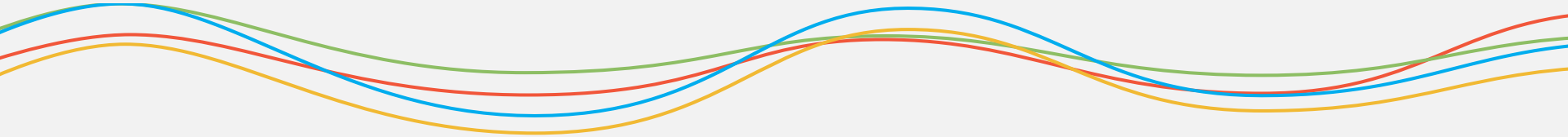
APMCon

映客直播 iOS App 性能优化实践

刘凯 @映客



- 刘凯
- 2014.4 ~ 2015.3 多米音乐，音视频 SDK开发
- 2015.3 ~ 至今 映客，iOS 开发

1、直播大厅的优化

2、直播间的优化

3、私信消息模块的优化

直播列表的刷新

问题：

- 作为 App 的入口，其高可用十分重要
- 用户点击进入直播间后，发现直播已经结束，体验不好
- 主播们及时露出、吸引粉丝的需求

实践方案：

1、多种刷新机制相结合

- 全量刷新
- 顶部刷新，按滑屏的偏移
- 局部刷新，对直播流进行分组，只更新用户当前界面的直播流；便于服务端实施缓存策略
- 刷新的时间间隔可配置

实践方案：

2、用户体验方面

- 用户当前正在滑动屏幕时，延迟刷新当前界面

问题：进入直播间速度的优化

常见现象：

- 用户点击某个直播之后，进入直播间页面较慢
- 直播间内视频迟迟加载不出来
- 主播开播后反映人气一直上不去

问题：进入直播间速度的优化

原因分析：

- 获取视频流时建立网络连接的耗时
- 直播间内业务模块众多，界面初始化耗时

问题：进入直播间速度的优化

解决思路：

- 在直播间大厅预先解析视频流地址，从而加快获取视频数据
- 对房间内部分UI模块使用懒加载方式，如公聊、用户列表等

业务模块划分

直播间内的优化

1. 定时器事件调度

1. 房间内长连接消息的处理

1. 动画展示

1. 主播端的体验优化

统一的定时器事件调度

公聊消息列表刷新

问题现象：

- 热门直播间大量的公聊消息使得列表持续快速刷新，导致主线程 CPU 占用很高，影响整体界面交互

公聊消息列表刷新

优化思路：

- 将处理长连接消息的 SocketIO库中的 block 执行线程设置为非主线程，缓解主线程的 CPU 占用

```
// Going to set a specific on the queue so we can validate we're on the work
// queue
dispatch_queue_set_specific(_workQueue, (__bridge void *)self,
                            maybe_bridge(_workQueue), NULL);

//     _delegateDispatchQueue = dispatch_get_main_queue();
_delegateDispatchQueue = dispatch_queue_create(
    "com.inke.webSocket.delegateQueue", DISPATCH_QUEUE_SERIAL);
```

公聊消息列表刷新

优化思路：

- 实时统计单位时间内收到的消息数目，在性能较低的机型上，动态调整公聊列表的刷新频率

公聊消息列表刷新

优化思路：

- 使用队列暂存消息，每条公聊消息到来时不直接刷新列表
- 批量的 pop 队列中的消息，只保留最近收到的一些消息进行滚动刷新

公聊消息列表刷新

优化效果：

- 热门直播间内 FPS 从最初的10上升至20以上，界面卡顿明显改善

问题：

大量的礼物动画如何进行流畅的展示

优化思路：

- 根据礼物类型，使用不同的队列存放待展示的礼物消息，队列区分优先级
- 使用内存池，缓存点赞和弹幕中执行动画的 view/layer，以重复使用
- 全屏动画都使用 Core Animation动画，取代 Gif 动画

问题：

大量的礼物动画如何进行流畅的展示

测试优化效果：

- 测试环境中，服务端配置自动送礼脚本，对各种送礼动画进行压力测试
- 和其他业务模块结合起来测试，比如开启美颜和连麦等功能，测试整体界面的流畅性

问题：

大量的礼物动画如何进行流畅的展示

测试优化效果：

- 导入线上热门直播间的广播消息数据，进行真实环境的测试

直播端体验的优化

问题：主播直播了很久，好不容易上了热门，发生App 崩溃如何处理

解决思路：

- 开始直播时保存推流地址等信息
- App 崩溃后，再次启动可以继续直播
- 恢复效果上相当于一次切后台的操作

直播端体验的优化

问题：如何应对运营活动时的房间内的消息压力

问题现象：

- 短时间内服务器广播消息压力非常大
- 公聊和礼物消息持续刷屏，主播直播严重受干扰

直播端体验的优化

问题：如何应对运营活动时的房间内的消息压力

解决思路：

- 服务降级，下发限制客户端公聊和礼物消息发送频率的命令
- 限制点赞和部分礼物的动画展示

减轻服务端并发请求方面的优化

问题场景：

- ➢ 用户关注关系的拉取
- ✓ 每次进入直播间，都需要请求用户列表中的关注关系
- ✓ 主播结束直播时，用户都需要拉取和主播的关注关系
- ➢ 推送消息
- ✓ 明星用户开播全量推送时，大量客户端同时拉取大厅直播列表

减轻服务端并发请求方面的优化

解决思路：

- 对用户profile、关注关系等信息进行缓存
- 收到直播推送时，不立即刷新直播列表，延迟到退出直播间后再启动大厅刷新机制

客户端首次启动方面的优化

问题场景：

- 用户安装映客后首次启动App，点击登录按钮出现卡顿

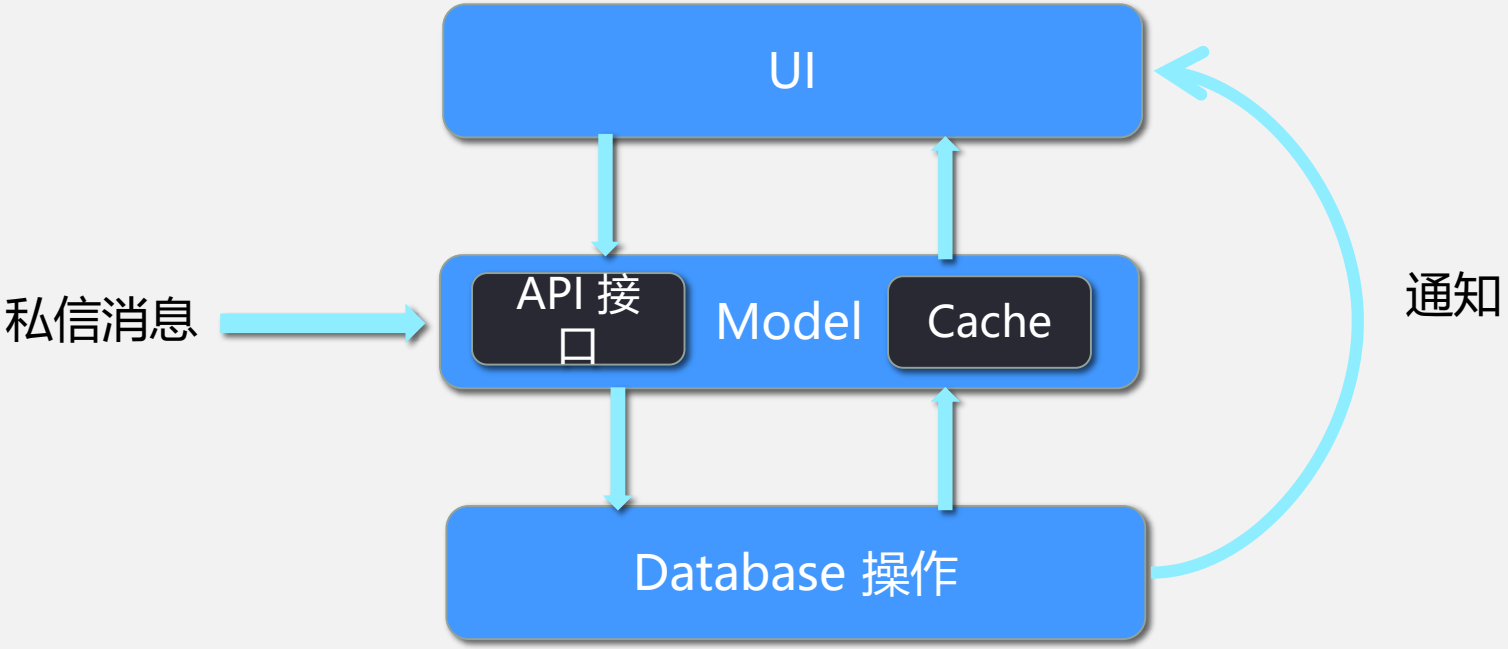
原因：

- 客户端启动时，需要下载大量的图片资源，网络请求和文件操作频繁

解决思路：

- 对部分礼物和等级图标资源进行内置，增量下载更新

私信消息模块的整体设计



优化点：私信聊天页面的打开速度和消息浏览的体验

思路：

- 内存中只缓存少量的几十条数据用于展示，滑屏浏览时，动态更新缓存内容

THANK YOU

kai.liu@meelive.cn