六大Web负载均衡原理与实现

2017-06-05 [架构之家](https://mp.weixin.qq.com/s?__biz=MzIyNjE4NDcyMA==&mid=2247484092&idx=1&sn=20484038dc12661dd4c6f0969a4e09c8&chksm=e87519c3df0290d5c8725d93ed36ccf2dc797b3a3a0f40565f815c2f68ca489551e2a4323630&scene=0&key=25d3550e373c8a764e915de43ca6f03d4b4ce9dede738776a1de3c65cb5158a7c8ccf88db76a6495fb88bb7e934f45e4a3964ccfb1916522b6c4be69295f1c1a07e1630322b967c8c6937dfa28ced1c4&ascene=1&uin=Mjg1Mzg5MzU%3D&devicetype=Windows+UnKnow&version=62040525&pass_ticket=DnykyS%2BwUoMPryB%2FpsE3Y1dLnzsop%2B2IGARhab3jjT4%3D&winzoom=1##)

开头先理解一下所谓的“均衡”

不能狭义地理解为分配给所有实际服务器一样多的工作量，因为多台服务器的承载能力各不相同，这可能体现在硬件配置、网络带宽的差异，也可能因为某台服务器身兼多职，我们所说的“均衡”，也就是希望所有服务器都不要过载，并且能够最大程序地发挥作用。

一、http重定向

当http代理（比如浏览器）向web服务器请求某个URL后，web服务器可以通过http响应头信息中的Location标记来返回一个新的URL。这意味着HTTP代理需要继续请求这个新的URL，完成自动跳转。

性能缺陷：

1、吞吐率限制

主站点服务器的吞吐率平均分配到了被转移的服务器。现假设使用RR（Round Robin）调度策略，子服务器的最大吞吐率为1000reqs/s，那么主服务器的吞吐率要达到3000reqs/s才能完全发挥三台子服务器的作用，那么如果有100台子服务器，那么主服务器的吞吐率可想而知得有大？相反，如果主服务的最大吞吐率为6000reqs/s，那么平均分配到子服务器的吞吐率为2000reqs/s，而现子服务器的最大吞吐率为1000reqs/s，因此就得增加子服务器的数量，增加到6个才能满足。

2、重定向访问深度不同

有的重定向一个静态页面，有的重定向相比复杂的动态页面，那么实际服务器的负载差异是不可预料的，而主站服务器却一无所知。因此整站使用重定向方法做负载均衡不太好。

我们需要权衡转移请求的开销和处理实际请求的开销，前者相对于后者越小，那么重定向的意义就越大，例如下载。你可以去很多镜像下载网站试下，会发现基本下载都使用了Location做了重定向。

二、DNS负载均衡

DNS负责提供域名解析服务，当访问某个站点时，实际上首先需要通过该站点域名的DNS服务器来获取域名指向的IP地址，在这一过程中，DNS服务器完成了域名到IP地址的映射，同样，这样映射也可以是一对多的，这时候，DNS服务器便充当了负载均衡调度器，它就像http重定向转换策略一样，将用户的请求分散到多台服务器上，但是它的实现机制完全不同。

使用dig命令来看下"baidu"的DNS设置

可见baidu拥有三个A记录

相比http重定向，基于DNS的负载均衡完全节省了所谓的主站点，或者说DNS服务器已经充当了主站点的职能。但不同的是，作为调度器，DNS服务器本身的性能几乎不用担心。因为DNS记录可以被用户浏览器或者互联网接入服务商的各级DNS服务器缓存，只有当缓存过期后才会重新向域名的DNS服务器请求解析。也说是DNS不存在http的吞吐率限制，理论上可以无限增加实际服务器的数量。

特性:

1、可以根据用户IP来进行智能解析。DNS服务器可以在所有可用的A记录中寻找离用记最近的一台服务器。

2、动态DNS：在每次IP地址变更时，及时更新DNS服务器。当然，因为缓存，一定的延迟不可避免。

不足：

1、没有用户能直接看到DNS解析到了哪一台实际服务器，加服务器运维人员的调试带来了不便。

2、策略的局限性。例如你无法将HTTP请求的上下文引入到调度策略中，而在前面介绍的基于HTTP重定向的负载均衡系统中，调度器工作在HTTP层面，它可以充分理解HTTP请求后根据站点的应用逻辑来设计调度策略，比如根据请求不同的URL来进行合理的过滤和转移。

3、如果要根据实际服务器的实时负载差异来调整调度策略，这需要DNS服务器在每次解析操作时分析各服务器的健康状态，对于DNS服务器来说，这种自定义开发存在较高的门槛，更何况大多数站点只是使用第三方DNS服务。

4、DNS记录缓存，各级节点的DNS服务器不同程序的缓存会让你晕头转向。

5、基于以上几点，DNS服务器并不能很好地完成工作量均衡分配，最后，是否选择基于DNS的负载均衡方式完全取决于你的需要。

三、反向代理负载均衡

这个肯定大家都有所接触，因为几乎所有主流的Web服务器都热衷于支持基于反向代理的负载均衡。它的核心工作就是转发HTTP请求。

相比前面的HTTP重定向和DNS解析，反向代理的调度器扮演的是用户和实际服务器中间人的角色：

1、任何对于实际服务器的HTTP请求都必须经过调度器

2、调度器必须等待实际服务器的HTTP响应，并将它反馈给用户（前两种方式不需要经过调度反馈，是实际服务器直接发送给用户）

特性：

1、调度策略丰富。例如可以为不同的实际服务器设置不同的权重，以达到能者多劳的效果。

2、对反向代理服务器的并发处理能力要求高，因为它工作在HTTP层面。

3、反向代理服务器进行转发操作本身是需要一定开销的，比如创建线程、与后端服务器建立TCP连接、接收后端服务器返回的处理结果、分析HTTP头部信息、用户空间和内核空间的频繁切换等，虽然这部分时间并不长，但是当后端服务器处理请求的时间非常短时，转发的开销就显得尤为突出。例如请求静态文件，更适合使用前面介绍的基于DNS的负载均衡方式。

4、反向代理服务器可以监控后端服务器，比如系统负载、响应时间、是否可用、TCP连接数、流量等，从而根据这些数据调整负载均衡的策略。

5、反射代理服务器可以让用户在一次会话周期内的所有请求始终转发到一台特定的后端服务器上（粘滞会话），这样的好处一是保持session的本地访问，二是防止后端服务器的动态内存缓存的资源浪费。

四、IP负载均衡(LVS-NAT)

因为反向代理服务器工作在HTTP层，其本身的开销就已经严重制约了可扩展性，从而也限制了它的性能极限。那能否在HTTP层面以下实现负载均衡呢？

NAT服务器:它工作在传输层，它可以修改发送来的IP数据包，将数据包的目标地址修改为实际服务器地址。

从Linux2.4内核开始，其内置的Neftilter模块在内核中维护着一些数据包过滤表，这些表包含了用于控制数据包过滤的规则。可喜的是，Linux提供了iptables来对过滤表进行插入、修改和删除等操作。更加令人振奋的是，Linux2.6.x内核中内置了IPVS模块，它的工作性质类型于Netfilter模块，不过它更专注于实现IP负载均衡。

想知道你的服务器内核是否已经安装了IPVS模块，可以

有输出意味着IPVS已经安装了。IPVS的管理工具是ipvsadm，它为提供了基于命令行的配置界面，可以通过它快速实现负载均衡系统。这就是大名鼎鼎的LVS(Linux Virtual Server，Linux虚拟服务器)。

1、打开调度器的数据包转发选项

echo 1 > /proc/sys/net/ipv4/ip\_forward

2、检查实际服务器是否已经将NAT服务器作为自己的默认网关，如果不是，如添加

route add default gw xx.xx.xx.xx

3、使用ipvsadm配置

ipvsadm -A -t 111.11.11.11:80 -s rr

添加一台虚拟服务器，-t 后面是服务器的外网ip和端口，-s rr是指采用简单轮询的RR调度策略（这属于静态调度策略，除此之外，LVS还提供了系列的动态调度策略，比如最小连接（LC）、带权重的最小连接（WLC），最短期望时间延迟（SED）等）

ipvsadm -a -t 111.11.11.11:80 -r 10.10.120.210:8000 -m

ipvsadm -a -t 111.11.11.11:80 -r 10.10.120.211:8000 -m  

添加两台实际服务器（不需要有外网ip），-r后面是实际服务器的内网ip和端口，-m表示采用NAT方式来转发数据包

运行ipvsadm -L -n可以查看实际服务器的状态。这样就大功告成了。

实验证明使用基于NAT的负载均衡系统。作为调度器的NAT服务器可以将吞吐率提升到一个新的高度，几乎是反向代理服务器的两倍以上，这大多归功于在内核中进行请求转发的较低开销。但是一旦请求的内容过大时，不论是基于反向代理还是NAT，负载均衡的整体吞吐量都差距不大，这说明对于一睦开销较大的内容，使用简单的反向代理来搭建负载均衡系统是值考虑的。

这么强大的系统还是有它的瓶颈，那就是NAT服务器的网络带宽，包括内部网络和外部网络。当然如果你不差钱，可以去花钱去购买千兆交换机或万兆交换机，甚至负载均衡硬件设备，但如果你是个屌丝，咋办？

一个简单有效的办法就是将基于NAT的集群和前面的DNS混合使用，比如５个100Mbps出口宽带的集群，然后通过DNS来将用户请求均衡地指向这些集群，同时，你还可以利用DNS智能解析实现地域就近访问。这样的配置对于大多数业务是足够了，但是对于提供下载或视频等服务的大规模站点，NAT服务器还是不够出色。

五、直接路由(LVS-DR)

NAT是工作在网络分层模型的传输层（第四层），而直接路由是工作在数据链路层（第二层），貌似更屌些。它通过修改数据包的目标MAC地址（没有修改目标IP），将数据包转发到实际服务器上，不同的是，实际服务器的响应数据包将直接发送给客户羰，而不经过调度器。

1、网络设置

这里假设一台负载均衡调度器，两台实际服务器，购买三个外网ip，一台机一个，三台机的默认网关需要相同，最后再设置同样的ip别名，这里假设别名为10.10.120.193。这样一来，将通过10.10.120.193这个IP别名来访问调度器，你可以将站点的域名指向这个IP别名。

2、将ip别名添加到回环接口lo上

这是为了让实际服务器不要去寻找其他拥有这个IP别名的服务器，在实际服务器中运行：

另外还要防止实际服务器响应来自网络中针对IP别名的ARP广播，为此还要执行：

echo "1" > /proc/sys/net/ipv4/conf/lo/arp\_ignore

echo "2" > /proc/sys/net/ipv4/conf/lo/arp\_announce

echo "1" > /proc/sys/net/ipv4/conf/all/arp\_ignore

echo "1" > /proc/sys/net/ipv4/conf/all/arp\_announce

配置完了就可以使用ipvsadm配置LVS-DR集群了

ipvsadm -A -t 10.10.120.193:80 -s rr

ipvsadm -a -t 10.10.120.193:80 -r 10.10.120.210:8000 -g

ipvsadm -a -t 10.10.120.193:80 -r 10.10.120.211:8000 -g

-g 就意味着使用直接路由的方式转发数据包

LVS-DR 相较于LVS-NAT的最大优势在于LVS-DR不受调度器宽带的限制，例如假设三台服务器在WAN交换机出口宽带都限制为10Mbps，只要对于连接调度器和两台实际服务器的LAN交换机没有限速，那么，使用LVS-DR理论上可以达到20Mbps的最大出口宽带，因为它的实际服务器的响应数据包可以不经过调度器而直接发往用户端啊，所以它与调度器的出口宽带没有关系，只能自身的有关系。而如果使用LVS-NAT，集群只能最大使用10Mbps的宽带。所以，越是响应数据包远远超过请求数据包的服务，就越应该降低调度器转移请求的开销，也就越能提高整体的扩展能力，最终也就越依赖于WAN出口宽带。

总的来说，LVS-DR适合搭建可扩展的负载均衡系统，不论是Web服务器还是文件服务器，以及视频服务器，它都拥有出色的性能。前提是你必须为实际器购买一系列的合法IP地址。

六、IP隧道(LVS-TUN)

基于IP隧道的请求转发机制：将调度器收到的IP数据包封装在一个新的IP数据包中，转交给实际服务器，然后实际服务器的响应数据包可以直接到达用户端。目前Linux大多支持，可以用LVS来实现，称为LVS-TUN，与LVS-DR不同的是，实际服务器可以和调度器不在同一个WANt网段，调度器通过IP隧道技术来转发请求到实际服务器，所以实际服务器也必须拥有合法的IP地址。

总体来说，LVS-DR和LVS-TUN都适合响应和请求不对称的Web服务器，如何从它们中做出选择，取决于你的网络部署需要，因为LVS-TUN可以将实际服务器根据需要部署在不同的地域，并且根据就近访问的原则来转移请求，所以有类似这种需求的，就应该选择LVS-TUN。

出处：http://m.blog.csdn.net/article/details?id=41478111