**网络学习：bilibili高并发实时弹幕的架构**

高并发实时弹幕是一种互动的体验。对于互动来说，考虑最多的地方就是：高稳定性、高可用性以及低延迟这三个方面。

高稳定性，为了保证互动的实时性，所以要求连接状态稳定；

高可用性，相当于提供一种备用方案，比如，互动时如果一台机器挂了，此时必须保证可以和另外一台机器连接，这样就从侧面解决了，用户连接不中断的问题；

低延迟，弹幕的延迟周期控制在1秒以内，响应是比较快的，所以可以满足互动的需求。

B站直播弹幕服务架构（下面简称GOIM）的出现就是为了解决这一系列的需求。下面将对此进行详细的介绍。

**B站直播弹幕服务架构GOIM的出现**

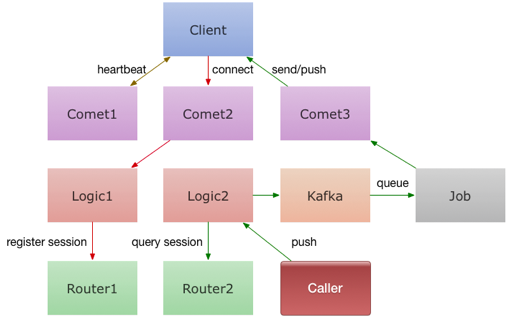


图 1

直播聊天系统本质上也是一种推送系统，所谓推送系统就是，当你发送一条消息时，它可以将这个消息推送给所有人。对于直播弹幕来说，用户在不断的发送消息，不断的进行广播，当一个房间里面有10万人时，一个消息就要发出10万次请求。在GOIM出现之前，也用过另一个名为Gopush的项目，这个项目推出的目的就是进行推送。在此之后，基于一些针对性的应用场景，GOIM对Gopush进行了优化，从而出现在我们视野当中。GOIM主要包含以下几个模块（图1）：

**1. Client**

客户端。与Comet建立链接。

**2. Comet**

维护客户端长链接。在上面可以规定一些业务需求，比如可以规定用户传送的信息的内容、输送用户信息等。Comet提供并维持服务端与客户端之间的链接，这里保证链接可用性的方法主要是发送链接协议（如Socket等）。

**3. Logic**

对消息进行逻辑处理。用户建立连接之后会将消息转发给Logic，在Logic上可以进行账号验证。当然，类似于IP过滤以及黑名单设置此类的操作也可以经由Logic进行。

**4. Router**

存储消息。Comet将信息传送给Logic之后，Logic会对所收到的信息进行存储，采用register session的方式在Router上进行存储。Router里面会收录用户的注册信息，这样就可以知道用户是与哪个机器建立的连接。

**5. Kafka（第三方服务）**

消息队列系统。Kafka是一个分布式的基于发布/订阅的消息系统，它是支持水平扩展的。每条发布到Kafka集群的消息都会打上一个名为Topic（逻辑上可以被认为是一个queue）的类别,起到消息分布式分发的作用。

**6. Jop**

消息分发。可以起多个Jop模块放到不同的机器上进行覆盖，将消息收录之后，分发到所有的Comet上，之后再由Comet转发出去。

以上就是GOIM系统实现客户端建立链接，并进行消息转发的一个具体过程。一开始这个结构并不完善，在代码层面也存在一些问题。鉴于这些问题，B站提供了一些相关的优化操作。在高稳定性方面，提供了内存优化、模块优化以及网络优化，下面是对这些优化操作的介绍。

**优化方向**

**内存优化**

内存优化主要分为以下三个方面：

一个消息一定只有一块内存

使用Job聚合消息，Comet指针引用

一个用户的内存尽量放到栈上

内存创建在对应的用户Goroutine（Go程）中

内存由自己控制

主要是针对Comet模块所做的优化，可以查看模块中各个分配内存的地方，使用内存池

**模块优化**

模块优化也分为以下三方面：

消息分发一定是并行的并且互不干扰

要保证到每一个Comet的通讯通道必须是相互独立的，保证消息分发必须是完全并列的，并且彼此之间互不干扰。

并发数一定是可以进行控制的

每个需要异步处理开启的Goroutine（Go协程）都必须预先创建好固定的个数，如果不提前进行控制，那么Goroutine就随时存在爆发的可能。

全局锁一定是被打散的

Socket链接池管理、用户在线数据管理都是多把锁；打散的个数通常取决于CPU，往往需要考虑CPU切换时造成的负担，并非是越多越好。

模块优化的三个方面，主要考虑的问题就是，分布式系统中会出现的单点问题，即当一个用户在建立链接后，如果出现故障，其余用户建立的链接不能被影响。

测试是实践过程中最不可缺少的一部分，同时，测试的数据也是用来进行参考比照的最好工具。

**网络优化**

最初B站的工作内容，主要是以开发为主，为了在结构上面得到扩展，所做的工作就是将代码尽量完善。但是在实际业务当中，也会遇见更多运维方面的问题，所以，在之后的关注重点上，B站添加了对运维的重点关注。

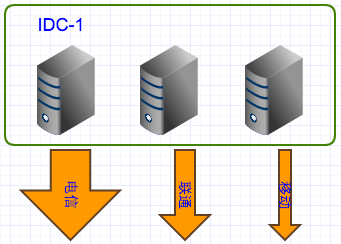


图 4

图4是B站早期的部署结构。最开始，整套服务是部署在一个IDC上面的（单点IDC），时间一长，这样的部署结构也逐渐显现出它的缺陷：

单线IDC流量不足、单点问题、接入率低

这样的网络部署往往会造成延迟高、网速卡顿等问题。

针对以上三点问题，B站也对部署结构进行了改善，图 5是改善过的网络部署结构，下面将对这个部署结构进行详细说明。

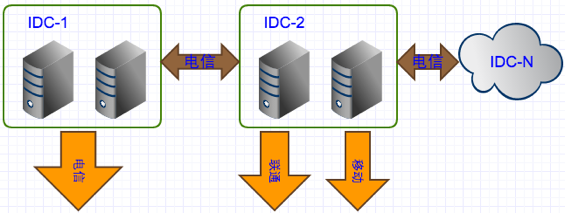


图 5

针对单点IDC流量不足的问题，B站采用了多点IDC接入的方案。一个机房的流量不够，那么就把它分散到不同的机房，看看效果如何。

对于多点IDC接入来说，专线的成本是非常高昂的，对于创业公司来说，是一块很大的负担，所以可以通过一些研发或者是架构的方式来解决多IDC的问题 。针对多IDC的问题，需要优化的方面还有很多，下面列举出一些B站现有的一些优化方案：

调节用户最优接入节点

使用Svrlist模块（图6.1 ）支持，选取距离用户最近的最稳定的节点，调控IP段，然后进行接入。

IDC 的服务质量监控：掉线率

判断一个节点是否稳定，需要不断收集大量的用户链接信息，此时就可以使用监控来查询掉线率，然后不断调优，收集最终的结果去做一个拓扑图（全国范围），在拓扑图当中就可以判断出城市到机房之间的最优线路。

自动切走“失联”服务器

消息100%的到达率（仍在实现中）

对于弹幕来说，低丢包率是非常重要的。比如，消息是价值上千块的礼物，此时一旦丢失某些消息，当用户发礼物时，起到的效果就是，实际在弹幕中显示出来的效果是，礼物数远远少于用户花费金钱买来的礼物数。这是一个很严重的问题。

流量控制

对于弹幕来说，当用户量到达一定级别时，需要考虑的问题还是流量控制，这也是对于花销成本的控制，当买的机房的带宽，是以千兆带宽为计费标准时，当有超标时，一定要将超标部分的流量切走，以此实现了流量控制的功能。

引入多点IDC接入之后，电信的用户依旧可以走电信的线路，但是可以将模块在其他机房进行部署，让移动的一些用户可以连接移动的机房。这样就保证了，不同地区不同运营商之间，最优网络选取的问题。

可是解决了最优网络的选取，却带来了跨域传输的问题。比如在数据收集时，Comet模块将数据反馈到Logic，Logic进行消息分发时，数据便会跨机房传输。有些公司的机房是通过专线进行传输，这样成本将会非常高。所以，为了节约成本就只能走公网的流量，但是公网的稳定性是否高、是否高可用，都是需要考虑的。当流量从电信的机房出去之后，经过电信的交换机，转到联通的交换机，然后到达联通的机房，就会存在跨运营商传输的问题，比如丢包率高，因此，跨运营商传输带来的问题还是非常严重的。

为了解决这个可能存在的风险，可以尝试在联通机房接入一条电信的线路（带宽可以小一点），“看管”电信的模块，让来自不同运营商的流量，可以走自己的线路。做了这样的尝试之后，不仅降低了丢包率，还满足了对稳定性的基本要求，并且成本消耗也不高。可是，这样的方案也不能说是百分百的完美，因为就算是同运营商之间的通讯，也会存在城市和城市之间某个交换机出现故障的情况，对于这样的情况，B站采取的方法是同时在IDC-1与IDC-2（图 5）之间部署两条电信线路，做了这样的备份方案之后，通畅程度以及稳定性都有非常明显的提升。

针对上述过程中出现的一些问题，前期，需要对每个线路的稳定性进行测试。为了测试每一条线路的稳定性，可以把Comet放入各个机房中，并将Comet之间的通讯方式汇总成一个链接池（链接池里可以放多个运营商的多条线路），作为网络链接可以将它配置成多条线路，用模块检测所有的Comet之间的通讯，以及任何线路传输的稳定性，如果说通畅的话，则保证这个链接是可以用的（这里面有很多线路，所以一定会选择通畅的那条线路进行传输，这样，就可以判断哪条线路是通畅的）这样一来，流量进行传输时，就有多条线路可以进行选择，三个运营商中，总有一个是可以服务的。

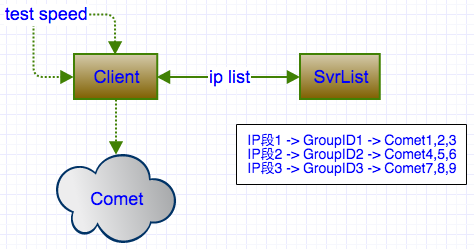


图 6.1

综合这些问题，B站又对结构进行了重新优化（这个结构刚刚做完，目前还没有上线，还需要经过一些测试）。

首先是Comet的链接，之前采用的是CDN、智能DNS。但实际上，有些运营商基站会缓存路由表，所以即便将机器迁移走，部分用户也并不能同时迁移走。而DNS解析这一块，也并非完全可靠，而且一旦遇上问题，解决的流程又很长，这样下来，体验效果是十分糟糕的。其次是List，将其部署在一个中心机房，客户端采用的是WEB接口的服务，让客户端访问这个服务，就可以知道该与哪些服务器进行连接。将IP List（ Comet ）部署在多个机房，可以将多个机房收集的值反馈给客户端（比如：哪些线路通畅）让客户端自己选择与那个机器进行连接。

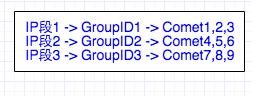


图 6.2

如图6.2，图中将IP段进行了城市的划分，将某一个城市的一些用户信息链接到一个群组（GroupID），群组下有一个或多个Comet，把属于这个群组的物理机全部分给Comet。

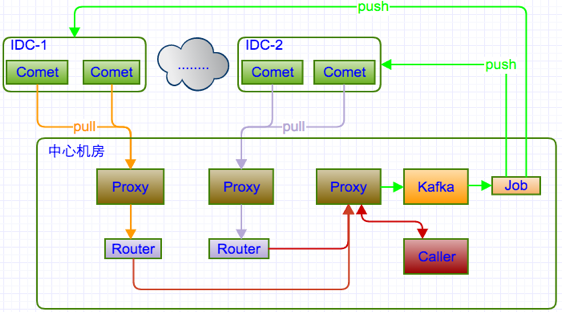


图 7

图7是再次优化的结构，还是将Comet全部放在IDC机房中，消息的传输不再使用push（推）的方式，而是通过pull（拉）的方式，将数据拉到中心机房（源站），做一些在线处理之后，再统一由源站进行数据推送。当然，这里要十分注意中心机房的选取，中心机房的稳定性是十分重要的。除此之外，B站在部署的时候还优化了故障监控这块功能，用来保证高可用的服务。故障监控主要为以下几项：

模拟 Client ，监控消息到达的速率

线上开启 Ppof ，随时抓图分析进程（CPU ）状况

白名单：指定人打印服务端日志

设置白名单，记录日志信息，收集问题反馈

标注重点问题，及时解决

防止消息重现

服务器负载监控，短信报警

对于GOIM来说，低成本、高效率一直是B站所追求的标准，B站也将对系统进行持续优化和改进，以给用户最好的直播弹幕体验。