msup

微博直播互动架构的热点挑战和应对实践

新浪微博 — 单戈

讲师简介

单戈

直播及通讯平台 - 架构师

- 2017年加入微博
- 负责的主要业务
  - 消息附件服务
  - 基于Trace的智能保障系统
  - 全链路压测系统
  - 直播平台基础服务

直播业务简介

媒体直播

商业直播

答题直播

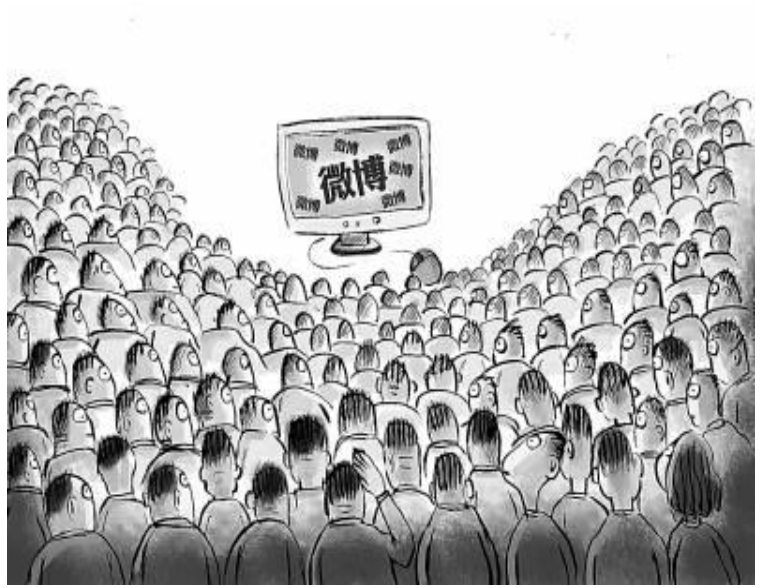
三方互通直播

微博热点特性

微博作为一个公开的社交媒体平台，往往是各类重大消息的首发平台。

不可预测

爆发增长



马太效应

目录

- 案例分析：热点的形成、增长和挑战
- 如何抗住突增的海量同时在线用户
- 如何解决瞬时进场的蜂拥问题
- 如何处理超高并发的消息量
- 总结与展望

案例分析：热点的形成、增长和挑战

《孟晚舟，欢迎回家！》

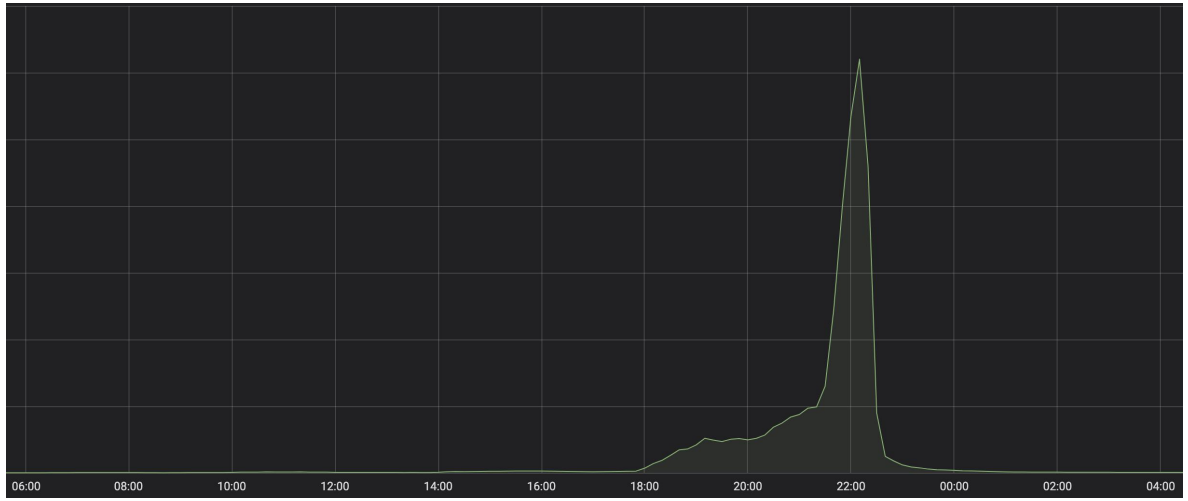
热点案例 – 孟晚舟回国

- 累计观看量PV：1.4亿+
- 最高同时在线：几百万级
- 进场速度峰值：几万级/秒
- 上行互动峰值：十万级/秒
- 下行消息峰值：几千万级/秒

直播热点的形成和增长

◆ 内容爆点引发热点

关注度高+内容爆点+平台推广，热度指数级增长。

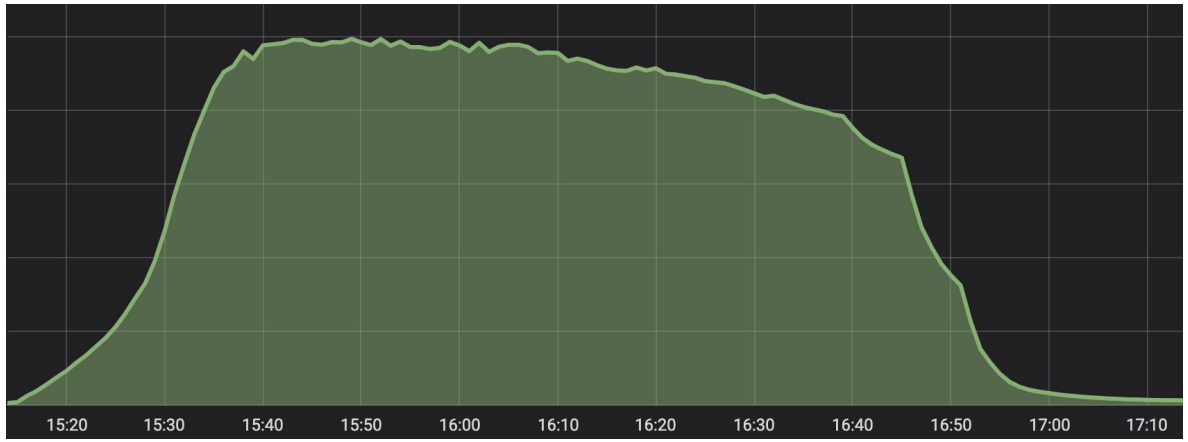


同时在线：孟晚舟回国

◆ 提前造势，开播即巅峰

① 独家直播，有明确的观看+打卡要求的直播

② 直播内容或主播有很大的影响力



同时在线：全国中小学消防云课堂

微博直播互动热点的挑战

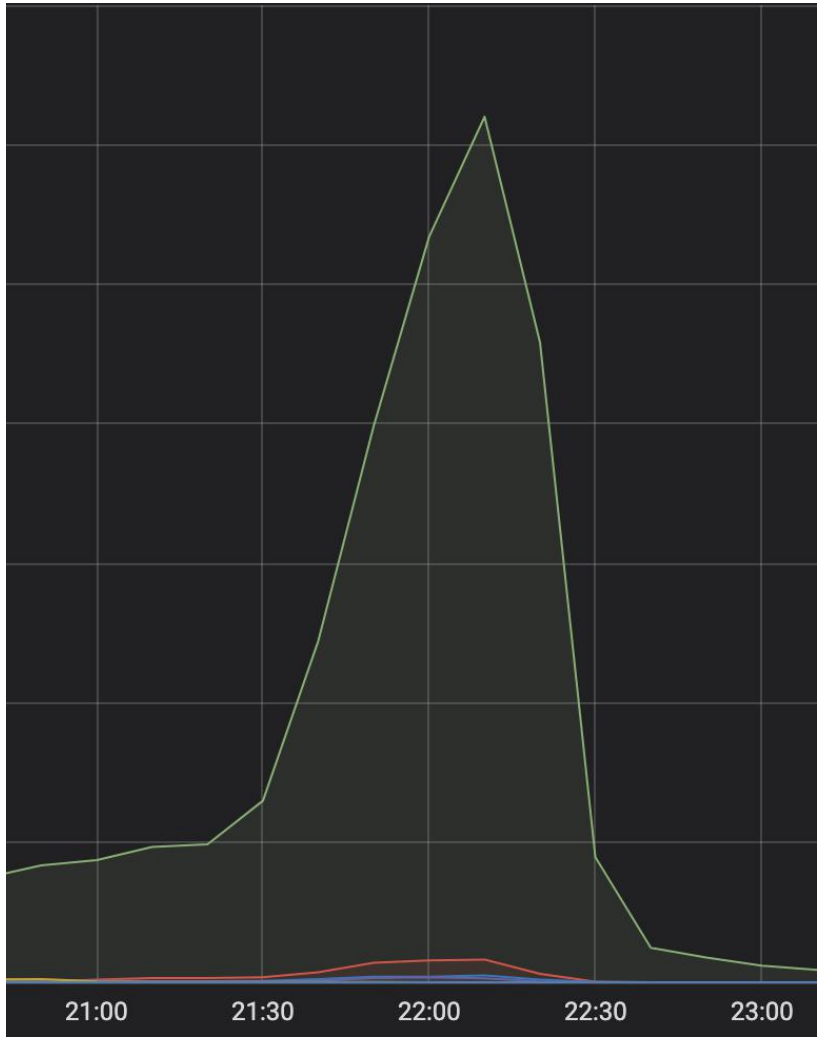
1. 突增几十倍的同时在线量
2. 单直播间超快的进场速度
3. 超高并发的消息处理

随着业务的发展，直播热点的趋势是PCU屡创新高，准备时间也从提前报备到随时一触即发。

如何抗住突增的海量同时在线用户

全链路架构梳理，确定应用和资源的横向伸缩性

海量的在线用户



同时在线：Top10直播间

海量在线用户

- 孟晚舟直播PCU = 几十倍 x Top2房间PCU

海量在线用户的影响

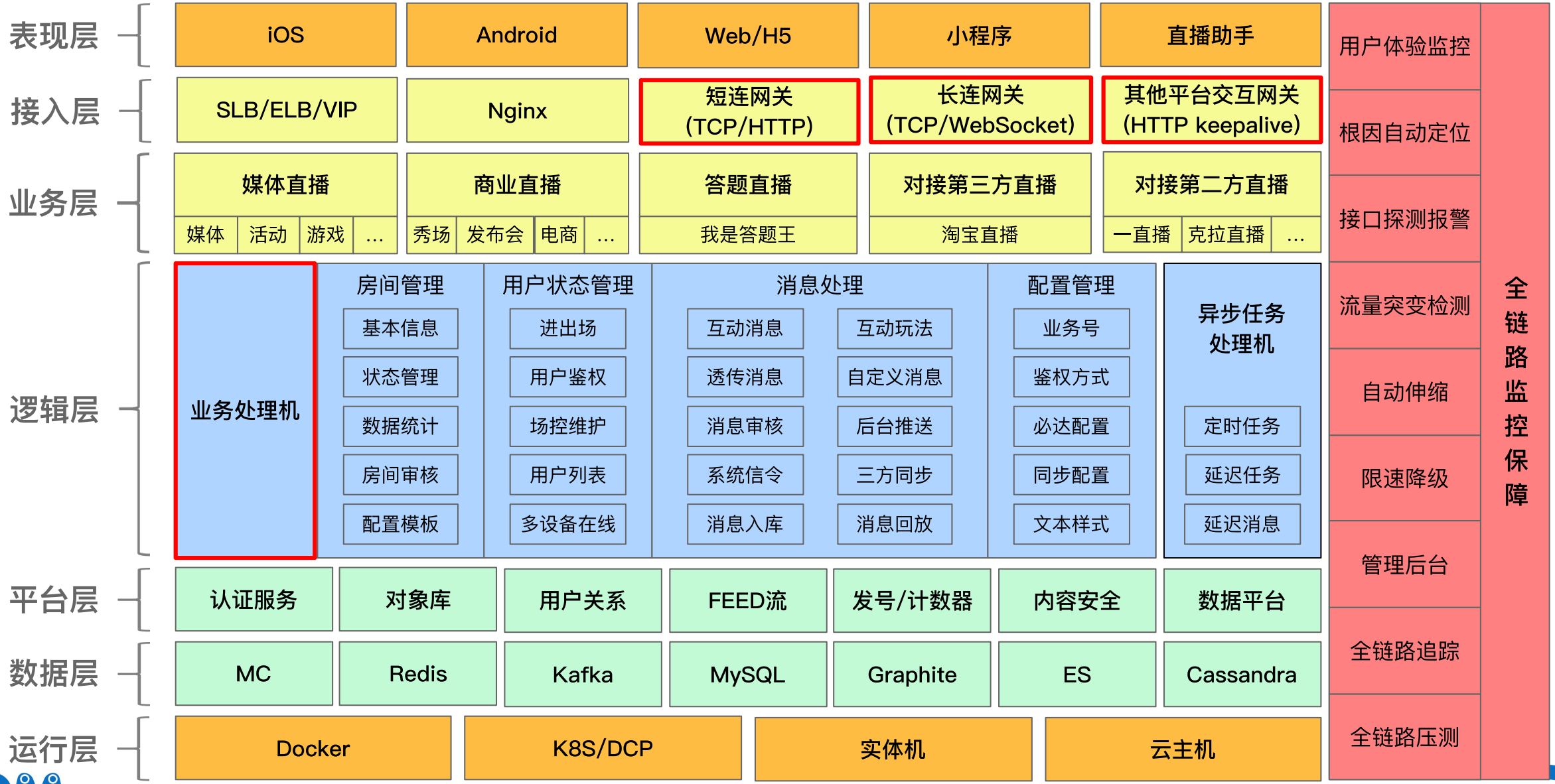
① 海量的用户状态需要维护

② 海量的上行请求需要处理

③ 海量的下行消息需要及时推送

④ 大量的服务器、资源、应用状态需要监控保障

分层架构图

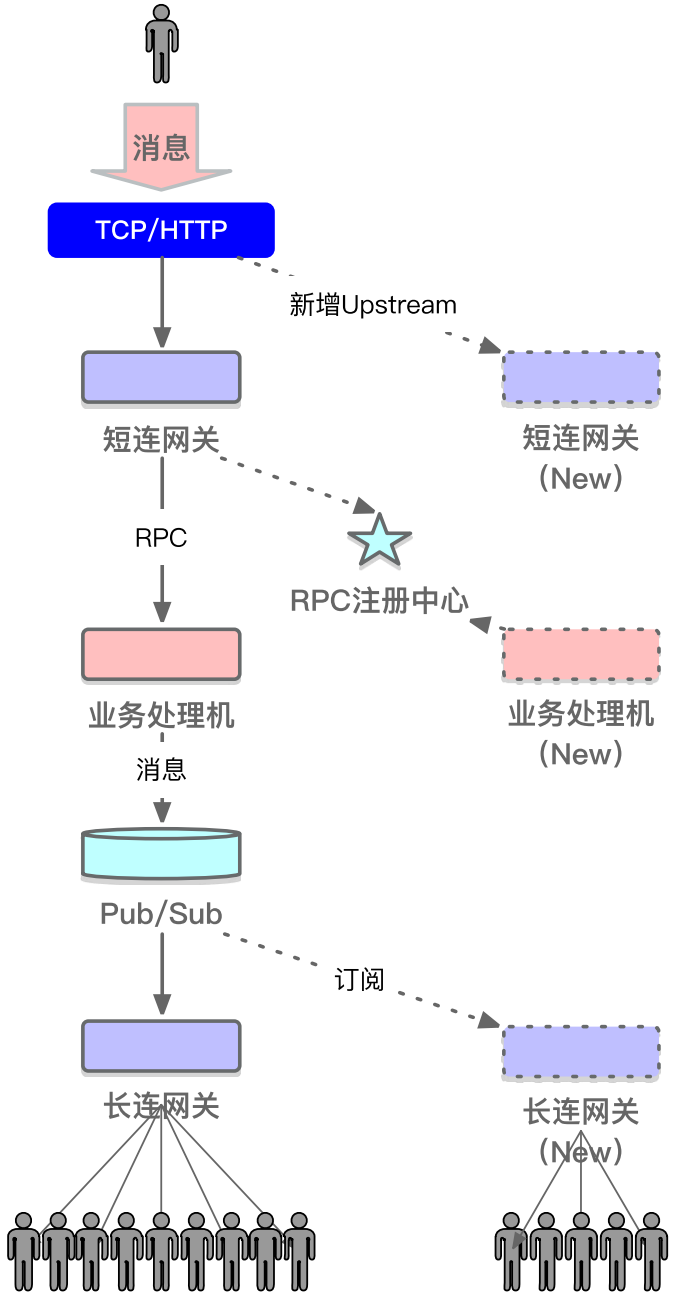

表现层
iOS
Android
Web/H5
小程序
直播助手
接入层
SLB/ELB/VIP
Nginx
短连网关 (TCP/HTTP)
长连网关 (TCP/WebSocket)
其他平台交互网关 (HTTP keepalive)
业务层
媒体直播
媒体
活动
游戏
…
商业直播
秀场
发布会
电商
…
答题直播
我是答题王
对接第三方直播
淘宝直播
对接第二方直播
一直播
克拉直播
…
逻辑层
业务处理机
房间管理
基本信息
状态管理
数据统计
房间审核
配置模板
用户状态管理
进出场
用户鉴权
场控维护
用户列表
多设备在线
消息处理
互动消息
互动玩法
透传消息
自定义消息
消息审核
后台推送
系统信令
三方同步
消息入库
消息回放
配置管理
业务号
鉴权方式
必达配置
同步配置
文本样式
异步任务处理机
定时任务
延迟任务
延迟消息
平台层
认证服务
对象库
用户关系
FEED流
发号/计数器
内容安全
数据平台
数据层
MC
Redis
Kafka
MySQL
Graphite
ES
Cassandra
运行层
Docker
K8S/DCP
实体机
云主机
用户体验监控
根因自动定位
接口探测报警
流量突变检测
自动伸缩
限速降级
管理后台
全链路追踪
全链路压测
全链路监控保障

应对海量在线：(1)全链路可扩容

msup

◆核心链路容量保障

- 网关接入层：扩容SLB/ELB、长/短连网关
- 业务处理机：向注册中心注册RPC服务
- 任务处理机：扩容同group消费任务队列
- 依赖的资源：扩充端口数，增加从库



应对海量在线：(2)无人值守

msup

◆ 自动扩缩容

- 根据接入层的TCP连接数+每秒过包量(PPS)设置可用度指标。
- 长连总体可用度 = Avg (1.0 - Max (Conns / 7k, PPS / 14w))

◆ 保证扩容速度

- 梯度扩容：当前可用度越低、同时在扩容的台数越多。
- K8S + 云主机结合扩容

◆ 保证扩容效果

- 负载均衡：可用度加权随机法

应对海量在线：(3)全链路监控报警

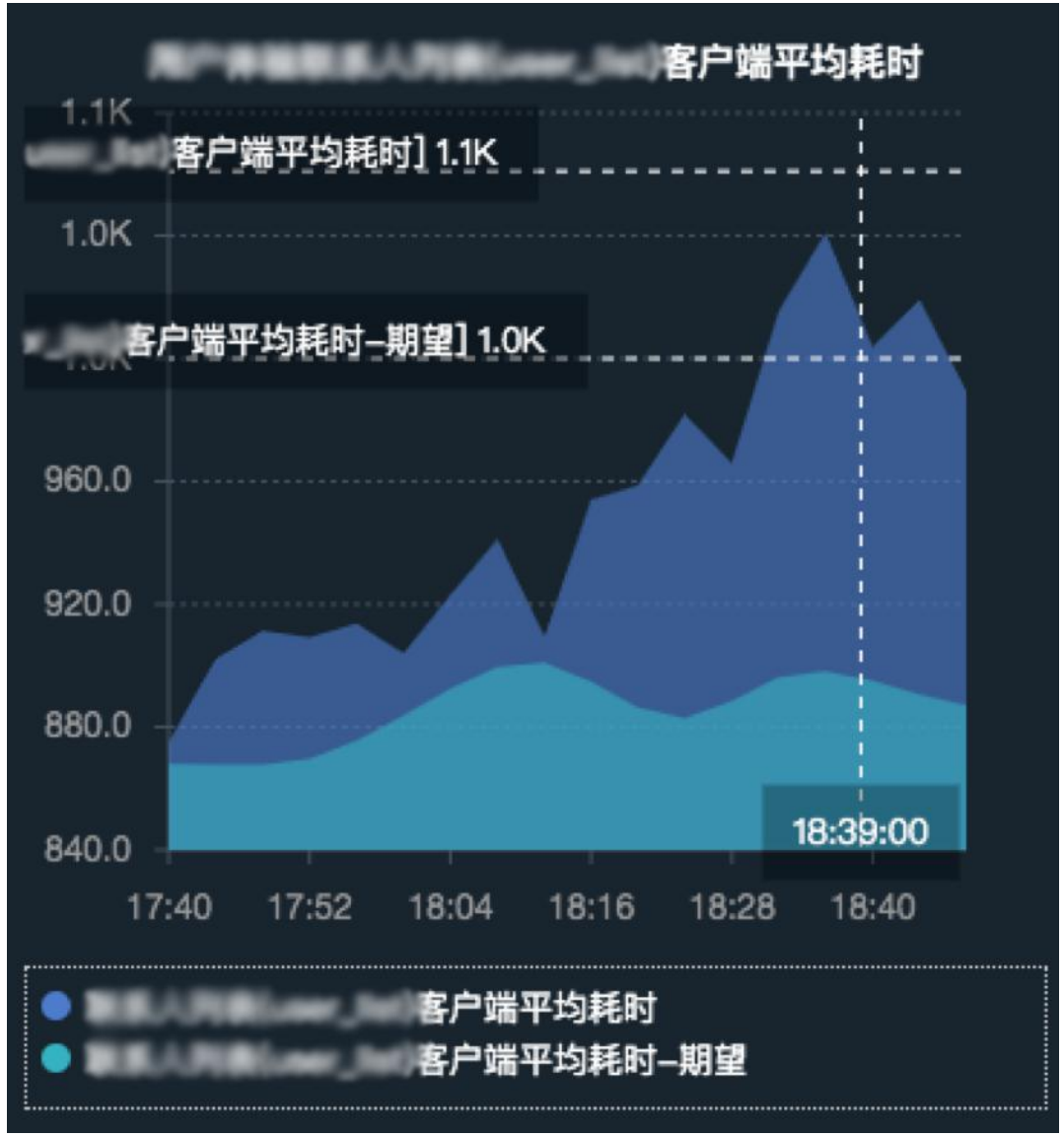
① 核心流程定时探测

② 全网流量和耗时监控

③ 服务状态和可用度检测

④ 异地多机房网络检测

⑤ 基于Trace的用户体验监控和根因分析



全链路压测验证

msup®

◆ 扩容验证

通过扩容可以支撑百倍于日常峰值的用户连接数，但进场速度和推送消息量有瓶颈。

◆ 关键瓶颈

① 单房间热点时，hash到的资源热点端口有写入瓶颈，影响最大进场速度。

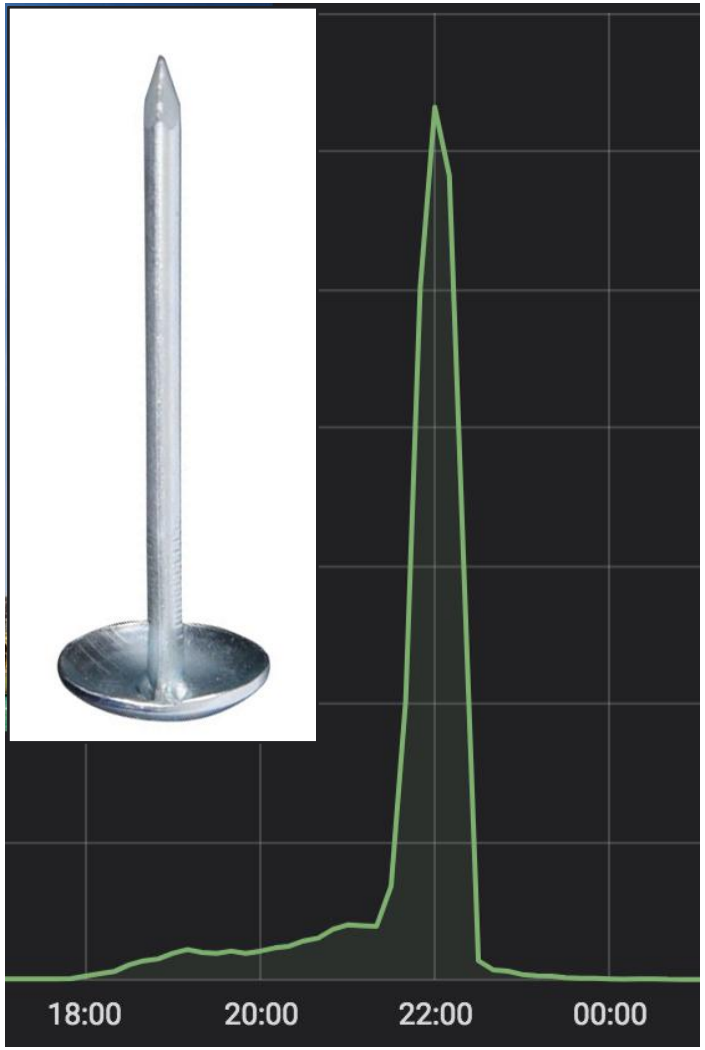
② 海量的定向消息占用了广播通道，在消息扇出环节造成了很大的浪费。

③ 全网消息量过大会造成扇出组件的CPU、内存和带宽的压力倍数增长。

如何解决瞬时进场的蜂拥问题

单房间过热时，要突破所依赖服务和资源的单点瓶颈

蜂拥的问题和影响



进场QPS：孟晚舟回国

◆ **蜂拥现象**：“倒立的图钉”式的进场QPS

内容爆点+平台推广造成热点，进场速度飙升10+倍

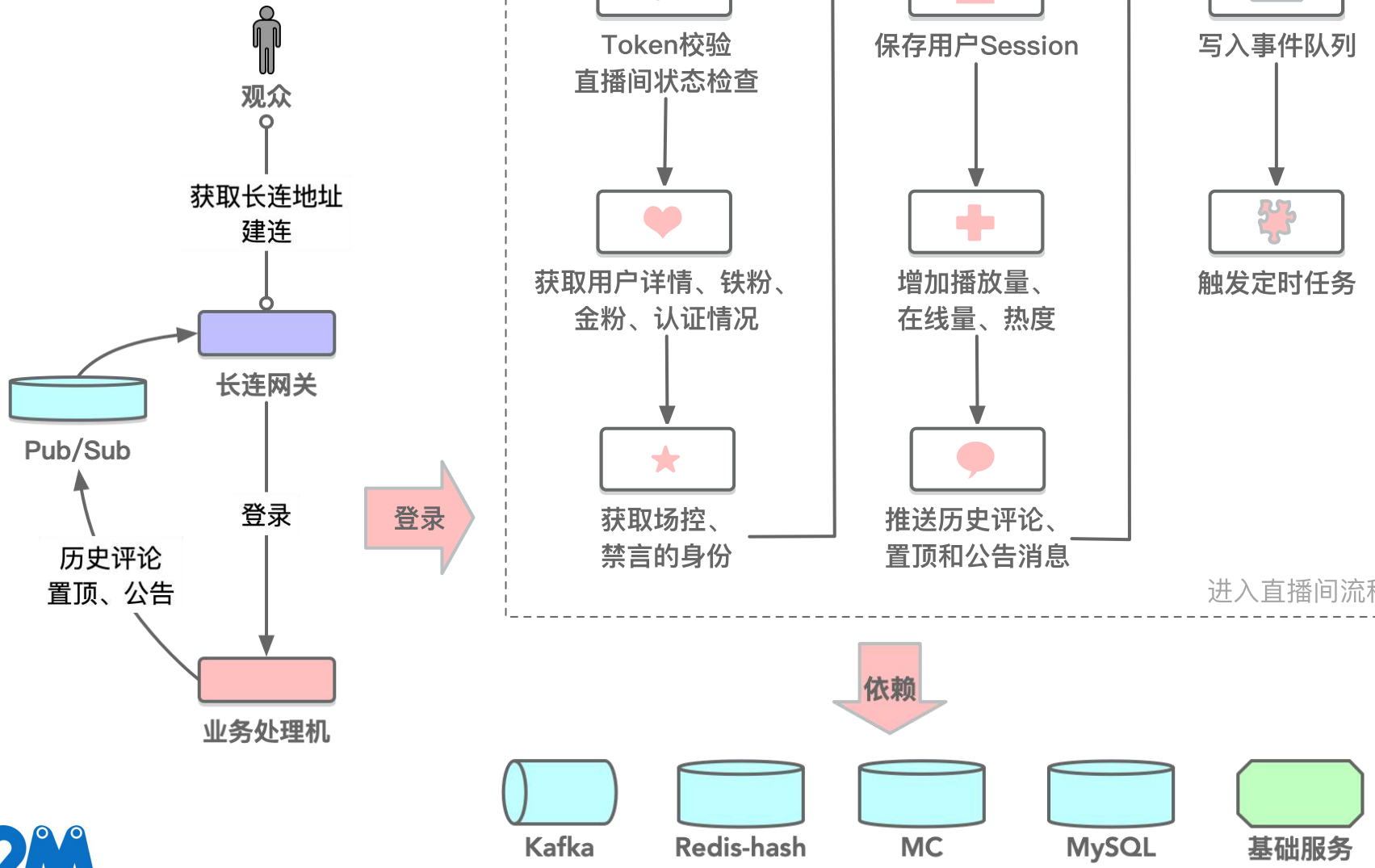
◆ **蜂拥的影响**

① 用户进场逻辑复杂，有单端口过热的问题

② 进场后为每个人下推房间数据和历史消息，定向的消息量大

③ 当失败和超时过多，客户端的无限重试可能导致雪崩

观众进场的流程

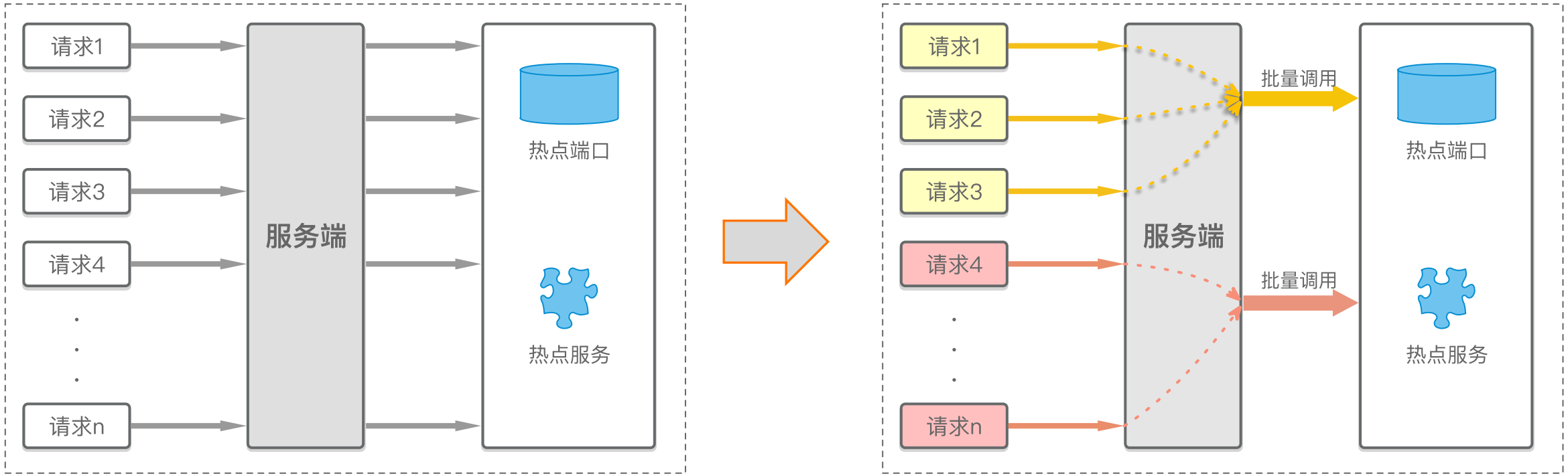


① 处理流程较长

② 有多次对资源的读写

③ 依赖多个基础服务

应对蜂拥：(1)请求合并+异步提交

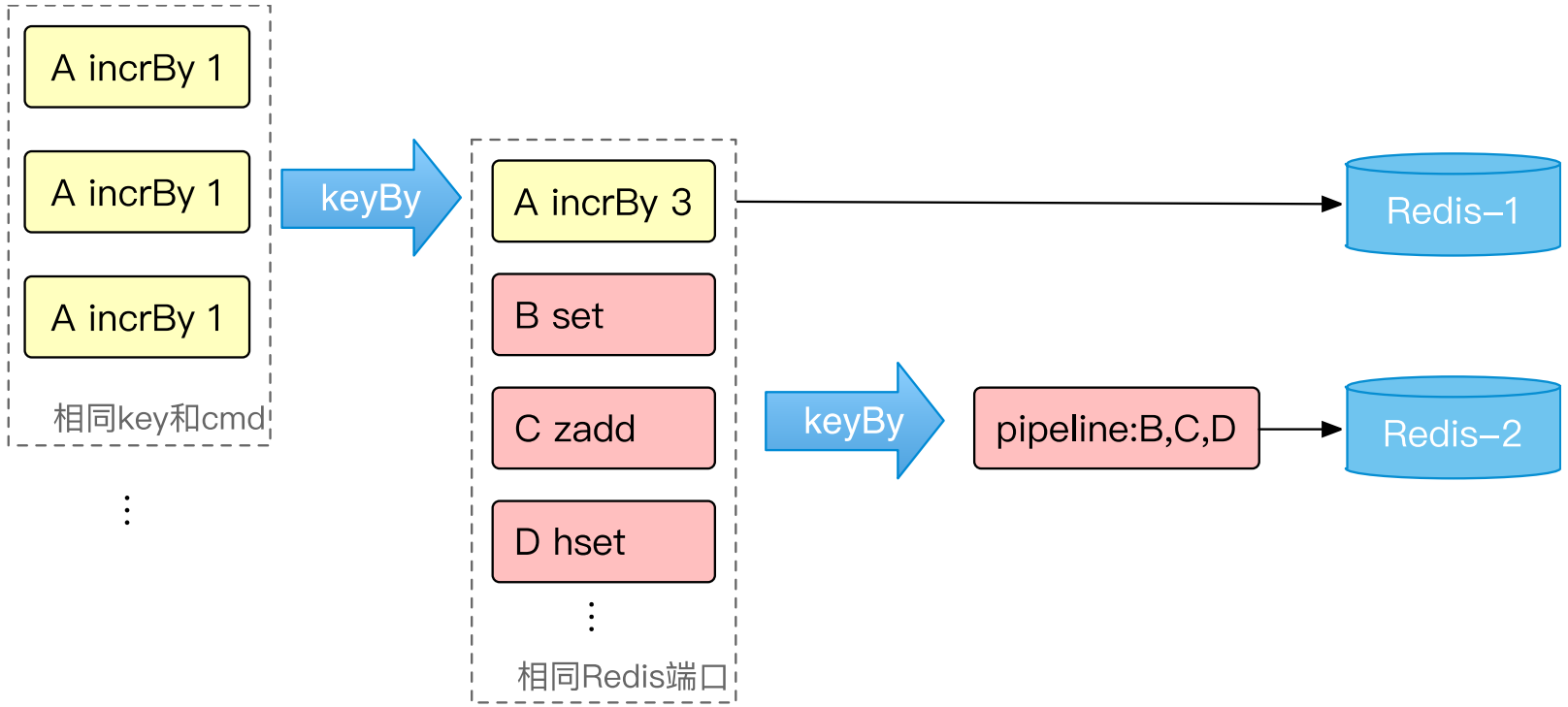


将并发的孤立请求通过内存合并为批量调用。既能有效减少RTT和IO调用次数，又能有效利用基础服务所提供的批处理接口的性能优势。

应对蜂拥：(2)指令级聚合和打包 (BufferSlayer - Redis)

msup

将热点资源端口的压力转化为应用集群的本地计算，继而通过扩容应用来解决。

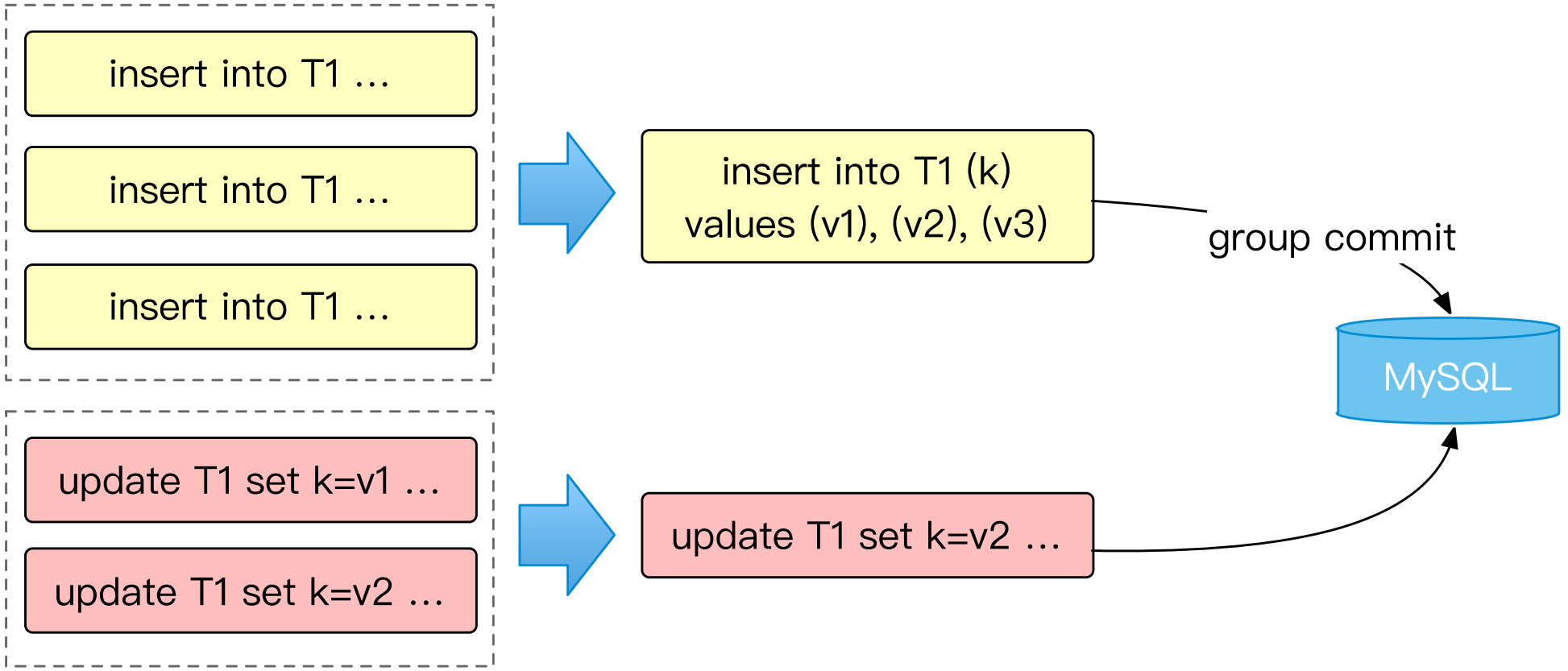


① 指令自动聚合：incrBy / hincrBy。

② 指令自动打包：将孤立cmd合并为pipeline。

成果：热点时减少95%+的实发指令数，整体提升2~3倍吞吐量。

应对蜂拥：(2)指令级聚合和打包 (BufferSlayer - MySQL)



在保存互动消息时，将孤立的insert命令转换为batch insert。

成果：热点时提升30倍+的吞吐量。

应对蜂拥：(3) 本地缓存抗读

msup

允许部分数据从本地缓存读取，减少资源的读压力。

◆ 适用的场景

① 写少读多的数据：房间基本信息，网安消息，置顶消息，场控和禁言用户列表

② 可接受短暂延迟：历史评论，在线观众列表，各类展示计数

成果：热点时可以减少95%+对资源的读qps，无需扩容资源。

如何处理超高并发的消息量

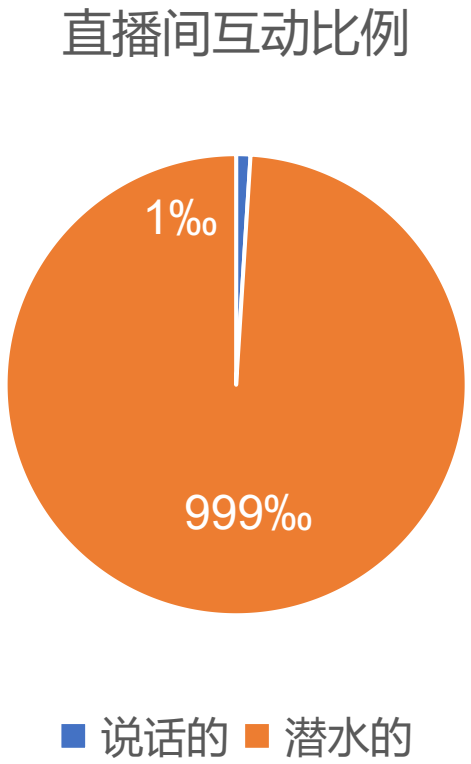
消息推送量 = 消息发送量 x 在线人数

超高并发的消息推送量

msup

◆ 笛卡尔积式的推送量

随着在线人数的不断增长，消息的扇出压力是按指数扩大的。



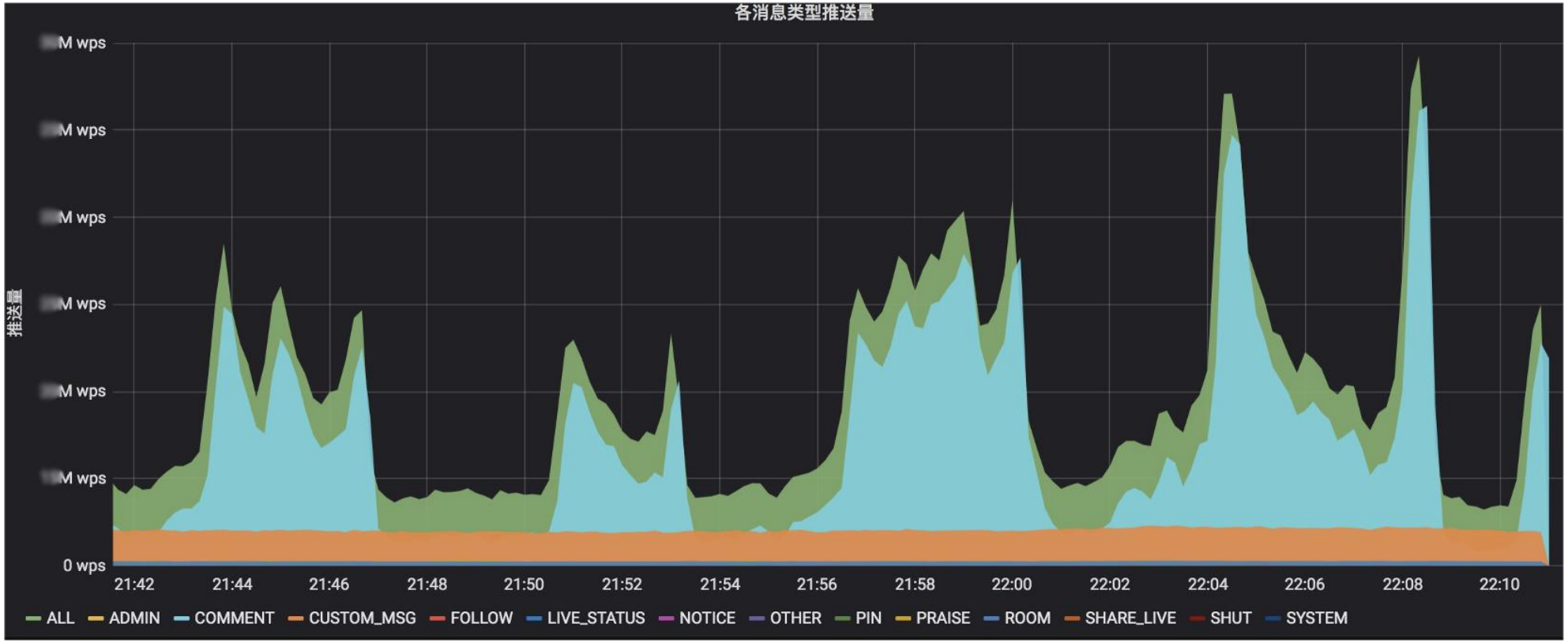
在线人数	理论发消息量	理论推送量
1万	10/秒	10万/秒
10万	100/秒	1千万/秒
100万	1000/秒	10亿/秒

100万在线用户 → 1万台长连网关?

踩坑经历：失控的第三方互通直播间

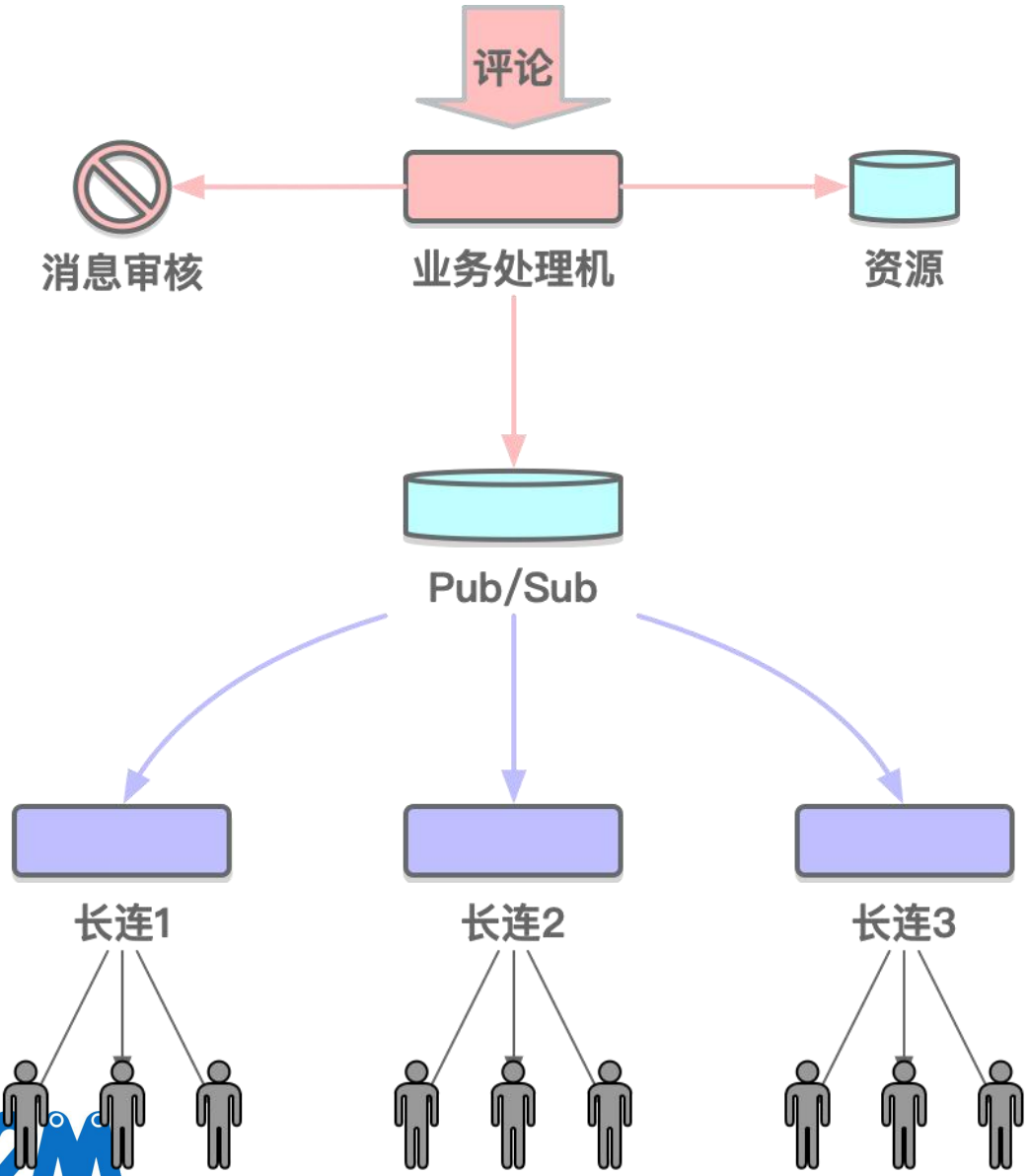
◆ 第三方原始消息未限速

消息推送量随着主播口令和观众的回应产生了剧烈的波动，峰值几千万级/秒。



某年双十一带货直播

超高并发的消息对系统的压力



◆ **业务处理机**

① 消息处理和内容填充

② 对依赖的基础服务和资源的压力

◆ **扇出组件**

① CPU，内存

② 连接数、带宽

◆ **长连网关机**

① 机器负载

② 带宽，每秒过包量（PPS）

应对海量消息：(1)上行预处理

msup

◆ 消息审核和分级限速

① 评论内容和用户身份基于审核模型的过滤，重要场次使用人工审核。

② 配置所有类型消息的限速额度，保障必达和高优先消息，优化阅读体验。

◆ 同类消息合并

① 将多人的并发点赞消息合并为1条，将其计数设置为累计值。

② 仅推送合并后的消息，客户端接收后读取计数值，渲染为多条效果。

应对海量消息：(2)改良的中央限速器

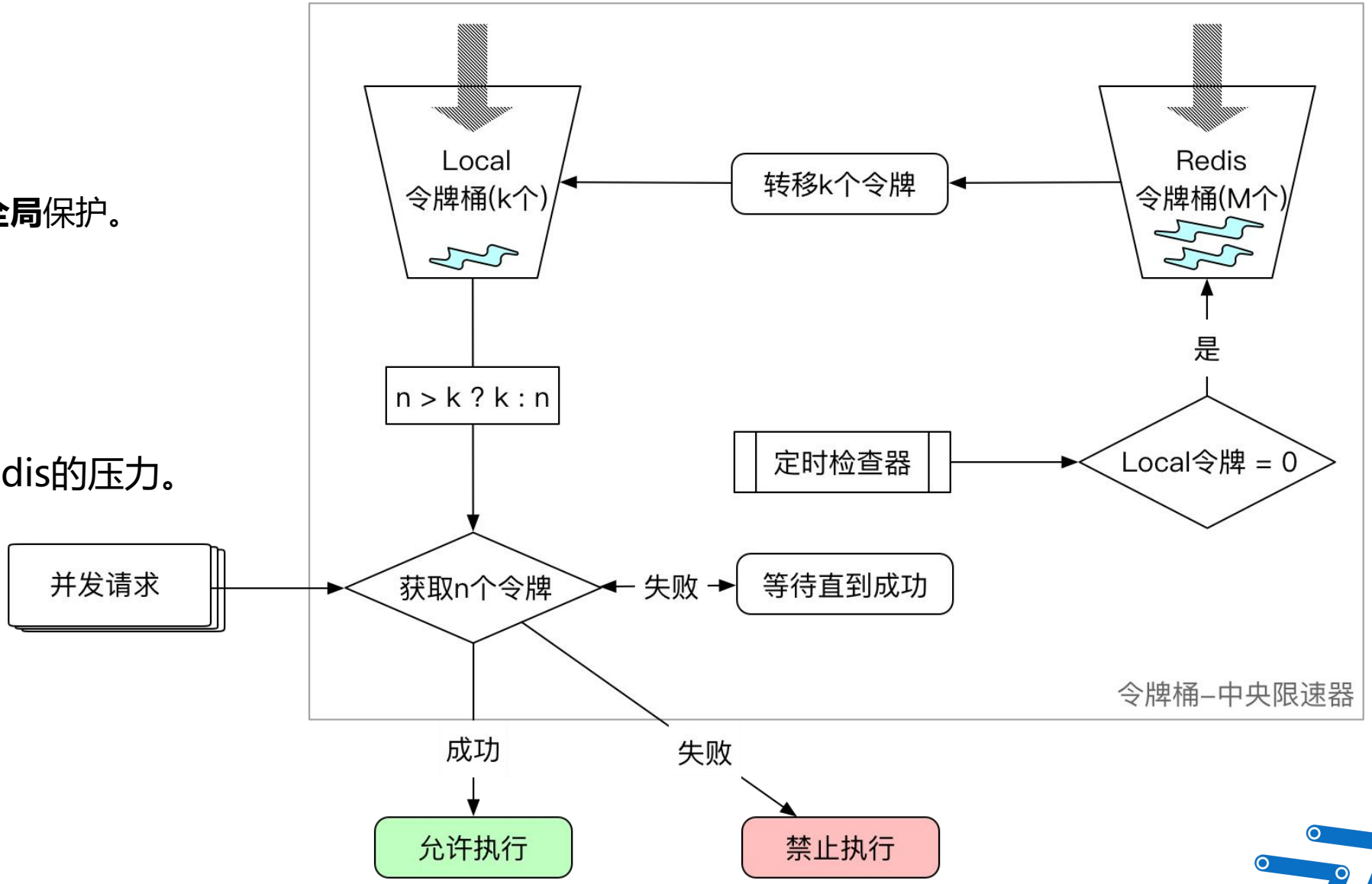
msup

◆ **基于Redis的中央令牌桶**

对集群整体限速，提供有效的**全局**保护。

◆ **扩充了本地缓冲令牌桶**

通过本地桶可以有效降低Redis的压力。



热应对海量消息：(3)定向推送链路优化

- 广播消息：用户评论、点赞、分享、礼物、红包等
- 定向消息：历史评论、答题结果、观看成就达成等

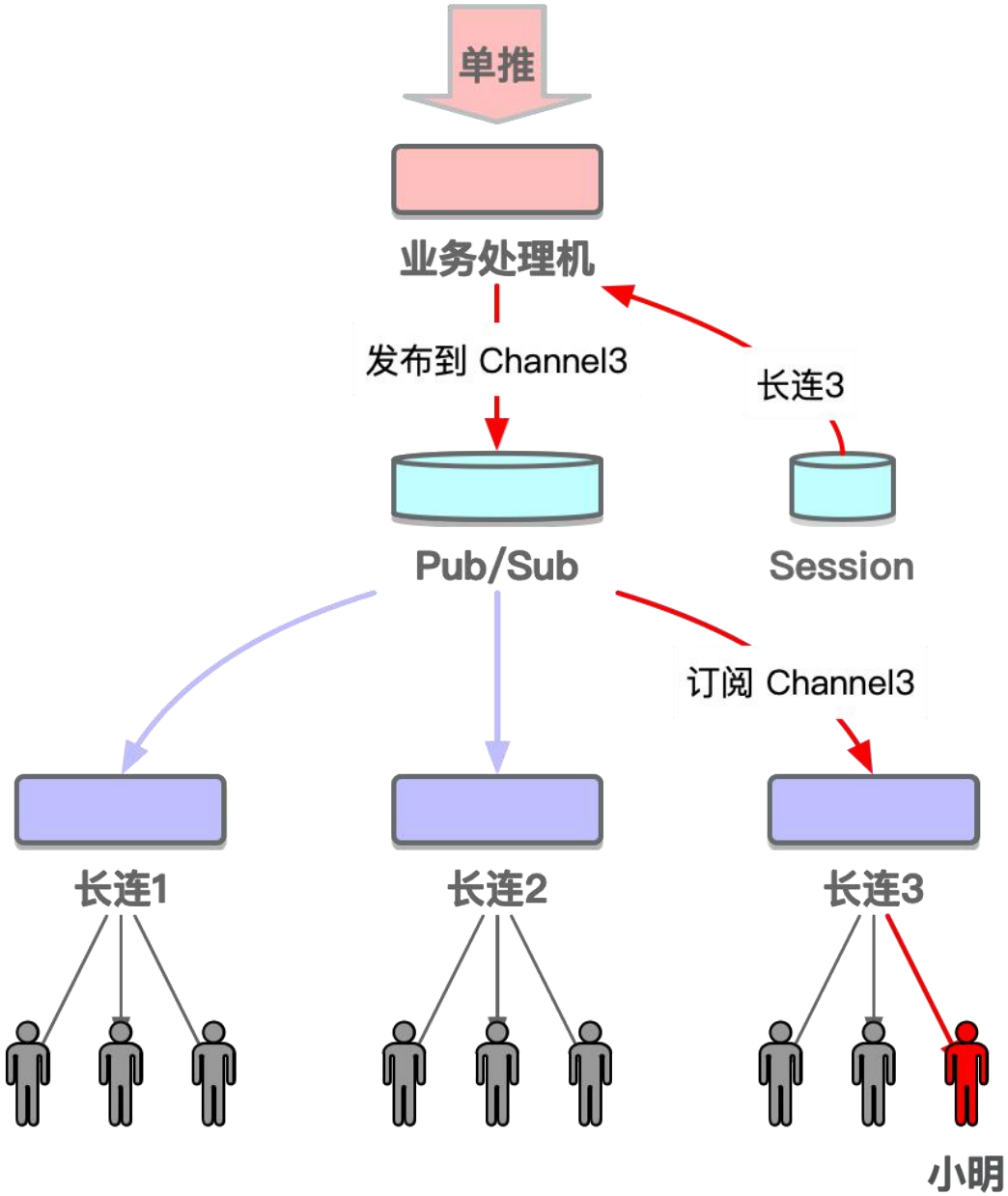
◆ 定向消息推送流程

① 小明登录后，保存其状态Session

② 定向推送时，先查Session获取其所在的长连编号

③ 仅“长连3”订阅了“Channel3”

④ 向“Channel3”发布该定向消息



应对海量消息：(4)消息下推重排序策略

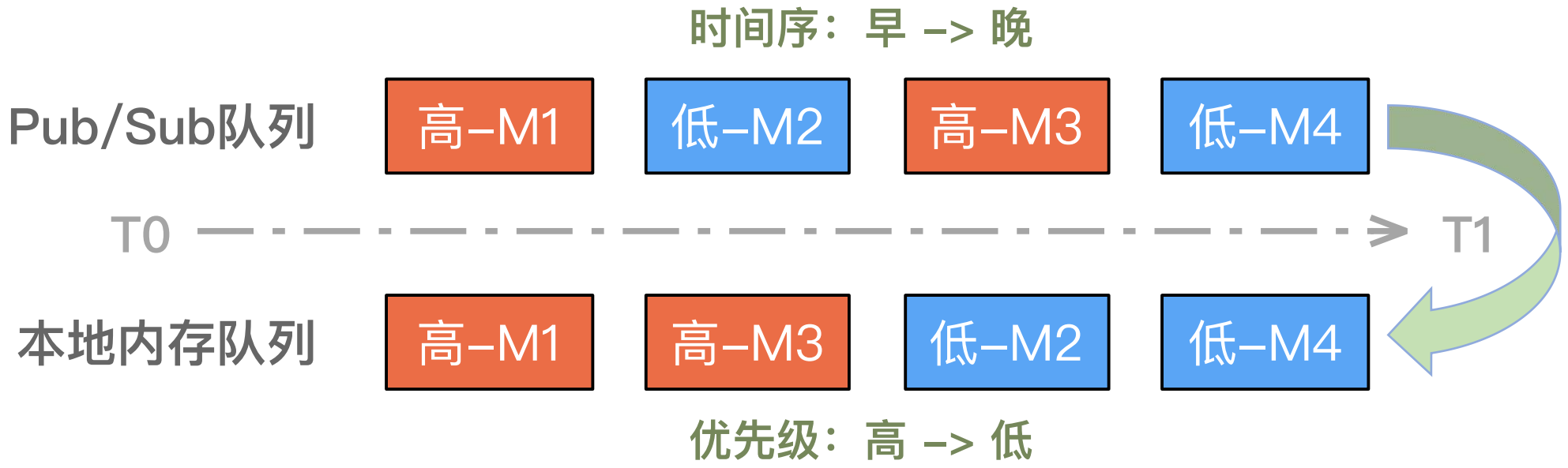
msup

◆ 优先级背压法

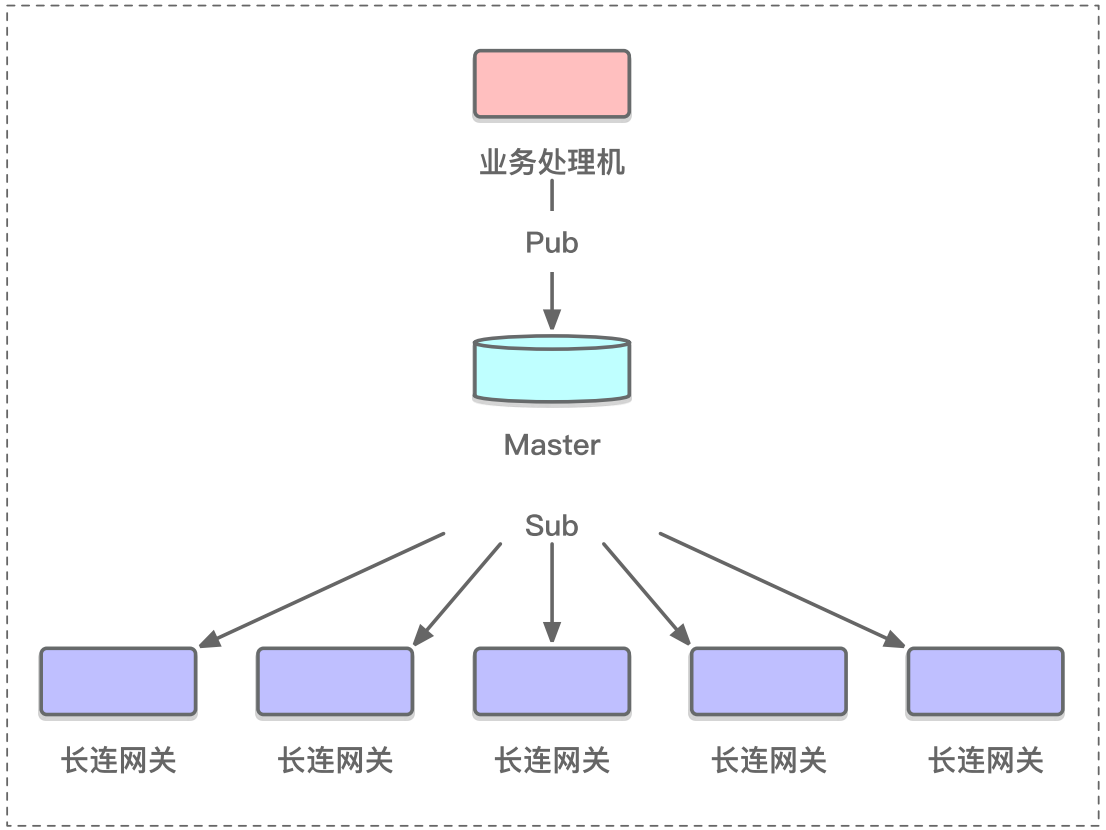
① 业务处理机按时间序列将消息pub到扇出组件

② 长连顺序接收后，将消息从时间序列转化为优先级序列并缓存

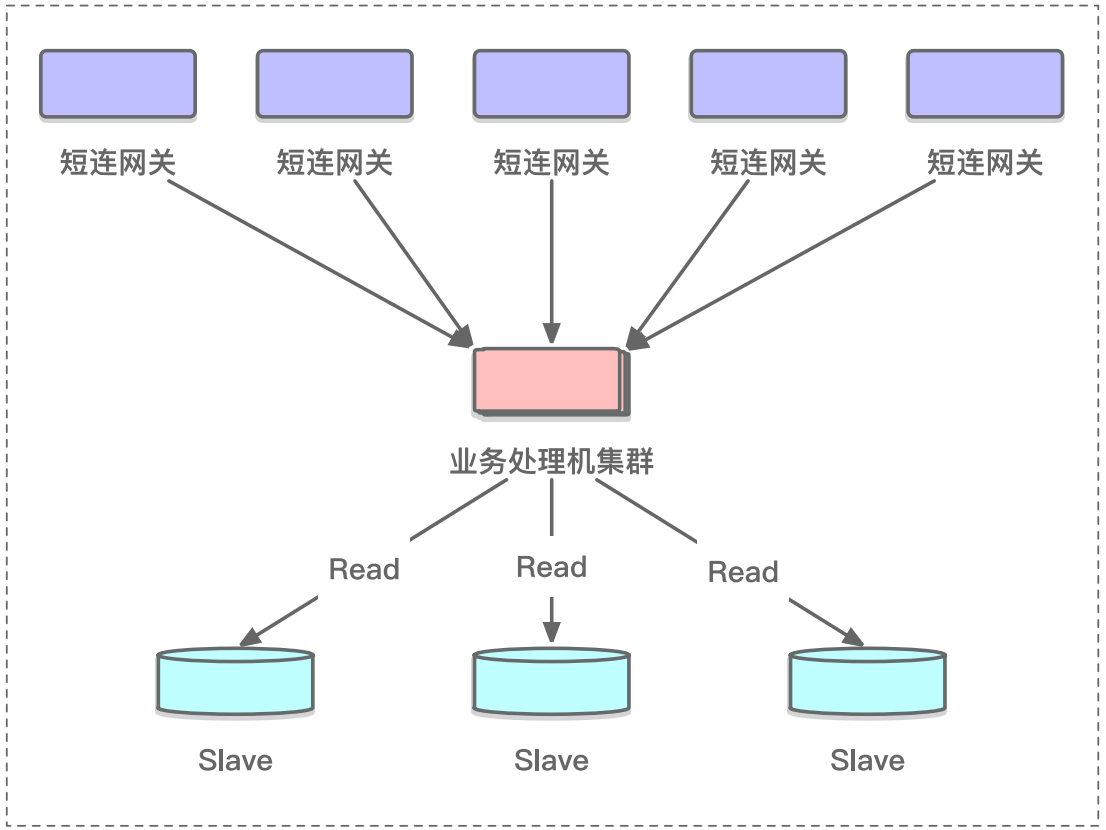
③ 推送高优先级的消息，丢弃队尾低优先的过期消息



应对海量消息：(5)拉模式预案



推模式：全网消息体量大时有瓶颈



拉模式：扩Slave或用LocalCache

应对海量消息：(6)防止雪崩

msup

◆ 增强限速

① 扩大本地缓存的覆盖范围

② 扩充BufferSlayer的内存队列长度，进一步减小对资源的压力

③ 修改不同种类消息的分级限速策略

◆ 降级预案

① 当互动热度较高时，可以降级历史消息的推送

② 进场用户不获取铁粉、金粉、认证等身份标识

③ 对依赖服务和资源的快速失败和熔断

④ 限制新用户进入房间

总结与展望

总结

msup

海量的在线用户

① 可用度检测+自动扩缩容
② 分级扩容保证速度和容量
③ 压测验证了扩容效果

超快的进场速度

① 远程请求合并
② 缓冲层抗写量
③ 本地缓存抗读量

超高并发的消息处理

① 上行消息审核、合并和限速
② 优化定向推送链路
③ 消息下推重排序策略
④ 拉模式备案
⑤ 限速、降级和熔断

全链路压测结果符合预期

msup

① 百万用户的10秒进场成功率为99.9992%

② 消息的整体到达率为99.998%

③ 99.99%的消息接收延迟为1.5秒

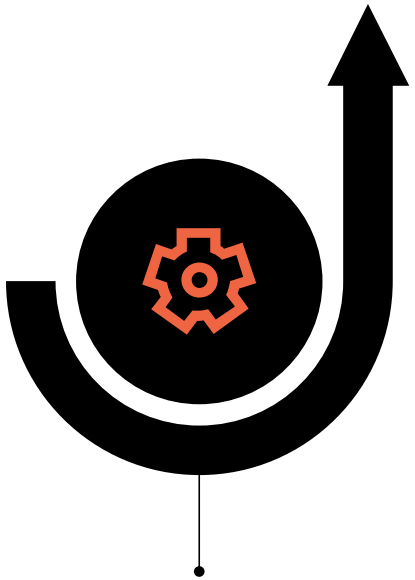
④ 消息的成功率、到达率和延迟符合预期

⑤ 客户端无崩溃、卡顿或发热问题

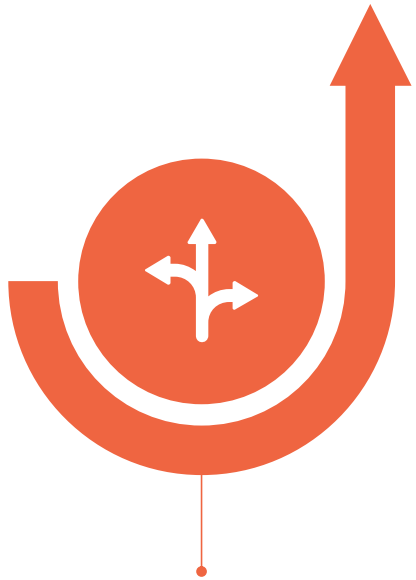
服务端压测结果

客户端压测效果

下一步计划



弱网优化



消息智能投放

关注msup公众号

获取更多AI落地实践

麦思博(msup)有限公司是一家面向技术型企业的培训咨询机构，携手2000余位中外客座导师，服务于技术团队的能力提升、软件工程效能和产品创新迭代，超过3000余家企业续约学习，是科技领域占有率第1的客座导师品牌，msup以整合全球领先经验实践为己任，为中国产业快速发展提供智库。