章文嵩博士和他背后的负载均衡帝国

> 评价一个技术人，你会从哪些方面入手？评价章文嵩博士这样的技术前辈，你又该如何开口？也许这样一个追赶者后辈，认识得更深刻。

前言

@正明，阿里集团技术大牛、淘宝基础核心软件研发负责人、LVS 创始人、阿里云首席科学家章文嵩博士从阿里离职，去追求技术人生另一段历程，让阿里像我一样的很多热爱技术的工程师都有一丝牵动和感触。

国内 IT 业界在底层基石技术层面有所建树，达到 Linux 标准内核模块层面的应该就只有 LVS，而且就广泛影响力方面，LVS 在 Linux 逐渐取代 IBM Aix, Sun Solaris,HPUX 这些 Unix 们的过程中，扮演了很重要一块的角色。

在阿里中间件，我所在的团队恰好是专注在”软负载“（软件负载网络）方面，提供的产品和服务，跟负载均衡产品有很大一部分交集，在集团内，我们的产品 VIPServer 可以说跟 LVS 有一些直面的竞争，在集团内越来越多的业务逐渐从 LVS 迁往 VIPServer 上，但正因为如此，所以在负载均衡领域我们对于章博士的江湖地位和影响力才有更深刻的了解。

在这个技术人目光再次聚焦在章博士身上的时候，我想借此机会将我们团队对于负载均衡产品这一块的一些思考和实践经验，总结分享给需要的人，希望帮助到正在思考这一块的技术人。也许未来我们永远也不能像章博士这样做出广泛影响

世界技术圈的产品，但是作为后辈，我们会看着前辈的身影，追寻着往这个方向一直努力下去。

目录

阿里 Load Balance 之旅

什么是负载均衡（Load Balance）

“负载(load)”惹得祸

首先让我们从原点出发。

某一天清晨起床，你突然有了一个美妙的想法，这个想法让你兴奋不已，因为你发现了一个具有广阔市场前景的处女地，这个处女地还完全没有被开垦，这是一项新业务，这项业务能满足世界上大部分人的某个方面他们自己都不知道的潜在的需求，更妙的是，到目前为止，它还没有被任何人发现。

你简直被自己的天才惊呆了，被自己的天才想法感动的内牛满面，你确信你的人生会因为这个想法而改变，宇宙也会因为你这个想法而变的大不相同，你确定这个想法的实现将能帮助你走上人生巅峰，上上个时代是比尔盖子，上个时代是拉里配齐，下个时代毫无疑问将是你，你迫不及待的想要将这项业务通过一个系统来表达和实现，于是你奋战了好几个通宵，有了下面这个能支撑这项新业务的非常牛逼的系统:

是的，你的系统已经有了第一个用户，这个用户就是你自己。接下来你将你的系统通过 X 信朋友圈、钉钉、各大论坛广邀各路亲朋好友试试这个系统，结果是，非常棒！好评如潮，他们不停的拉自己的亲朋好友开始访问这个系统，因此注册、活跃用户越来越多，此时，你兴奋不已，感觉自己马上就要走上人生巅峰。

但等等，不知道哪里出了问题，你的这个新系统居然上了 ccav 财经频道，而且被各路媒体小报纷纷报导，瞬间所有人都想看看这个牛逼的系统，于是乎,悲剧发生了:

怎么办？怎么办？很快你想到了 3 个解决方案。

纵向扩展（Scale Up）

没啥好说的，也许只是服务器还不够 NB，买！买！买！宇宙最好的服务器来上一台，可惜，创业刚开始，投资人的钱要花在刀刃上，比如广泛营销上，比如路边扫个码啊，顺便送个内衣啊、牙刷啊什么的，服务器？买不起！而且看了一下宇宙上最好服务器的网卡配置，更泄气，这就不是有钱买个 NB 服务器就能解决的事！

业务拆分

你仔细审视之后，发现其实你的系统的 2 个页面是 2 个不同的业务，用来满足不同的需求的，于是啊，你就想，是不是能把这 2 个业务分开到 2 个系统中去，这样用户们自然乖乖的被引导、分流到 2 个子系统去了，这样每个系统的压力就减少了啊。这就好比一个饭店，生意火爆，原先来吃火锅和吃大饼卷牛排的都在一起，挤不下之后，现在分成了 2 个店，1 个是火锅店，1 个是大饼卷牛排店。从某种角度来说，这种其实也是一种负载均衡，但怎么做业务拆分并且通过组织文化和人事架构去保障和适应这种业务拆分，是有很多的业务考量和业务属性在里面的，每个老板和每个行业的答案可能都是不同的，此文讨论的不在这个方向上。

横向扩展（Scale Out）和 副本(Replica)

做完上面的垂直拆分之后，可能你会发现还是不行啊，这世界上拥有独特口味的人太多，吃大饼卷牛排的人还是太多了，怎么办？开分店吧，每个上档次的商场都开一个大饼卷牛排店，这样每个地域的人又被分流到了附近的店。这个提供一模一样服务的“分店”就是系统的副本（Replica），分布式系统中的副本(Replica)除了满足数据冗余，容灾的需要等之外，横向扩展，通过开多个分店（Replica）分流的行为,就是负载均衡,做到多副本之间分流是一个重要的目的。而这个正是本文讨论的范围所在，如下简图:

接下来，关于如何让你的系统实现负载均衡，做到 Scale Out，你开始走上选型之路，

常用的负载均衡技术比较

DNS 轮询

DNS 本身的机制不再赘述，这里主要看一看基于 DNS 的负载均衡，其大致原理很清楚，DNS 系统本身支持同一个域名映射到多个 ip (A 记录）,例如

```
niubility.com.          IN      A       172.168.1.101
                        IN      A       172.168.1.102
                        IN      A       172.168.1.103
                        IN      A       172.168.1.104
```

这样每次向 DNS 系统询问该域名的 ip 地址时（Tell Me The IP Address of niubility.com.），DNS 会轮询(Round Robin)这个 ip 列表，每次给一个不同的 ip，从而达到负载均衡的效果。

来看看这种负载均衡解决方案的优缺点

优点

易于实现

对于应用系统本身几乎没有任何侵入，配置也很简单，在某个文本里多加几行 A 记录就可以。尤其对于一个基于 Web 的系统来说更是如此，用户在浏览器里输入的 URL host 部分天然就是域名，所以在某个环节你必然有起码一台 DNS 服务器记录着这个域名对应的 ip，所以可以说是基于已有域名系统(资产)就做到了负载均衡。

缺点

会话粘连 (Session Sticky)

客户端与目标系统之间一般存在会话的概念(不止是 web 系统的 http session)，其本质在于 server 端会或多或少的存一些客户端整个会话期间交互的身份识别以及数据信息，为了防止 server 端每次都对同一个客户端问一下，你是谁？系统会希望客户端在一个会话期间粘连在某个特定的 serer 上，除非这个 server 失败才 failover 到其它的 server 上，这种粘连特性对于 server 处理客户端请求处理的性能和客户端看到的数据一致性是有很大好处的。但是 DNS 负载均衡不能保证下一次请求会再次落在同一个 server 上。

DNS 解析缓存和 TTL 带来的麻烦

dns 记录的缓存以及缓存失效时间都是个问题，在无线时代，通常来自手机的访问会经过称为行业网关的代理服务器，由于代理服务器会将域名解析的结果缓存一段时间，所以所有经由这个代理服务器的访问请求就全被解析到同一台服务器上去了，因此就可能无法实现均等分配需要处理的请求了。另外在后端集群的拓扑结构(副本数、部署位置、健康状态等)发生变化之后，dns 配置的变化要等到网络上所有节点的缓存失效才能反馈出来，这带来的问题起码有 2 个，1 是在等待失效过程中，完全不可控，没有办法加快这个进程，中美切换要花 10 分钟，因为要等网络所有几点对某些域名的 TTL 失效，2 是滞后，有时候这种滞后是致命的，比如仍然有部分流量打到已经挂掉的那部分服务器上。

容错

一个大型数据中心，每天都有机器坏了是很正常的事情，尤其是在虚拟化大行其道的今天，更是如此，相信你对虚拟主机又崩溃了一个，或者总是被同宿主机的猪一样的队友“挤”死这种情况一定不陌生。dns 负载均衡的一大问题就在于这种情况下的容灾很麻烦，一是需要人工干预或者其他软件配合做健康监测，从 dns 配置中将无响应的机器或者崩溃的机器的相应的 A 记录删掉。一是删掉之

后也要等到所有网络节点上的 dns 解析缓存失效，在这端时间内，很多访问系统的客户会受到影响。

数据热点

dns 是在域名层面做负载均衡，如果从 web 系统的请求 URL 角度讲，不同的 URL 对后端 server 的压力强度不一样，dns 负载很可能会出现所有高强度的请求全都被打到小部分服务器甚至同一台上去了的情况，这个问题的可怕性不在于风险，而在于风险完全不可控。

引入负载均衡器

如图: 客户端 -> LB -> replica1,replica2,replica3

LB 负责客户端流量到后端服务集群的分发，一般 LB 也会负责后端所有 server 的健康监测，关于健康监测这一块我们在稍后一点再做具体分析。

优点

可以在 LB 里面做集中的分发逻辑，可以有多种分发方式，例如最常见的 Round Robin, Random Robin, Weight-Based Round Robin 等。

缺点

LB 是单点，这里其实是有 2 个问题，下面具体分解一下。

问题 1: 所有流量(请求流、响应流）都要经过 LB，量太大，LB 这小身板扛不住啊，尤其是响应流，一般远大于请求流，为什么? 我们想想一般一个 http 请求报文大小和响应的报文大小的对比就明白了。

解决方案:

响应流不走 LB 了!

问题 2: LB 是单点啊，挂了，所有流量都损失了

解决方案之一(LB Active-Standby 模式):

现在一些 LB 也会支持 Active-Active 模式，这里不再介绍。

四层 LB vs 七层 LB

四层 LB 的特点一般是在网络和网络传输层(TCP/IP)做负载均衡，而七层则是指在应用层做负载均衡。2 者的区别还是比较大的，各有优缺点，四层 LB 对于应用侵入比较小，这一层的 LB 对应用的感知较少，同时应用接入基本不需要针对 LB 做任何的代码改造。七层负载均衡一般对应用本身的感知比较多，可以结合一些通用的业务流量负载逻辑和容灾逻辑做成很细致的负载均衡和流量导向方案，但是一般接入时，应用需要配合做相应的改造。

互联网时代因为流量就是钱啊，所以对于流量的调度的细致程度往往是四层 LB 难以满足的，所以可以说七层负载均衡的解决方案现在是百花齐放，百家争鸣，中间层负载均衡(mid-tier load balancing)正当其时。

健康检测，负载均衡的伴侣

对于 LB 或者一个 LB 解决方案来说，健康检测是 LB 固有需要一起实现的需求之一，LB 要能帮助业务实现业务、服务器、容器的优雅上、下线，帮助业务实现透明的扩容、缩容等一系列很现实的功能，如下简图:

在云计算时代弹性计算(Elastic Compute Service)，按需伸缩(On-Demand Allocation)大行其道的今天，对于 LB 健康检测方案的灵敏度，准确性，多层次等等都提出了非常高的要求，别看健康检测这个功能貌似很简单，但是实现过程中如果有很多地方没有想到，很容易就造成系统的重大的故障，我们的一些系统在心跳检测上栽的跟头并不少，踩过很多的坑。

为什么我们要做 VIPServer

在有@正明（章文嵩博士的花名）这样的技术大牛，LVS 的原作者本尊坐镇的阿里，为什么阿里会再做一个负载均衡产品?其实很简单，为了解决实际的业务问题。

在集团,阿里技术团队常常面临的都是世界级难题，中间件团队更是如此，我们不敢吹牛逼说对这些问题我们解得多好，但我们真的在踏实的认真的解这些问题，而且用我们自己特有的方式，而不是 COPY 国外技术栈的方式在解这些难题，其实我们很多时候也很想 COPY 啊，但是放眼全球相关领域确实是没得抄。

言归正传，上面我们过了一遍常见的负载均衡方案之后，我们可以发现传统 LB，如 LVS 的解决方案中并不是把所有的问题都已经解决了，我们检视一下，起码还遗留了 6 个问题:

如果过 LB 的请求量就大到把 LB 给打挂了怎么办?互联网的流量，尤其是中国互联网的流量，我们要有足够的自信啊，而且参与过春节买票的，春晚修一修抢红包的都能想象得到。

LB 虽然可以有 standby 的方案或者有小规模集群能力，但如果 active/standby 同时挂了怎么办? 1 个蛋蛋很危险，但 2 个蛋蛋也未必就多安全。比如在 active-standby 方案中，既然 active 撑不住请求流量，那么作为其 clone 的 standby 身上当然也不会出现任何奇迹，那么是不是 LB 前面还应该再架一层 LB 呢?能不能 LB 集群全挂了的情况下，不影响正常的业务?

请求方和目标机器之间总是要过一次 LB,这在网络链路上是多了 1 跳，我们都知道多一跳可不光是 rt 的损耗那么简单，链路上从 1 跳到 2 跳，链路和连接出故障的概率也翻了一倍,这要怎么解？

多机房，多区域的异地多活与容灾，国际化战略的跨国流量的容灾对于负载均衡提出的挑战怎么解，在阿里集团内部，现在断网、断电、断机房的演习如日常喝水、像办公大楼消防演习一样随意，据说要达到，马老师半夜起来上个厕所，顺便断个电的能力，这些容灾场景下业务流量的负载均衡怎么解？

每次在一些如“秒杀”，“大促”等营销热点场景下，业务为了应对可以预期的流量洪峰，评估 LB 这一块的容量够不够、要扩多少的痛点又如何解决?LB 的弹性在哪里?

成本。虽然 LVS 比一些传统硬件 LB 的成本已经有很大的优势，但是在一个大型互联网系统级别的流量和业务发展面前,LVS 的使用成本还是太高了一点。

VIPServer 就是为了解决这些实际问题而生，所以上面介绍的这些我们耳熟能详的机制全都不是 VIPServer 解决问题的思路。

VIPServer 简介

VIPServer 是阿里中间件团队开发的一个中间层负载均衡(mid-tier load balancing)产品，VIPServer 是基于 P2P 模式，是一个七层负载均衡产品。

VIPServer 提供动态域名解析和负载均衡服务，支持很多诸如多业务单元同单元优先、同机房优先、同区域优先等一系列流量智能调度和容灾策略，支持多种健康监测协议，支持精细的权重控制，提供多级容灾体系、具有对称调用、健康阈值保护等保护功能的解决方案。是阿里负载均衡体系、域名服务体系的一个非常重要的组成部分。

目前阿里包括阿里妈妈、钉钉、搜索、AE、1688、阿里云、高德等几乎所有的业务线均在使用 VIPServer，在一些重大的项目例如历年双 11，支付宝春晚红包项目都发挥了重大的作用。

VIPServer 的实现原理，限于篇幅，这里不做详细介绍，具体可以参考我们发表的 VIPServer 的 Paper:

VIPServer: A System for Dynamic Address Mapping and Environment Management

为什么 VIPServer 在阿里发展这么快？

讲了这么多平实的技术的东西，让我们接着上面的故事往下叙述，帮助您理解 VIPServer 为何在发挥越来越重要的作用。我们看看上面的故事中，随着业务的发展还会发生一些什么事，遇到哪些挑战。

现在你的公司已经各方面都已经走上正轨，马上就要敲钟上市了，在业内，世界范围内已经有了广泛的社会影响力，已经成为一个新兴行业的风向标之一，很多

人都盯着你，你这个公司有任何的风吹草动，新闻从业人员肾上腺激素都会往外狂飙，你的任何一个系统挂了都会影响几亿的用户，产生千万的资产损失，那么问题来了，你的系统还敢随意的挂掉么？挂掉哪怕 1 分钟都是给竞争对手的一场狂欢，让自己的公关团队夙夜难眠！

好，像这么牛逼的公司你觉得你开始更多的需要考虑的是什么？1 个机房，全在 H 城？那怎么可以？!!且不说地球这么危险，地震、海啸频发，错峰用电、火灾时有发生，有时候周围的化工厂还会爆个炸啥的，有时候就是工地上的铲车看起来都那么可怕，尤其是 blueshi(r)t 人开的铲车尤其的可怕。好吧，看来你需要在多个城市，多个国家，多个地球上建设很多个机房。有了多个机房万事就 OK 了么？不，事情其实要比你想象的复杂的多，你的公司的已有应用都在一开始就支持异地容灾能力了么，做架构的都知道无状态的应用可能好一点，但是那些有状态的应用在这一块绝对是重灾区。

再从另一个视角来看这个问题，你的公司再过去的几年发展过程中会有各种的新需求，贵公司会有越来越多的新业务来满足这些新需求，而针对这些新业务当然会有更多的各种系统或者叫应用来满足和服务这些客户，但这些应用就像我们的手指头，并不是均衡的，并不是都一样长，这些系统，用户数不均衡，流量不均衡，耗费的计算资源不均衡，数据重要性、特征分布，容灾能力，跨地域部署能力等等全都不均衡。

某一天你好奇到底有多少个系统，调用链路是怎样的，他们的机房分布是怎样的？想评估一下 A 地的机房全挂了会影响多少系统，多少流量，产生多少资损？于是梳理了一下，这下傻眼了，可能是这个样子的：

如果更准确一点的话，这个图其实应该是个动态的 jpeg，因为每周都有系统在出生和消亡，有的应用版图在扩大，有的应用版图在缩小，而且从一定程度上来讲，这个图的变化的速度一定程度上反应了贵公司的整个系统跟随业务快速变化的能力和业务的活力，除了应用本身，数据中心每天都有物理机器在崩溃和修复，虚拟化之后更明显。

如果有这么一种负载均衡器，能在宏观上，某个切面上让所有的应用之间的调用满足如下的智能调度策略，将是一个多么美好的事情，而且最好是接入几乎不用怎么让已有的业务做代码的改造，这种改造是多么蛋疼的事情，有经验的平台架构师肯定身有体会，无需赘述。

同机房优先 A 应用调用 B 应用，负载均衡器负责调度，如果 B 应用有跟 A 部署在同一个机房部署的话，就优先路由到同机房的 B。

流量打散的容灾需求 昨天同机房的 A 和 B 玩的还好好的，但同机房的 B 应用说没有就没有了，有时候是凌晨睡的正香的时候没有了，手机上报警哗哗的，这时候你就想要是能流量自动打散到最近的机房就好了。

贵公司要的是千里之外的容灾，国际化的战略，而用户需要的是毫秒级的请求响应速度，跟随战略，这网络延迟蹭蹭的往上涨，网络质量蹭蹭的往下降，但另一个方面，现在用户的时间碎片化这么严重，毫秒级的延迟就可能导致用户失去耐心全去你的竞争对手的网站上去了，所以如何去弥合所有已有应用之间的跨机房、区域、国家的调用导致这 2 个方向上的裂隙?

你作为老板，想了一下，一个小小负载均衡器挂了，怎么可以影响到所有的业务呢？而作为有梦想的程序猿啊，出来混，系统要"稳"字当头，一个 3 天 2 头宕机的系统即使有价值也有限的很，而且很快会被更稳定的系统所替代，所以 LB 的 server 挂了还能保障业务继续运行的 LB 才是好的 LB.

VIPServer 并不完美

作为一个诞生刚几年的家伙，虽然发展确实很快，但它还只是个孩子，我们想说的是不光是用户这样看，其实在我们自己的眼里，它还有很多很多的不足甚至缺陷，但有@正明这样的前辈在前面的孜孜不倦、追求卓越的身影，我们这些后辈怎么可以懈怠，尸位素餐，怎么可以不持续改进自己的产品，在国内技术领域的技术积累贡献自己一点微薄的力量呢！《汉书·程序猿传》有云："今我辈程序猿，上不能匡主，下不能益民，皆可以称之为尸位素餐。"

-

本文由阿里中间件团队博客原创首发，戳阅读原文即可访问。

-
-

感谢“云栖社区”授权转载，ID：【yunqiinsight】

-