已有平台引入直播的实战之路

张嵩　20160813

目录

一. 背景

二. 引入直播的问题

三. 开发过程中的坑与优化的经验

背景

- 开发周期给得较为短暂
- 没有任何直播流及IM的技术积累
- 现有系统全服务化结构
- 物理机房无机柜可扩容
- 设计目标为至少能容纳当前日活用户同时观看

Q00

没有直播流及IM的技术积累

直播流采用七牛，IM采用GO语言编写

优

1. 直播流节约大量时间，可以把时间花在业务开发上
2. 基于GO语言本身特性，不用临时招聘人员，边学边写
3. IM自研较容易处理在线排行榜等需要长连接处理的业务

劣

1. 服务的稳定全压在第三方身上，自身几乎无法调优
2. IM被压缩到一周内完成，实际上是三天
3. 没有时间去尝试，也没有优化的时间

Q01

物理机房已没有可用的机柜以及机器

云服务

优

1. 已有非核心业务在云平台上，多机房间有ipsec隧道
2. 能够满足快速上线的要求【申请及采购流程耗时久】
3. 碰到高峰期扩容方便

劣

1. 部分未提供的服务需要自己在云平台的虚拟主机上搭建
2. 引入新的服务扩机房调用问题

Q02

跨机房调用使得服务SLA无法保证

预同步数据、强制同步数据、异步化调用

分布式事务？最终一致性？

1. 无分布式时，业务逻辑通过事务实现一致性
2. 事务型的消息队列来保证
3. 基于弱消息队列+定时补偿
4. 异步的double-check机制

优

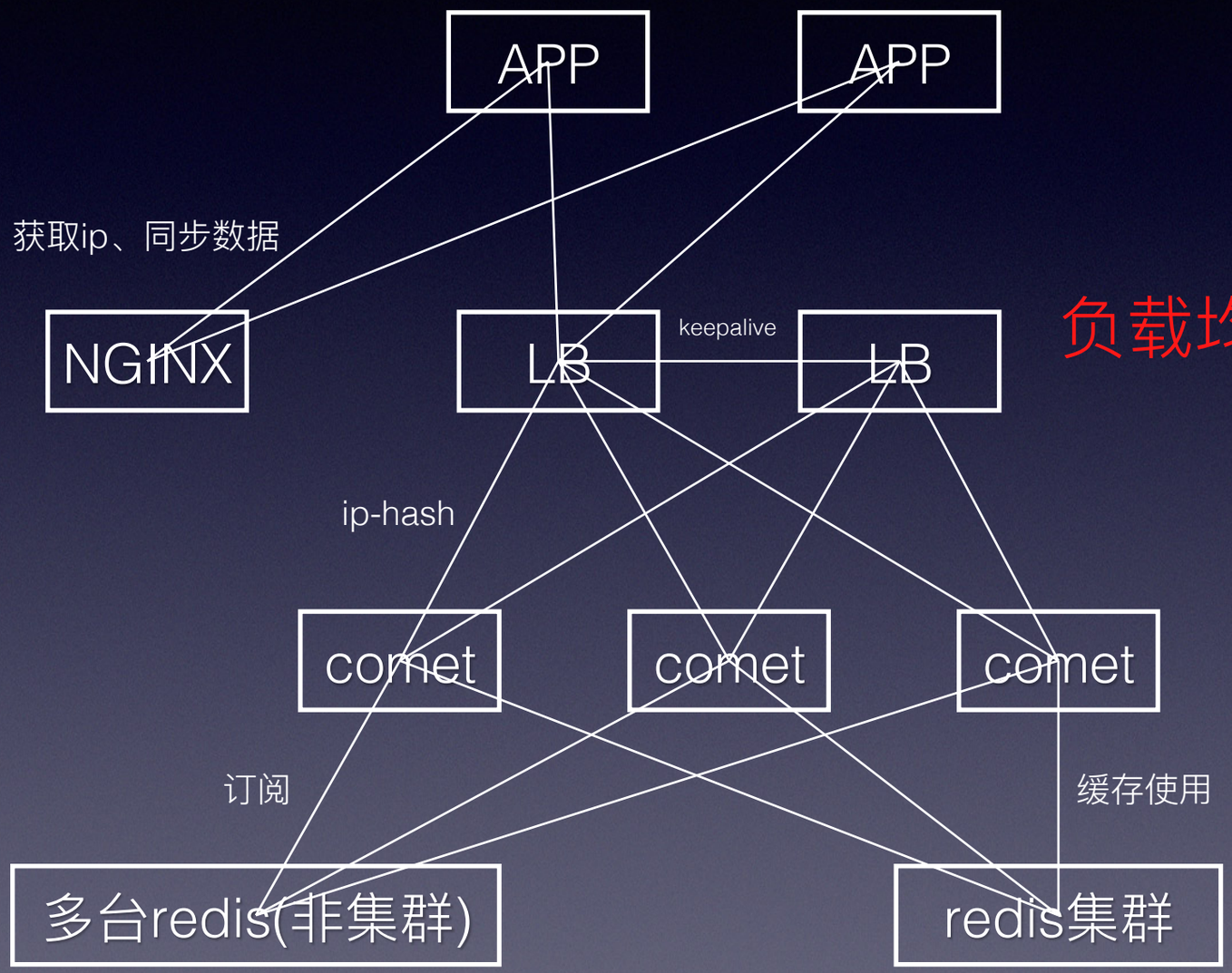
1. 最低成本实现，不用修改机房调用结构
2. SLA基本可控
3. 当前已有服务可实现复用

劣

1. 没有专线来得简单
2. 需要思考哪些业务需要特殊处理，编写较为复杂

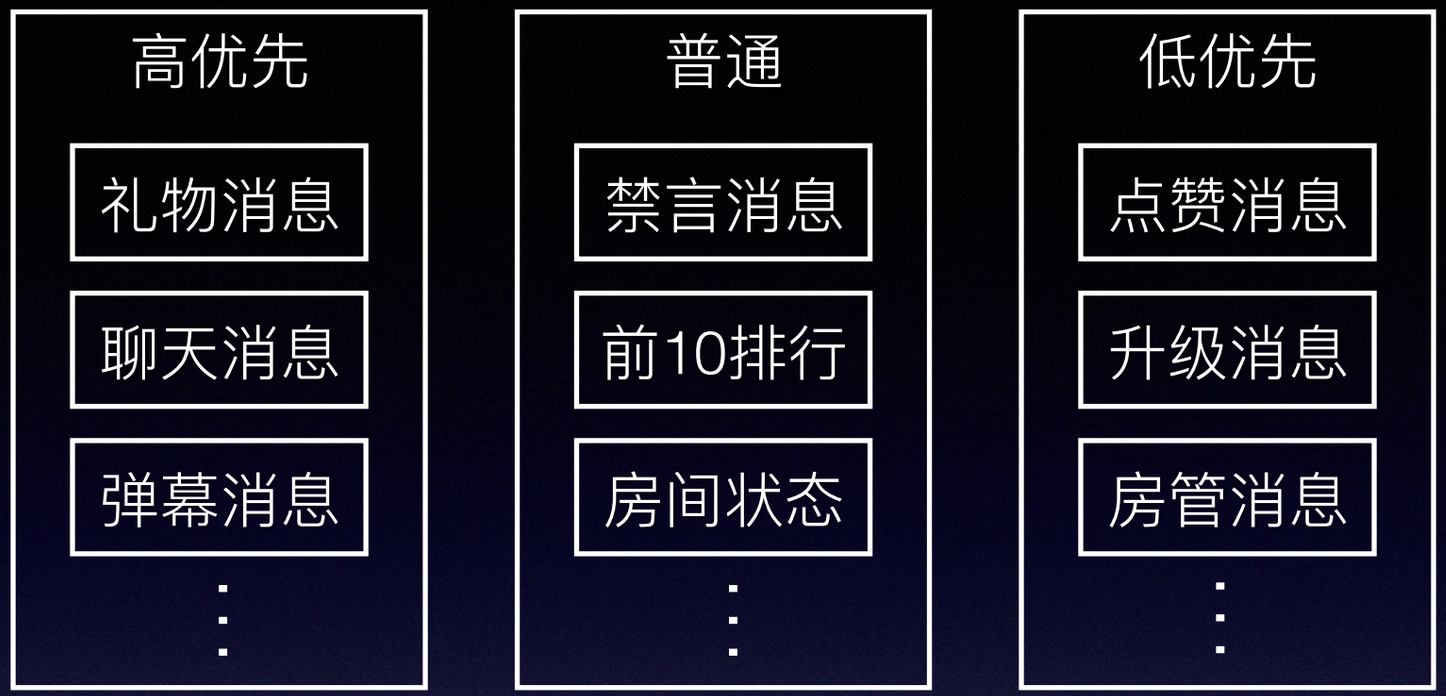
IM结构

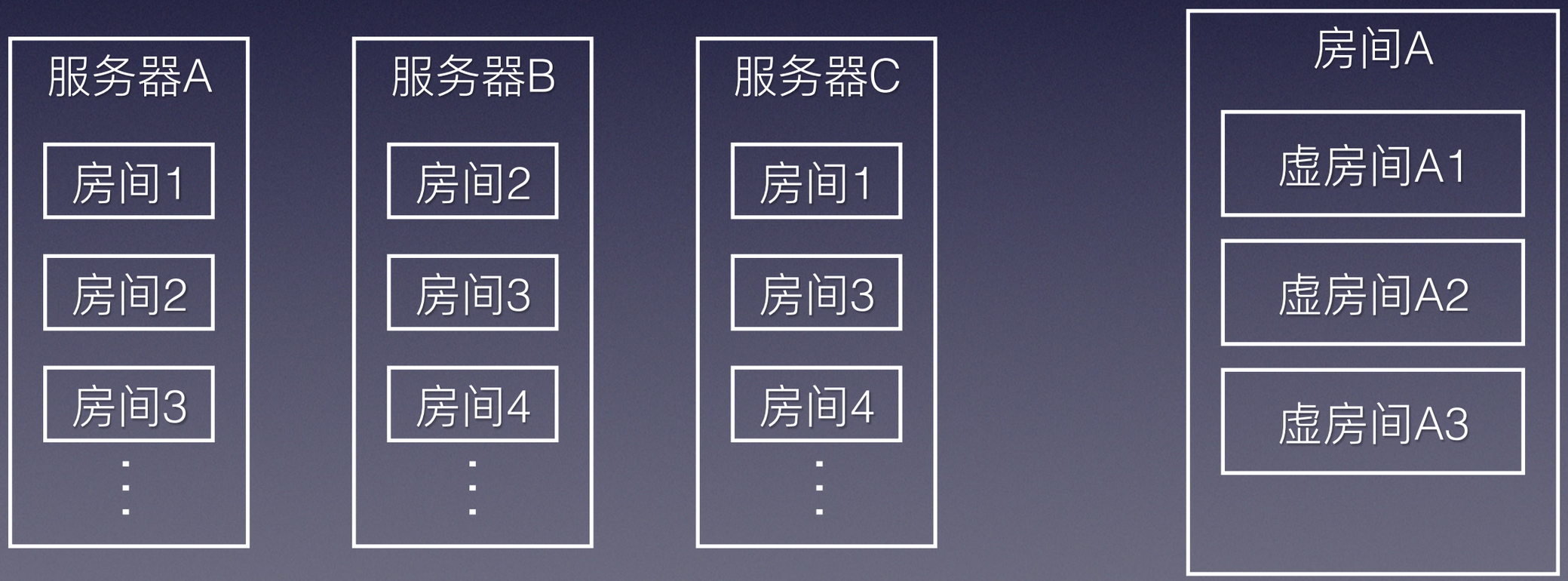
- 用户根据ip-hash分配到各个comet服务器
- comet服务器不保存任何状态
- 房间与服务器是多对多的关系
- 拆分不同类型的消息以保证优先级及扩展性
- 消息量过大时使用同房拆分策略



负载均衡节点包量的极限

redis的极限


高优先
礼物消息
聊天消息
弹幕消息
普通
禁言消息
前10排行
房间状态
低优先
点赞消息
升级消息
房管消息


服务器A
房间1
房间2
房间3
服务器B
房间2
房间3
房间4
服务器C
房间1
房间3
房间4
房间A
虚房间A1
虚房间A2
虚房间A3

IM相关优化

- 弱网络下的优化：
  包不分片、SACK、SOCKET-BUFFER、断线重连发现等
- 控制消息大小，正常MSS-头-websocket分帧消耗
- 采用较长心跳时要考虑到运营商的NAT回收
- 采用较短心跳时要注意到场景的适用性

ip.addr eq 106.7[redacted] and ip.addr eq 192.168.2.9

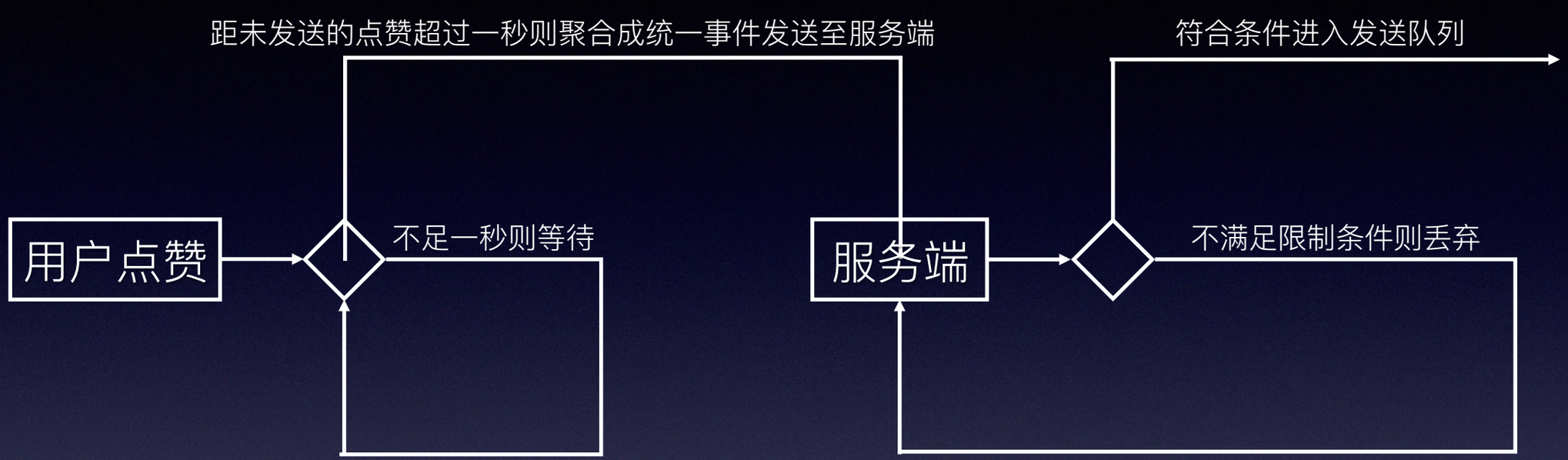
No.	Time	Source	Destination	Protocol	Length	Info
1372	11.185665	192.168[redacted]	106.75[redacted]	TCP	78	50569 → 1314 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1565213944 TSecr=0 SACK_PERM=1
1373	11.231228	106.75.[redacted]46	192.16[redacted].9	TCP	74	1314 → 50569 [SYN, ACK] Seq=0 Ack=1 Win=14020 Len=0 MSS=1412 SACK_PERM=1 TSval=2742972314 TSecr=1565213944 WS=128
1374	11.234487	192.168[redacted]	106.75.[redacted]246	TCP	66	50569 → 1314 [ACK] Seq=1 Ack=1 Win=131584 Len=0 TSval=1565213991 TSecr=2742972314
1375	11.243478	192.168[redacted]	106.75.[redacted]246	HTTP	445	GET /socket.io/?transport=websocket&ticket_id=9c394a175508edfcd3d1db3b7bc8008d HTTP/1.1
1378	11.287888	106.75.[redacted]246	192.168[redacted]9	TCP	66	1314 → 50569 [ACK] Seq=1 Ack=380 Win=15104 Len=0 TSval=2742972371 TSecr=1565214000
1379	11.288716	106.75.[redacted]246	192.168[redacted]9	HTTP	195	HTTP/1.1 101 Switching Protocols
1380	11.288720	106.75.[redacted]246	192.168[redacted]9	WebSocket	164	WebSocket Text [FIN]
1381	11.288777	106.75.[redacted]246	192.168[redacted]9	WebSocket	70	WebSocket Text [FIN]
1382	11.294555	192.168[redacted]9	106.75.[redacted]246	TCP	66	50569 → 1314 [ACK] Seq=380 Ack=130 Win=131456 Len=0 TSval=1565214048 TSecr=2742972372
1383	11.294563	192.168[redacted]9	106.75.[redacted]246	TCP	66	50569 → 1314 [ACK] Seq=380 Ack=228 Win=131360 Len=0 TSval=1565214048 TSecr=2742972372
1384	11.294565	192.168[redacted]9	106.75.[redacted]246	TCP	66	50569 → 1314 [ACK] Seq=380 Ack=232 Win=131360 Len=0 TSval=1565214048 TSecr=2742972372
1385	11.303159	192.168[redacted]9	106.75.[redacted]246	WebSocket	73	WebSocket Text [FIN] [MASKED]
1386	11.347543	106.75.[redacted]246	192.168[redacted]9	WebSocket	69	WebSocket Text [FIN]
1387	11.378196	192.168[redacted]9	106.75.[redacted]246	TCP	66	50569 → 1314 [ACK] Seq=387 Ack=235 Win=131360 Len=0 TSval=1565214130 TSecr=2742972431
1390	11.576001	192.168[redacted]9	106.75.[redacted]246	TCP	54	[TCP Keep-Alive] 50569 → 1314 [ACK] Seq=386 Ack=235 Win=131360 Len=0
1391	11.619905	106.75.[redacted]246	192.168[redacted]9	TCP	66	[TCP Keep-Alive ACK] 1314 → 50569 [ACK] Seq=235 Ack=387 Win=15104 Len=0 TSval=2742972703 TSecr=1565214130
1471	36.347675	106.75.[redacted]246	192.168[redacted]9	WebSocket	69	WebSocket Text [FIN]
1472	36.384300	192.168[redacted]9	106.75.[redacted]246	TCP	66	50569 → 1314 [ACK] Seq=387 Ack=238 Win=131360 Len=0 TSval=1565239085 TSecr=2742997429
1474	36.585597	192.168[redacted]9	106.75.[redacted]246	TCP	54	[TCP Keep-Alive] 50569 → 1314 [ACK] Seq=386 Ack=238 Win=131360 Len=0
1477	36.629092	106.75.[redacted]246	192.168[redacted]9	TCP	66	[TCP Keep-Alive ACK] 1314 → 50569 [ACK] Seq=238 Ack=387 Win=15104 Len=0 TSval=2742997710 TSecr=1565239085
1479	36.763569	192.168[redacted]9	106.75.[redacted]246	WebSocket	73	WebSocket Text [FIN] [MASKED]
1481	36.808200	106.75.[redacted]246	192.168[redacted]9	WebSocket	69	WebSocket Text [FIN]
1482	36.810848	192.168[redacted]9	106.75.[redacted]246	TCP	66	50569 → 1314 [ACK] Seq=394 Ack=241 Win=131360 Len=0 TSval=1565239508 TSecr=2742997890
1483	37.066758	192.168[redacted]9	106.75.[redacted]246	TCP	54	[TCP Keep-Alive] 50569 → 1314 [ACK] Seq=393 Ack=241 Win=131360 Len=0
1484	37.110840	106.75.[redacted]246	192.168[redacted]9	TCP	66	[TCP Keep-Alive ACK] 1314 → 50569 [ACK] Seq=241 Ack=394 Win=15104 Len=0 TSval=2742998192 TSecr=1565239508
1525	61.804454	192.168[redacted]9	106.75.[redacted]246	WebSocket	73	WebSocket Text [FIN] [MASKED]
1526	61.808187	106.75.[redacted]246	192.168[redacted]9	WebSocket	69	WebSocket Text [FIN]
1527	61.837187	192.168[redacted]9	106.75.[redacted]246	TCP	66	50569 → 1314 [ACK] Seq=401 Ack=244 Win=131328 Len=0 TSval=1565264515 TSecr=2743022887
1528	61.849416	106.75.[redacted]246	192.168[redacted]9	WebSocket	69	WebSocket Text [FIN]
1529	61.851444	192.168[redacted]9	106.75.[redacted]246	TCP	66	50569 → 1314 [ACK] Seq=401 Ack=247 Win=131328 Len=0 TSval=1565264528 TSecr=2743022929

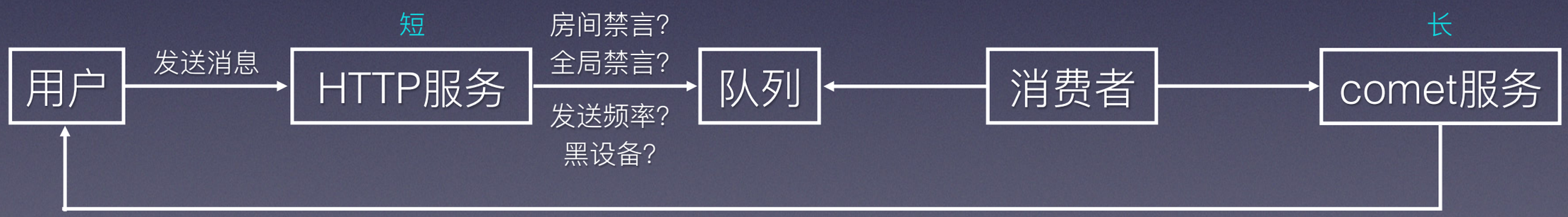
```
    [TCP Flags: *******A**S*]
  Window size value: 14020
  [Calculated window size: 14020]
▶ Checksum: 0x50c8 [validation disabled]
  Urgent pointer: 0
▼ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
  ▼ Maximum segment size: 1412 bytes
      Kind: Maximum Segment Size (2)
      Length: 4
      MSS Value: 1412
  ▶ TCP SACK Permitted Option: True
  ▼ Timestamps: TSval 2742972314, TSecr 1565213944
      Kind: Time Stamp Option (8)
      Length: 10
      Timestamp value: 2742972314
      Timestamp echo reply: 1565213944
  ▶ No-Operation (NOP)
  ▶ Window scale: 7 (multiply by 128)
```

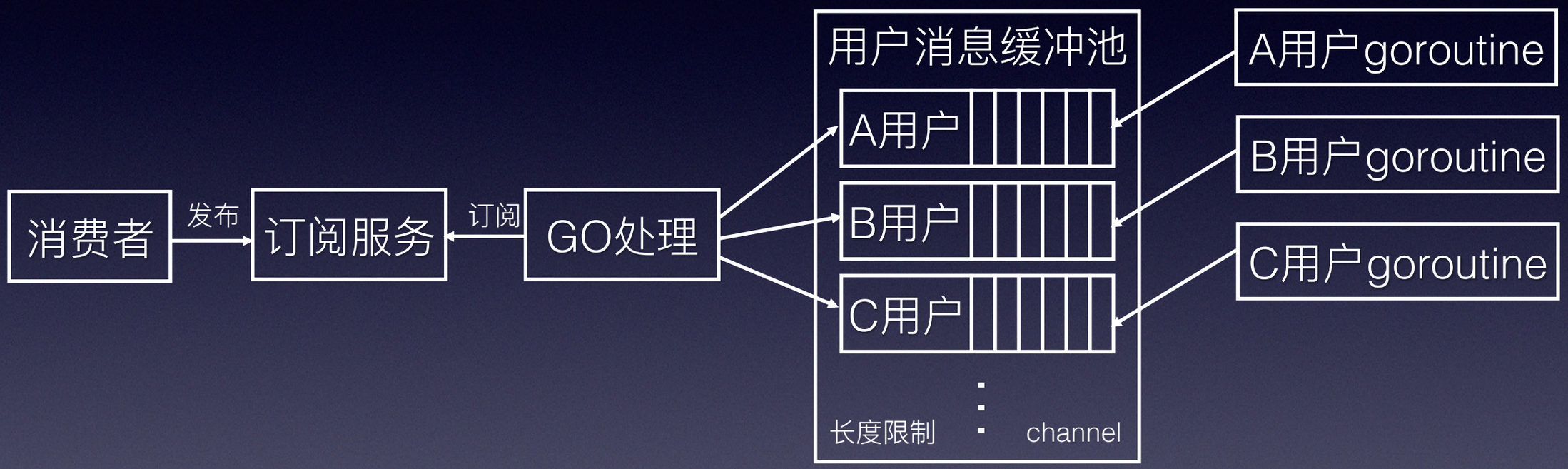
```
0000  b4 8b 19 67 7f 5e 12 dd  b1 2a 85 64 08 00 45 00   ...g.^.. .*.d..E.
0010  00 3c 00 00 00 00 35 06  39 ca 6a 4b 1e f6 c0 a8   .<....5. 9.jK....
0020  02 09 05 22 c5 89 db 7e  4c c5 60 9e 6b b6 a0 12   ..."...~ L.`.k...
0030  36 c4 50 c8 00 00 02 04  05 84 04 02 08 0a a3 7e   6.P..... .......~
0040  6f 9a 5d 4b 44 f8 01 03  03 07                     o.]KD... ..
```

业务消息扩展及优化

- 合并点赞消息等
- 消息异步化处理，多条线程同时处理
- 发送消息的验证、限流需求
- 单用户不卡住广播消息的发布也不影响其他用户
- 每个用户的缓冲队列应该有上限【心跳延迟问题】

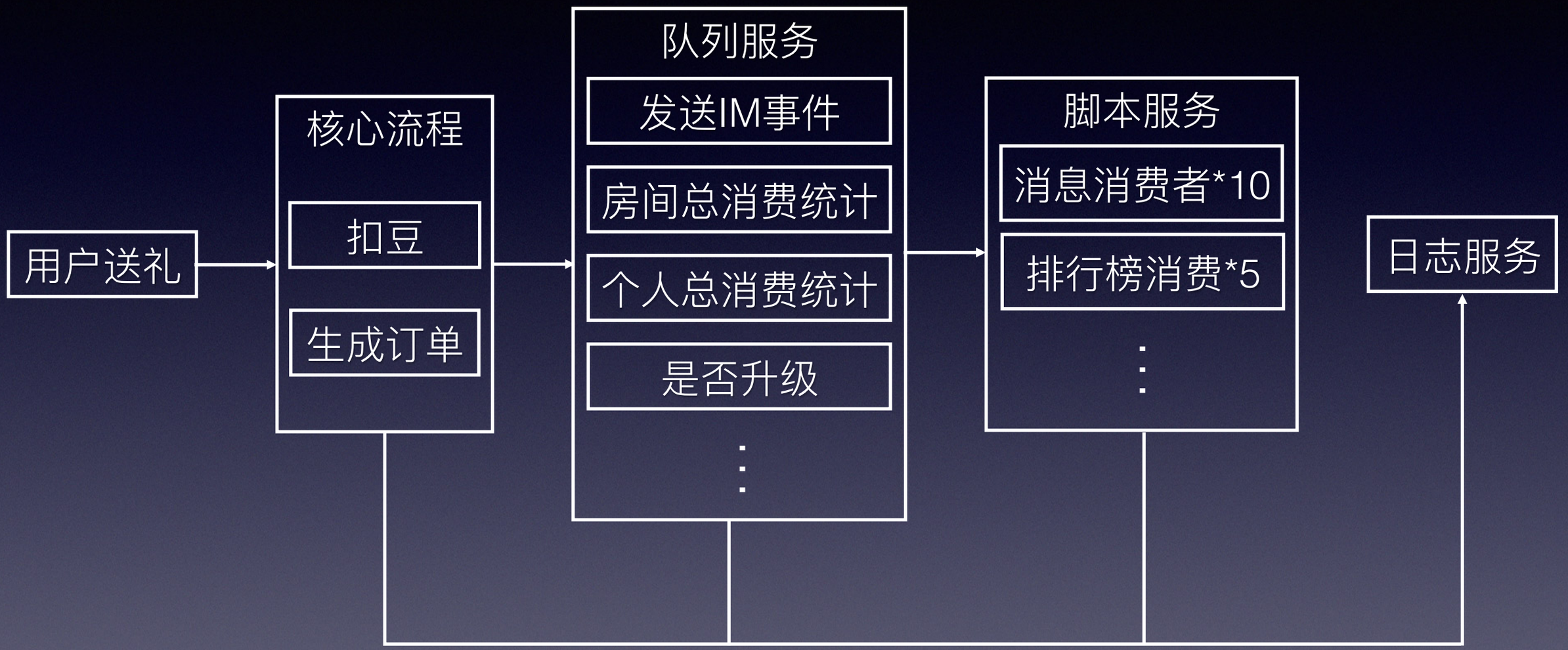

距未发送的点赞超过一秒则聚合成统一事件发送至服务端
符合条件进入发送队列
用户点赞
不足一秒则等待
服务端
不满足限制条件则丢弃

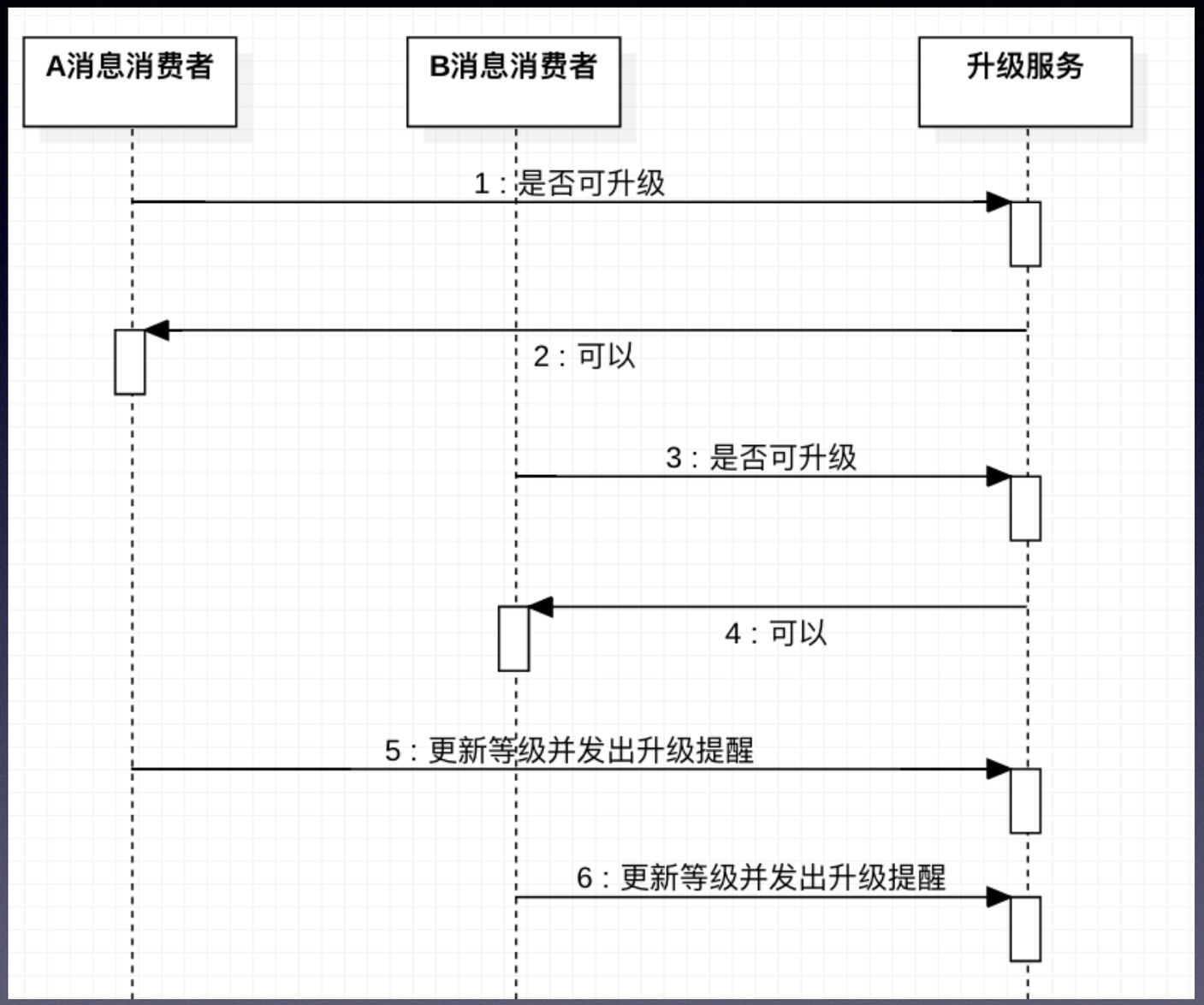

短
房间禁言?
全局禁言?
长
用户
发送消息
HTTP服务
队列
消费者
comet服务
发送频率?
黑设备?

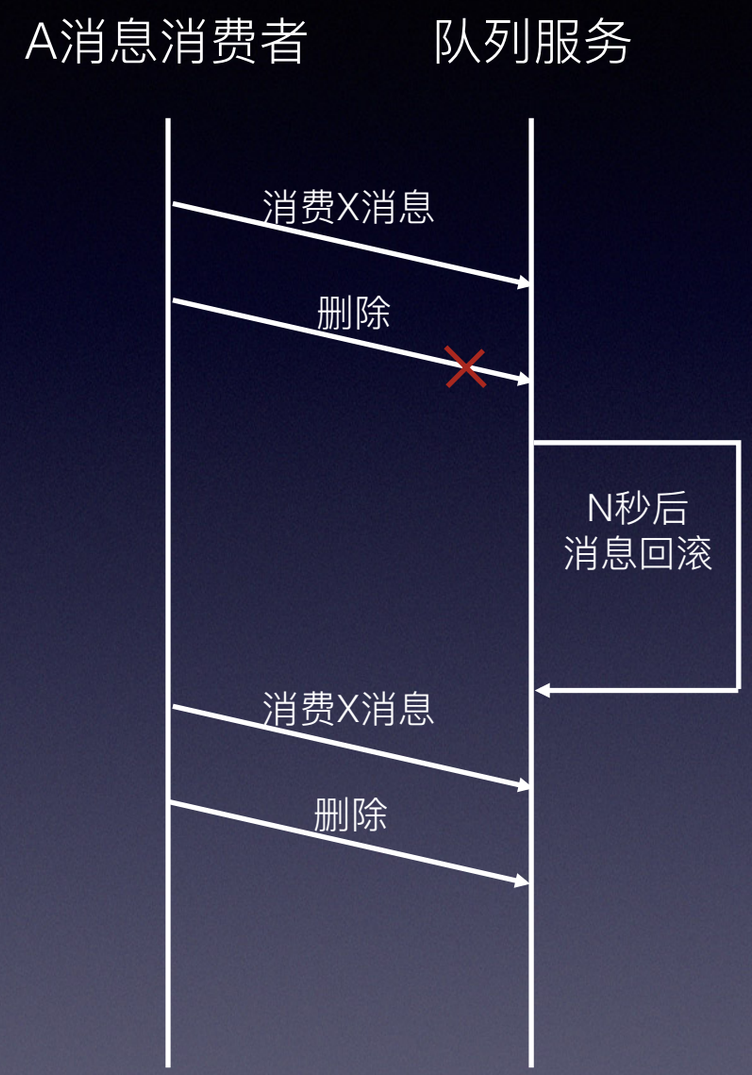

用户消息缓冲池
A用户
B用户
C用户
长度限制
channel
A用户goroutine
B用户goroutine
C用户goroutine
消费者
发布
订阅服务
订阅
GO处理

业务优化

- 每个礼物都是一笔交易，对比电商业务峰值高
- 送礼加入队列后再进行处理
- 消息的优先级可配置
- 原子与幂等


用户送礼
核心流程
扣豆
生成订单
队列服务
发送IM事件
房间总消费统计
个人总消费统计
是否升级
脚本服务
消息消费者*10
排行榜消费*5
日志服务


A消息消费者
B消息消费者
升级服务
1：是否可升级
2：可以
3：是否可升级
4：可以
5：更新等级并发出升级提醒
6：更新等级并发出升级提醒


A消息消费者
队列服务
消费X消息
删除
N秒后
消息回滚
消费X消息
删除

谢谢