QCon 北京 2014

移动时代

——QQ后台架构的演化与启示(二)

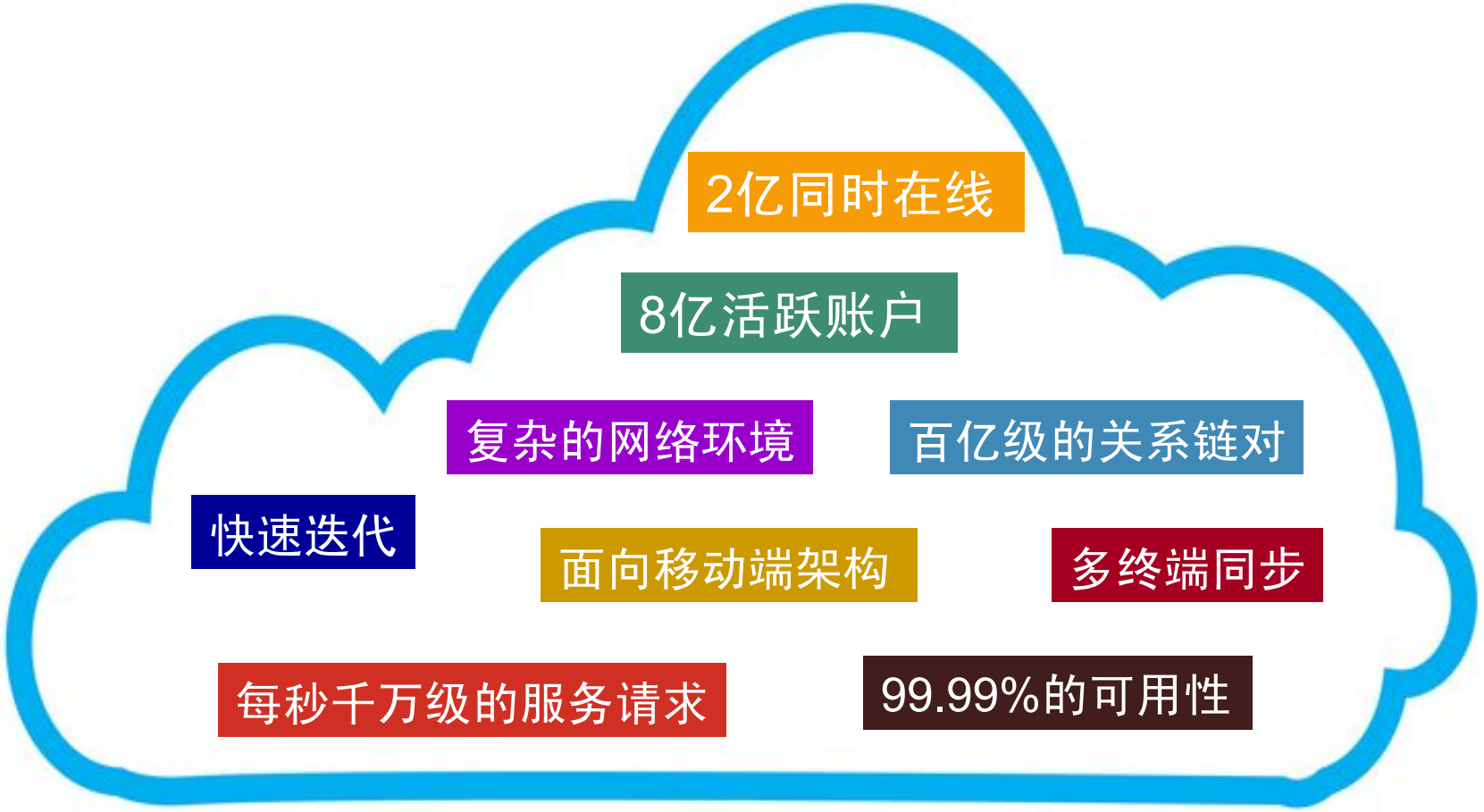
腾讯 韦彬
mikewei@tencent.com

自我介绍

- 韦彬 （mikewei）
- 2006-浙江大学计算机学院硕士毕业，加入腾讯
- 2006-至今，专注于QQ后台的架构研发运营
  - 后台开发通道 T4专家
  - 即通平台部 技术副总监
  - 经历了QQ从2千万到2亿在线、从PC到移动时代的过程
  - 擅长方向：海量分布式后台架构、通信系统、linux内核等

QQ后台(?)

在挑战中不断发展演进



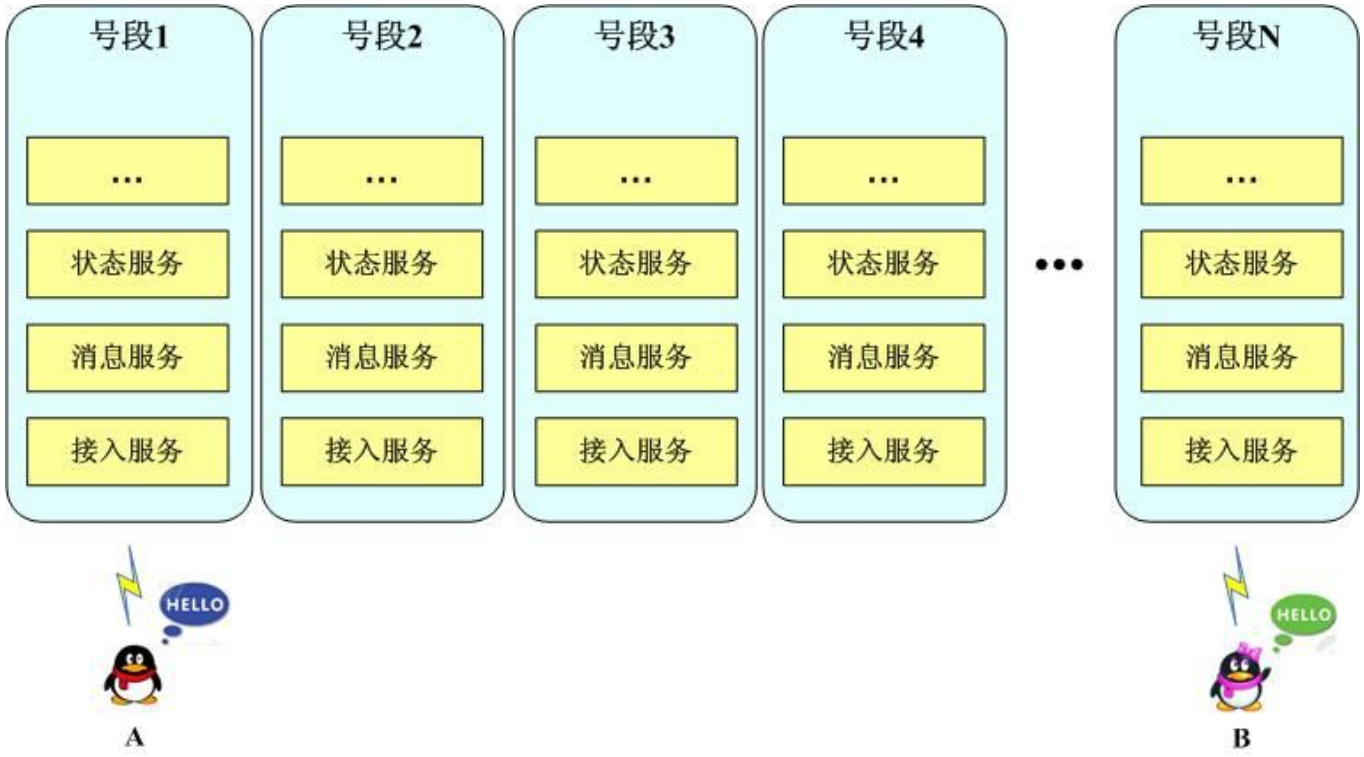
目录

- 从2千万到2亿在线
- 从单点登录到多终端在线
- 从桌面环境到移动环境

从2千万到-

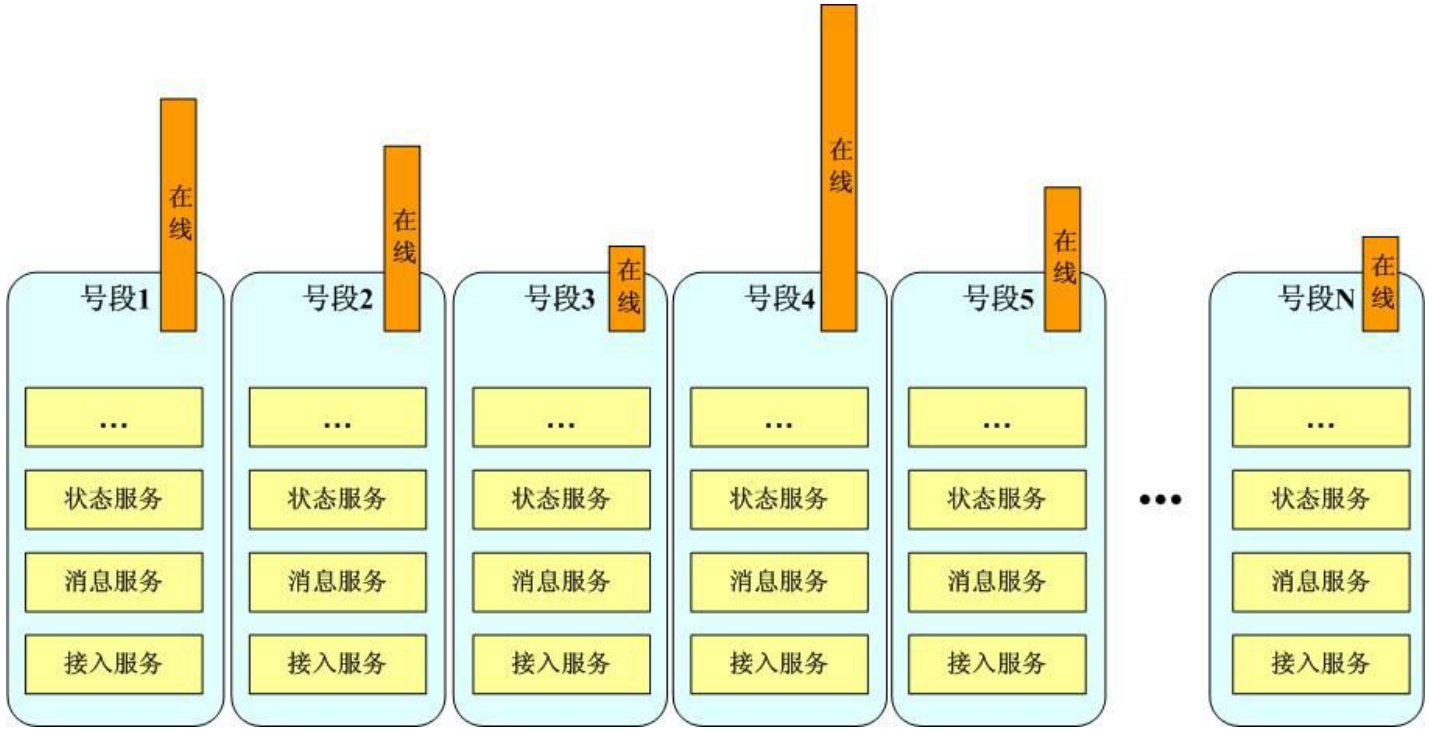
原千万级架构

- 水平伸缩：按QQ号段划分集群
- 从百万级在线水平扩展到千万级在线



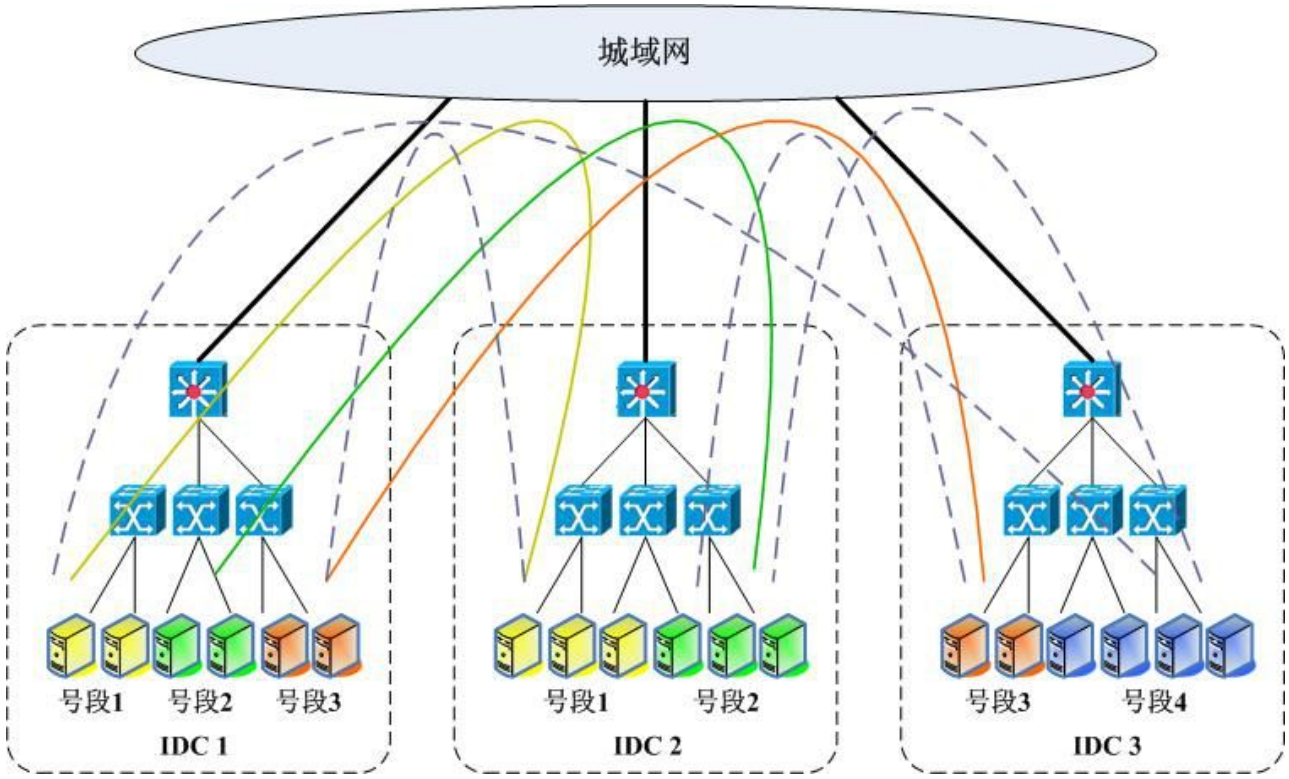
原架构的挑战

- 伸缩灵活性



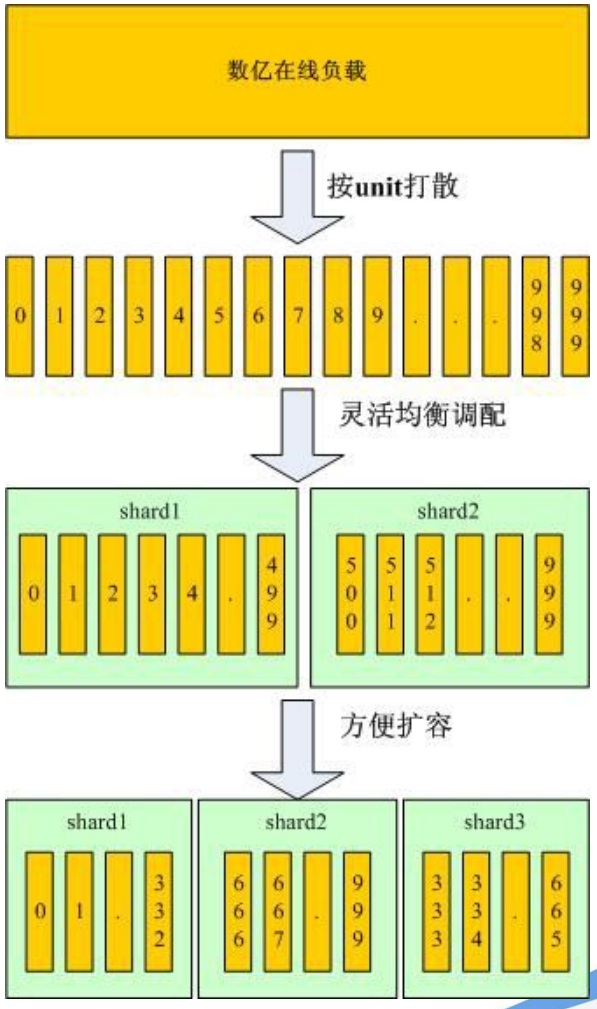
原架构的挑战

- 物理网络瓶颈
- 运维成本



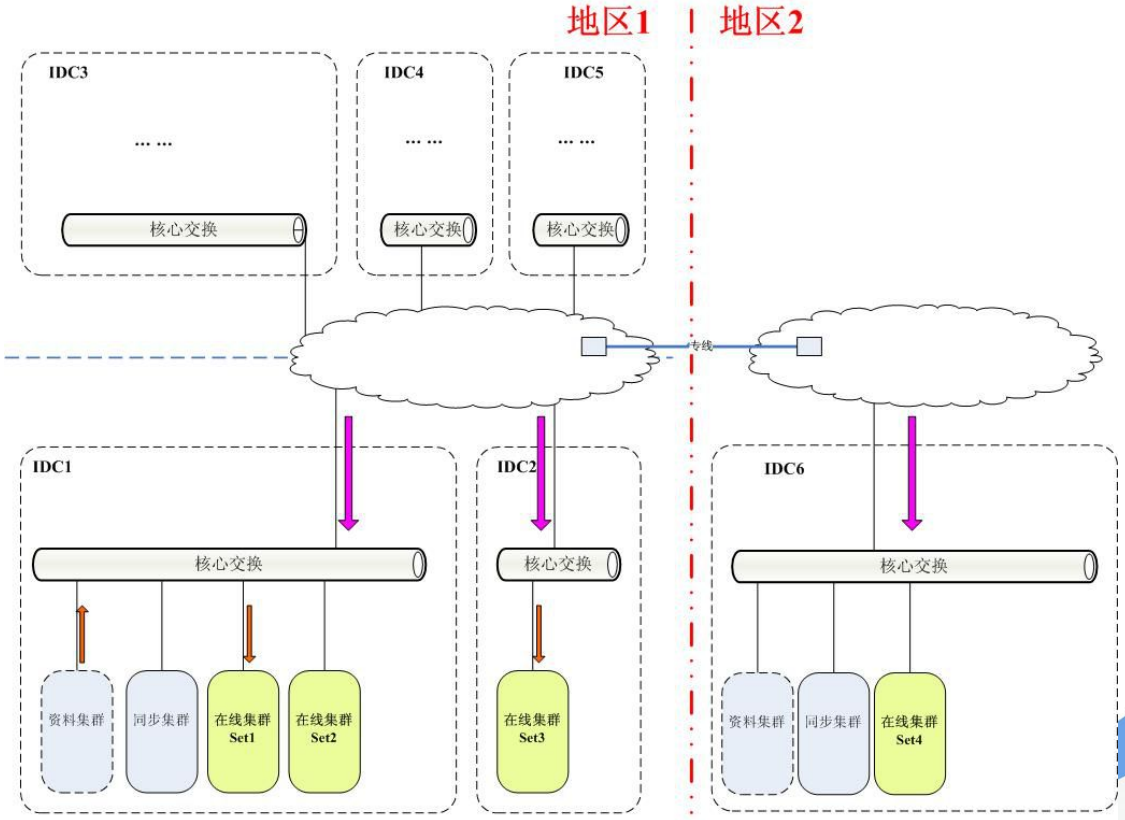
基于unit映射的sharding设计

- 灵活水平伸缩
- 方便扩容和运维



基于在线的set集群设计

- 在线为容量指标
- 按需伸缩



关于架构伸缩性(Scalability)设计的启示

- 不同量级下伸缩性设计会不同
- x10 vs x100
- 亿级IM系统为例
- 有效方法：建立Set模型

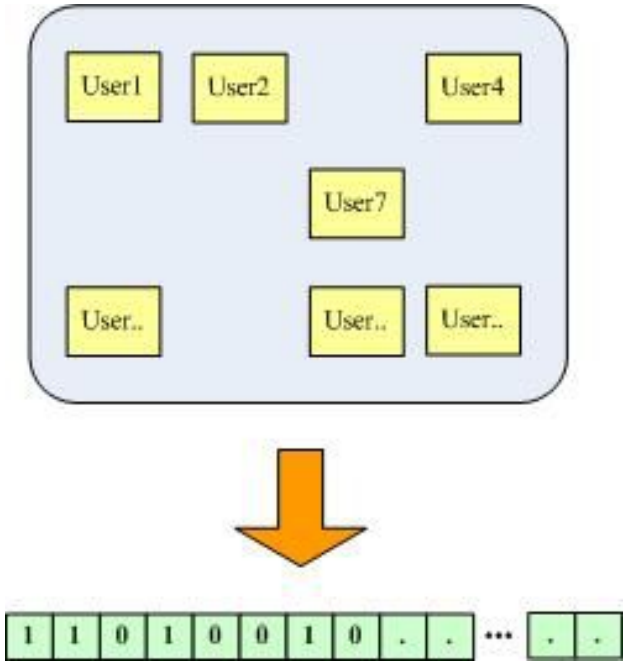
目录

- 从2千万到2亿在线
- 从单点登录到多终端在线
- 从桌面环境到移动环境

状态系统-原设计

- 固定数据结构/策略
- 支持多终端、多业务？

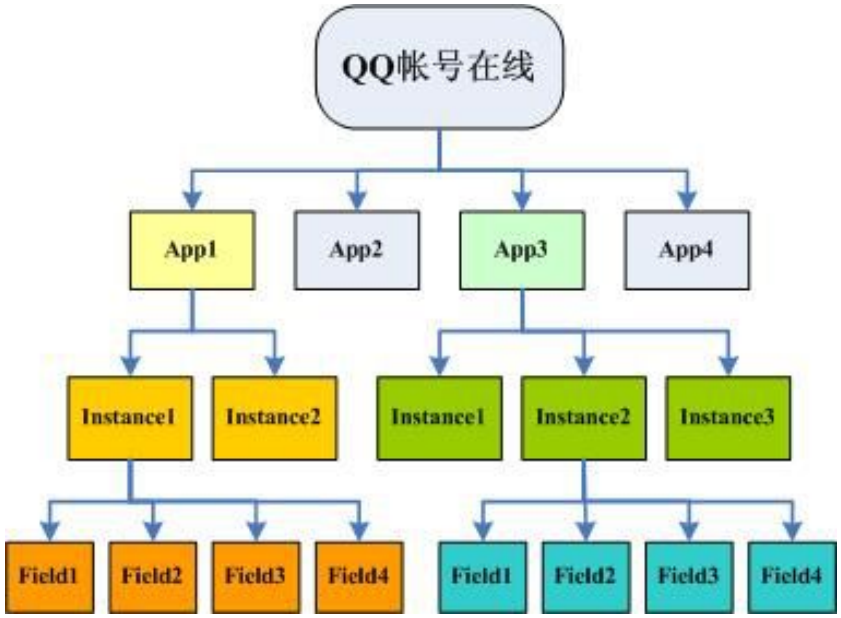
```
typedef struct
{
    unsigned C2_INT32 lUin;
    unsigned C2_INT32 lFlag;
    char  sTeaKey[16];
    char    sIP[16];
    char    sPort[6];
    unsigned short shPtl;
    unsigned short shStatus;
    unsigned short shSubStatus;
    C2_TIME  lAlive;
    char    cReLoginFlag;
    unsigned short shConnFlag3;
    unsigned short shConnFlag4;
    unsigned short shServerID;
    C2_TIME  lStatusTime;
    unsigned short   shConnFlag1;
    unsigned char cIspId;
    unsigned char cMode;
    char    cClientIndex;
    unsigned short shFriendNum;
    unsigned short shFriendLevel;
    unsigned C2_INT32 lFriendPos;
    C2_TIME lLoginTime;
    char sIdentifyBitmap[4];
    char sServiceBitmap[8];
    char cAllow;//yradd1030 是否允许其他人加为好友
    short shQQLevel;// 在线时长等级，负数(-1)表示没有获取等级信息
    unsigned char cClientType; // 客户端接入类型，比如手机、Service等
} OnlineRecord;
```



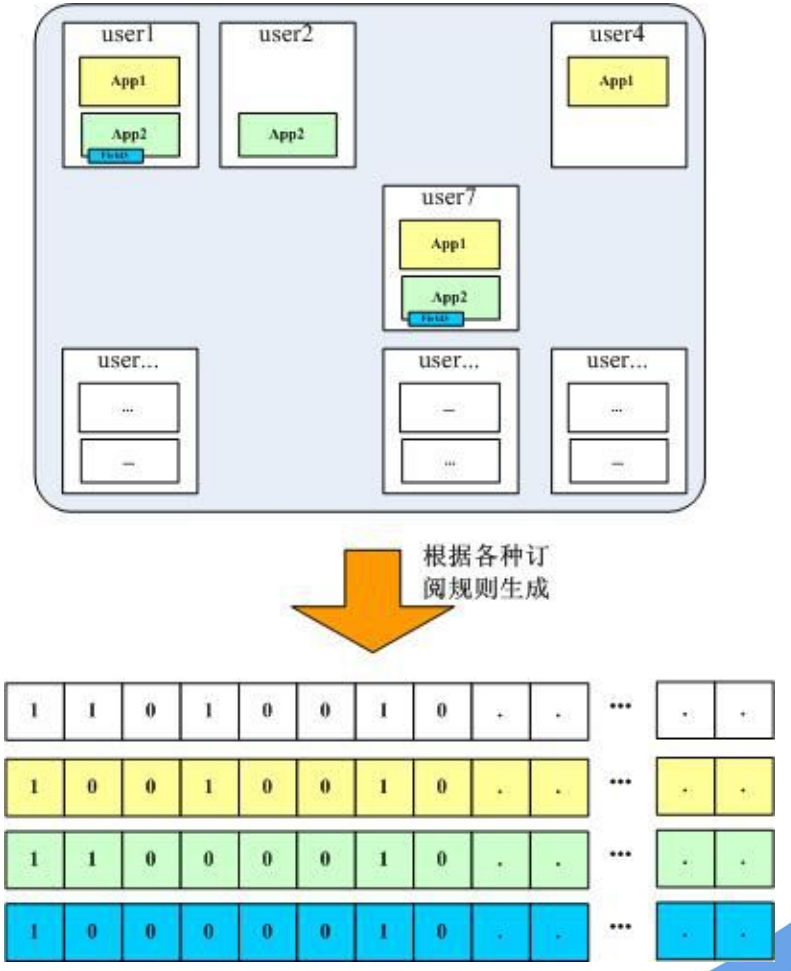
状态摘要Bitmap

状态系统-支持多终端在线

- 更灵活的在线状态结构



在线状态数据结构



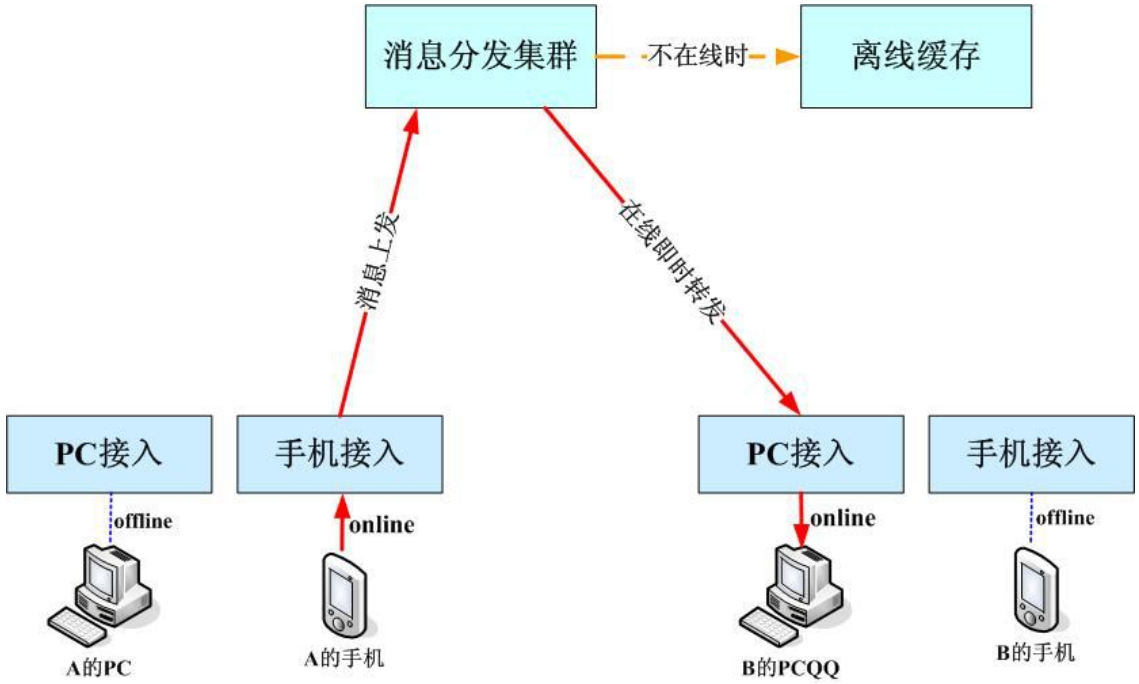
状态摘要Bitmap

关于扩展性(Extendability)设计的启示

- “先不用考虑这么多，目前的设计够用了”
- Design for the future, because it will be here sooner than you think —— 《Unix编程艺术》
- 复杂系统的分层设计
- 初始框架的实现质量

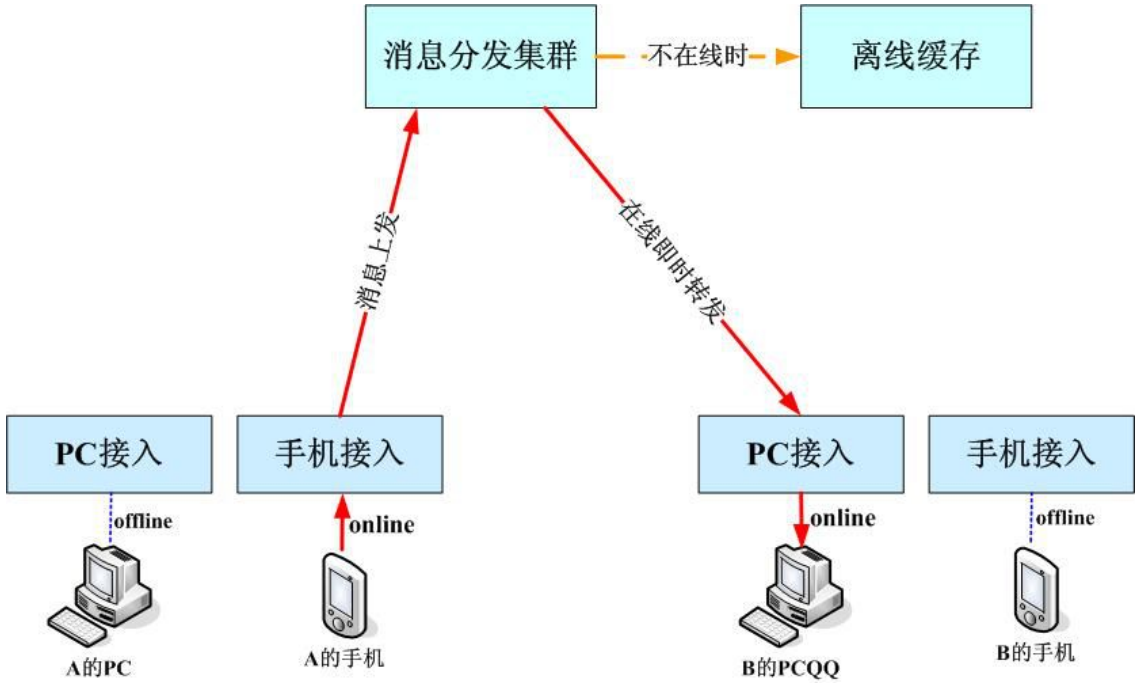
消息系统-单点登录的架构

- “交换机”式转发
- 送达即清除
- 已读未读客户端处理



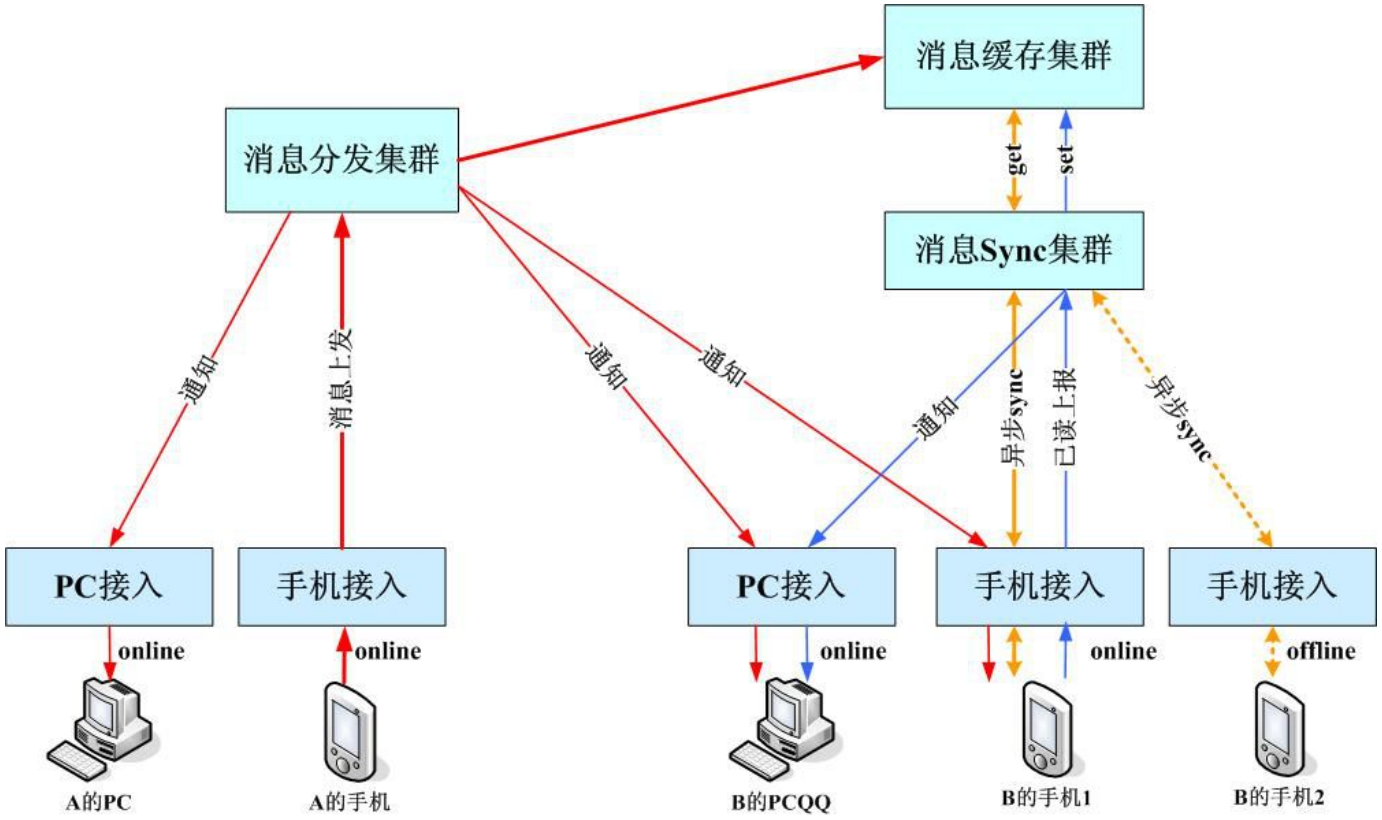
消息系统-单点登录的架构

- 多终端场景下的问题？
  - 漏！消息不连贯
  - 烦！重复提醒



消息系统-多终端消息体验

- 本质是消息处理架构问题
- 异步化处理架构



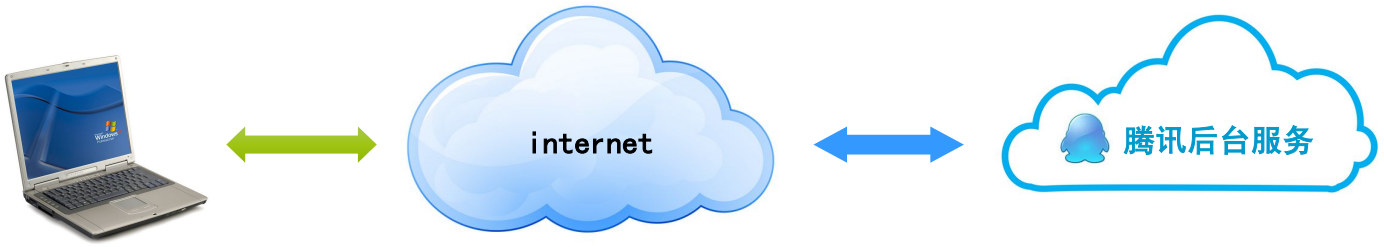
关于异步化(Asynchronization)设计的体会

- 传统IM通信（同步） <---> 移动端服务（异步）
- 基于传输（同步） <---> 基于存储（异步）
- 融合异步与同步的混合架构

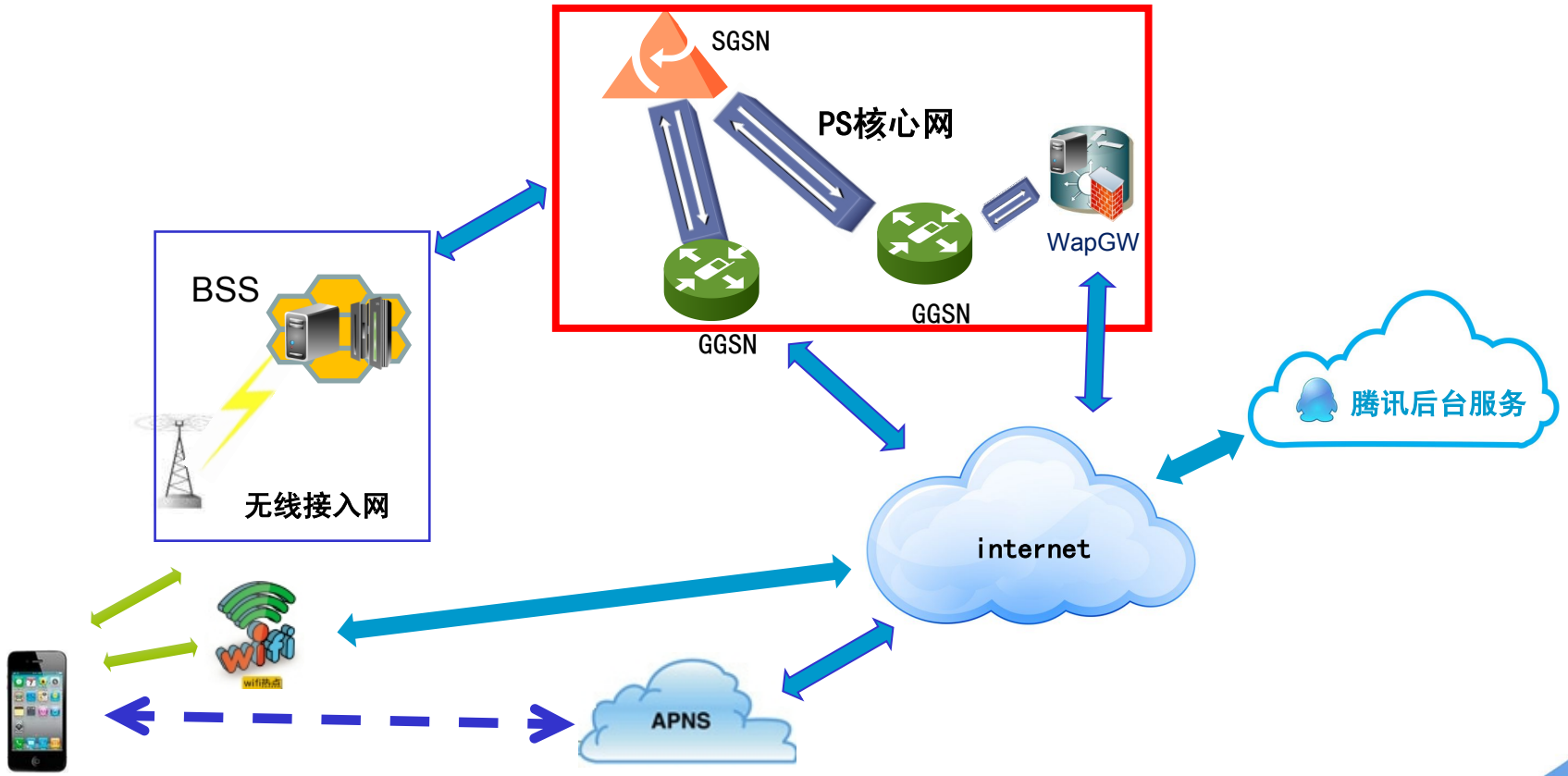
目录

- 从2千万到2亿在线
- 从单点登录到多终端在线
- 从桌面环境到移动环境

PC时代的网络



移动网络的复杂性

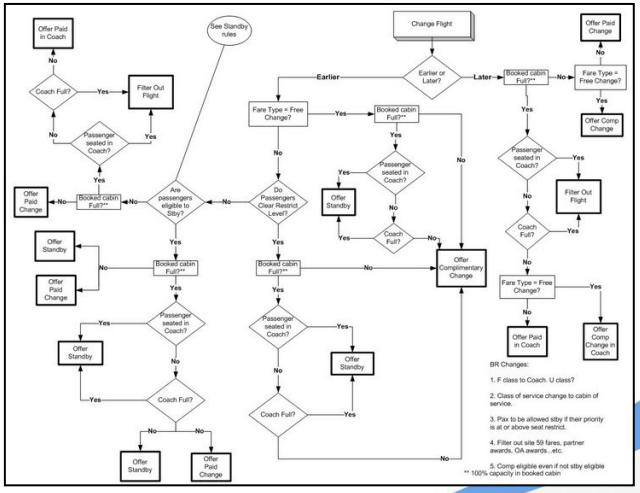


移动环境下的接入层技术

- 适应复杂环境的接入调度
- 域名解析机制优化
- 对网关高穿透性协议设计
- 针对移动网络的传输优化

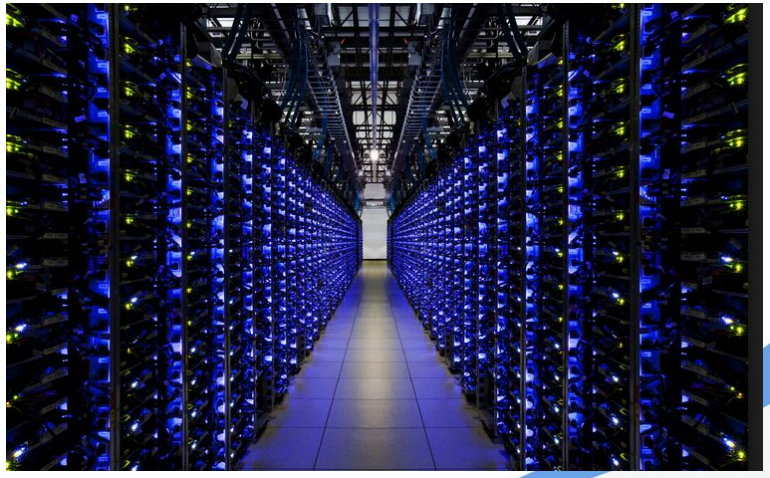
移动环境下的逻辑层技术

- 轻前端重后台模式
- 高效率的开发框架
  - RPC、Actor模型
- 复杂性优化
  - 解耦、隔离
- 云端配置



移动环境下的存储层技术

- 状态云端化
- 通用存储组件
  - 多维结构、高性能、多地分布、自动化
- 前后端最小化传输
  - 单调一致性
  - 统一增量同步机制



总结

- 亿级在线：弹性伸缩架构重设计
- 多终端在线：灵活扩展性/处理异步化
- 移动环境：接入/逻辑/存储全方位优化

- 未来挑战还很多...

谢谢~

有志之士，欢迎加入