

1. 三次握手四次挥手**

1.1 为什么连接的时候是三次握手，关闭的时候却是四次握手？

答：因为当Server端收到Client端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。但是关闭连接时，当Server端收到FIN报文时，很可能并不会立即关闭SOCKET，所以只能先回复一个ACK报文，告诉Client端，“你发的FIN报文我收到了”。只有等到我Server端所有的报文都发送完了，我才能发送FIN报文，因此不能一起发送。故需要四步握手。

1.2 为什么TIME_WAIT状态需要经过2MSL(最大报文段生存时间)才能返回到CLOSE状态？

MSL (MaximumSegment Lifetime) ，TCP允许不同的实现可以设置不同的MSL值。第一，保证客户端发送的最后一个ACK报文能够到达服务器，因为这个ACK报文可能丢失，站在服务器的角度来看，我已经发送了FIN+ACK报文请求断开了，客户端还没有给我回应，应该是我发送的请求断开报文它没有收到，于是服务器又会重新发送一次，而客户端就能在这个2MSL时间段内收到这个重传的报文，接着给出回应报文，并且会重启2MSL计时器。第二，防止类似与“三次握手”中提到了的“已经失效的连接请求报文段”出现在本连接中。客户端发送完最后一个确认报文后，在这个2MSL时间中，就可以使本连接持续的时间内所产生的所有报文段都从网络中消失。这样新的连接中不会出现旧连接的请求报文。

1.3 为什么建立连接是三次握手，关闭连接确是四次挥手呢？

建立连接的时候，服务器在LISTEN状态下，收到建立连接请求的SYN报文后，把ACK和SYN放在一个报文里发送给客户端。而关闭连接时，服务器收到对方的FIN报文时，仅仅表示对方不再发送数据了但是还能接收数据，而自己也未必全部数据都发送给对方了，所以己方可以立即关闭，也可以发送一些数据给对方后，再发送FIN报文给对方来表示同意现在关闭连接，因此，己方ACK和FIN一般都会分开发送，从而导致多了一次。

1.4 为什么不能用两次握手进行连接？

答：3次握手完成两个重要的功能，既要双方做好发送数据的准备工作(双方都知道彼此已准备好)，也要允许双方就初始序列号进行协商，这个序列号在握手过程中被发送和确认。现在把三次握手改成仅需要两次握手，死锁是可能发生的。作为例子，考虑计算机S和C之间的通信，假定C给S发送一个连接请求分组，S收到了这个分组，并发送了确认应答分组。按照两次握手的协定，S认为连接已经成功地建立了，可以开始发送数据分组。可是，C在S的应答分组在传输中被丢失的情况下，将不知道S是否已准备好，不知道S建立什么样的序列号，C甚至怀疑S是否收到自己的连接请求分组。在这种情况下，C认为连接还未建立成功，将忽略S发来的任何数据分组，只等待连接确认应答分组。而S在发出的分组超时后，重复发送同样的分组。这样就形成了死锁。

1.5 如果已经建立了连接，但是客户端突然出现故障了怎么办？

TCP还设有一个保活计时器，显然，客户端如果出现故障，服务器不能一直等下去，白白浪费资源。服务器每收到一次客户端的请求后都会重新复位这个计时器，时间通常是设置为2小时，若两小时还没有收到客户端的任何数据，服务器就会发送一个探测报文段，以后每隔75分钟发送一次。若一连发送10个探测报文仍然没反应，服务器就认为客户端出了故障，接着就关闭连接。

1.6 为什么要三次握手？

保证可靠的核心就是双方都需要确认自己发送和接受信息的功能正常,但因为网络环境的不稳定性,这一秒能收发下一秒可能网络核心就发生严重拥塞,所以世界上不存在完全可靠的通信协议.两次握手会怎样?若建立连接只需两次握手,客户端并没有太大的变化,在获得服务端的应答后进入ESTABLISHED状态,即确认自己的发送和接受信息的功能正常.但如果服务端在收到连接请求后就进入ESTABLISHED状态,不能保证客户端能收到自己的信息,此时如果网络拥塞,客户端发送的连接请求迟迟到不了服务端,客户端便超时重发请求,如果服务端正确接收并确认应答,双方便开始通信,通信结束后释放连接.此时,如果那个失效的连接请求抵达了服务端,由于只有两次握手,服务端收到请求就会进入ESTABLISHED状态,等待发送数据或主动发送数据.但此时的客户端早已进入CLOSED状态,服务端将会一直等待下去,这样浪费服务端连接资源。

1.7 为什么要四次挥手?

TCP连接的释放一共需要四步,因此称为『四次挥手』.我们知道,TCP连接是双向的,因此在四次挥手中,前两次挥手用于断开一个方向的连接,后两次挥手用于断开另一方向的连接。**第一次挥手:**若A认为数据发送完成,则它需要向B发送连接释放请求.该请求只有报文头,头中携带的主要参数为:FIN=1,seq=u.此时,A将进入FIN-WAIT-1状态。1, FIN=1表示该报文段是一个连接释放请求。

2, seq=u,u-1是A向B发送的最后一个字节的序号。

第二次挥手:

B收到连接释放请求后,会通知相应的应用程序,告诉它A向B这个方向的连接已经释放.此时B进入CLOSE-WAIT状态,并向A发送连接释放的应答,其报文头包含:ACK=1,seq=v,ack=u+1.

ACK=1: 除TCP连接请求报文段以外,TCP通信过程中所有数据报的ACK都为1, 表示应答。

1, seq=v, v-1是B向A发送的最后一个字节的序号。

2, ack=u+1表示希望收到从第u+1个字节开始的报文段,并且已经成功接收了前u个字节.A收到该应答,进入FIN-WAIT-2状态,等待B发送连接释放请求。

第二次挥手完成后, A到B方向的连接已经释放, B不会再接收数据, A也不会再发送数据。但B到A方向的连接仍然存在, B可以继续向A发送数据。

第三次挥手:

当B向A发完所有数据后,向A发送连接释放请求,请求头中包含: FIN=1,ACK=1,seq=w,ack=u+1.随后B进入LAST-ACK状态。

第四次挥手:

A收到释放请求后,向B发送确认应答,此时A进入TIME-WAIT状态.该状态会持续2MSL时间,若该时间段内没有B的重发请求的话,就进入CLOSED状态,撤销TCB.当B收到确认应答后,也便进入CLOSED状态,撤销TCB。

1.8 为什么TCP客户端最后还要发送一次确认呢?

一句话, 主要防止已经失效的连接请求报文突然又传送到了服务器, 从而产生错误。如果使用的是两次握手建立连接, 假设有这样一种场景, 客户端发送了第一个请求连接并且没有丢失, 只是因为网络结点中滞留的时间太长了, 由于TCP的客户端迟迟没有收到确认报文, 以为服务器没有收到, 此时重新向服务器发送这条报文, 此后客户端和服务器经过两次握手完成连接, 传输数据, 然后关闭连接。此时此前滞留的那一次请求连接, 网络通畅了到达了服务器, 这个报文本该是失效的, 但是, 两次握手的机制

将会让客户端和服务端再次建立连接，这将导致不必要的错误和资源的浪费。如果采用的是三次握手，就算是那一次失效的报文传送过来了，服务端接受到了那条失效报文并且回复了确认报文，但是客户端不会再次发出确认。由于服务器收不到确认，就知道客户端并没有请求连接。

1.9 简述TCP三次握手的过程？

答：在TCP/IP协议中，TCP协议提供可靠的连接服务，采用三次握手建立一个连接。第一次握手：建立连接时，客户端发送syn包(syn=j)到服务器，并进入SYN_SEND状态，等待服务器确认。第二次握手：服务器收到syn包，必须确认客户的SYN (ack=j+1)，同时自己也发送一个SYN包 (syn=k)，即SYN+ACK包，此时服务器进入SYN_RECV状态。第三次握手：客户端收到服务器的SYN + ACK包，向服务器发送确认包ACK(ack=k+1)，此包发送完毕，客户端和服务端进入ESTABLISHED状态，完成三次握手。完成三次握手，客户端与服务器开始传送数据简版：首先A向B发SYN（同步请求），然后B回复SYN+ACK（同步请求应答），最后A回复ACK确认，这样TCP的一次连接（三次握手）的过程就建立了。三次握手我们先明确两个定义：

1， client为数据发送方

2， server为数据接收方

好，下面进行三次握手的总结：

1， client想要向server发送数据，请求连接。这时client想服务器发送一个数据包，其中同步位（SYN）被置为1，表明client申请TCP连接，序号为j。

2， 当server接收到了来自client的数据包时，解析发现同步位为1，便知道client是想要简历TCP连接，于是将当前client的IP、端口之类的加入未连接队列中，并向client回复接受连接请求，想client发送数据包，其中同步位为1，并附带确认位ACK=j+1，表明server已经准备好分配资源了，并向client发起连接请求，请求client为建立TCP连接而分配资源。

3， client向server回复一个ACK，并分配资源建立连接。server收到这个确认时也分配资源进行连接的建立。

那么问题来了为什么需要第三次握手？第三次握手失败了怎么办？三次握手有什么缺陷可以被黑客利用，用来对服务器进行攻击？

怎么防范这种攻击？

接下来进行一一解答。

1.9.1 为什么需要第三次握手？

答：如果没有第三次握手，可能会出现如下情况：如果只有两次握手，那么server收到了client的SYN=1的请求连接数据包之后，便会分配资源并且向client发送一个确认位ACK回复数据包。那么，如果在client与server建立连接的过程中，由于网络不顺畅等原因造成的通信链路中存在着残留数据包，即client向server发送的请求建立连接的数据包由于数据链路的拥塞或者质量不佳导致该连接请求数据包仍然在网络的链路中，这些残留数据包会造成如下危害危害：当client与server建立连接，数据发送完毕并且关闭TCP连接之后，如果链路中的残留数据包才到达server，那么server就会认为client重新发送了一次连接申请，便会回复ACK包并且分配资源。并且一直等待client发送数据，这就会造成server的资源浪费。

1.9.2 第三次握手失败了怎么办？

答：当client与server的第三次握手失败了之后，即client发送至server的确认建立连接报文段未能到达server，server在等待client回复ACK的过程中超时了，那么server会向client发送一个RST报文段并进入关闭状态，即：并不等待client第三次握手的ACK包重传，直接关闭连接请求，这主要是为了防止泛洪攻击，即坏人伪造许多IP向server发送连接请求，从而将server的未连接队列塞满，浪费server的资源。

1.9.3 三次握手有什么缺陷可以被黑客利用，用来对服务器进行攻击？

答：黑客伪造IP大量的向server发送TCP连接请求报文包，从而将server的半连接队列（上文所说的未连接队列，即server收到连接请求SYN之后将client加入半连接队列中）占满，从而使得server拒绝其他正常的连接请求。即拒绝服务攻击

1.9.4 怎么防范这种攻击？

1，缩短服务器接收客户端SYN报文之后的等待连接时间，即SYNtimeout时间，也就是server接收到SYN报文段，到最后放弃此连接请求的超时时间，将SYNtimeout设置的更低，便可以成倍的减少server的负荷，但是过低的SYNtimeout可能会影响正常的TCP连接的建立，一旦网络不通畅便可能导致client连接请求失败2，SYNcookie + SYN proxy 无缝集成（较好的解决方案）SYNcookie：当server接收到client的SYN之后，不立即分配资源，而是根据client发送过来的SYN包计算出一个cookie值，这个cookie值用来存储server返回给client的SYN+ACK数据包中的初始序列号，当client返回第三次握手的ACK包之后进行校验，如果校验成功则server分配资源，建立连接。SYNproxy代理，作为server与client连接的代理，代替server与client建立三次握手的连接，同时SYNproxy与client建立好了三次握手连接之后，确保是正常的TCP连接，而不是TCP泛洪攻击，那么SYNproxy就与server建立三次握手连接，作为代理（网关？）来连通client与server。（类似VPN了解一下。）

2. 七层模型

应用层 (Application)：网络服务与最终用户的一个接口。协议有：HTTP FTP TFTP SMTP SNMP DNS TELNET HTTPS POP3 DHCP
表示层 (Presentation Layer)：

数据的表示、安全、压缩。（在五层模型里面已经合并到了应用层）

格式有，JPEG、ASCII、DECOIC、加密格式等

会话层 (Session Layer)：

建立、管理、终止会话。（在五层模型里面已经合并到了应用层）

对应主机进程，指本地主机与远程主机正在进行的会话

传输层 (Transport)：

定义传输数据的协议端口号，以及流控和差错校验。

协议有：TCP UDP，数据包一旦离开网卡即进入网络传输层

网络层 (Network)：

进行逻辑地址寻址，实现不同网络之间的路径选择。

协议有：ICMP IGMP IP (IPV4 IPV6) ARP RARP

数据链路层 (Link):

建立逻辑连接、进行硬件地址寻址、差错校验等功能。（由底层网络定义协议）

将比特组合成字节进而组合成帧，用MAC地址访问介质，错误发现但不能纠正。

物理层 (Physical Layer) :

是计算机网络OSI模型中最低的一层。

物理层规定：为传输数据所需要的物理链路创建、维持、拆除而提供具有机械的，电子的，功能的和规范的特性。

简单的说，物理层确保原始的数据可在各种物理媒体上传输。局域网与广域网皆属第1、2层。物理层是OSI的第一层，它虽然处于最底层，却是整个开放系统的基础。物理层为设备之间的数据通信提供传输媒体及互连设备，为数据传输提供可靠的环境。如果您想要用尽量少的词来记住这个第一层，那就是“信号和介质”。

3. 简述raid0 raid1 raid5 三种工作模式的工作原理及特点

RAID，可以把硬盘整合成一个大磁盘，还可以在大磁盘上再分区，放数据 还有一个大功能，多块盘放在一起可以有冗余（备份） RAID整合方式有很多，常用的：0 1 5 10

RAID 0，可以是一块盘和N个盘组合 其优点读写快，是RAID中最好的 缺点：没有冗余，一块坏了数据就全没有了

RAID 1，只能2块盘，盘的大小可以不一样，以小的为准 10G+10G只有10G，另一个做备份。它有100%的冗余，缺点：浪费资源，成本高

RAID 5，3块盘，容量计算 $10 * (n-1)$ ，损失一块盘 特点，读写性能一般，读还好一点，写不好

冗余从好到坏：RAID1 RAID10 RAID 5 RAID0 性能从好到坏：RAID0 RAID10 RAID5 RAID1 成本从低到高：RAID0 RAID5 RAID1 RAID10

单台服务器：很重要盘不多，系统盘，RAID1 数据库服务器：主库：RAID10 从库 RAID5 RAID0（为了维护成本，RAID10） WEB服务器，如果没有太多的数据的话，RAID5, RAID0（单盘） 有多台，监控、应用服务器，RAID0 RAID5

我们会根据数据的存储和访问的需求，去匹配对应的RAID级别

4. LVS、Nginx、HAProxy有什么区别？工作中你怎么选择？

LVS：是基于四层的转发 HAProxy：是基于四层和七层的转发，是专业的代理服务器 Nginx：是WEB服务器，缓存服务器，又是反向代理服务器，可以做七层的转发

区别：LVS由于是基于四层的转发所以只能做端口的转发 而基于URL的、基于目录的这种转发LVS就做不了

工作选择：

HAProxy和Nginx由于可以做七层的转发，所以URL和目录的转发都可以做 在很大并发量的时候我们就要选择LVS，像中小型公司的话并发量没那么大 选择HAProxy或者Nginx足已，由于HAProxy由是专业的代理服务器 配置简单，所以中小型企业推荐使用HAProxy

5. Tomcat和Resin有什么区别，工作中你怎么选择？

区别：Tomcat用户数多，可参考文档多，Resin用户数少，可考虑文档少 最主要区别则是Tomcat是标准的java容器，不过性能方面比resin的要差一些 但稳定性和java程序的兼容性，应该是比resin的要好

工作中选择：现在大公司都是用resin，追求性能；而中小型公司都是用Tomcat，追求稳定和程序的兼容

6. 讲述一下Tomcat8005、8009、8080三个端口的含义？

8005==》关闭时使用 8009==》为AJP端口，即容器使用，如Apache能通过AJP协议访问Tomcat的8009端口 8080==》一般应用使用

7. 什么叫CDN？

- 即内容分发网络
- 其目的是通过在现有的Internet中增加一层新的网络架构，将网站的内容发布到 最接近用户的网络边缘，使用户可就近取得所需的内容，提高用户访问网站的速度

8. 什么叫网站灰度发布？

灰度发布是指在黑与白之间，能够平滑过渡的一种发布方式 AB test就是一种灰度发布方式，让一部用户继续用A，一部分用户开始用B 如果用户对B没有什么反对意见，那么逐步扩大范围，把所有用户都迁移到B上面 来 灰度发布可以保证整体系统的稳定，在初始灰度的时候就可以发现、调整问题，以保证其影响度

9. 简述DNS进行域名解析的过程？

用户要访问www.baidu.com，会先找本机的host文件，再找本地设置的DNS服务器，如果也没有的话，就去网络中找根服务器，根服务器反馈结果，说只能提供一级域名服务器.cn，就去找一级域名服务器，一级域名服务器说只能提供二级域名服务器.com.cn,就去找二级域名服务器，二级域服务器只能提供三级域名服务器.baidu.com.cn，就去找三级域名服务器，三级域名服务器正好有这个网站www.baidu.com，然后发给请求的服务器，保存一份之后，再发给客户端

10. 讲一下Keepalived的工作原理？

在一个虚拟路由器中，只有作为MASTER的VRRP路由器会一直发送VRRP通告信息, BACKUP不会抢占MASTER，除非它的优先级更高。当MASTER不可用时(BACKUP收不到通告信息) 多台BACKUP中优先级最高的这台会被抢占为MASTER。这种抢占是非常快速的(<1s)，以保证服务的连续性 由于安全性考虑，VRRP包使用了加密协议进行加密。BACKUP不会发送通告信息，只会接收通告信息

11. 讲述一下LVS三种模式的工作过程？

LVS 有三种负载均衡的模式，分别是VS/NAT (nat 模式) VS/DR(路由模式) VS/TUN (隧道模式)

一、NAT模式 (VS-NAT) 原理：就是把客户端发来的数据包的目的地址，在负载均衡器上换成其中一台RS的IP地址 并发至此RS来处理,RS处理完后把数据交给负载均衡器,负载均衡器再把数据包原IP地址改为自己的IP 将目的地址改为客户端IP地址即可期间,无论是进来的流量,还是出去的流量,都必须经过负载均衡器 优点：集群中的物理服务器可以使用任何支持TCP/IP操作系统，只有负载均衡器需要一个合法的IP地址 缺点：扩展性有限。当服务器节点（普通PC服务器）增长过多时,负载均衡器将成为整个系统的瓶颈 因为所有的请求包和应答包的流向都经过负载均衡器。当服务器节点过多时 大量的数据包都交汇在负载均衡器那，速度就会变慢！

二、IP隧道模式 (VS-TUN) 原理：首先要知道，互联网上的大多Internet服务的请求包很短小，而应答包通常很大 那么隧道模式就是，把客户端发来的数据包，封装一个新的IP头标记(仅目的IP)发给RS RS收到后,先把数据包的头解开,还原数据包,处理后,直接返回给客户端,不需要再经过 负载均衡器。注意,由于RS需要对负载均衡器发过来的数据包进行还原,所以说必须支持 IPTUNNEL协议，所以,在RS的内核中,必须编译支持IPTUNNEL这个选项 优点：负载均衡器只负责将请求包分发给后端节点服务器，而RS将应答包直接发给用户 所以，减少了负载均衡器的大量数据流动，负载均衡器不再是系统的瓶颈，就能处理很巨大的请求量 这种方式，一台负载均衡器能够为很多RS进行分发。而且跑在公网上就能进行不同地域的分发。 缺点：隧道模式的RS节点需要合法IP，这种方式需要所有的服务器支持“IP Tunneling” (IP Encapsulation)协议，服务器可能只局限在部分Linux系统上

三、直接路由模式 (VS-DR) 原理：负载均衡器和RS都使用同一个IP对外服务但只有DR对ARP请求进行响应 所有RS对本身这个IP的ARP请求保持静默也就是说,网关会把对这个服务IP的请求全部定向给DR 而DR收到数据包后根据调度算法,找出对应的RS,把目的MAC地址改为RS的MAC（因为IP一致） 并将请求分发给这台RS这时RS收到这个数据包,处理完成之后，由于IP一致，可以直接将数据返给客户 则等于直接从客户端收到这个数据包无异,处理后直接返回给客户端 由于负载均衡器要对二层包头进行改换,所以负载均衡器和RS之间必须在一个广播域 也可以简单的理解为在同一台交换机上 优点：和TUN（隧道模式）一样，负载均衡器也只是分发请求，应答包通过单独的路由方法返回给客户端 与VS-TUN相比，VS-DR这种实现方式不需要隧道结构，因此可以使用大多数操作系统做为物理服务器。 缺点：（不能说缺点，只能说是不足）要求负载均衡器的网卡必须与物理网卡在一个物理段上。

12. 如何重置mysql root密码？

一、在已知MYSQL数据库的ROOT用户密码的情况下，修改密码的方法： 1、在SHELL环境下，使用mysqladmin命令设置： `mysqladmin -u root -p password "新密码"` 回车后要求输入旧密码

2、在mysql>环境中,使用update命令，直接更新mysql库user表的数据： `Update mysql.user set password=password('新密码') where user='root'; flush privileges;` 注意：mysql语句要以分号”；”结束

3、在mysql>环境中，使用grant命令，修改root用户的授权权限。 `grant all on . to root@'localhost' identified by '新密码';`

二、如查忘记了mysql数据库的ROOT用户的密码，又如何做呢？ 方法如下： 1、关闭当前运行的mysqld服务程序： `service mysqld stop`（要先将mysqld添加为系统服务） 2、使用mysqld_safe脚本以安全模式（不加载授权表）启动mysqld 服务 `/usr/local/mysql/bin/mysqld_safe --skip-grant-table &` 3、使用空密码的root用户登录数据库，重新设置ROOT用户的密码 `# mysql -u root Mysql> Update mysql.user set password=password('新密码') where user='root'; Mysql> flush privileges;`

lvs/nginx/haproxy优缺点

Nginx的优点是： 1、工作在网络的7层之上，可以针对http应用做一些分流的策略，比如针对域名、目录结构 它的正则规则比HAProxy更为强大和灵活，这也是它目前广泛流行的主要原因之一 Nginx单凭这点可利用的场合就远多于LVS了。

2、Nginx对网络稳定性的依赖非常小，理论上能ping通就能进行负载功能，这个也是它的优势之一 相反LVS对网络稳定性依赖比较大，这点本人深有体会；

3、Nginx安装和配置比较简单，测试起来比较方便，它基本能把错误用日志打印出来 LVS的配置、测试就要花比较长的时间了，LVS对网络依赖比较大。

4、可以承担高负载压力且稳定，在硬件不差的情况下一般能支撑几万次的并发量，负载度比LVS相对小些。

5、Nginx可以通过端口检测到服务器内部的故障，比如根据服务器处理网页返回的状态码、超时等等，并且会把返回错误的请求重新提交到另一个节点，不过其中缺点就是不支持url来检测。比如用户正在上传一个文件，而处理该上传的节点刚好在上传过程中出现故障，Nginx会把上传切到另一台服务器重新处理，而LVS就直接断掉了 如果是上传一个很大的文件或者很重要的文件的话，用户可能会因此而不满。

6、Nginx不仅仅是一款优秀的负载均衡器/反向代理软件，它同时也是功能强大的Web应用服务器 LNMP也是近几年非常流行的web架构，在高流量的环境中稳定性也很好。

7、Nginx现在作为Web反向加速缓存越来越成熟了，速度比传统的Squid服务器更快，可考虑用其作为反向代理加速器

8、Nginx可作为中层反向代理使用，这一层面Nginx基本上无对手，唯一可以对比Nginx的就只有lighttpd了 不过lighttpd目前还没有做到Nginx完全的功能，配置也不那么清晰易读，社区资料也远远没Nginx活跃

9、Nginx也可作为静态网页和图片服务器，这方面的性能也无对手。还有Nginx社区非常活跃，第三方模块也很多

Nginx的缺点是： 1、Nginx仅能支持http、https和Email协议，这样就在适用范围上面小些，这个是它的缺点 2、对后端服务器的健康检查，只支持通过端口来检测，不支持通过url来检测 不支持Session的直接保持，但能通过ip_hash来解决

LVS：使用Linux内核集群实现一个高性能、高可用的负载均衡服务器 它具有很好的可伸缩性 (Scalability)、可靠性 (Reliability)和可管理性 (Manageability)

LVS的优点是： 1、抗负载能力强、是工作在网络4层之上仅作分发之用，没有流量的产生 这个特点也决定了它在负载均衡软件里的性能最强的，对内存和cpu资源消耗比较低

2、配置性比较低，这是一个缺点也是一个优点，因为没有可太多配置的东西 所以并不需要太多接触，大大减少了人为出错的几率

3、工作稳定，因为其本身抗负载能力很强，自身有完整的双机热备方案 如LVS+Keepalived，不过我们在项目实施中用得最多的还是LVS/DR+Keepalived

4、无流量，LVS只分发请求，而流量并不从它本身出去，这点保证了均衡器IO的性能不会收到大流量的影响。 5、应用范围较广，因为LVS工作在4层，所以它几乎可对所有应用做负载均衡，包括http、数据库、在线聊天室等

LVS的缺点是： 1、软件本身不支持正则表达式处理，不能做动静分离 而现在许多网站在这方面都有较强的需求，这个是Nginx/HAProxy+Keepalived的优势所在 2、如果是网站应用比较庞大的话，LVS/DR+Keepalived实施起来就比较复杂了 特别后面有Windows Server的机器的话，如果实施及配置还有维护过程就比较复杂了 相对而言，Nginx/HAProxy+Keepalived就简单多了。

HAProxy的特点是： 1、HAProxy也是支持虚拟主机的。 2、HAProxy的优点能够补充Nginx的一些缺点，比如支持Session的保持，Cookie的引导 同时支持通过获取指定的url来检测后端服务器的状态

3、HAProxy跟LVS类似，本身就只是一款负载均衡软件 单纯从效率上来讲HAProxy会比Nginx有更出色的负载均衡速度，在并发处理上也是优于Nginx的

4、HAProxy支持TCP协议的负载均衡转发，可以对MySQL读进行负载均衡 对后端的MySQL节点进行检测和负载均衡，大家可以用LVS+Keepalived对MySQL主从做负载均衡

5、HAProxy负载均衡策略非常多，HAProxy的负载均衡算法现在具体有如下8种： ①roundrobin，表示简单的轮询，这个不多说，这个是负载均衡基本都具备的； ② static-rr，表示根据权重，建议关注； ③leastconn，表示最少连接者先处理，建议关注； ④ source，表示根据请求源IP，这个跟Nginx的IP_hash机制类似 我们用其作为解决session问题的一种方法，建议关注； ⑤ri，表示根据请求的URI； ⑥rl_param，表示根据请求的URI参数'balance url_param' requires an URL parameter name； ⑦hdr(name)，表示根据HTTP请求头来锁定每一次HTTP请求； ⑧rdp-cookie(name)，表示根据cookie(name)来锁定并哈希每一次TCP请求。

14. mysql数据备份工具

mysqldump工具 mysqldump是mysql自带的备份工具，目录在bin目录下

面： /usr/local/mysql/bin/mysqldump 支持基于innodb的热备份，但是由于是逻辑备份，所以速度不是很快，适合备份数据比较小的场景 Mysqldump完全备份+二进制日志可以实现基于时间点的恢复。

基于LVM快照备份 在物理备份中，有基于文件系统的物理备份（LVM的快照），也可以直接用tar之类的命令对整个数据库目录 进行打包备份，但是这些只能进行冷备份，不同的存储引擎备份的也不一样，myisam自动备份到表级别 而innodb不开启独立表空间的话只能备份整个数据库。

tar包备份

percona提供的xtrabackup工具 支持innodb的物理热备份，支持完全备份，增量备份，而且速度非常快，支持innodb存储引起的数据在不同 数据库之间迁移，支持复制模式下的从机备份恢复备份恢复，为了让xtrabackup支持更多的功能扩展 可以设立独立表空间，打开 innodb_file_per_table功能，启用之后可以支持单独的表备份

15. 统计ip访问情况，要求分析nginx访问日志，找出访问页面数量在前十位的ip

```
cat access.log | awk '{print $1}' | uniq -c | sort -rn | head -10
```

16. 使用tcpdump监听主机为192.168.1.1，tcp端口为80的数据，同时将输出结果保存输出到tcpdump.log

```
tcpdump 'host 192.168.1.1 and port 80' > tcpdump.log
```

17. 你对现在运维工程师的理解和以及对其工作的认识

运维工程师在公司当中责任重大，需要保证时刻为公司及客户提供最高、最快、最稳定、最安全的服务 运维工程师的一个小小的失误，很有可能会对公司及客户造成重大损失 因此运维工程师的工作需要严谨及富有创新精神

18. 实时抓取并显示当前系统中tcp 80端口的网络数据信息，请写出完整操作命令

```
tcpdump -nn tcp port 80
```

19. Linux系统中病毒怎么解决

1) 最简单有效的方法就是重装系统 2) 要查的话就是找到病毒文件然后删除 中毒之后一般机器cpu、内存使用率会比较高 机器向外发包等异常情况，排查方法简单介绍下

top 命令找到cpu使用率最高的进程 一般病毒文件命名都比较乱，可以用 ps aux 找到病毒文件位置 rm -f 命令删除病毒文件 检查计划任务、开机启动项和病毒文件目录有无其他可以文件等

3) 由于即使删除病毒文件不排除有潜伏病毒，所以最好是把机器备份数据之后重装一下

20. 你常用的Nginx模块，用来做什么

rewrite模块，实现重写功能 access模块：来源控制 ssl模块：安全加密 ngx_http_gzip_module：网络传输压缩模块 ngx_http_proxy_module 模块实现代理 ngx_http_upstream_module模块实现定义后端服务器列表 ngx_cache_purge实现缓存清除功能

21. 请列出你了解的web服务器负载架构

- Nginx
- Haproxy
- Keepalived
- LVS

22. 查看http的并发请求数与其TCP连接状态

```
netstat -n | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a}]'
```

还有ulimit -n 查看linux系统打开最大的文件描述符，这里默认1024 不修改这里web服务器修改再大也没用，若要用就修改很几个办法，这里说其中一个： 修改/etc/security/limits.conf

- soft nofile 10240
- hard nofile 10240 重启后生效

23. 用tcpdump嗅探80端口的访问看看谁最高

```
tcpdump -i eth0 -tnn dst port 80 -c 1000 | awk -F"." '{print $1"."$2"."$3"."$4}' | sort | uniq -c | sort -nr | head -20
```

24. 写一个脚本，实现判断192.168.1.0/24网络里，当前在线的IP有哪些，能ping通则认为在线

```
#!/bin/bash
for ip in seq 1 255
do
{
ping -c 1 192.168.1.$ip > /dev/null 2>&1
if [ $? -eq 0 ]; then
echo 192.168.1.$ip UP
else
echo 192.168.1.$ip DOWN
fi
}&
done
wait
```

25. 如何优化 Linux系统（可以不说太具体）？

- 不用root，添加普通用户，通过sudo授权管理
- 更改默认的远程连接SSH服务端口及禁止root用户远程连接
- 定时自动更新服务器时间
- 配置国内yum源
- 关闭selinux及iptables（iptables工作场景如果有外网IP一定要打开，高并发除外）
- 调整文件描述符的数量
- 精简开机启动服务（crond rsyslog network sshd）
- 内核参数优化（/etc/sysctl.conf）
- 更改字符集，支持中文，但建议还是用英文字符集，防止乱码
- 锁定关键系统文件
- 清空/etc/issue，去除系统及内核版本登录前的屏幕显示

26. 每天晚上 12 点，打包站点目录/var/www/html 备份到/data 目录下（最好每次备份按时间生成不同的备份包）

```
cat a.sh
/bin/bash
cd /var/www/ && /bin/tar zcf /data/html-date +%m-%d%H.tar.gz html/

crontab -e
00 00 * * * /bin/sh /root/a.sh
```

27. 写一个sed命令，修改/tmp/input.txt文件的内容，要求：

1. 删除所有空行；
2. 一行中，如果包含“11111”，则在“11111”前面插入“AAA”，在“11111”后面插入“BBB”，比如：
将内容为00001111122222的一行改为：0000AAA11111BBB2222

```
[root@~]# cat -n /tmp/input.txt
1 000011111222
2
3 000011111222222
4 11111000000222
5
6
7 11111111111112222222222
8 2211111111
9 112222222
10 1122
11
删除所有空行命令
[root@~]# sed '/^$/d' /tmp/input.txt
000011111222
000011111222222
11111000000222
11111111111112222222222
2211111111
112222222
1122
插入指定的字符
[root@~]# sed 's#\([11111\)\)#AAA\1BBB#g' /tmp/input.txt
0000AAA11111BBB222
0000AAA11111BBB222222
AAA11111BBB000000222
AAA11111BBBAAA11111BBB1112222222222
22AAA11111BBB111
112222222
1122
```

28. 写出如何给apache增加virtualhost，让访问<http://www.test.com>和<http://www.test.cn>的时候，都打开/var/www/html目录下面的文件：

```
ServerAdmin admini@abc.com
DocumentRoot "/var/www/html"
ServerName www.test.com
ServerAlias test.cn
ErrorLog "logs/bbs-error_log"
CustomLog "logs/bbs-access_log" common
```

29. 用一条命令显示本机eth0网卡的IP地址，不显示其它字符

方法一：

```
[root@apache ~]# ifconfig eth0|grep "inet addr"|awk -F '[:]+' '{print $4}'
192.168.1.22
```

方法二：

```
[root@apache ~]# ifconfig eth0|awk -F '[:]+' 'NR==2 {print $4}'
192.168.1.22
```

方法三：

```
[root@apache ~]# ifconfig eth0|sed -n '2p'|sed 's#^.*addr:##g'|sed 's#Bc.*$##g'
192.168.1.22
```

方法四：

```
[root@apache ~]# ifconfig eth0|sed -n '2p'|sed -r 's#^.*addr:(.*) Bc.*$#\1#g'
192.168.1.22
```

30. 写出一个curl命令，访问指定服务器61.135.169.121上的如下URL：<http://www.baidu.com/s?wd=test>，访问的超时时间是20秒：

```
curl -connect-timeout 20 http://61.135.169.121/s?wd=test
```

31. 编写脚本实现将/usr/local/test目录下大于100K文件，将它拷贝到/tmp目录下

```
#!/bin/bash
for file in `ls /usr/local/test`
do
if [ -f $file ];then
    if [ `ls -l $file`|awk ' {print $5} ' -gt 10000];then
        mv $file /tmp/
    fi
fi
done
```