火爆的直播应用，你了解背后的技术架构吗?

技术琐话 5月22日

以下文章来源于IT人的职场进阶 ， 作者Rockets.Luo

IT人的职场进阶

前亚马逊工程师，现58转转技术总监。专注于高质量原创，持续分享疑难技术点、复杂...

直播行业大概在10年多前就开始兴起了，秀场直播和游戏直播是PC时代比较成功的应用场景，直到16年，随着移动互联网的大规模普及，直播行业迎来了真正的元年，成百上千的直播APP出现在大众视野，大概在18年年初，直播答题当时火了一把，那算是直播类应用的第一次全民普及，然后是19年短视频网站掀起的直播电商。回顾直播行业的发展历程，直播类应用在各个领域遍地开花，那么它背后的技术架构你是否了解?

两年前，我参与了一个支持100万用户同时在线、20万并发的直播答题系统的架构设计，下文将以『直播答题』的应用场景为例，带你了解当前疫情下火爆的直播应用背后的技术架构。内容分成以下4个部分:

- 产品功能简介
- 面临的技术挑战
- 技术选型及依据
- 架构设计方案

01 产品功能简介

直播答题能在当时成为风口，得益于它的玩法足够简单，用户教育成本几乎为0。简单来说就是：全民在线PK、10秒答题、超时或答错即退出游戏、成功答对所有题即可平分奖金，俗称在线版的开心辞典。

在了解系统设计方案和架构之前，先看看直播答题应用有哪些核心功能？下面是APP端的几张产品截图:

1、答题以活动的形式展开，每场活动都会预先公布直播开始时间、答题总奖金、红包雨奖金等信息。

2、活动时间到后，用户就可以进入直播间，看到实时的视频流，有专业的主持人进行串词口播，场控人员会配合主持人同步进行发题、公布答案等操作。

3、题目下发后，用户有10秒的作答时间，答错或者超时即退出游戏，如果用户有复活道具在答错时会自动使用，用户能继续进行答题。

4、为了留住答错用户，活动期间有多场红包雨，用户点击屏幕就有概率抢到。

5、活动期间用户可发弹幕，同时会有弹幕滚屏轮播，以营造热闹的直播氛围。

6、其他功能：邀请新人、活动榜单、奖金提现等。

其他直播类应用在产品功能上和直播答题类似，基本也是这两类：

1、直播的基础功能：连麦互动直播（支持多码率、多协议，多主播同框）、美颜特效、弹幕、IM聊天、点赞、屏幕共享等功能性需求，以及防盗链、涉黄涉政鉴别等非功能性需求。

2、应用本身的个性化功能：比如答题场景中的发题目、作答、公布答案，电商场景中的商品展示、一键下单购买，网红直播场景中的礼物打赏。

02 面临的技术挑战

当时我们做直播答题应用时，面临以下技术挑战：

1、音视频处理及传输：涉及音视频编码、实时美颜、视频推流、CDN加速分发、终端适配和播放，流量统计等诸多技术点，而且我们当时的技术团队没有专门做音视频方面的专家。

2、高并发请求：参考了冲顶大会等答题竞品的用户量级，预计会有100W用户同时在线，1秒内有20W用户同时答题；1秒内单个用户可触屏发起4-5次抢红包请求，并发最大可达到500W QPS.

3、高带宽压力：按照标清视频的标准，观看直播的码流至少为1Mbps，如果100W用户在线，光视频流的出口带宽能达到976.56G bps。1条弹幕可达到130字节，1秒要滚屏20条弹幕，如果需要同时推送给100W用户，弹幕的出口带宽也将达到19.37G bps.

4、高计算压力：1道题目的对错判断涉及到答案比对、复活道具的使用、判断用户是否创了新纪录，同时还涉及一系列的反作弊策略（比如前面的题目答错了将无法继续作答），在主持人口播公布答案的瞬间，如何快速完成100W用户的答题结果计算?

5、资金流的正确性和安全性：单个用户最多抢3个红包如何不多领？财务上如何保证答题奖金、红包奖励不出现1分钱的误差?

6、低延迟性要求：直播场景下如何整合视频流和业务数据流，做到声音、主播画面和题目同步，以保证用户体验?

7、对商城交易业务的低干扰：直播答题仅作为商城的一个运营活动，核心目标是导流，它依托商城原有的用户体系，运营系统，大数据系统，提现通道等，如何做到对商城现有交易系统的低干扰?

可见答题这种泛娱乐的直播场景还是有挺多技术挑战的，它取决于直播应用的用户量级和业务流程。就算是直播电商这种低频交易转化的场景，要是李佳琪在带货，同样也会面临瞬时抢购的高并发挑战，所以**架构设计时必须考虑业务最高峰时的压力，并且系统的各个环节必须具备动态可伸缩的能力**。

03 技术选型及依据

一. 音视频处理及传输的方案选型

音视频处理及传输，因为技术团队不具备这方面的能力，所以当时调研了各种云厂商的付费解决方案，最终采用了**腾讯云的直播解决方案**。主持人侧：通过演播室的专业摄像设备，搭载腾讯云提供的obs推流软件，即可进行视频录制和推流。用户侧：APP端集成腾讯云的SDK，动态拿到推流地址后即可观看直播。

二. 业务数据流的方案选型

业务数据是指除音视频以外的，和答题应用场景相关的数据（比如题目、答案、弹幕、红包等）。腾讯云提供了两种可选方案:

1、题目预先设置好，直接由腾讯云的SDK通过音视频通道下发，打入直播流中。
2、让题目先通过腾讯云的IM通道快速送达观众端APP，在观众端先缓存下来，等待播放器通知了预期的NTP 时间戳之后，再把题目显示出来。

腾讯云的这两种方案都可以提供“音-画-题”的完美同步，但是存在以下局限：用户的答案必须以HTTP请求方式汇总到答题服务器，这一块必须自己开发，另外公布答案、抢红包、弹幕这些业务腾讯云系统都是不支持的，在底层通信通道上仍然需要自行开发。

考虑上述局限以及业务的多变性，我们最终打算**自研业务数据流通道**。这样视频流和业务数据流会分两个通道下发，因为业务流相对视频流的数据量很小，只要能保证业务逻辑的处理速度和业务数据的下行速度，

“音-画-题”的延迟是可以接受的。毕竟当时已经是4G时代，如果用户侧的网速不行，视频流可能都无法正常观看了。

为了做到业务数据流的独立传输，需要实现一个长连接、高性能的网关服务器（支持100W用户同时在线，20W并发答题，弹幕实时推送等要求），**我们的技术选型是：Netty、ProtoBuf、WebSocket**，选型理由：

1、Netty：Netty是当时最流行的高性能和异步NIO框架，直播答题的业务场景中，涉及到题目下发、弹幕、下红包雨等非常多的推送场景，而且一场答题活动中，客户端和服务端的通信频繁，长连接比短连接在性能上更优。

2、ProtoBuf：作为客户端和服务端的数据交换格式，PB是一种效率和兼容性都很优秀的二进制数据传输格式，在码流和序列化速度上明显优于JSON、XML、hessian等主流格式，同时支持向前向后兼容以及各种主流语言。

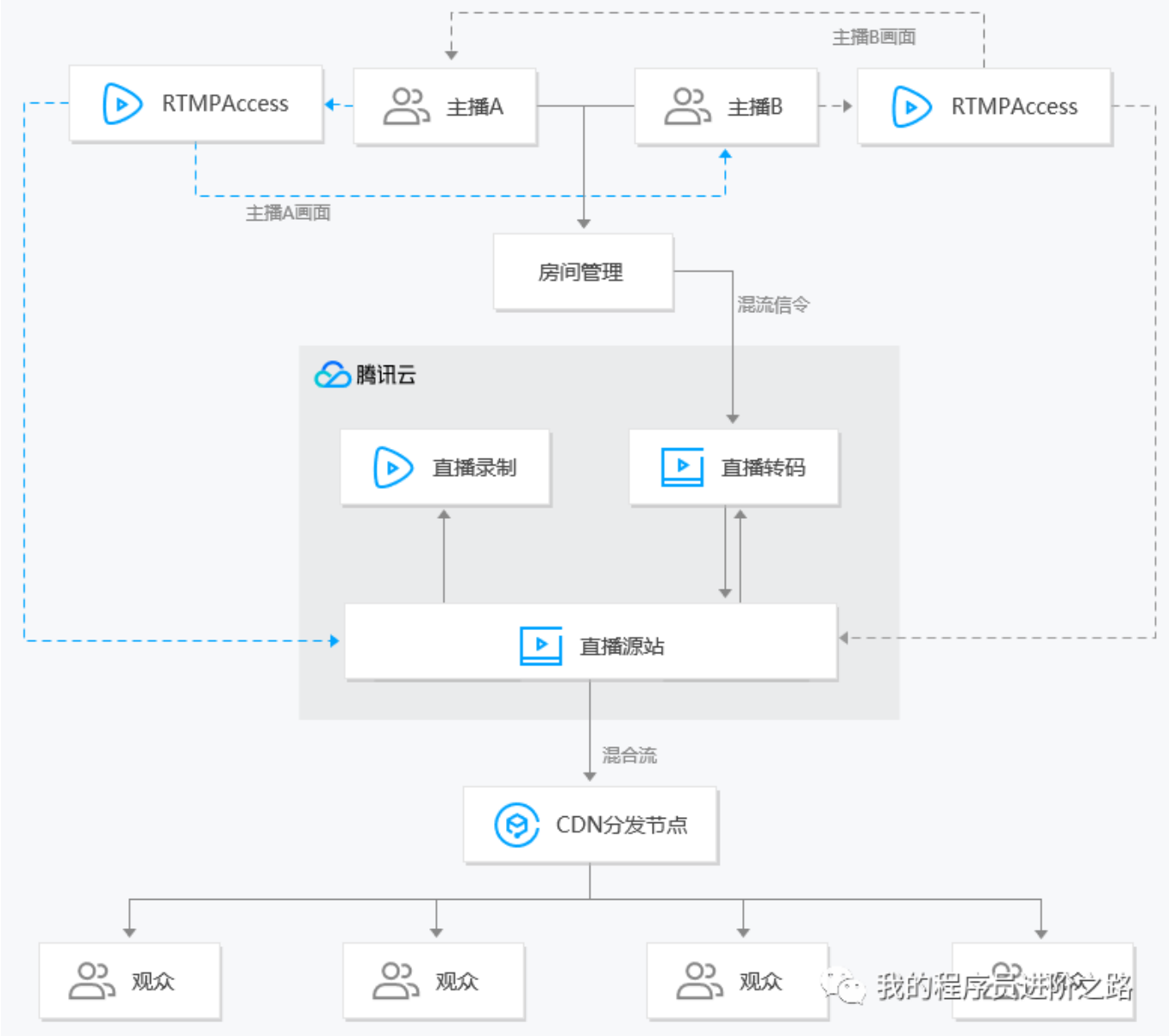
3、WebSocket：是 HTML5 一种新的协议，用来实现客户端与服务器端的长连接通讯。

三. 服务端的部署方案选型

前面提过，直播答题仅作为商城的一个运营活动，它依托商城原有的用户体系，运营系统，大数据系统，提现通道等。现有的商城系统部署在我们自建的机房中，为了降低对商城现有交易系统的低干扰，**我们采用了『私有云+公有云』的混合部署方案**。将高并发的答题系统以及它所依赖的缓存、MQ等公共组件部署在公有云上，方便弹性扩展，同时降低对商城交易系统的流量冲击。

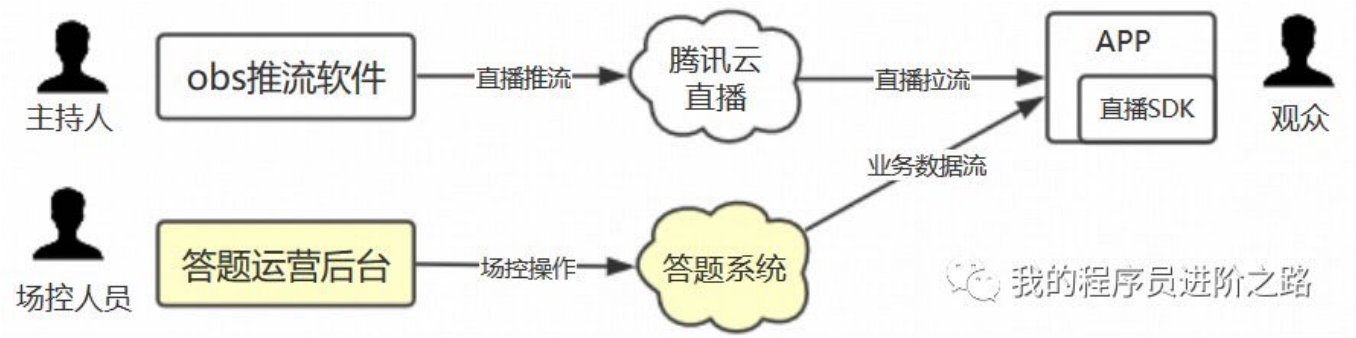
04 架构设计方案

一. 音视频直播架构



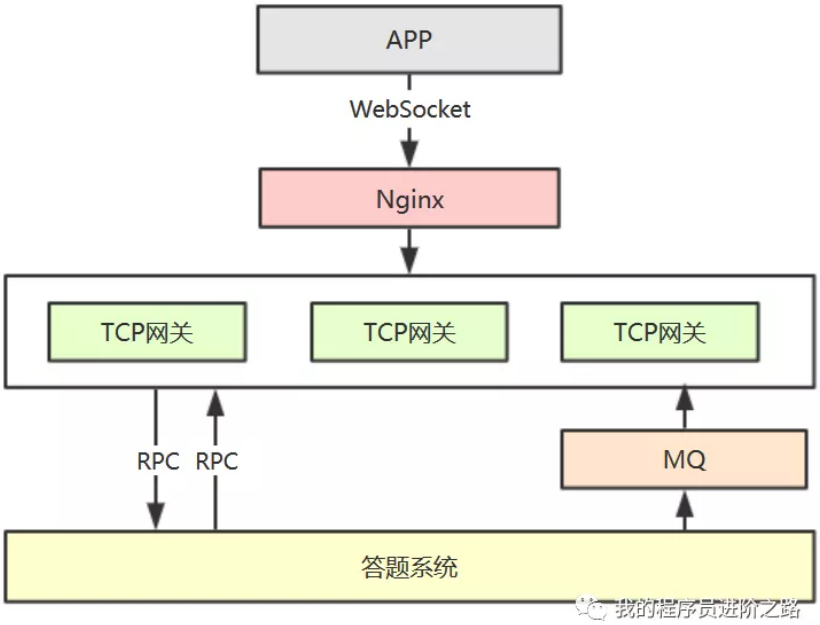
上面是腾讯云直播解决方案的架构图，其他云厂商（比如阿里云）的直播解决方案，技术架构类似。感兴趣的同学可以直接去腾讯云官网上详细了解，这里不做展开。

二. 数据流方案



音视频流采用了腾讯云的直播解决方案，而业务数据流（活动、题目、答案、弹幕、红包等）则采用了自研的长连接方案。架构图中的答题系统和答题运营后台也均为自研。客户端会分开处理两个通道的数据，以控制用户交互上的变化。

三. 基于TCP长连接的通信架构



上面的通信架构用于业务数据流的传输，流程如下：

1、客户端使用websocket与服务端进行通讯，用户进入答题直播间时建立连接，退出直播间时断开连接。

2、Nginx对websocket做负载均衡。

3、TCP网关基于netty实现，用于维持长连接和转发业务请求，不负责具体的业务逻辑，它和下层业务系统（答题系统）通过RPC接口进行交互，主要考虑后续其他业务可以复用TCP网关层，所以将业务下沉。客户端和网关之间通过心跳机制保证连接的有效性以及检测僵尸连接。

4、消息推送（比如弹幕、下发题目、公布答案等诸多场景）由下层业务（答题系统）通过MQ通知TCP网关，再由TCP网关推送给客户端。

四. 长连接通信中的数据传输格式定义

通信架构中另外比较重要的是数据传输格式的定义，客户端和TCP网关之间，TCP网关和答题系统之间均采用的是protobuf格式。下面分开说一下比较关键的几个格式定义。

4.1 客户端请求消息的格式

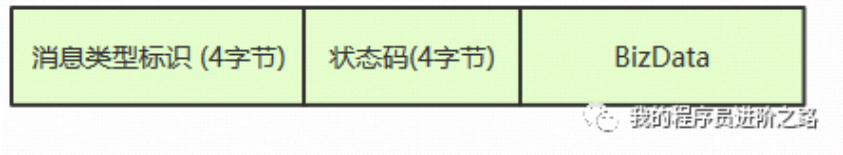
1、消息类型标识：-1表示心跳消息、0表示用户验证消息、>0 表示业务类消息

2、用户ID：用户的唯一标识，在前置的登录流程中已经获得

3、Token：用户登录APP后的授权令牌，在前置的登录流程中已经获得

4、BizData：真正的业务数据，protobuf序列化后的byte数组，由下层业务自行定义更具体的格式

4.2 客户端响应消息的格式



1、消息类型标识：和请求中的消息类型保持一致

2、状态码：0表示处理失败，1表示处理成功

3、BizData：真正的业务数据，如果状态码为0，该字段为4字节的异常码（100表示token验证失败，101表示请求业务层失败），如果状态码为1，该字段为业务层返回的protobuf序列化后的byte数组，同样由下层业务自行定义更具体的格式

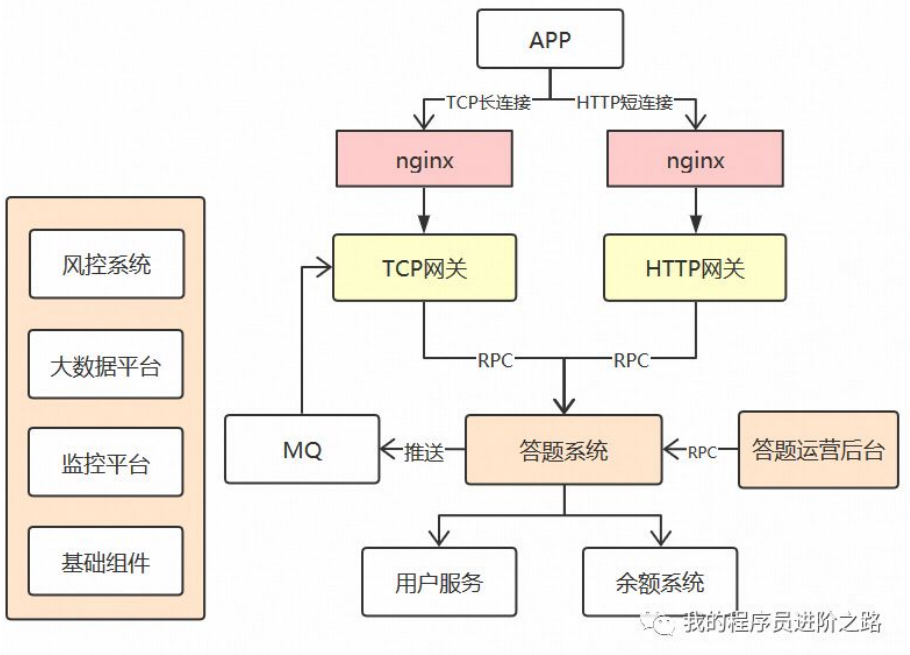
4.3 业务数据的ProtoBuf格式定义

```
message Message
{
   MessageType messageType = 1; // 消息类型，枚举值
   string sequence = 2; // 消息序号

   Request request = 3; // 请求类消息
   Response response = 4; // 响应类消息
   Notification notification = 5; // 推送类消息
}
```

这里的格式设计比较巧妙，通过Message顶层消息把Request、Response、Notification这3类消息包装起来，并且定义一个MessageType枚举值，用来表示消息类型，另外sequence字段表示消息的唯一序列号，需要保证发送端内唯一，这样发送端收到Response后可以进行匹配处理。有了上面的统一消息格式，答题系统就可以针对不同的MessageType定义不同的消息处理器，配置好映射关系后，即可实现消息路由。

五. 系统总体架构

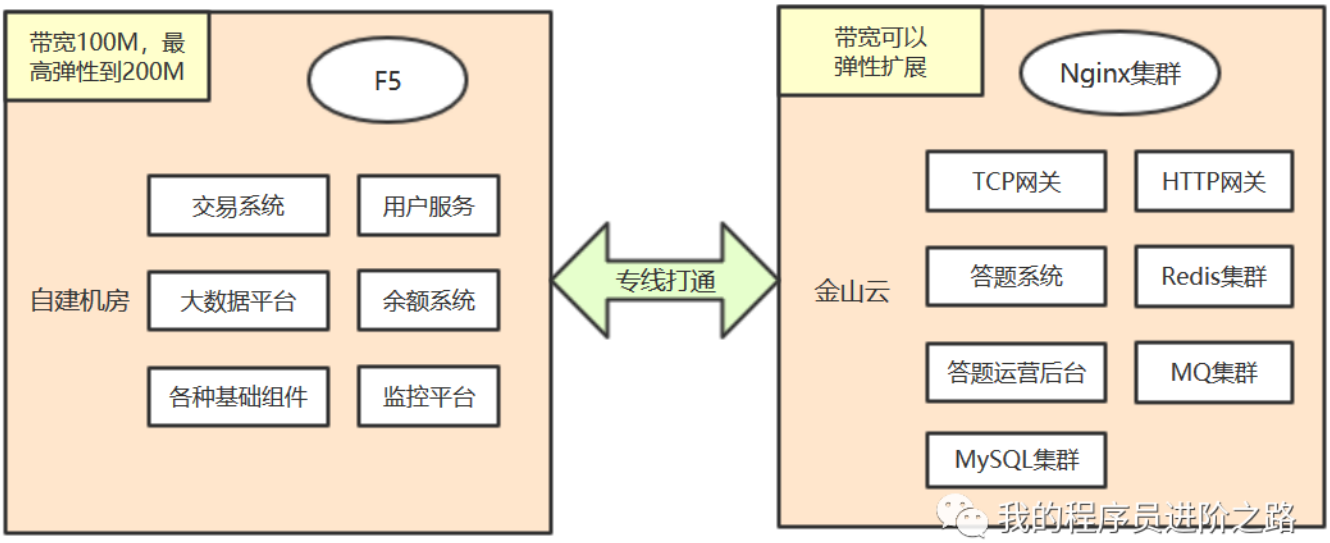


1、网关层：架构上同时采用了TCP网关和HTTP网关，TCP网关上一章节已经讲解，它主要用于维持百万用户同时在线，以及答题期间的实时数据交互（包括加入直播间、推送题目、答题、推送答案、抢红包、弹幕推送等）。HTTP网关为APP端提供Restful接口，用于低频的数据请求，比如活动首页、排行榜、个人奖金明细、新人邀请、分享等场景。

2、答题系统：Dubbo实现，最核心的业务服务，同时面向C端和B端系统提供RPC接口，包括活动管理、题库管理、直播间管理、以及面向C端的高并发接口（比如加入直播间、答题、抢红包等）。这个集群用到了很多高并发设计的通用方法：比如服务的横向扩容能力、多级缓存、异步化、限流等，后面的章节再做具体介绍。另外，通过MQ的事务消息+对账机制保证资金流在答题系统和余额系统的一致性。

3、答题运营系统：后台系统，供运营人员和直播时的场控人员使用，可以管理活动和题目，以及直播过程中的各种业务操作（比如配合主持人的口播下发题目、公布答案、发红包等）。

六. 部署架构



直播答题的在线用户量以及并发量远超过商城的交易系统，为了减少对主交易流程的影响，采用了上述『私有云+公有云』的混合部署方案，自建机房和云机房之间通过网络专线打通。直播答题有关的应用服务器、存储服务器、网络带宽都在云端，可以根据流量监控做到快速扩容，随着运营计划的调整，也可以随时增减服务器，灵活性高。

七. 答题系统的高并发设计方案

7.1 答题接口的高并发设计方案

答题接口的并发预计20W QPS，在说设计方案之前，先简单列一下答题接口的判断逻辑：

> 1、需要判断答题活动是否仍在进行中
> 2、需要判断用户是否已经答错出局
> 3、需要判断用户输入的题目是否是当前正在作答的题目
> 4、需要判断用户上一道答对的题和当前这题是否连续
> 5、需要判断用户作答是否已经超时
> 6、需要判断用户的答案是否是正确答案
> 7、如果答错了，需要判断用户是否有复活卡
> 8、如果有复活卡，需要判断前面的题是否用过复活卡了

除了上述判断逻辑外，还有一些写操作，比如更新每个答案选项的选择人数，更新用户的答对题号或者出局题号，更新复活卡的使用记录等，总的来说，答题是一个读多写少的接口。为了应对高并发，我们采取了以

下技术方案：

1、答题接口的所有逻辑只操作缓存，不操作数据库。采用Write Behind Caching的更新模式（即只更新缓存，在答题结束后再异步批量更新数据库）。

2、多级缓存：本地缓存+Redis缓存。同时在答题活动开始之前，会先将活动的配置信息，题目，答案等所有静态数据都预热到本地缓存中。

3、Redis 一主多从 + 哨兵的部署架构，确保高并发和高可用，同时分业务模块配置了多套Redis实例（用户、弹幕、直播作答、奖金榜单）做进一步分流。

4、调整判断逻辑的顺序，上面提到的8个判断逻辑，其中前4个都属于安全校验，因为客户端已经通过交互做了拦截，如果用户不采取作弊手段肯定都能校验通过。所以我们将第5、6、7步逻辑做了前置，通过后再进行前4步校验，这样大大减少了计算量。

7.2 答题结果推送的设计方案

如何在主持人口播公布答案的瞬间，将答题结果推送给几十万作答用户？因为用户的作答结果有非常多的状态：有答对的和答错的，有使用复活卡和未使用复活卡的，有出局的，不同状态下APP端的交互均不同，完全是一种依赖服务端的有状态推送。为了解决这个并发计算问题，我们采用了一个比较巧的设计方案，很好地将有状态推送转变成了无状态推送：

1、将答题结果的计算前置到用户提交答案的时候同步完成（上一节的方案），这样相当于将瞬时的计算压力分散到10秒作答时间里了。

2、用户的作答结果也在第1步计算完成后立刻推送给用户了，不用等到主持人口播公布答案的瞬间，否则的话仍然涉及到并发读取作答结果以及有状态推送，对存储服务器以及带宽的瞬时压力仍然存在。

3、但是作答结果如果提前推送给客户端必然存在安全问题，黑客抓包就能提前知道是否答对了，可以利用批量账号进行作弊。为了解决这个问题，我们对作答结果采用XOR进行了对称加密，同时将秘钥延迟到公布答案的瞬间才下发，而且每道题的秘钥均是随机生成的。这样就很好地解决了安全问题。

4、公布答案的瞬间，我们只需要推送一个非常小的数据包给客户端即可，而且这个数据包所有用户都是一样的。

7.3 抢红包接口的高并发设计方案

屏幕下红包雨，用户触屏点中红包就直接拆了，所以抢红包接口的并发非常高，1秒内单个用户可触屏发起4-5次抢红包请求，按照百万用户在线，并发最大可达到上百万QPS，下面说下技术方案：

1、红包金额提前计算好并加载到redis中：在创建活动时，运营可配置红包总金额以及红包总个数，系统在创建活动时就会提前计算出各个红包的金额并存储到redis中，相当于计算前置。

2、客户端限流：在接口设计时，我们就预埋了一个限流因子参数，可由服务端动态控制客户端的限流比例，在通知客户端抢红包的接口中，我们根据当前的在线人数以及红包总个数动态算出限流因子，控制最多只有10W QPS的请求能发到服务端。

3、服务端限流：根据服务器数量以及红包总数量提前计算出每秒令牌的生成数量，基于guava的RateLimiter进行二次限流。

4、前面3步基本把并发降下来了，再通过lua脚本保证原子性即可，抢红包的结果也是在活动结束后再异步刷新到数据库。

7.4 其他高并发的优化点

1、Redis缓存的对象要尽可能简化（用不到的字段不要存），key的长度要尽可能短（高并发下的瓶颈在于IO），善于利用pipeline组装多个命令（但是命令个数不能过多）

2、各种连接池和JVM参数的调整：涉及redis连接池、dubbo的线程池、JVM内存大小，可以在压测环境下找到合理值。

3、答题系统可水平扩展（scale out），同时通过dubbo的分组配置将ToB和ToC接口进行隔离部署，避免相互影响。

最后，关于直播架构再简单总结下:

1、音视频编码和传输，这些基础性的直播功能，除非公司有钱有实力，否则建议直接用腾讯云或者阿里云的解决方案（斗鱼、蘑菇街这些知名的直播应用都还用的腾讯云）。

2、架构设计重点放在应用本身，根据直播应用的用户量级和业务特性先确定通信架构（长连接还是短链接，或者两者混用）。

3、要根据业务高峰来做方案设计，如果是高并发场景，要把高并发当做一个系统性问题去对待：从客户端到服务端，从网络带宽到部署架构，甚至产品设计等各个维度全盘考虑，要抠各种细节。

往期推荐

支付宝全局架构师曹刚：为12亿用户设计架构是什么体验?
季琦：林彪给我的启发！创业者要多看人文的东西来扩大视野！
马蜂窝支付中心架构演进
从新型冠状病毒想到清单革命
银行系统 - [账户体系]演进历程梳理及账户层次解析与关键点