主题：平行论坛9-中国技术开放日之移动应用开发技术前沿

时间：2016年12月9日上午9：30

地点：广州四季酒店5楼会议室2

主持：黛拉

内容：

主持人：广州的朋友们大家早上好，在座的应该都是工程师，广州的工程师们大家早上好，欢迎来到中国技术开放日之移动技术开发前沿这个论坛，这是由IDG和极客邦联合主办的活动，主要为广州的开发者提供交流的平台。

先提一个小问题，在场谁听说过极客邦科技的吗？那在场谁听说过InfoQ？谁是从“移动开发前线”这个公众号了解到这个活动？我先介绍一下极客邦科技。

极客邦科技是为技术人员和企业提供成长服务的。InfoQ是他旗下的一个品牌，同时还有StuQ和EGO两种服务。EGO是针对高端用户群体提供的一个小型封闭的社交活动，StuQ是为企业定制系统化的课程培训，可以想象是IT教育当中的斯坦福。

自从InfoQ进入中国以来，我们一直为开发者们提供线下的高质量技术交流活动，这个形式我们也将以“中国技术开放日”这个品牌持续运营，希望通过我们的努力，能够让更多的开发者有更多的机会结识业内的技术大牛、收获更多的实践经验。这个活动分为国内版和国外版，国外的活动目前正在硅谷展开。

今天的活动，我们邀请了来自微信、淘宝、百度和唯品会的四位嘉宾，来给我们讲解移动开发技术的最新实践。

接下来，我们欢迎来自微信的工程师，来讲讲微信Mars- 移动互联网下的高质量网络连接探索？首先问一下周工程师，微信Mars是什么？

周志杰：大家好，我是周志杰，微信Mars是我们在微信终端中使用的跨平台的组件，今天的分享中，我会做详细的介绍。Mars是因为我们主要提供可靠的网络通道，因此我们就用Mars，就是地球跟火星之间，我们微信的标志也就是星球的形状，我们就命名为Mars，意为地球与火星的连接。

主持人：接下来的时间交给周工程师。

周志杰：谢谢大家，在接下来的30分钟时间里面，我将给大家介绍一下微信Mars，并且着重介绍微信Mars在探索移动互联网下的探索，希望这次介绍希望大家对我们微信Mars有一定的了解。

先简单介绍一下，做一个自我介绍，我2010年进入腾讯，2011年初的时候转战到微信版本中，是微信的塞班版本，2012年末的时候我转战了新方向，就是微信跨平台开发，经过4年的发展，我们微信目前有3个重要组成部分，第一是网络、第二时间、第三监控。网络是我们微信内部的大部分业务、文本、语音都是在网络里面组成的。

时间就是传输；统计监控就是提供了一个能力，提供了验证，验证跟监控自己的统计渠道率，可以保证数据是可靠的这个数据我们搭建了微信终端质量平台，使我们数据不止是说，只能从微信后台中看到，而且可以看到微信前端数据营运情况，最后就是说经过这段时间的发展，我们希望跟行业内做更多的交流，从事促进移动互联网的发展，因此我们提出了Mars开元项目。

今天的分享会分成5个部分，第一部分会介绍Mars的概况，接下来三个部分是着重介绍Mars的STN的模块，第三是Mars的开元计划，首先是Mars的情况。大家都知道2012年的时候我们微信只有安卓和IOS两个版本，但在2012年的时候微信处在高速发展的时期，同时用户量在急遽增加，我们微信服务器框架也在慢慢摸索和成熟过程中，这样导致了我们也碰到了不少问题，因为多平台的实现不统一，导致我们服务器要做擦屁股的动作。网络只是一个软件里面很核心的部分，如果网络出现问题，如果在微信版本上面出现问题，这对微信是很大的伤害第三我们想开展很多基础方面的研究，第四大家可以知道，我们微信现在发展到现在有6大平台，有Mars平台、windows平台等，为了在技术研究上做的更好，我们有跨平台的主线，微信Mars就是源于这样的主线，这是Mars的大体框架。

首先Mars是建立在6大平台的基础上，基础于这个平台我们构建了技术库，包括携程，第二个部分这是Mars开元的核心就是STN的模块，这一部分我们在后面有相对详细一点的介绍。

第三是日志，这个提供的能力是在运行时的时候，而不会引起APP的卡顿这方面我们做了很多时间，大家可以看我们的文章，第四部分就是SDT，就是网络检测能力，可以供大家使用，基于这四个部分及我们构建了业务的逻辑。

微信现在还没有开放出来的APP，已经在使用Mars项目。Mars来源于微信，未来Mars将回归于微信。目前我们我们的Mars慢慢回流到微信当中，目前有上千万的用户在用，未来全平台都会用这个代码，我们微信平台提供了很多微信的验证，就是说明我们的Mars经过微信的验证，并且可以非常可靠。

刚才介绍Mars的时候，我们提到了STN模块这里着重介绍STN模块是怎样的内容。刚才说到是细腻传输网络，是网络解决方案，现在各个平台上已经有很多开元的解决方案，跟其他开元的解决方案相对比有怎样的特点？如果要做更深入的研究跟优化的话我们必须在收费层提供更灵活的能力。

STN目前适用于小数据的传输，至于图片、视频等大数据传输，因为大数据的传输在设计上跟小数据网络有很大的区别，因此是独立于socket的模块，我们的STN究竟能干什么？有三个主要特点，第一提供两种网络通道的能力，长连接和短连接的能力，长连接是很有作用的，可以长连接上面可以探索到用户，可以在长连接上做对性能要求比较高的请求，短连接我们做的是一来一回的请求模式。

第二个特点是STN中比较核心的一个点，就是STN是在微信使用四年多的网络主线过程中而引起的开元，这意味着我们微信在这个过程中针对移动互联网做的很多移动以及网络下的问题探索，因此这是一个贴合移动互联网的网络特征的网络解决方案。

第三有很多适用于移动平台的特性，我们微信有一个前后台的概念，还有休眠、省电、省流量的机制，我们STN中还包含了APP中包含的东西。

最后我们提供了数据监控能力，因为微信也要用数据监控，我们产生了很多连接数据，网络的请求数据我们会提供给上层。

我们刚才提到MarsSTN是网络解决方案尤其是适合于移动互联网的解决方案究竟做了怎样的内容，今天我们分享两个点，今天的分享主题是高质量的网络连接，我们看一下网络连接上的思考设计，首先网络连接是一个网络操作，它有什么可设置的点其实有端口还有组合和排序，因为我们操作有很重要的连接参数。

超时大家有没有想过，你可以做很多种选择，但不同的选择在不同的场景下会做什么影响？第一连不通场景，不管做什么操作都差不多我们服务器出了部分故障，或者说我们的故障被人劫持了，因为这是部分的故障或者部分的劫持，我们可以通过更换IP端可以查找到可用的资源。

第三是基站繁忙连接信号弱、丢包率高，我们在每一个角落都可以发生这种操作，这个时候你希望你的操作是长的还是短的？当然是长一点的，从而增高成功率。

我们刚才介绍了这几个场景我们总结一下，第一就是可用性、第二就是网络敏感性、第三就是用户体验。

我们是怎么做的？我们的协议是基于TCP层的，提供了很多连接超时的机制，我们看一下ios中，在ios中的话，我们可以看到其实它做了很多机制尝试，有这样一个特点，首先在ios中做了优化，其次是递增的间隔，第三是总的时间，就是connect操作失败，已经过了一分钟，这一分钟我们只是尝试了STP端口资源，这是我们连接超时的选择，这里并不是说STN中的选择是最好的，可能有一些应用需要更高的敏感性，可能有不同的选择，我们之所以选择10秒，首先是因为占用到前面，其次的话就是从10秒之后，安卓平台需要经过5秒发起下一次的重启，有一个长一点的无效等待时间，不同的应用有不同的选择，这里只是分享一下关于设计方面的思考。

刚才讲的是连接超时，我们超时连接就是为了找到可用的端口资源，我们对比一下，并发连接与串行连接的可用性，如果串行连接的话，是要一个一个找，其次在选择上是有肯定的地方有没有一种连接机制可以方便使用？就是符合连接，其实这个方案很简单，一开始的时候，发起对IP的访问，第四秒的时候没有成功的话，以此类推。

这是在统一的情况下进行访问的我们在更快的，在正常的情况下不增加服务器负载这样我们的成功率可以得到不少的提升。

除了刚才提到的建立连接方面的策略我们还有其他的策略，这里不一一讲了。

刚才讲的建立连接，高质量的网络连接不仅只包含建立连接的部分，还包含着维持连接的部分，其中很重要的参数就是读写超时设计。我们使用的是TCP，它在不同的分层上都提供了可靠机制，提供了ARQ方案，这种方案是毫秒方案。

TCP方案在安卓平台上，这个超时指的是读写超时，这是安卓平台的方案，在前面的时候会更积极一点，第三就是IOS平台方案我们可以看到不管安卓平台方案还是安卓平台方案，都在移动端进行了优化，我们是不是应用层，不需要设计读写超层。

我们在应用层的超时处理机制是什么样子的？就是很简单，这样子带来的好处，首先是可以减少等待时间越到后面的话，隔了很久才可以传，第二就是切换连路。

我们有其他的并发因素的影响，这是总的读写超时，但我们可以看到，这个完全满足不了高性能要求，因为这里面包括了很多主观因素，服务器约定的最大耗时也是纠结的值，如果我们涉及到第三方，可能这个服务器号没法确认，我们按照步骤，在每一个步骤上需分开，首先是手包超时，包含几个组成部分，这样子就少了服务器回包的服务成分，包含了很多主观因素。为了使性能更好，我们用了动态超时，我们收到了手包之后，已经确认了从客户端到服务器以及服务器回到客户端整个链路是通的，这样子使包包超时的时候，主观因素就没有了这样我们就可以设定一个较小的固定值，从而有一个固定值，而没有主观因素。

第二动态超时，我们即使引用了前面那些超时，但还是没有性能不好的因素，我们在使用的过程中，已经有很多的请求要完成。

最后就是发送超时，就是我们的请求包发出去之后，到了系统以后，系统里面有多少数据是收到发送超时。

我们刚才介绍了STN中如何做高质量互联网下的探索，未来说一下Mars未来开元的计划和未来运营的方式。我们预计在春节前开元，预计12月底，但具体的公布时间，我们以微信的公众号的时间为准，我们的运营方式的话，就是说我们把这个东西开元出去，绝对不会说开元出去之后，不管不顾独立的东西我们的微信里面将持用Mars代码，并且微信有任何更新都同步在Mars中，Mars中有好的也回流到微信当中，希望未来可以得到更多的支持。

未来发展上我们也讲了我们希望跟行业进行忙，更多的交流促进移动网络的发展，我的分享就到这里，谢谢。

提问：我问一下你这个纯粹是STN解决的方案吗？

嘉宾：STV层的话，但是并没有完整的包含STV方案，在STV方面，我们也正在做这方面的尝试，目前的话还是没有。

提问：就是我用这个的话，需要在上面做协议的分装？

嘉宾：对。

提问：我对STN不是很了解，我想问的是，如果之后要使用你们Mars系统，是不是服务端也要做？

嘉宾：其实微信Mars里面我们也会提供相应的技术，我们开元的是终端的首先我们开元的项目是跟业务无关的，其次服务端的话，是协议有不同，其他的跟日常用的差不多。

提问：现在APP已经做了一段时间了，如果要用Mars的话，是不是也要做se（音）？

嘉宾：只需要我的数据包上去，你的不需要改变。

提问：我想问一下我们现在，分享一下微信那边做网络优化的样板？就是说如果在弱网络下面，通常有哪些原因导致网络是通的，但连接不成功，另外Mars系统还有哪些网络是现在没有完成的？

嘉宾：就是说我们弱网络情况下，为什么网络通的，但连接不成功，这是移动互联网，移动手机跟基站之间的问题，就是基站资源是有限的，可能你想不到资源，就是繁忙的情况；还有就是无线通讯，如果基站跟你的距离相对较远，或者你快速移动，基站不断切换过程中，可能导致有网络但你连不上，还有一个就是说丢包率高的情况下，就是手机跟基站之间有丢包率的情况，也是在移动互联网下经常有的问题。

第二个问题，其实刚才那位朋友已经提到了，我们现在提供的是一个不完备的STVP的方案同时在移动互联网效率上面的探索，我们还在进行，我们还会研究谷歌方案，因为大家都知道，CTB层有比较久的历史，最开始的设计是基于有限网络，其实里面谷歌也推出来，其实我们也在我们少部分的业务中不停做这样的尝试，就是这样子的。

提问：请问一下，可不可以说Mars方面的数据？

嘉宾：因为性能方面的话，我们比较难提供一个纯粹的环境跟其他的开元软件进行对比我们的性能数据主要体现在微信业务当中，所以也不是很好公开，其实我们没有做性能的对比，我们只是跟业界的对比，就是我们在微信的发展中，我们会跟同行业的APP进行对比，所以没有提供一个实验数据。

提问：你刚才一直讲消费这一块，我想问一下有没有想跟net（音）进行对比？它也是用智能连接的我们在用的时候，听你这么讲，你的项目跟net差不多，但就是说你说了超时，我们每次定的都是5秒，就是固定时间，我在想有2个问题，一个是你们Mars会不会涉及到后台常驻，还有另外一个就是说，你们会不会有菜包和黏包的处理？

嘉宾：我们在实质上，我们有这样的处理，但我们有常规的方案，所以没有说这个问题，你说的net方案，我还没有了解，至于常驻连接，就是根据你的使用情况。

提问：另外一个问题，你们常驻的时候，你们超时是怎么处理的？

嘉宾：更核心设计是心跳包的设计上，我们有智能心跳的方案，但这里因为时间关系没法分享，你可以关注我们的公众号，我们有发过。

提问：你好，我是来自YY直播海外开发的同事，我有两个问题，一个问题是DNS劫持，想让你详细介绍一下，还有您说的多级超时，包包超时、动态超时有什么影响？有什么优化？

嘉宾：我们在手包超时里面，数据包回去之后，从服务器回来还有一个RTT时间，相对于其他的两个主观值来说是两个相对小的值，所以我们会忽略，因为包包超时我们讲过，我们处理包包超时的时候，我们已经收到了首要数据，就证明我我们前面两个步骤没有，服务器前面的两个步骤是没有的，这样子使的我们不会依赖于主观因素。

关于DNS劫持，国内遇到的比较多，就是域名解析，我们发现反馈回来的不是我们的服务器，而是其他的，比如某某运营商或者某某广告这样的东西，我们是处理这种情况的，通常行业内的做法是STN，其实我们在公众号里面有说这个事情，这里我不详细展开了。

主持人：大家还有疑问的话可以扫码进行这个群，如果加不进去的同学可以让周边的同学拉一下，下一位来自手机淘宝自身团队的查老师。

2014年，我第一次听说过手机淘宝容器是2015年容器大会现场，当时绝大多数讲师是来自于阿里的团队，应该算在国内非常早的，在容器上面相当有时间的团队了接下来的时间交给你。

手淘灵活的容器框架-Atlas

李龙（查郁）：今天我给大家带来的题目是手淘灵活的容器框架ATLAS，首先介绍一下ATLAS产生的技术背景和现状。

在2014年无线之后，在淘系有几十个APP都要在手淘上落地，这样就带来了同时有几十个团队，几十个业务集合在手淘，安卓大家都知道有535的问题，方法数也会积聚更多，线上出现问题也要采用发报的形式。

从那时候开始手淘开始想设计一种灵活的框架，需要解决这些问题，首先需要就是说，各个业务需要独立的发布，第二个就是说相互模块之间去集约。于是从2014年ATLAS就诞生了，现在ATLAS已经在淘系中占了80%的应用，现在的话我们从之前的每个月发布到现在的每周发布，而且遇到问题的话，每天都发布，手淘里面具有60多个应用，每一个APP都可以独立跑。

下面我给大家详细介绍一下手淘ATLAS的容器化实现。

这是手机淘宝整个模块的划分，其实大致可以分为三层，最上面是业务层，包括聚划算、天猫、支付定这一些，可以看作一个单独的APP，大家可以看到上面，我们内部叫bundle，中间是底层中间件，包括网络库，再往下就是ATLAS容器，上面这些业务都是运行在ATLAS容器之上，再下面就是启动。

这一页大致列出了ATLAS整个容器架构，大家可以看一下，最下面是hack层，我们会把安卓底层的一些，我们需要用的安卓原先的方法，都先拿出来，稍微往上一层是bundle framework，这一层包括安装、生命周期、安全检查这一部分，再往上就是运行期管理，包括监控、还有调试，就是整个在安卓上面运行的时候，我们会在这一层做辅助；再往上是business层，也会带ATLAS自己的业务；最上面是应用接入层，因为ATLAS提供了给外部APP灵活接入的接口，这一层的话，主要是给外面APP做接入的时候，能够很轻松的接入它。

这一个大家可以看一下经过ATLAS改造的手淘的APP结构，左边是正常的APP结构，我们有很多代码，有很多码词，其实你启动的时候，你需要做安装，必然会导致其中的耗时，还会导致启动非常慢。

整个APP更新，我们可以看成是bundle的更新，因为现在ATLAS有了动态化特点，就是单个业务是一个APP，这一个APP也可以进行更新；第二就是应用的application，其实跟正常APP一样；第三就是pathclassloader，我们安装每一个bundle的时候都有，android os，我们会帮助解析，可以在ATLAS上做掉，相当于是安卓的ios。

因为ATLAS最初的设计理念是基于os的理念，我们会为每一个bundle设计生命周期，因为现在我们的手淘里面的每一个应用都是随时随地，如果甬道它的时候我们才会对应bundle，bundle有生命周期，因为每一个讲了，每一个APP在膨化完成之后，我们会把它们靠背到bundle的安装目录，比如里面有SO，我们可以做解压，会把so解压出来，跟安装原生态APP是一样的。

安装的时候，我们会创件bundle copy file extract lib load

最后一步就是bundle可以启动了，可以通过自己反射，这样的话bundle就起来了。

bundle在运行的时候，我们主要用了以下几个技术点

mmessage

activitythreadhook

application

我们的bundle class load

我们现在设计ATLAS容器，其实各个bundle在运行之间是相互独立的，但是有一些强偶合的业务，如果想用bundle的公众库，可以通过（）引，这样的话，在手淘的目录下有一个配置文件，但这种情况是尽量避免的，为什么呢？因为有bundle的时候，你会做N多次，你进这个业务的时候，你会发现会转圈，所以这种情况下要尽量避免。

刚才讲了在ATLAS下的加载机制，下面我讲一下在resource方面讲一下

当我们安装bundle做注入的时候，这时候我们会通过delegateresources

assetmanagernative

刚才我们也说过了，每一个bundle都是独立的APP，其实在构建的时候我们会为每一个bundle分一个ID，打包的时候我们构建PK的时候，我们会为bundle生成一个IP，保证不重复，其实说到重这里，手淘有自己的构架插件，其实也是我们自己写出来的，其实也是要符合ATLAS架构，

刚才讲了ATLAS的容器化架构，上面的业务是怎么运行的，后面我给大家讲一讲ATLAS的动态化实现。动态化实现说明白点就是你怎么在APP里面做某一个业务的更新，怎么实现快速的迭代发布，包括你的周发布、天发布是怎么实现的。现在周发布的话基本通过动态部署的方式，就是让用户无感知的更新，打包的时候，比如构建diff的时候，我们会裁减一些信息，这样我们保证两次构建的，我们都是保证一致的，diff dex，优化主要是优化三个点。

string 我们现在要把string干掉，第二要把没有方法的干掉，还有就是删除类，比如上一次构建APP，这一次怎么把

我们算了一下，会比的的diff dex

在客户端的话，我们需要接入dexmerge，通过反面工具看它的画，在客户端，我们通过dexmerge，单独起一个进程，重新设计一个merge collision policy，

刚刚讲的就是说，我们在做bundle动态部署的时候，在主dex部署之pathclassloader

dex

很多人说我是不是不让它做preverify，只做dex优化的时候，会创建一个ID值，就是virtualmethod，

现在手淘里面只有bundledex，下去一个dex，

今天发布安卓7.0之后，我们的动态部署会有不成效的问题，因为我们后来通过分析源头，其实安卓7.0上在后台会帮我们做热代码，会生成APP动态码。大家知道7.0提高了安卓启动的性能，会有热代码，这会给动态部署带来什么影响？

我们现在也做了远程下载，因为手淘业务非常多，现在有六七个APP在手淘里面运行，如果APP比较大，我们会放在服务端，通过后端看存在不存在，如果不存在的话，我们就H5页面提供你下载，这样也减少一些IO的错误。

现在客户端运行起来比较稳定，是怎么做到的呢？竞争隔离，非正常启动，也会让用户无感知，包括会插入校验代码，包括工具方面我们也做了一些方法删除。

我们的ATLAS预计在2017年1月份到2月份的时候会开元，后续请大家持续关注，谢谢。

提问：

嘉宾：里面的内容完全一样，手淘很多业务，因为在手淘里面要跑，自己也要做APP，用的其实完全是一套代码。

提问：我这边独立开发，需要不需要做你们的STK？

嘉宾：需要接入，可能需要接入ATLAS，没有ATLAS，你的APP运行不起来。

提问：安卓版本有很多，国内有很多，怎么做到兼容？

嘉宾：为什么ATLAS从2014年做这个东西，后面一直做优化，现在手淘有完好的优质策略，有小米、索尼，都在现场运行以后，拿到代码进行分析，里面加了各种各样的系统、集成型号的判断。

提问：听起来维护成本非常大。

嘉宾：现在没什么成本了，现在接入非常简单，里面的实现对安卓线上现有的机型都做了适配，现在已经非常稳定了。

提问：我有一个问题，你们bundle那么多版本，那么多团队维护，是怎么发布的？各个团队独立发布吗？

嘉宾：现在我们采用快速迭代，一周一个版本，一周一个发布点，周四，周四你们的业务自己看，赶不上这个发布点就下一个周四，如果发上去你的业务有问题，你的业务再单独申请再发一个版本。

提问：因为是独立发布的，bundle相当于独立的APP，手淘是不是自己也有一个库，他们相互之间是独立的？

嘉宾：手淘有中间件，是公用的，在所有的主dex里面，所有的bundle都可以调用到，但bundle与bundle之间是相对隔离的。

提问：因为每一个bundle不一样，会不会手淘APP会越来越大？

嘉宾：不会越大，只会说业务越来越多，其实公共的中间件其实都在主dex上面，不会导致总的APP非常大，你说版本管理，我们真正构建的时候，插件都是我们自己做的，开发器版本或者是SDT的很老的版本，最终在构建整一个APP的时候，我们有版本系统，如果说集成最新的，我们会拿一个最新的，可能你有接入性问题，后续我们会修改，就是里面的调用是不存在的。

提问：你好，我想问一个问题，IOS和H5还有安卓三个平台，我印象中他们的发布是基于vks（音），不需要写安卓的代码，这个系统跟vks之有什么关系？

嘉宾：据我了解，vks在上11表现比较好，他们主要做安卓页面的更新，现在除了vks页面，还有bundle里面也有你自己的中间层，后面就是说vks更偏向于活动上面快速的页面上线，通过那边的动态发布，可能动态部署更适合于快速的迭代发布，可能多个业务同时要在这个版本上做更新，就是替代掉了以前APP更新。

提问：按照你说的我的理解就是VKS跟你刚说的系统实际上是有一个平行的关系，功能是可以相互替代的？

嘉宾：功能不能完全替代，vks主要投入页面，现在主要是推页面，vks基于后台发布，三端更快，但ATLAS在安卓上做的更全，阿里现在vks在用，动态部署也在用。

提问：我想问一下ATLAS里面有一个动态部署的概念。

嘉宾：时间有限，今天我没有介绍手淘 其实现在和ATLAS也是并行的，我们现在构建的时候会做抉择，线上有问题了，能才能hokfks（音），我们也在做优化，现在资源部署二期也做，包括版权平台会开放出去。

提问：手淘现在有哪些内置插件，对于外部下载插件，有网络错误的话，你们采取了什么措施？

嘉宾：你刚讲的第一个就是远程下的包，如果有异常情况，这种情况下的话，第一种做法就是H5，至少你能用，下单不会在远程，现在放在远程的就是版本比较大，我们下载也会限制，仅仅是在WIFI环境下才会让它下，如果出现下载错误，我们会给你到H5。

提问：内置插件会不会影响你们包体的大小？

嘉宾：现在还好，我们做中间件，做容器这一部分，我们也会梳理，哪一些在里面可以用了，其实包也是很大的，我们会分析它，现在在手淘里面，究竟用到的是哪一些，我们会做静态扫描，没用的话，我们会定时清理。

提问：我的理解是你们有基础库的建立？

嘉宾：对。

提问：这样在推动上面，是基于自己的库搞的，比如你有网络库，但你用的不是一个网络库，你们负责推动吗？

嘉宾：我们会负责推动，我们自己集成在手淘中间件里面的部分，我们尽量让业务方使用，因为很多业方都关心自己的APP大小，包括每次打包出来的时候，都会及时打，我们很明显的会提示出来，及时通知业务方，包括我们这边有版本发布。

提问：这样的话涉及开发步骤不一致的问题，有的业务使用的比较旧的版本，这一方面，推动成本比较大，本身插件开发也是为了。

嘉宾：你说一部分用的比较旧，一部分用的比较新，如果你们需要用到SDK，我们会放在主dex里面，就是SDK组会持续的做同步，更新到最新的版本。

提问：就是业务在开发APP的时候，觉得这个东西将来要作为插件集成的话，必须要有同样的东西？

嘉宾：对，你现在在主dex里面加东西的话，我们这边会进行审核，因为现在手淘业务非常多。

提问：我觉得你们可以把包大小的控制分享出来。

嘉宾：可以，私下可以交流。

提问：你好，我想问一下你们现在手淘有六七十个业务，怎么去判断说这些业务用到的模块抽取出来成为一个公共模块，抽取出来之后又怎样保证有用到这些东西的业务能够迁移过去？

嘉宾：迁移非常容易，如果说你们有几个业务，就是用哪一个公众SDK，抽取非常容易，其实无非就是类加载的问题，这个只要放在主dex里面就可以了，你们工程怎么依赖都可以，我们最终只是做版本的裁决。

提问：你们是怎么判断的？业务放提出说我要加还是说你们要监控每一个业务，达到一定的量，然后把那个模块变成公共模块？

嘉宾：这个我们会做包扫描，就是会扫描重复类，比如说你这个业务里面用到了SDK，另一个业务也用了SDK，我们会进行重复扫描，我们再决定SDK放在了主dex里面，而不是你的业务里面。

提问：

嘉宾：你要说内部实现不一样的话，现在还没有特别好的方式去辨别里面的方法实现是不是一样的其实这一个方法他们也可以改，开元SDK，你也可以改，你也不知道他用的什么功能，扫描也扫不出来，因为我们会定期向业务组做普及、做推广，如果大家在手淘基础上做开发的话，现在也有这个意识，不会就是说拿第三方的安装在上面，我不会用你的，就在网络上面自己下载，其实这个很难控制的我们只是尽量告诉业务方，不要这么搞。

提问：手淘IOS有没有容器？

嘉宾：在任修复方面，IOS线上修复我们一直用我们自己的技术，现在还没达到安卓的周发布、日发布的快速迭代的发布模式。

主持人：接下来 我们欢迎来自百度IOS输入法的工程师。

胡官钦：大家好，我是来自百度手机输入法的胡官钦，虽然标题上带着百度输入法，里面介绍的优化方法是通用的，简单看一下现在的主要内容，首先介绍一下输入法的性能问题，然后花一定的篇幅介绍一下通用的性能优化方法，最后通过输入法中具体的实例具体说明一下具体怎么做有优化的。

输入法作为工具类的APP性能问题对于输入法来说可以说是衡量输入法是否好用的关键技术指标，第三方输入法现在面临的问题是IOS第三方输入法不稳定，在座的如果是IOS用户，上面的用户或多或少都会出现，比如使用国家中键盘弹不起来了，或者突然变成了null等情况，其实这些问题更多是苹果公司的问题，每一个版本发布出去都有用户反馈说输入法不好用或者性能不行，对于我们来说，我们只能提升本身的输入法稳定性，以及提高它的性能来让上面这些问题发生的概率得到一定的降低。

除了框架不稳定，还有一个问题就是现在输入法在IOS系统上限制比较大，主要是时间方面和内存方面的限制，时间方面，如果长时间没反应，IOS会自动关闭你这个APP，对于输入法来说，这个时间比较苛刻，虽然没有正规的文档说明这一点，但我们做过一些实验。键盘起不来的概率会大大的提升，除了时间方面的限制，还有内存方面，每一个APP使用的最大内存是不一样的，具体的值是有一个配置文件去设置的，如果大家是越狱手机可以找到具体的文件。

我们的输入法相对于一般的APP小很多，这个问题我们向苹果的工程师反馈过，对方给我们的答复是他们系统默认的输入法只有20兆，所以对我们第三方输入法比较慷慨了，后续可能会有进一步放松的迹象。这一个值，每一个设备上不一样，如果是IPAD的话，通常是66兆，这里要说的是，用户崩溃的时候经常会看到一类叫jetsamevent，你自己的APP里面没法捕获的，对于我们自己的输入法来说，我们要提升性能解决这个问题，所以性能是非常重要的问题。

下面看一些具体的性能优化方法，首先声明一点，内存优化也是性能优化的范畴。首先性能优化第一步就是要找到性能衡量的方法，第一个问题要在真机还是模拟器上做性能优化？这里我们要在用户的真实环境做测试，模拟器跟真机的状况相差非常大，所以建议大家在真机上面做性能优化，这里强调一下，不光是真机，而且是手头上性能最差的那个机器做性能分析，通常最烂的机器才是性能的评定。

性能衡量方法有两种，特别是适合打定点的分析，我们通常在开发过程中需要时刻关注这样一个关键的指标，通过这种方法，我们在日常开发过程中，一旦出现性性能问题，就立刻检查。

大家不要猜，一定要通过具体的衡量手段发现问题的瓶颈手段，再做进一步的分析下面具体介绍一下，如何做性能优化，这里我总结了5种方法。

不该做的不要做，不要重复做，用更快的方式做，提前做，连续做。

第一个就是不要做，比较明显的例子就是日志跟错误处理的方法，大家在开发的过程中，为了方便调试，会打日志或者使用断言，同样它们对性能的影响非常大，我们提交APP store之前要做好一下。

比如在我们的项目中针对assertions

除了前面的代码，还有一些不太明显的操作，也会引起性能问题，在我们实际开发中也发生过，大家可能会想，我们不会无缘无故做一个非整数数值，通常产生这些的原因是你可以产生一些自适应布局或者缩放预算，一不小心就会出现这种情况，这种情况对大家来说，如果用眼睛找是很难发现的我们可以利用这个模板，这个模板可以帮我们找试图渲染的题。

其中有一个选项比较关键，就是礼品渲染，通常比如左边有这么一个土层，要渲染到屏幕上去的时候，是在屏幕的缓冲区进行渲染。

通常礼品渲染都是靠性能的，第一个要额外分配一块内存出来作为缓冲区，另外一块是之间上下文切换代价，大家在具体的开发中要注意这一类的性能问题。

下面说一下第二条优化规则，不要重复做。我简单介绍一下，首先我们缓冲的肯定是频繁使用的对象，另外就是开销比较大的对象我们要放进去，一个文件经常用到，频繁的打开关闭，性能肯定非常差。

大家的APP对内存的要求非常高，不建议用uii mage如果你的内存达到一定的限制要去释放它，你没有办法，最好的办法就是控制uiimage周期，除了缓存对象以外，我们手写输入法来说，有手写轨迹的显示，他测试的时候，在上面乱涂，在涂的过程中，性能会越来越差，在你画完第一笔，我们把中间的件保存下来，这样性能会提升。

接下来第三个优化方法，大家所理想的是数据结构跟算法这一块的，我个人认为移动开发数据结构算法要求不是非常高，每一种操作的复杂度，掌握到这种程度就可以了，这里面列了几个，通常情况下C元的实现比OC的实现还要性能好不少，不是要求大家都把代码替换到C。

手写模块里面有矩阵运算，这个就是CPU的运算方式，当我们处理大量的运算的时候，CPU运算非常大，其实我们不需要自己做硬件加速苹果有一个我们可以直接用，我们手写模块用这个框架重写以后大概提高了120%的样子，除了软硬件盲从硬化，下面还有一些投机取巧的方式这里面举了一个字符串长度的例子，IOS里面是通过数字曲线表示的，写的时候要会聚成一张张图片，同样IOS关于性能的考虑，这部分会缓存起来，至少到目前为止，我们还找到合适的方式把这部分的内存清希望掉，这里只是想计算一下字符串的长度，这个会把整个字符串汇聚起来，所以说字符串长度计算，是既耗内存也耗性能的操作，对于我们来讲性能影响是非常大的。

怎么用近似计算提高性能，比如我们要算一下字符串的长度我们要先把中国技术开放日的中字先算出来，算出来的结果大家可以看一下，但如果你的字符串里面有一些特殊表情或者是英文字符，计算出来的结果有可能出入就非常大，这里有一个例子，如果用英文算，两种技术差别是非常大的。

下面再看一下do di before

客户端需要去显示一张图片，首先要从服务端把它拉下来，先处理完，当客户端真正要显示的时候，就可以加快整个过程，另外一个例子就是xcode，压缩率、渲染时的性能，都会提前帮你做好。我们以前做粤语输入法的时候，我们优化的方式就是用了这个。

还有一个预加载，做照片类的APP，这个是很常用的，比如我们要展示中国技术开放日，如果用户没有找到侯选词，他需要翻页。我想说的是，如果是预处理加载，策略就非常重要，包括加载的时机以及页数，这会根据用户的使用状况决定的，事实上用户找翻页词是很低频的遭到，当用户进行快速点击操作过程中，你每次都做加载第二页、第三页操作的话，反而对性能有很大的影响。

如果你检测到用户有这个动作，你再有翻页，这样用户可以有优化的体验。

最后一个是do it、later，这个是延迟加载，一八布加载，这个用到了极致在项目中用到它们的技术还是可以提升不少的渲染性，最后有一个例子，内存初始化的例子，实际上如果你分配的是一块很小的内存，这两种方式实际上效果是一模一样的，但这个例子你可以看到，我们看到的是四道。

讲到这里，前面的那5个优化方法已经讲完了，下面看一下我们输入法的皮肤案例，具体看一下怎么做这些性能优化的。

首先简单介绍一下我们输入法的皮肤结构，整个皮肤结构分了三块，一个是布局文件，还有样式文件，还有图片资源，布局文件就是指定了某一个案件指定了，大小是多少，这个案件是怎么显示的，背景是什么最后我们要显示到外面的资源文件这个对于大家来说，不需要太了解。

整个皮肤显示的流程我们要先把前面的配置文件显示出来，整个过程中，其实这三步是比较耗性能的我们想到的方法就是把整个皮肤文件先中间缓存一次，就是把原始的皮肤文件转化出来一个中间格式，这个中间格式显示的时候，性能比原先好很多的，用到的前面优化思想就两条。

下面看一下整个过程中最耗时的部分，因为输入法中间的图片比较特殊，我们输入法刚开始设计就是这样子的，就是有一张很大的图片，里面放了很多小图片，我们显示的时候需要把里面的小图抠出来，我们做皮肤缓存的时候想跳过这部分，我们显示的话直接显示这个小图片。整个过程是图片处理比较密集的操作，另外涉及到IO操作。

看一下我们是怎么做优化的，我们首先决定看一下格式，把解码后的图片讲一下，整张图片通常情况下，事实上只是得到一个空的对象，IOS底层只是是提供了内存关系，

在这个过程中才会真正做图片解码的操作，首先第一步会分出一些内存，用于解码的操作，第三步用cpu解码图片数据，最后把这些数据渲染到图层上面，整个过程中最占性能的就是图片解码的过程，我们在考虑的时候，我们想直接是解码完的图片，最后生成的时候就不需要做解码的操作，通常这样的性能更加好，因为有一定的副作用，如果保存成解码后的图片，那空间就会增大，在IOS上对于空间敏感度不是非常大，最后我们决定还是用解码完的皮肤数据。

下面看一下我们第一步做的初始方案，相对比较简单，首先把原图加进来，调用image方法，把具体的图片数据抠出来，整个过程在代码逻辑上一点问题都没有，但这个性能非常差，第一个问题切片的性能非常差，为什么这么说，我前面说过了，真正解码是解码出来的图片数据是用IOS系统缓存的。

我们先在操作之前，强行让这张图片解码掉，我们再对生成出来的图片做后续的抠图操作，从而达到性能的提升。

第二个问题就是内存方面的效率非常低，大家前面看到了，首先把原始数据抠一块出来，本身是耗内存的操作，这个内存会销毁，这些方案有两种，我这边举的例子就是申请context 的时候，最后我们不需要生成中间对象，直接对数据进行操作因为我们知道一张小图在一张大图中的位置是怎样的大家可以看一下，底下可以把小图的数据读出来，写到文件中去，中间不需要再生成一些对象。

最后一个问题IO性能差，针对写文件的方式我们采用了内存影射的方式，需要把用户态的数据靠背到内盒态里面相对来说IOS性能会好一点，另外还有一个小秘密可以告诉大家，用硬盒内存不会加入到IOS内存里面。

关闭文件相对比较简单我们用了延迟关闭的方式，不是纯粹提升了多少性能，而是可以提前结束，大家不要小看这个操作这个操作大概可以提升10%的性能。

前面讲的优化方法，现在讲一下性能优化的tips，我们说了要在比较烂的真机上面做测试，在提交app store之前，把代码全部移除掉，通常过早做这些事情，会束手束脚，会严重影响项目的推进时间。

性能优化当中，大家追求的是一种够好，而不是最好，通常情况你在性能优化过程中会发现及对代码的可读性，可维护性有一定的伤害，最后就是性能优化，大家还是要跟多职能角色沟通一下，这样可以事半功倍。

我今天的演讲就到这里。

提问：我想问一下建行这一块除了内存这一块，我想问一下，百度输入法有没有对用户的点击和数据输出有量化的标准？

嘉宾：我们每一个版本都会做这方面的衡量，还有跟竞品的对比，我可以告诉大家，其实点击键盘操作通常都不是性能的主要问题，现在的IOS设备都已经比较先进了，性能比较好，我们的问题还是启动这一块。

提问：我问一个比较大的问题，对我们用户来说，其实你们做的足够好了，从客户那边反馈给你们问题，其实比较少，你们开发的动力在哪里？你们是自上而下是领导提要求还是客户提要求，激发你们开发的动力？

嘉宾：你说用户反馈少，其实也不少，我们每天也会收到差不多200条的建议，我们也是有自己的反馈渠道，还有微博、公众号，每天都有人去做部分的信息收集，另外你说上头是不是有指示，那是肯定的，每年的技术方向、任务也是不一样的。

提问：我咨询一个问题，我们测试这方面，都是基于真机，我们的程序执行是自动化吗？

嘉宾：我们目前是用自动化方式做的，大公司里面应该都是通过这种方式做的。

提问：你好，在日常生活中有UI的设计。

嘉宾：其实TTP中也有介绍，我删掉了，你做原角处理，我的建议你可以跟ui沟通一下，第二种就是你的边缘跟你的底色是一样的，直接覆盖上去，让用户看上去这是原操作，就是你在你的图层上面，再覆盖一层，虽然会加大一定的渲染过程，有一定的性能消耗，但比起礼品渲染，性能还是会好很多的。

提问：就是交一张图片，中间是透明的。

嘉宾：对，中间是透明的，是一个原角，用户看到的就是透明的。

提问：你说你们做了很多优化，导致只需要做一次，最后都不需要做了，你们这个时机是怎么选择的？

嘉宾：刚才说过了，我们的皮肤是在另外设置当中，通常皮肤是设置下另外一个栏中，需要有一定的等待时间，对于用户来说是要接受的。

提问：我想问一下在输入法里面，我们使用输入法的时候，应该会有一个搜索引擎帮我们匹配输入的字。因为我们在输入一个字的时候，输入法会提示我们很多联想词，这个我想知道，比如网络比较热的情况下，这个怎么做联想搜索，是否客户端也做了某一些缓存机制处理这些场景？

嘉宾：目前缓存这一块没有做，网络要求也是比较高的，原先我们考虑到用户流量的问题，只会在WIFI底下做，现在这两年3G、4G比较普遍了，大家对于流量也不是很敏感，最近我们也放开了，在4G底下也可以做云输入。

提问：你好，我有一个问题想问一下，按我理解的话，IOS启动的时候，就是输入法APP是不是有常驻内存？

嘉宾：怎么说呢，跟普通APP来说，是独立运行的进程，你说的常驻进程是什么概念？其实输入法常驻是比较特殊的，一旦用户把键盘收起来，这个时间非常短，我估计只有几秒钟的时间，我们的进程就停止运行了，不会长时间在后台运行的，如果长时间在后台运行，对于我们来说会实现很多功能。

提问：启动很快结束也很快？

嘉宾：对，而且无法预计，有时候生命周期的含数都没执行完，你这个代数已经结束了。

提问：根据你刚才说的，你会做图片缓存，做这些有没有意义呢？

嘉宾：你刚才说的缓存是皮肤缓存吗？用原始数据显示，你往往需要做很多的解码操作，首先皮肤文件里面的格式化数据，可以理解为节省数据，如果加载到具体的对象当中，本身就是耗时的操作，我们先把这个操作转化成中间的格式，你可以认为是，我们原先做的是C的结构体，直接影射到了OC对象。

主持人：现在非常感谢来自百度的工程师的分享，最后一个分享的是来自唯品会的技术副总。

蔡镜明：今天给大家带来的主要是说和时间赛跑的移动电商技术演进，主要讲讲唯品会做的技术探究最近新的动向，这个标题的原因是说，首先会跟大家讲一下目前在移动端，我们的架构是什么，但我会特别聚焦原生和H5这个部分。

大家可以看一下，如果有装唯品会APP的同学可以看一下，这是我昨天晚上截图的，大家过来酒店的路上应该有看到，就在楼下的很多公交车站牌都可以看到周杰伦给我们带来的活动，这四个页面就是唯品会APP打开，流量最高的页面。

在所有的电商当中，唯品会在原生上的布局是最复杂的，大家可以想中间这张大图并没有用任何方案，我们完全靠一些动态的设计来画出来的，更早时期我们还做不到这一点，所以每次我们做大促就换成H5页面。这个流量入口完全是由H5做的。

大家都知道唯品会是电商，而且是特卖，所以经常有大规模的促销活动，如果多数功能用原生做的话，大家可以看到一些事情，我们的整个运营节奏，比如定在12月8日，我们做一个活动可能我们提前一周到两周发到各大应用商店里面，既然是强运营的事情，在座的技术人员都比较了解，如果你是强运营的，你的变化特别多，这样就决定所有的动态内容、所有的跟运营相关的内容都要有灵活的方案支持这样就逼迫我们突破难点，所以我们整体上会采用H5的方案。

这里给大家看一下，我们的安卓和IOS的页面，大家可以看到我们大部分的功能其实都是nativ的方式，这是我们在这一部分的架构，因为我们比较聚焦在page，自上往下大家看到成色的部分就是我们跑在唯品会APP里面，往下是我们做的一些事情首先是H5我们和一些容器的交互协议，容器的现状是IOS现在跑的是WK，我们的H5页面可以调用一些接口，主要是用在安全系数比较高的产品上，第二就是对各组件的支持，里面包含了对H5的支持，还有大家都知道AHBS，我们知道H5页面每次从服务器下载的话，页面会跑的很慢。

这是APP里面非常大的创新，大家都知道，在中国的互联网情况下，在这个过程当中，对于做交易的公司，对于做支付的公司来说特别麻烦，因为我们多出的页面都是H5的，只能说别人突出不好的手段，交易和用户敏感的信息可能就流出了，所以在这个过程中，我们加入了智能陆由，陆由有问题的时候，我们会进行跟进。

我们原来做的协议非常简单，去年的时候我们的原生和H5交互都是这样调的，调的这些业口，每一次都是根据你的需求临时加一点，所以没有什么扩展性可言，我们是基于分装，这里面有一个数据，我们完全基于内部APP协议部分把接口开放出来我们差不多每半个月APP会崩溃一次。

我们用了新协议之后，由这部分带来的问题当中，过去的半年没发生过一次，这是我们分发的实现，中间这部分是购物车等，在上层其实会分装出，给H5用的原生，比如可以加载弹窗，最底层其实就是SDK，到了右边我们可以看到在H5上面，最上面的业务层就是我们用到所有组件的内容页，接下来我们又进行了更好的分装，所以我们有效进行H5的分状，进行这样交互的稳定性。

下面接下来就到了原生内部，我们首先安卓用的是web view这也是我们看到的简单对比，大概是这样的情况，web view controller层是基于独立建层的，不用跨页面操作。

还有另外一个问题，我们会发现，原来我们做老一些版本的时候，有一个UI的含数需要广告请求的，于是我们就是说，我们通过阅读，我们最终发现有一个办法是可以区别的。

wkwebview，

整体页面加载速度都是在H5的时候，差不多整体快的是1到2秒，安卓平台上我们采用了腾讯的H5机构，大家可以看到对比的数据简单对比的感觉，但这里说一下H5的早期版本，我们早期跟腾讯在H5上面进行合作，我们和他们的沟通、交流、迭代都是在很稳定的水平上。在我们所有的webview上面，这是我们用的H5之后的数据，现在我们可以就是说进去再出来差不多增加2K。

这是关于说我们H5和native的第一层交互，首先第一个就是说离线的稳定支持，大家都知道直接开一个问题，加一个null的话，再看你的服务器页面响应速度，这是很简单的交互，我们说用H5的原因是因为它灵活、可变、速度快，但是不是每一个H5页面都需要那么灵活，我想大家都知道答案。这个最早在Facebook的首页上，蓝色部分是这样，我们在APP上面实现的我们和大家介绍，本身就可以支持，对所有人的请求进行判断，如果这个请求是有缓存的，我会把这个请求发到服务器上，通常我们对资源的数据量会做一个校验，因为做电商少不了给用户的触达，很多用户打开的时候都有刷新，所以这两个时间点上，我们会把压缩好的离线文件推送到用户的APP里面，我们也用了插件办法。

数据上报，我刚才跟大家说有exception report，会把这些数据上报到服务器，一般来说我们可以很快响应一些问题，有一些比较说特别严重的问题我们很快会处理，因为多出的页面，当我们知道有问题的时候可以很快进行调整，这就是跟大家提到的智能陆由，如果纯原生方案里面，大家肯定不陌生，我们做的更深除一步，就是H5

对于H5来说，首先你要形成拦截，这就是我刚才说的在WK里面找到一个方法，因为我们既定的，就是H5的域名就是那么一些，我们会有离线方案，包括定期检测，从用户当前的节点和终端的健康状况，用户真的发生，比如某一个H5请求的时候，由此保证用户的流通效果，我们会发现就是说平均在建立连接的加载时间上，就是动态的接口上，我们H5的访问成功率，在没有达到页面超时率之前，我们提升到了0.6%，

最后说一下我们最近做的事情，我们所有的H5页面都不是我们开发出来的我们的运营也好、产品也好，我们后台都是采用模块化的组件开发出来的，我们每一天在线上有1万个，大家可以看到有了页面之后不能及时用，他们有预处理的过程，后续我会解释。

当用户加载页面的时候，我们可以通过渲染整个布局再到达里面，在网络连接上，我们走的比较激进。大家可以看一下我们的广告方案，从左至右就是我们的商品页和列表页，中间的也是H5，为什么是H5？特别简单，这个图片有一些不规则的热区，就是说你点热区的时候可以看到对应的商品上线，看起来是图片，其实不行的，大家都看到是一张图片，我看到的是红色的，但别的人看到的可能是男士，右边的是我们的一个旗舰店。

这里就是看一下移动方案，其实我们同时支持了H5的方案我们通过开关进行开发，我们左边内部是说，我们内部有协议头，如果这个页面做好了或者已经生好了这一些方案，我们的APP会加载对应的容器，用其他的静原生加载它。

社交领域用recyclerview，一个页面的商品可以多达数百个或者上千个，只要让用户很快的看到商品就可以了，我们发现随着访问深度不断往下下滑，内存是持续增长的只是最后崩溃掉，我们做了一系列的优化，优化方案页内都是一条路了，我们用了8个缓存单元，这个读写队列又有一个问题，列表很长，而且列表比较大，内存崩溃了，快速增长的时候，可视区域会出现闪屏的现象，大家可以去网上面一下，就是快速滑动的时候，内存虽然增长没那么快，但会出现闪屏，非常厉害，但试图刷新的时候都是异步的，所以在平台滑动的时候，而又由于刷新，这个本身很耗资源，心里会出现闪动的问题，我们解决思路很简单，就是把时间改为20秒，黑色的部分是后来，差不多降低原来的三分之一。

最后谈到mem leak，因为你进入一个页面内存不再释放了，如果你的APP里面，用到了这个技术，其实你的内存在持续往上涨的，可以看到红色的是里面自带的方案，我们测试了300行的列表页，内存的变化趋势，在这个地方解决的办法很简单。

谢谢大家。

提问：你们有没有遇到过什么问题？

嘉宾：没有。

提问：如果因此被拒的话，有什么措施？

嘉宾：我们可以关掉。现在比较好了，业内很多公司在用了，苹果对它的态度也是接受的。

提问：我刚才看到你们在IOS和安卓上面，你们都做了网络处理，现在这个是统一的一个网络库，还是各自有各自的？

嘉宾：分开的。

提问：另外的话，你说有网络预测，在WIFI环境下还好，如果在2G、3G、4G下探测的话，对用户也有流量消耗，这一块你们的策略是什么样子的？

嘉宾：我们对网络的探测采用2分值协议，关于探测这个事情，我们会通过服务器算，我们是基于，当你知道你当前的节点情况，其实你是可以通过一些公式算出来网络访问模型，其实就是说相当于客户端进行打点或者通过精准的方式传输给服务器，我们的服务器有一套算法，但不是百分之百，我们会有实验组，我们是一个迭代过程。

提问：你们的IP还是基于系统探索出来的？

嘉宾：这就是我们的实现方式，其实就是相当于在最底层，我们自己把IP实现了。

提问：SDK的协议应该加一个东西，那个是统一的协议还是安卓和IOS是不一样的？

嘉宾：统一的SDK是不同的，但是H5是相同的。

提问：关于数据页面缓存，是数据自己的缓存还是SDK有自己的协议？

嘉宾：在热加载的部分，这个部分会采用H5的探索机制。

提问：通过GS发起请求，让数据进行缓存，进行后期的渲染？就是NA和H5合并的那一种。

嘉宾：这部分是的

提问：你好，我想问一下，我们利用H5非常不顺利，就是说各种奇葩的机型，会不会有问题？有问题你们是怎么解决掉的？

嘉宾：这是随着时间自然会解决的问题，那时候我们的页面质量比较差，我们对工程师要求比较高，H5本身有一些建设性考虑，这个对于H5要求比较低一些，因为我们和腾讯沟通过。

提问：你们对低版本的没管它？

嘉宾：4.4版本以下的H5页面，我们有一个标准，我们要优化它，我们会看一些主流的机型，不会达到一个水平上。

提问：你好，我想问一下唯品会APP有没有做图片处理？

嘉宾：我们在线上有图片，但没有大规模使用。

提问：你们的WK陆由有没有做支持？

嘉宾：有。

提问：是怎么拦截的？

嘉宾：对于WK格式，我们有一个功能，我们把所有的问题先截住。

提问：H5前端的框架？

嘉宾：我们采用两种。

主持人：非常感谢四位老师今天的分享，非常感谢大家的参加，也感谢四位老师，大家可以扫这个二维码加群。