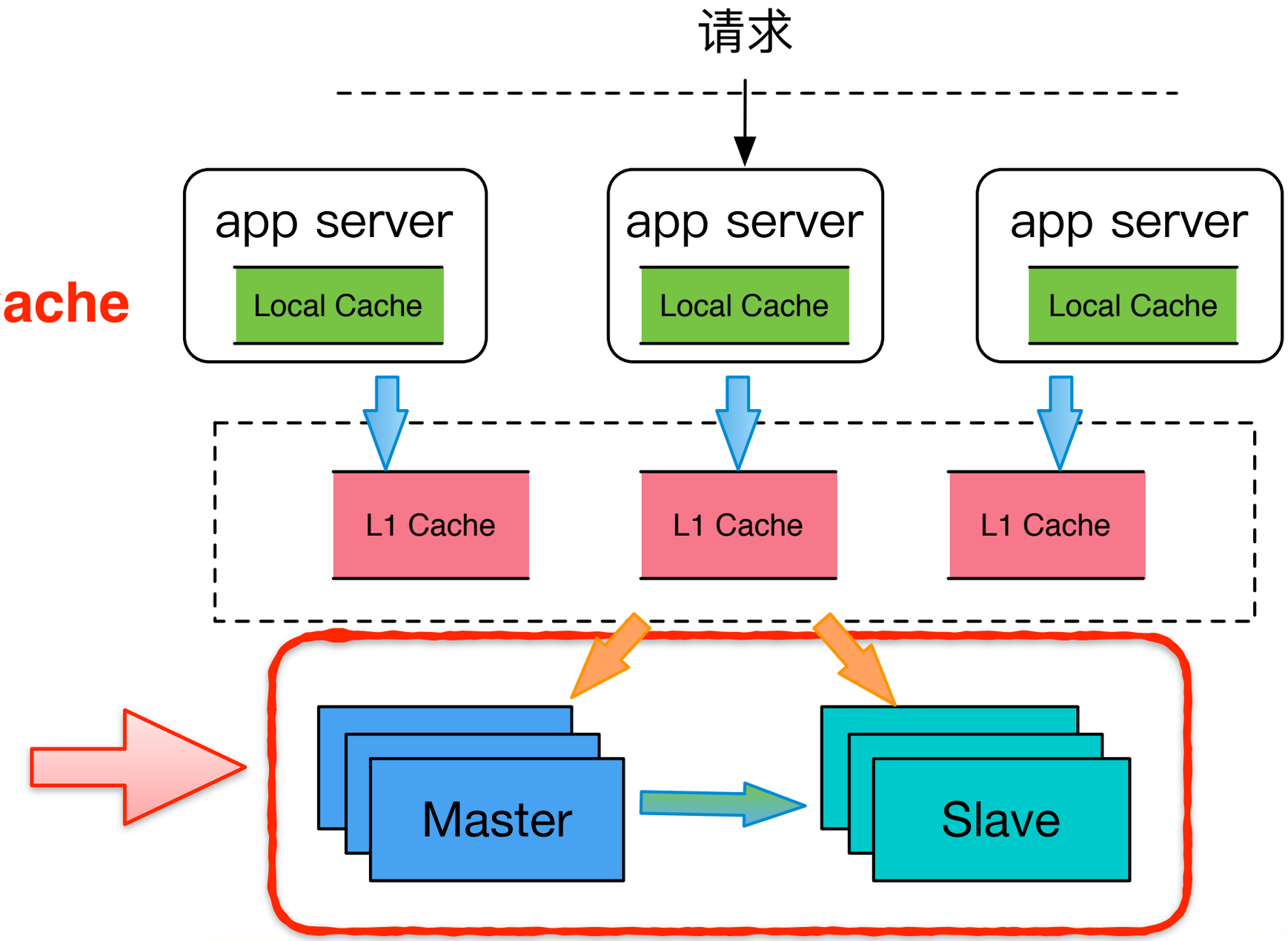
快速发展阶段-峰值应对

如何解决?

突发流量峰值应对

多级Cache防热点

- **Master-Slave Cache**
- L1 Cache
- LocalCache



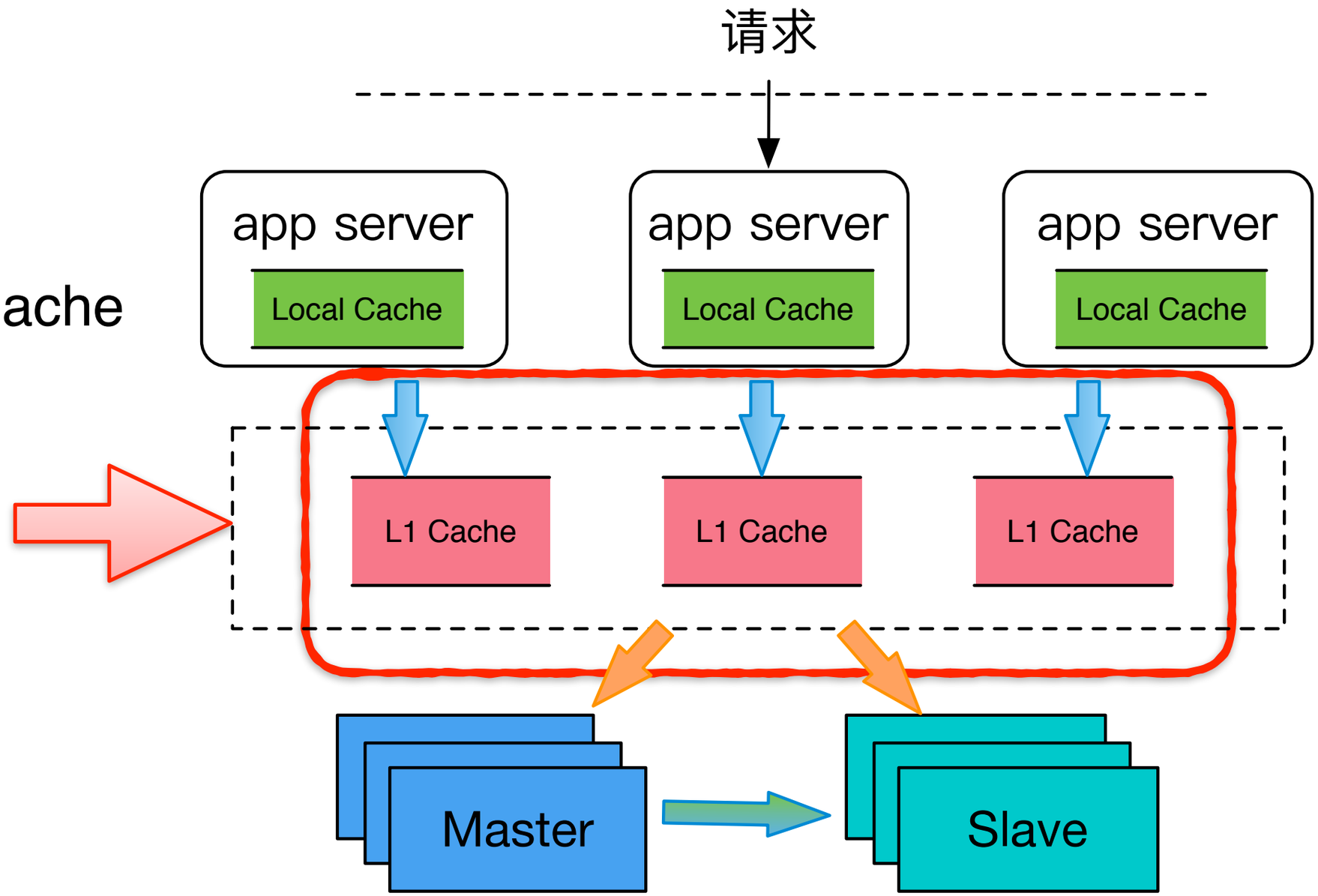
快速发展阶段-峰值应对

如何解决?

突发流量峰值应对

多级Cache防热点

- Master-Slave Cache
- **L1 Cache**
- LocalCache



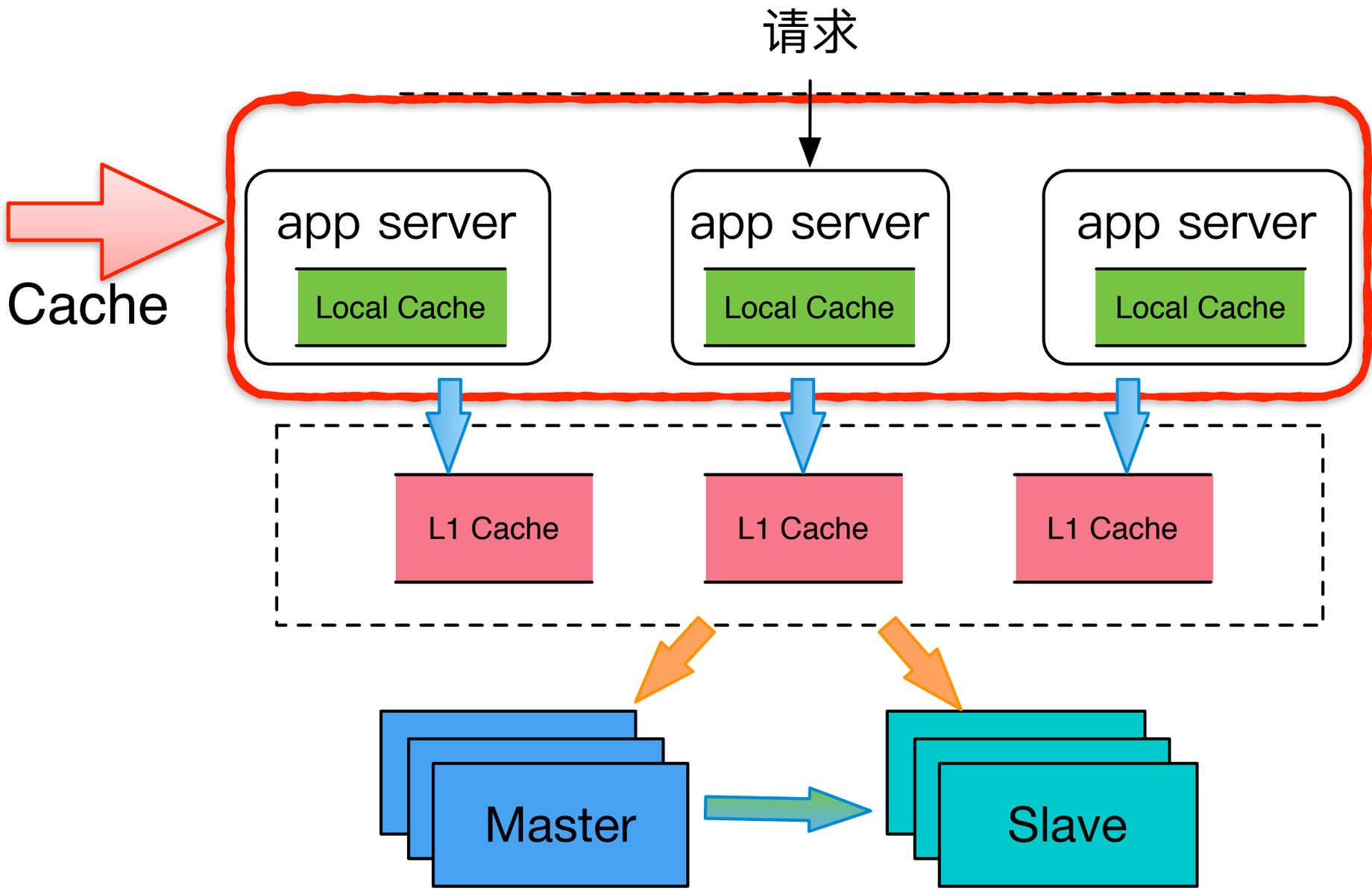
快速发展阶段-峰值应对

如何解决?

突发流量峰值应对

多级Cache防热点

- Master-Slave Cache
- L1 Cache
- **LocalCache**



快速发展阶段-性能优化

如何解决?

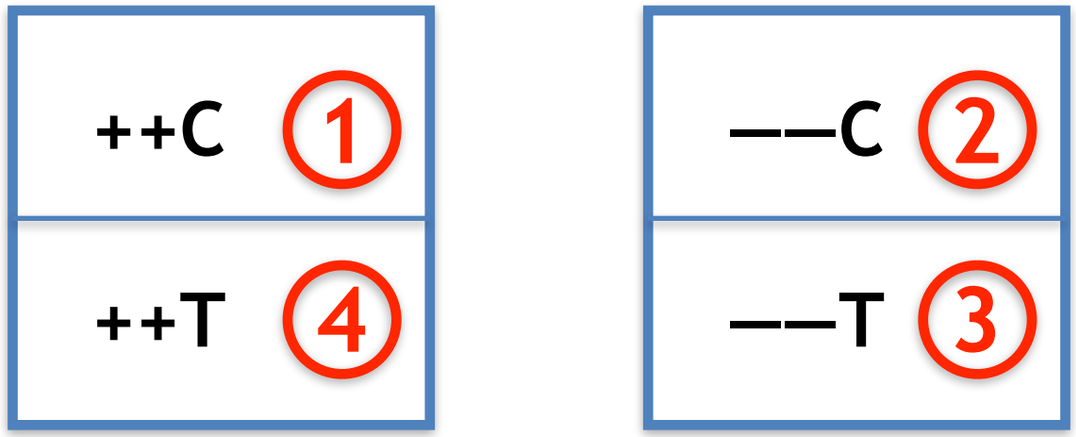
消息未读数高并发变更

问题描述

未读数变更需要保证原子性的同时满足高性能的需求

快速发展阶段-性能优化

并发更新导致不一致的场景



C为会话未读
T为总未读
起始 C=0 T=0

期望值： C==0 T==0
实际值： C==0 T==1

快速发展阶段-性能问题

难点：保证分布式事务性的同时满足高性能要求。

一些方案

分布式MC锁?

使用Redis提供的Transaction机制?

基于Redis的Lua实现?

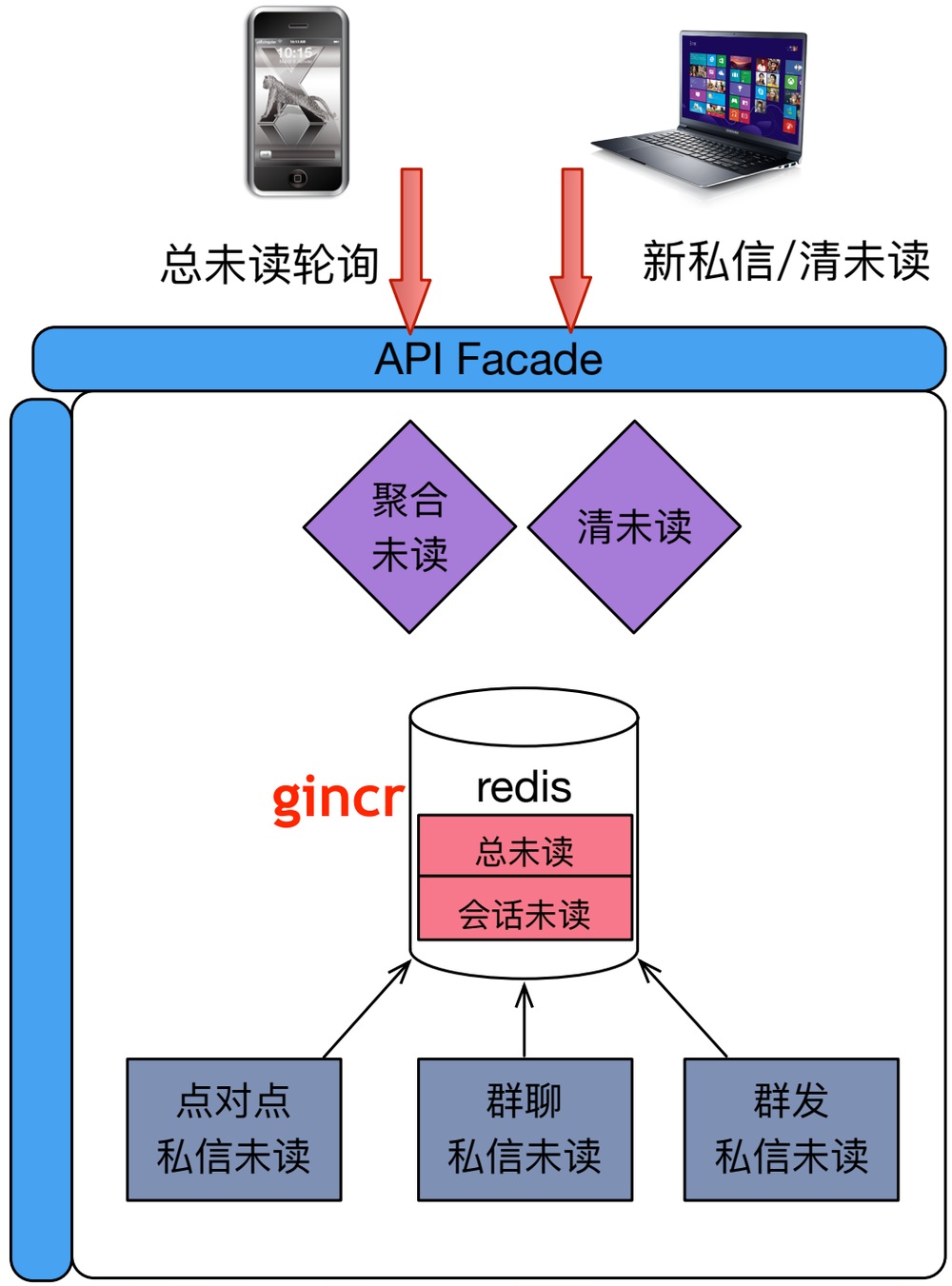
快速发展阶段-性能问题

最终采用方案

Redis源码修改

未读变更逻辑形成新命令—gincr

- 支持批处理。
- 性能强悍。**单端口50W key/s**
- 支持旧数据自动修复。



方案上线后整体一致性达7个9以上。

快速发展阶段

- **实践经验**

  ➢ 异步的方式提高接口吞吐量。

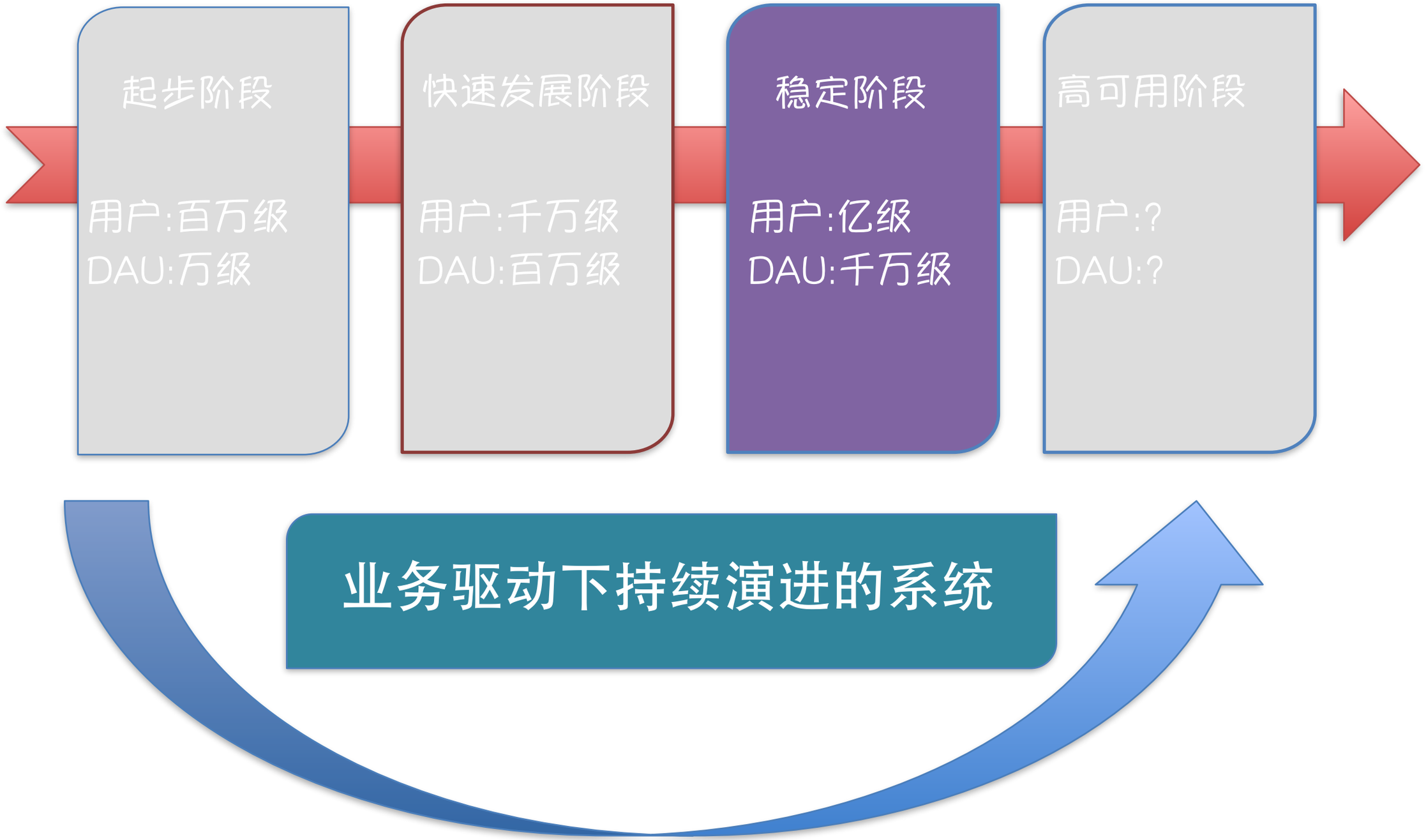
  ➢ 业务拆分、独立扩容提高热点业务的吞吐量。

  ➢ 分库分表解决数据容量问题和冷热分离。

  ➢ 多级Cache解决流量峰值和极热数据的问题。

  ➢ 针对通用组件进行业务定制化满足高性能需求。

消息系统演进


起步阶段
用户:百万级
DAU:万级
快速发展阶段
用户:千万级
DAU:百万级
稳定阶段
用户:亿级
DAU:千万级
高可用阶段
用户:?
DAU:?
业务驱动下持续演进的系统

稳定阶段-面临的问题

业务的变化

- 移动端用户占比越来越高
- 用户对消息到达实时性要求更高
- 用户对流量敏感

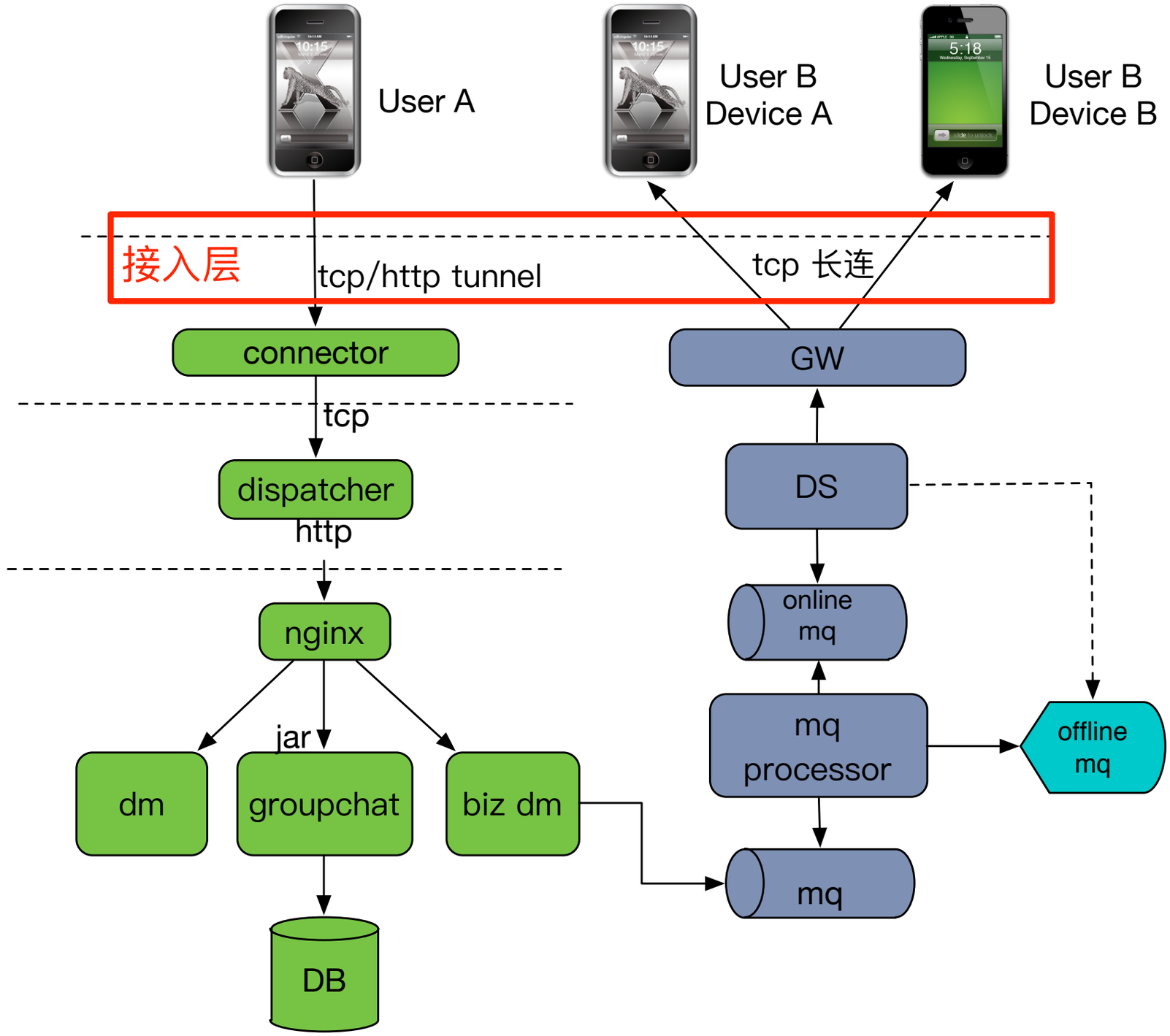
架构的挑战

- 面向移动端的整体架构

稳定阶段-移动端优化

主要改进点：

- 接入层改进
- 短连通道
- 消息推送优化

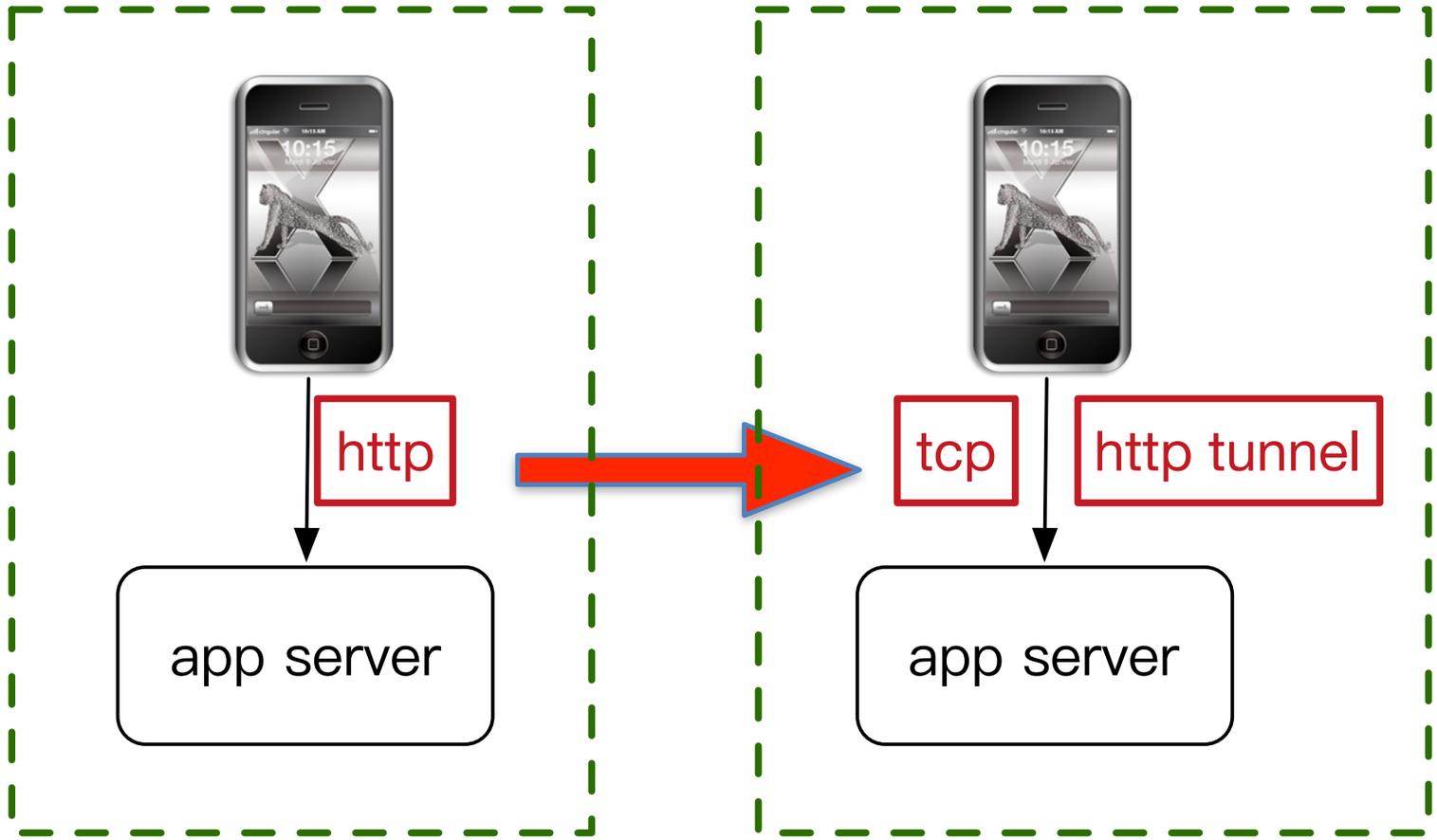


稳定阶段-移动端优化

接入层改进

- **wesync协议**
  - ——私有二进制
  - ——高可扩展
  - ——包头复用
  - ——包合并
  - ——gzip压缩

- **Http Tunnel**
  - ——wap网关限制
  - ——tcp通道的failover



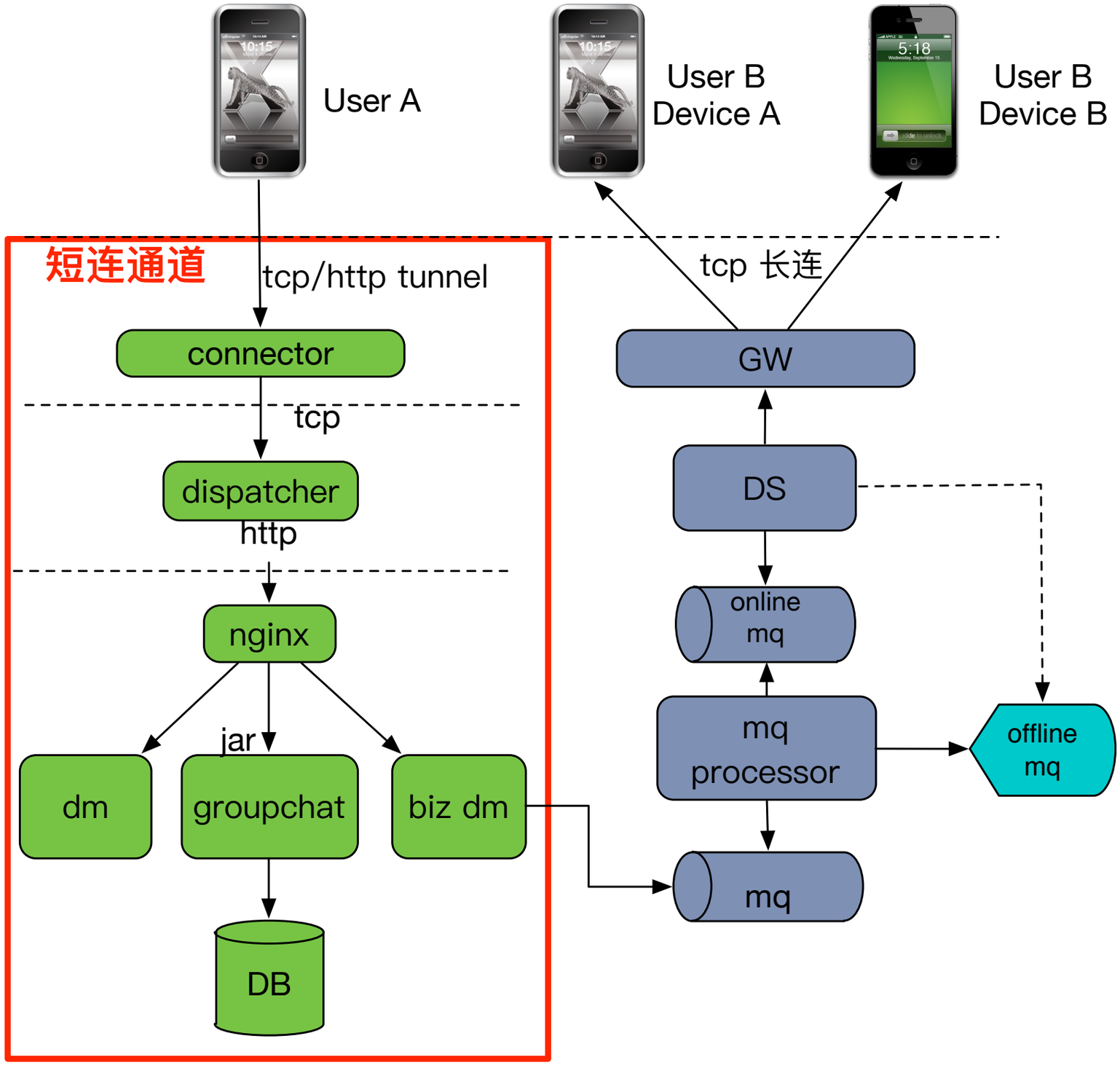
稳定阶段-移动端优化

主要改进点：

➢接入层改进

➢短连通道优化

➢消息推送优化



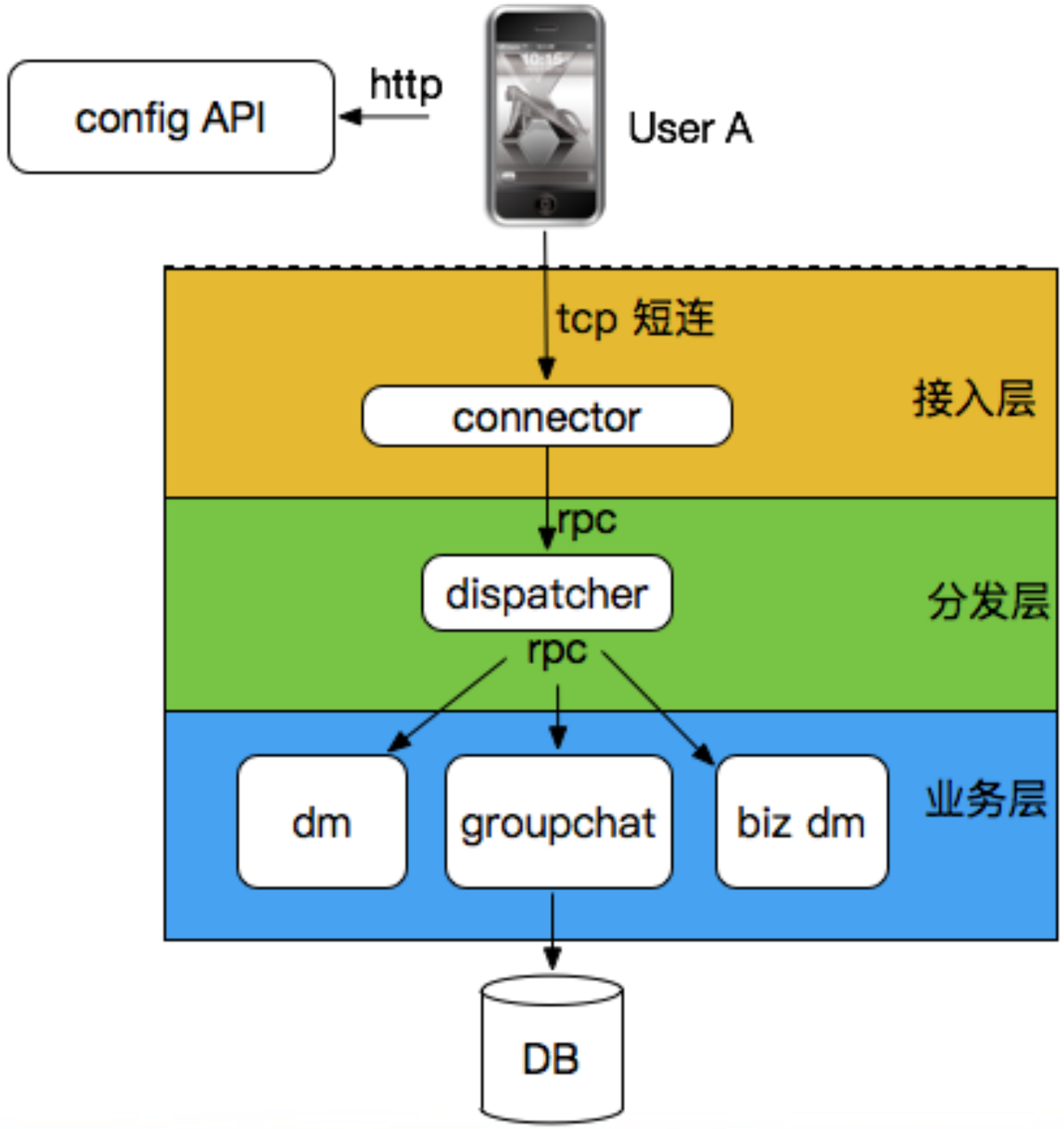
稳定阶段-移动端优化（短连服务）

短连通道架构优化

- ➢ 保证接入层稳定性
  ——业务无关

- ➢ 提高接入层可连通性
  ——入口下发通道
  ——客户端预埋IP

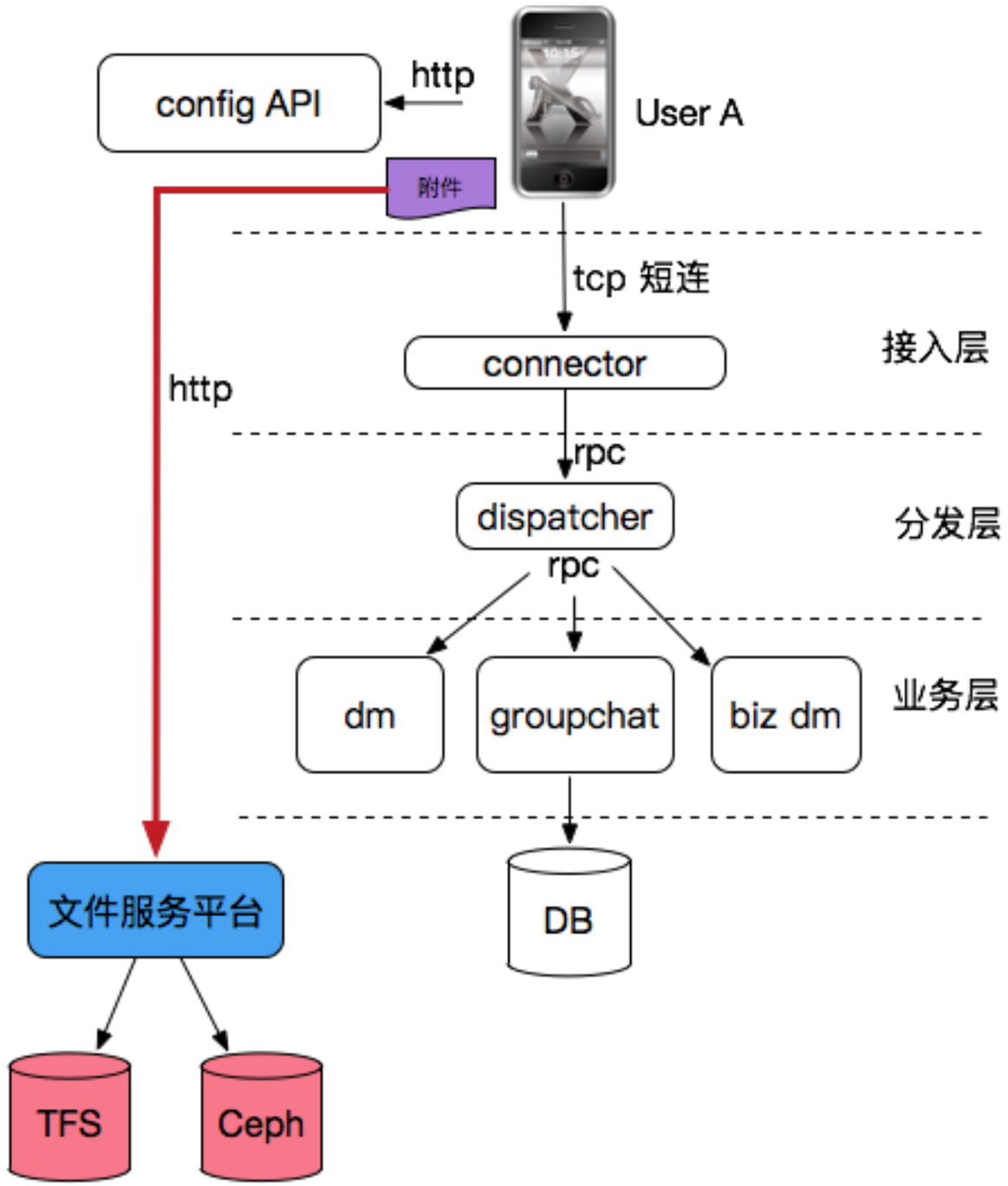
- ➢ 业务层服务化改造
  —— motan框架



稳定阶段-移动端优化（短连服务）

隔离附件业务

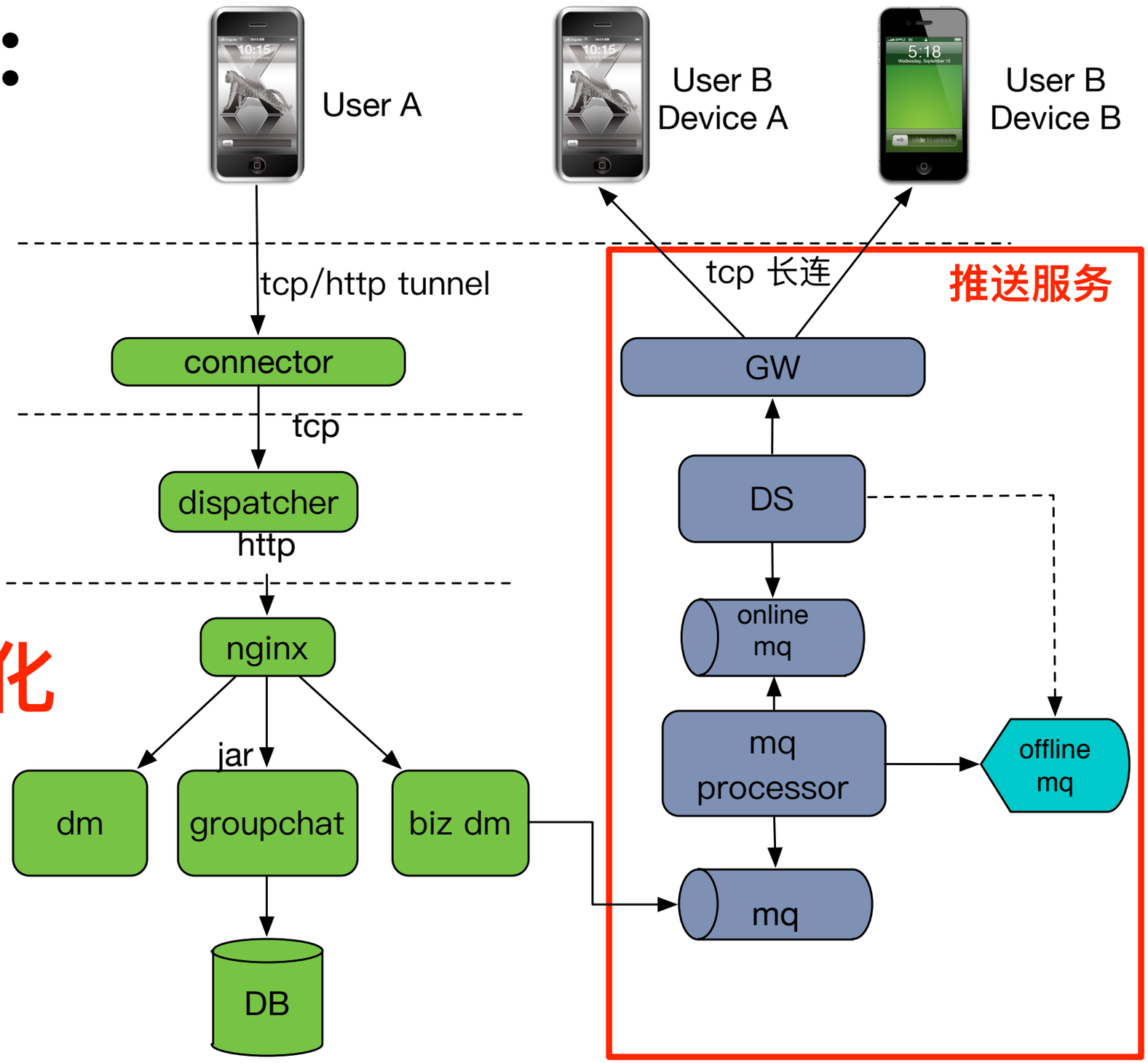
- 保护短连通道的核心业务。
- 独立存储服务屏蔽存储细节。



稳定阶段-移动端优化

主要改进点：

- 接入层改进
- 短连通道优化
- **消息推送优化**



稳定阶段-消息推送一致性

消息推送一致性问题：

- ➢ 推送过程丢消息客户端无感知。
- ➢ 消息乱序到达影响用户体验。

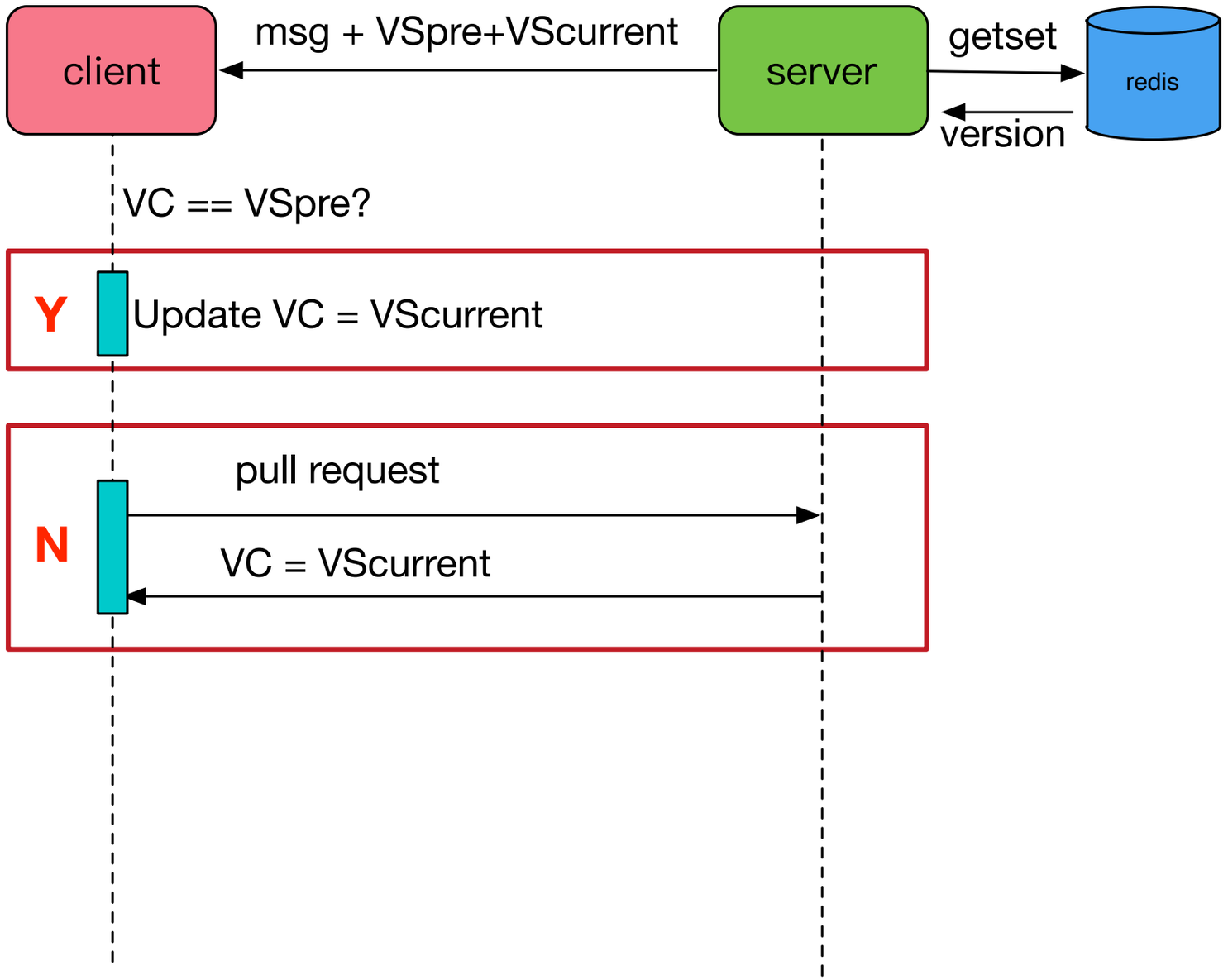
解决思路：

- ➢ 通过版本号机制解决多端不一致。

稳定阶段-消息推送一致性

基于版本号的推送一致性方案

- 服务端维护用户版本库。
- 客户端维护本地最新版本号。
- 每次变更产生新版本号。
- 推送时消息+版本号。
- 客户端版本比对。
- 支持服务端直接push。
- 支持离线队列。



稳定阶段

- **实践经验**

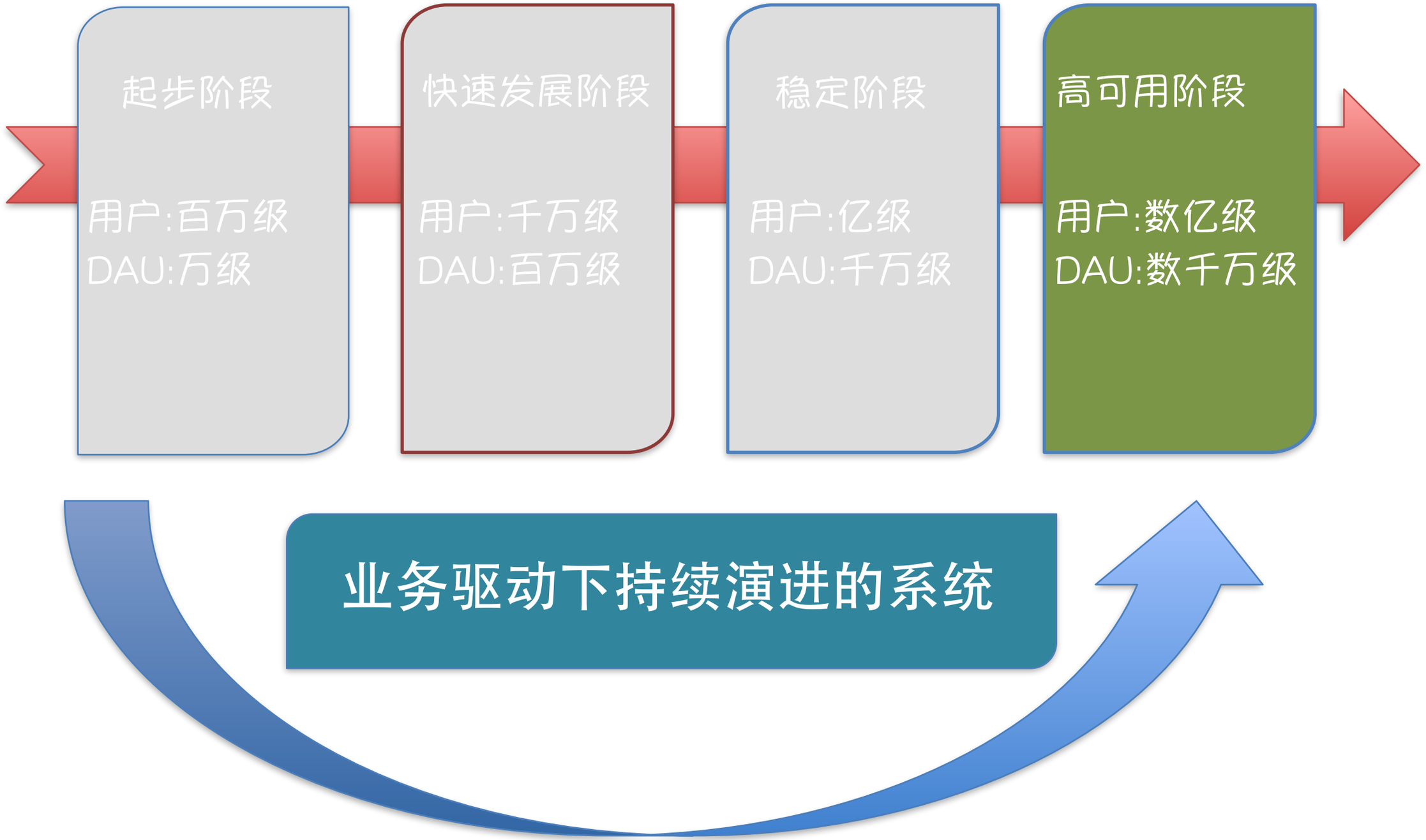
➢ 选择省流量、易扩展、兼容性好的协议。

➢ 保证接入层的稳定性和可用性。

➢ 隔离非核心业务，保护核心功能。

➢ 通过版本号机制保证多端一致性。

消息系统演进


起步阶段
用户:百万级
DAU:万级
快速发展阶段
用户:千万级
DAU:百万级
稳定阶段
用户:亿级
DAU:千万级
高可用阶段
用户:数亿级
DAU:数千万级
业务驱动下持续演进的系统

高可用阶段-面临的问题

业务的变化

➢ 用户成熟度提升的同时对系统可用性要求更高。

架构的挑战

➢ 提高系统隔离失败的能力。

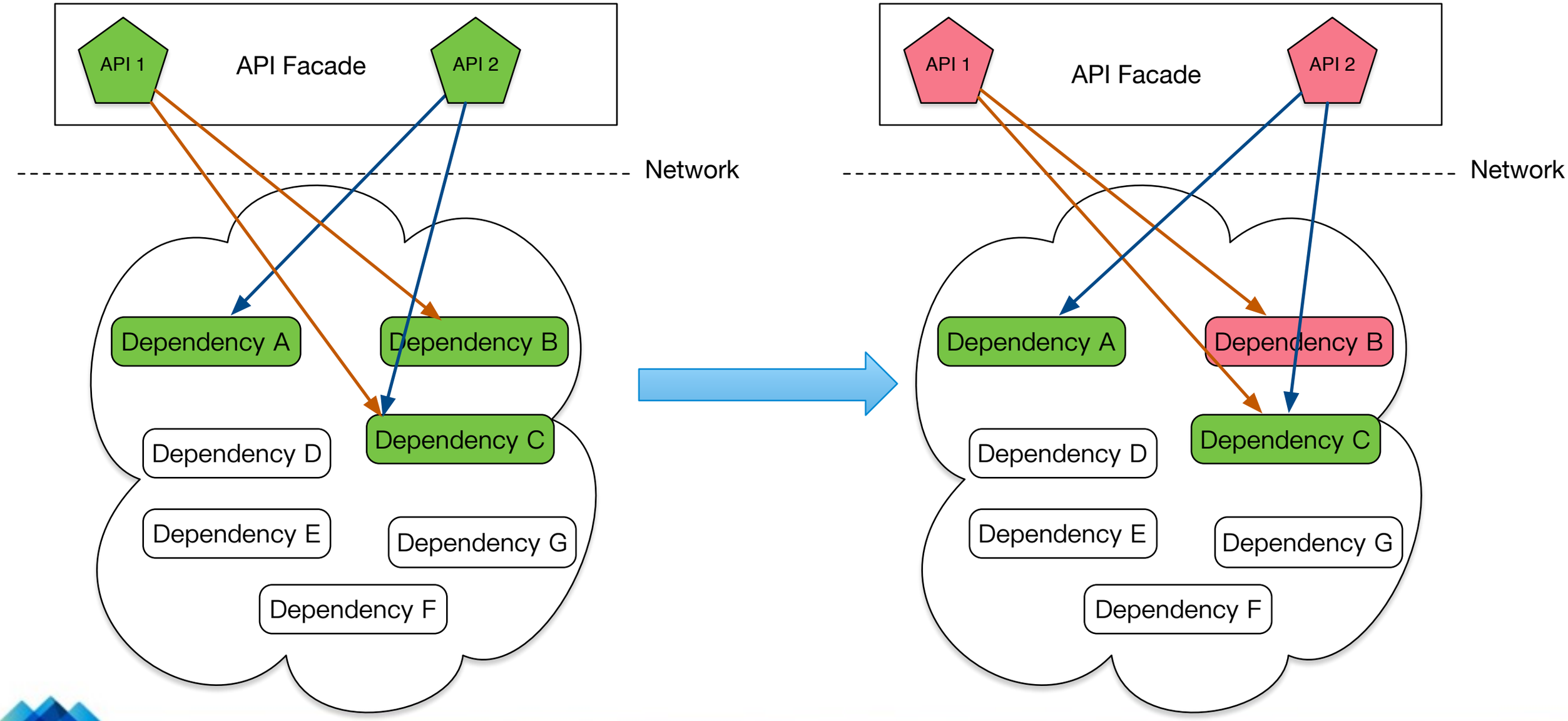
➢ 保护系统和资源。

➢ 多方位解决可用性问题。

高可用阶段-服务可用性

提高系统隔离失败的能力

框架级、细粒度的自动降级机制（design for failure)



高可用阶段-服务可用性

如何自动降级

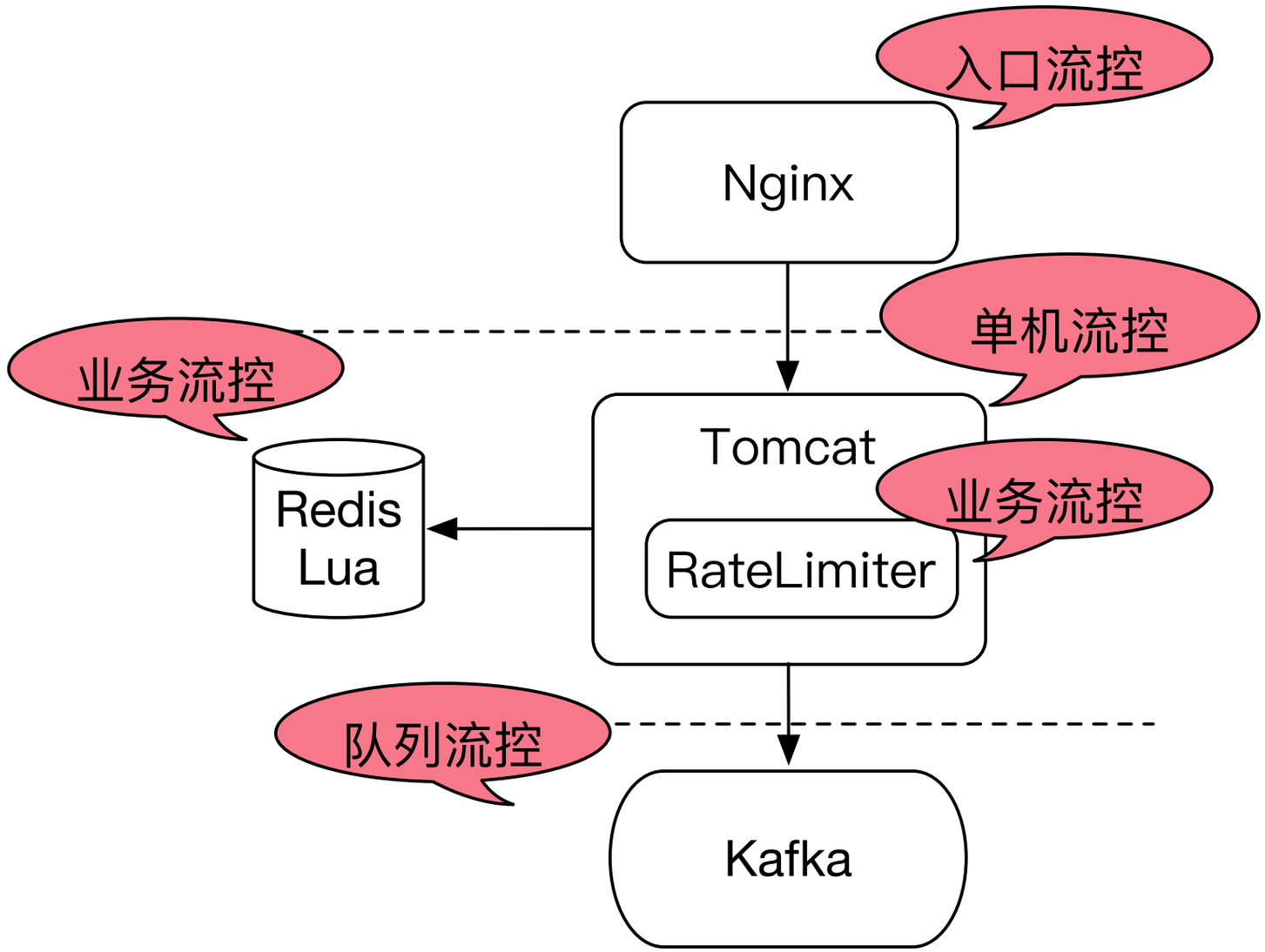
- ➢ 超时或异常时自动熔断&自动恢复机制。
- ➢ 熔断后自动降级处理。（eg.默认值or其它可替换依赖）。
- ➢ 通过故障模拟确认熔断阈值。
- ➢ 熔断的监控和告警。

高可用阶段-服务可用性

保护系统和资源

灵活的流控机制

- 入口流控
- 单机流控
- 队列流控
- 业务层流控



高可用阶段-运维层面

监控

资源监控

- ➢ 系统资源。
- ➢ 业务资源。

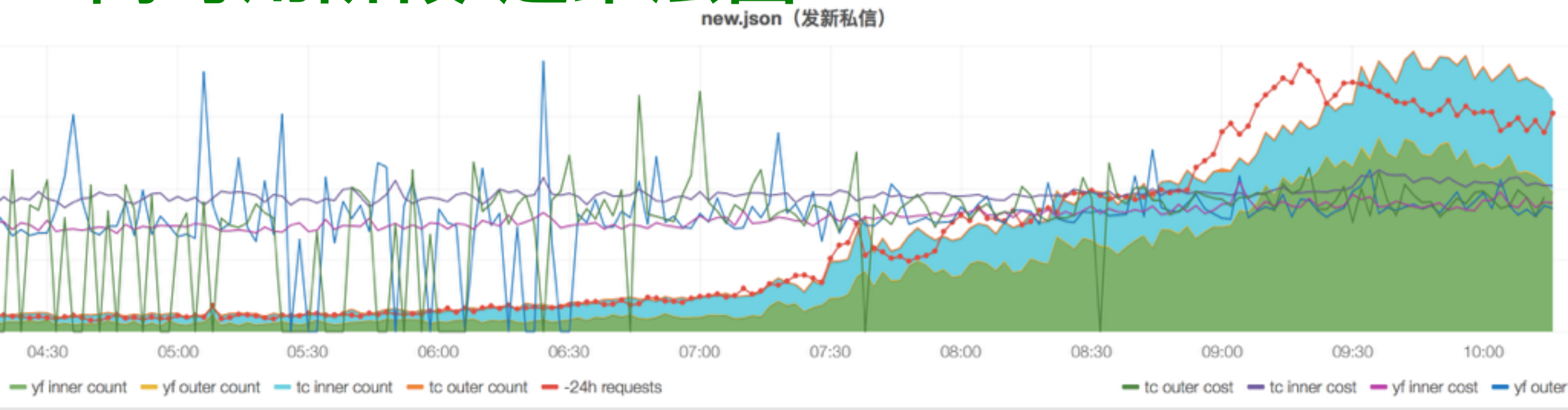
业务监控

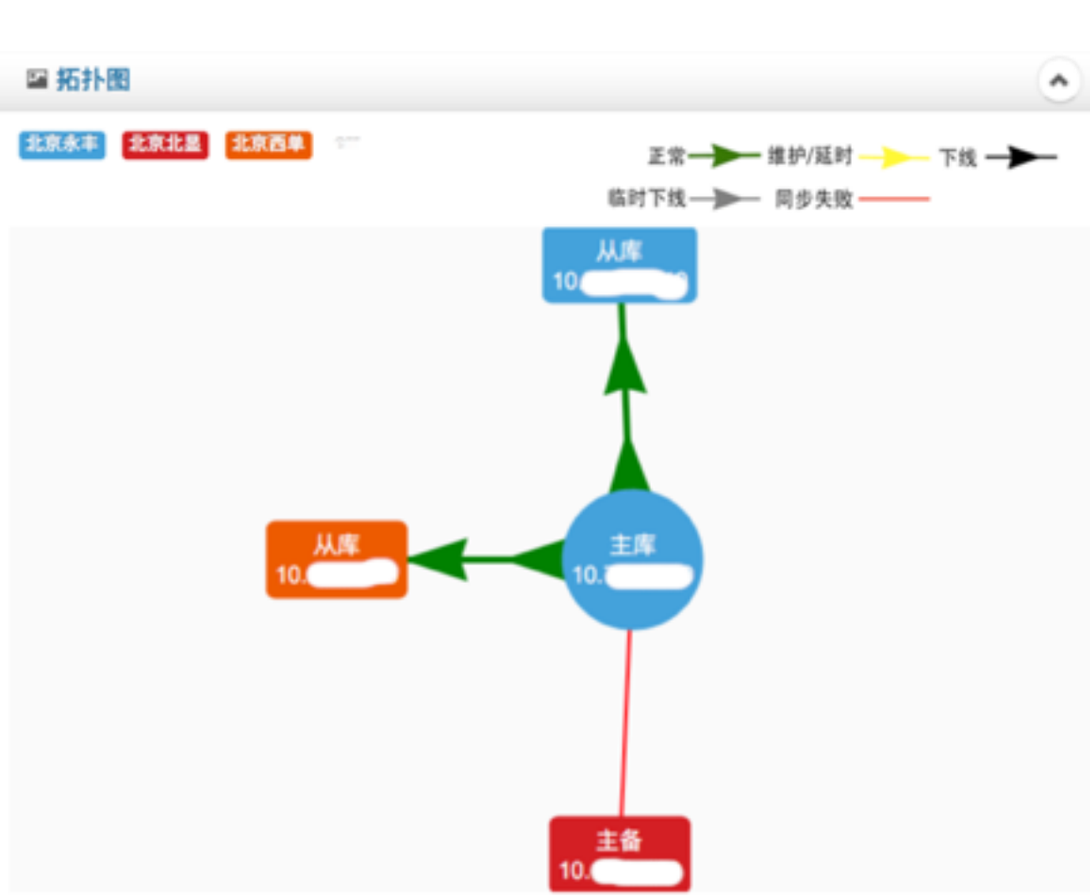
- ➢ 访问量、成功率、响应时间、异常情况等。
- ➢ 客户端埋点采集日志监控。

告警

- ➢ 基于日志的被动分析式告警。
- ➢ 基于接口的主动探测式告警。
- ➢ 多种告警方式。

高可用阶段-运维层面


new.json（发新私信）
04:30
05:00
05:30
06:00
06:30
07:00
07:30
08:00
08:30
09:00
09:30
10:00
yf inner count
yf outer count
tc inner count
tc outer count
-24h requests
tc outer cost
tc inner cost
yf inner cost
yf outer

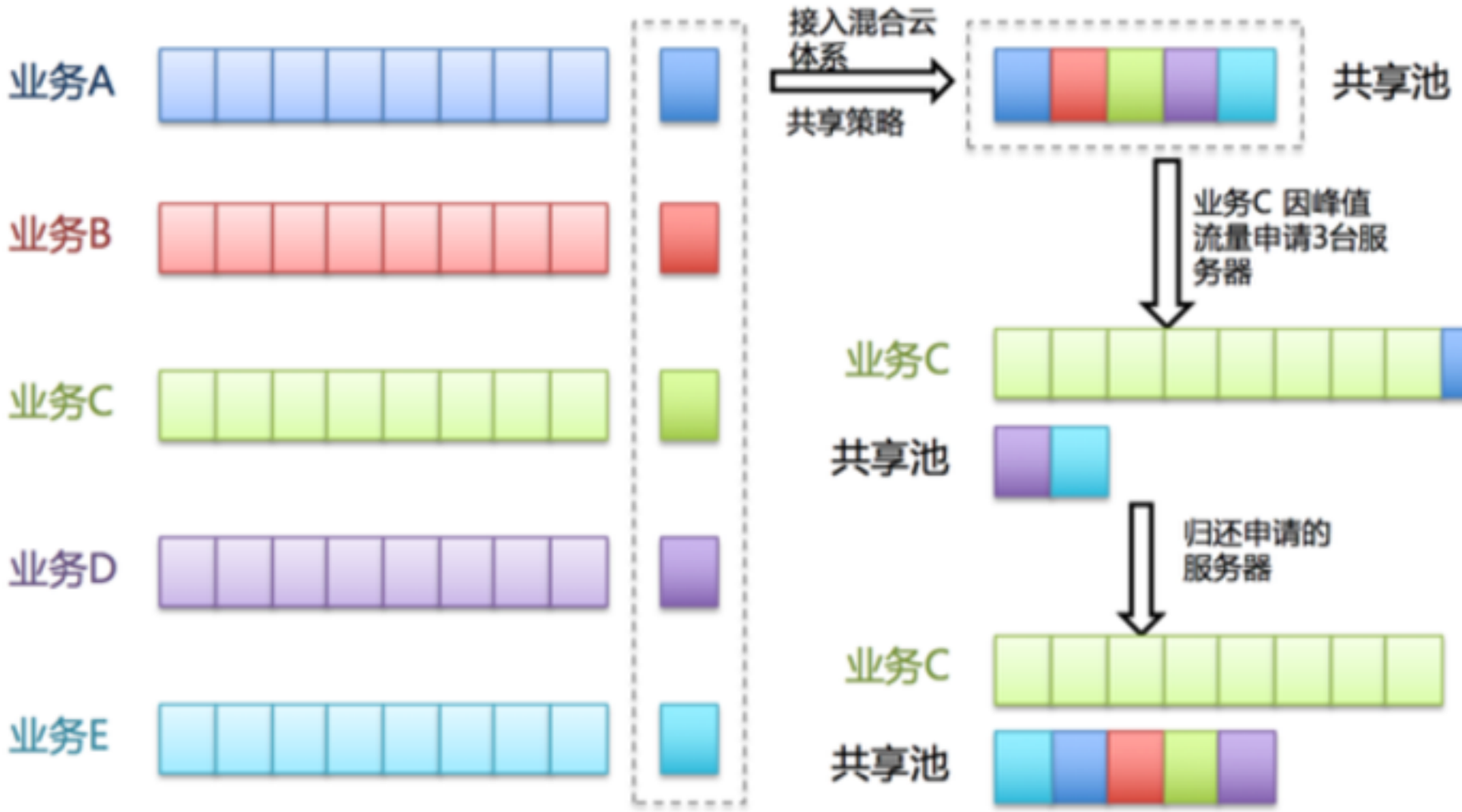

拓扑图
北京永丰
北京北显
北京西单
正常
维护/延时
下线
临时下线
同步失败
从库
10
从库
10.
主库
10.
主备
10.


CPU Idle
97 %
Load
0.83
已使用内存大小
9 GB
磁盘可用空间
220 GB
磁盘使用率
1.35
流量 - 进
7 mb/s
流量 - 出
4 mb/s
监控数据
实时
24小时
7天
30天
自定义监控图
100%
50%
0%
06:00
3. Jun
10:00
CPU - IDLE
286 GB
191 GB
95 GB
磁盘 - 可用空间大小
20 mb/s
10 mb/s
流量 - 进
流量 - 出
300
200
100
磁盘使用率
7.5
5
2.5
Load - 1分钟内的平均负载
19 GB
10 GB
总内存
已使用内存大小
监控报警
创建报警配置
查看报警配置

高可用阶段-运维层面

基于Docker的混合云弹性调度系统-DCP

- 跨云共享资源池
- 快速扩容能力1000/10min



高可用阶段

- **实践经验**

  ➢ 通过自动降级机制隔离失败。

  ➢ 通过多种限流机制保护系统和资源。

  ➢ 通过监控和告警提高系统可运维性。

  ➢ 通过弹性调度系统实现混合云上的快速扩容。

Thanks!