- **我们的网站/app/游戏登录访问很慢，怎么解决?**
- 访问很慢是一个很普遍的问题，根据我的经验来说，问题多半出在程序和数据库上，往往都是性能问题或者是优化不到位。
- 排错过程是这样的，假设我们的架构外围是CDN，源站前端是nginx+tomcat的负载均衡，后端是mysql的主从，从服务器还使用了haproxy的四层代理，再加上redis的数据缓存。
- 我们先通过hosts指定nginx的ip来访问，如果访问速度很快则是CDN出现问题，很慢的话，那么源站肯定有问题，需要进一步排查。往下面的tomcat指定访问，如果访问很快则是nginx出现问题，很慢则说明tomcat以下的架构出现问题，就这样依次将问题服务器找出来。
- 如果是CDN出现问题，我们需要联系CDN公司帮忙获取并分析日志，看看是否有异常情况。
- 如果是nginx出现问题，就查看一下访问和错误日志是不是有异常情况，有可能是遭到了爬虫，需要配置一下反爬虫，在配置文件里面加入if $http_user_agent 如果匹配百度蜘蛛啊、etao蜘蛛啊什么的就返回一个503拒绝连接回去。nginx还有可能是因为timewait过多导致访问变慢，这时候需要先将可用端口范围加大，在/proc/sys/net/ipv4/ip_local_port_range文件里面加入端口范围，加到最大就1024到65535这样，因为1024以内的端口一般是系统级的进程占用的。然后将回收打开，更改/proc/sys/net/ipv4/tcp_tw_recycle文件的内容为1，这样timewait占用的进程就会被回收，从而加快访问速度。
- 如果是tomcat出现问题，也需要查看tomcat的运行和错误日志。有可能是内存溢出的问题，需要将jvm的内存加大，在tomcat的bin下面的catalina.sh里面将java_opts的xms和xmx值改大，这两个参数代表着jvm的最小和最大内存数，以我的经验来说，应该要将它们调成一样的大小。还有可能是toomanyopenfiles的问题，需要将系统的文件句柄数加大，先使用ulimit -n临时改变文件句柄数的大小，然后在/etc/security/limit.conf文件里面写入以便永久生效。
- 如果是mysql出现问题，有可能是查询语句不够优化引起的，这个时候就需要去与开发人员进行沟通去优化查询语句。
- 当然，不管是哪个服务的问题，都需要检查系统的性能指标。free/vmstat命令查看内存情况，sar命令查看cpu情况，iostat/iotop命令查看io情况，ifstat/iftop查看带宽情况，uptime查看负载情况等等。
- **我们的网站访问有时候快有时候慢，怎么解决?**
- 如果这个快慢的时间是有规律的话，那么就应该是有个系统的可循环crontab计划任务或者程序循环执行什么任务去了。前者我们需要查看系统相应时间的计划任务情况，后者就需要去找开发人员咨询一下，看看是否有这样的业务程序在运行。

- 如果这个时间是没有规律的话，那么我们就应该监控变慢时候的所有系统的情况，看看哪些机器有问题，可能是内存、cpu、磁盘io、网络io、负载等有问题。定位到具体机器后就去查看访问变慢时的系统日志，服务访问/错误日志等。通常都是内存不够，cpu占用过大等，如果找不到的话，我们可以自定义一个后台计划任务，每隔五分钟将top命令的输出重定向到一个文件。使用top -b -n 1 重定向到一个文件，然后再去分析具体是哪个进程有问题。
- **squid的工作原理是什么？CDN的原理是什么？DNS服务器的工作原理是什么?**
- squid的工作原理是这样的，首先客户端向squid请求一个页面，squid会查询本地的缓存信息，如果本地有的话就会将页面直接发送给客户端，如果没有的话就会向源站进行请求，得到页面后将页面发送给客户端，同时在本地保存一份。
- 每一台squid代理服务器都有若干个硬盘，每个硬盘又被分成很多区，每一个分区又建立了很多个目录，目录下才放入文件，squid将其视为object。squid通过查询表的方式定位一个资源的位置，所查询的表为hash table和digest table。digest表可以称为索引，它记录了每个磁盘、分区、目录里的缓存摘要，hash表可以称为目录，它记录所有digest表的信息。squid接收请求后首先查询hash表，然后根据hash表指向的digest表查询需要的信息。
- CDN主要包括两部分内容，智能DNS和缓存节点。
- 说智能DNS前首先说一下普通DNS的工作原理。DNS是域名系统的缩写，它能将域名解析成为ip，起到一个域名解析的作用。客户端提出域名解析请求，本地DNS接收到请求后首先查看本地的缓存，如果有记录则直接将ip返回
- 如果本地没有记录，则本地DNS会将请求转发到根域名服务器，根域名服务器会把下级域名的信息发送给本地域名服务器，本地DNS根据信息请求下级域名服务器，直到得到正确的ip信息返回给客户端。
- 普通的DNS直接返回的是ip地址，而做了CDN的权威DNS呢返回的是一个cname记录，记录的是CDN厂商的域名，CDN厂商有一个智能DNS，会根据用户的ip返回给用户一个最近或者最佳的CDN节点的IP地址给本地DNS，然后返回给用户，从而完成访问。
    - **raid的原理，如何创建的，说出LVM逻辑卷扩容的具体操作，怎么把卷组的空间全部给逻辑卷?**
- RAID技术主要包含RAID 0到7等等数个规范，它们的侧重点各不相同，常见的规范有raid0、raid1、raid5、raid10等等。
- raid0需要两个以上的硬盘，它连续地分割数据并并行地读/写于多个磁盘上。因此具有很高的数据传输率。但是它的安全性很低，如果一个磁盘失效，那么所有的数据都将变得不可用。它适合网吧这类不依赖安全性的情况。

- raid1需要两个硬盘，它是通过数据镜像的方式在两个分离的磁盘上产生互为备份的数据。因此它的安全性很高，一个失效了另一个有完整的数据。但是由此产生的费用很高。它适合运行安全性需求高的业务，用来备份数据。
- raid5需要三个硬盘，它交叉地存储数据以及它们的奇偶校验信息，它的安全性很好，一个磁盘失效也可以根据奇偶校验信息将数据还原出来，但是它的写的性能很低，一次写操作将产生四个读写操作，两次读旧的数据和校验数据，两次写新的数据和校验数据。它适合运行CDN缓存业务机器上。
- raid10是raid1和raid0的结合体，它是利用奇偶校验实现条带集镜像，所以它继承了Raid0的快速和Raid1的安全。这种新结构的价格高，可扩充性不好。主要用于容易、不大，但要求速度和差错控制的数据库中。
- raid是使用mdadm命令来创建的，它是multiple devices admin 的简称。mdadm -C 表示创建阵列，-ayes表示同意创建，-l加数字表示创建的阵列的等级是raid几，-n表示使用几块活动中的磁盘，-x表示备用磁盘。
- 假设卷组的空间都用完了，我们需要将逻辑卷扩容。这时候就需要先将vg扩容，使用pvcreate命令将一个新的磁盘创建为pv，然后使用vgextend命令将新的pv加入到需要扩容的vg中。vg有空间了我们就可以将逻辑卷扩容了，使用lvextend -l表示想加大多少空间，后面接加100%FREE可以将卷组的空间全部给逻辑卷，但是这时lv并不会立即扩大，我们需要运行resize2fs命令才能将本次扩容生效。最后可以使用df -h命令查看是否扩容成功。
- **七层的负载均衡和四层的有什么区别?**
- 四层的负载均衡基于报文中的目标ip和端口，七层的负载均衡基于报文中的应用层信息，比如url，http头，也叫“内容交换”。
- 相对于四层来说，七层的灵活性更好。
- 它们和客户端建立tcp连接三次握手的方式不一样。四层负载均衡在收到客户端的第一个syn请求后，会将报文中的目标ip地址进行修改，直接转发给服务器，所以tcp的三次握手建立连接实际上是客户端直接与服务端建立的，四层负载均衡只是起到一个类似路由的转发作用；而七层负载均衡需要根据真正的应用层内容选择服务器，所以它会首先跟客户端三次握手建立连接之后，接收到信息之后再根据内容与相应的服务器进行连接。
- 所以面对ddos攻击时，七层负载均衡会更加安全，因为ddos攻击的原理是这样的，它利用了tcp实现上的一个缺陷，首先它发送大量的syn请求给服务端，但是在握手的第三步时不给服务端发送确认连接的信息，导致服务器不停地重试并等待一个同步超时的时间，这样就会占用服务器大量的资源和带宽等等，严重影响到正常用户的访问。如果是四层，ddos攻击会越过代理直接攻击到后端的服务器，但是七层的话攻击只会影响到代理服务器，从而达到保护后端服务器的作用。

- 当然，负载均衡还有二层和三层的。二层负载均衡也就是数据链路层负载均衡，它是基于mac地址的，先通过一个虚拟的mac地址接收请求然后再分配到一个真实的mac地址上。一般是通过链路聚合技术实现负载均衡的。三层负载均衡也就是网络层负载均衡，它是基于ip的，先通过一个虚拟ip接收请求，然后再分配到一个真实的ip上。常用的有ospf负载均衡和rip负载均衡。
- **haproxy的调度算法有哪些？lvs、nginx、haproxy的优缺点**
- haproxy的调度算法可以通过修改balance字段的值来改变。它跟nginx的调度算法相似，有权重轮询roudrobin算法，有类似于ip_hash的source算法，可以用来解决session问题，有类似于url_hash的uri算法，可以用来提高缓存命中率。还有leastconn算法，将请求转发到连接最少的服务上，hdr算法，根据http请求头来锁定每一次http请求。
- lvs的抗负载能力强，性能很高，对内存cpu等资源的消耗很低，它的稳定性和可靠性很高，但是它不支持正则处理，不能做动静态分离，而且它的配置相对复杂，对网络的要求较高。
- nginx的配置比较简单，测试起来很方便，它对网络的要求不高，而且抗负载能力不弱。但是它对后端的健康检查，只支持通过端口的方式，不支持通过url检测。
- haproxy支持url的后端检查。它的负载均衡效率要比nginx更好，但是它无法用做web服务器以及缓存服务器。
- **lvs的三种工作模式是怎么样的，它们的工作原理是？分别作用正在第几层？lvs的单点故障是怎么解决的?**
- lvs的三种工作模式分别是nat模式隧道模式和dr模式。
- nat即网络地址转换，工作在三层网络层，首先客户端将数据包发送给vip，调度器lb将会接收到信息并根据调度算法决定将请求转发给后端的某个真实服务器，它会将请求数据包的目标ip转换成为后端真实服务器的ip，这样后端服务器就会收到信息并进行处理，然后根据默认路由将响应发送到lb，lb会将响应的源地址转换成为vip，再发送给客户端。
- 隧道模式工作在三层网络层，它与nat不同的是，它在lb和真实服务器之间的传输不用改写ip地址。而是将客户的请求包封装在一个ip隧道里，然后再发送到真实服务器，这个隧道包的目标地址是vip，所以真实服务器需要在网卡上绑定vip的IP地址，如果绑定了就处理这个包，如果没有会直接丢掉，处理完请求后，真实服务器会将响应包通过自己的外网地址直接发送给客户而不经过lb。
- dr直接路由模式工作在二层数据链路层，这个模式下lb和真实服务器使用同一个ip对外服务，它将客户端发送的请求报文直接路由给真实服务器，不转换ip也不封装，而是根据各个真实服务器的负载、连接数情况动态的选择一台，将请求报文的目

标mac地址改为这台服务器的mac地址。服务器处理请求后根据路由信息将响应数据包发送给客户端。

- 章文嵩博士去了淘宝之后又开发出了第四种工作模式，就是fullnat模式。它和nat的区别在于，它需要做四次地址转换。客户端请求到vip需要一次dnat，将客户端目的ip转换为lbip，然后从lbip到真实ip又一次，这只是将数据包传到真实服务端，响应包也同样需要两次转换传到客户端。这种方式下，lvs和真实服务器可以不在一个vlan中，大大提高了运维部署的便利性。
- lvs的单点故障可以通过keepalived的高可用来解决。keepalived是借用vrrp协议来实现高可用的，vrrp的出现就是为了解决静态路由的单点故障问题的。
- **haproxy是怎么做负载均衡的?**
- haproxy可以做四层也可以做七层的负载均衡。四层负载均衡是这样的，修改配置文件，将默认的mod设置为tcp，在listen端bind本地ip和端口，然后再添加轮询方式balance，在下面添加server组，加入我们的数据库从服务器ip和端口，这样就能做到四层的负载均衡了。七层负载均衡的话需要将默认的mod设置为http，我们可以设置动静分离，在frontend下面加入acl访问控制，匹配所有的img、png等等静态文件的请求就转发到静态池，其他的请求转发到动态池。静态和动态池在bakend里面指定balance轮询方式、server 加上IP地址端口，就做到了七层的负载均衡。
- **osi七层，tcp三次握手，linux启动过程**
- osi七层参考模型是国际标准化组织制定的一个标准体系。第一层是物理层 。它主要起到建立、维护、断开物理连接的作用，协议有ieee 802系列标准。第二层是数据链路层。它主要有建立逻辑链接、进行硬件地址寻址、差错校验等功能，协议有arp、rarp等等。第三层是网络层。它主要是进行逻辑地址寻址，实现不同网络之间的路径选择。协议有ip协议，icmp协议等等。第四层传输层。它可以定义传输数据的协议端口号，以及控流和差错校验。协议有tcp、udp等等。第五层会话层。主要是对会话的建立、管理以及终止等，协议有ssl、rpc等等。第六层是表示层。主要功能是对数据的表示、安全和压缩，主要协议有lpp轻量级表示协议等等。第七层是应用层。它是网络服务与用户之间的一个接口。它的协议有很多，主要协议有http、ftp、dns、dhcp等等。
- tcp的三次握手过程我们可以通过tcpdump去抓包，然后将包用sz命令发送到专门的机器上用wireshark进行分析看到。它的过程是这样的，首先客户端发送请求位码为syn=1，随机产生seq number的包到服务端，表示请求建立连接，这就是第一次握手。第二次握手是服务端需要确认连接信息，向客户端发送ack number，值就是客户端发送的seq number的值加一，还有syn=1，ack=1，以及一个随机的seq数字。第三次握手客户端会检测ack number是否正确，ack是否等于1，如果通过就会再次

给服务端发送seq number，值为服务端发送过来的seq的值加一，还有ack=1，服务端收到后确认值是否正确，正确的话就建立连接成功。

- linux的启动过程大致可以分为五个步骤。第一个步骤就是内核引导。计算机开机后首先会启动bios系统，bios根据设置的硬盘启动操作系统，操作系统会根据grub文件读入/boot下的内核文件。第二个步骤是启动init进程。init进程是所有进程的起点，没有这个进程存在，系统中任何其它进程都不会启动，init进程需要读取/etc/inittab配置文件。第三个步骤进行系统初始化。init进程会先执行/etc/rc.d/rc.sysinit脚本，这个脚本会激活交换分区，检查磁盘，加载硬件模块等等一些需要优先执行的任务。init配置文件里面会定义系统本次的运行级别，比如此次的运行级别为5的话，就会去执行/etc/rc.d/rc5.d/目录下的脚本，这个下面定义了各种需要初始化开启的服务。第四个步骤就是建立终端。基本环境已经设置好了，init就会打开6个终端，这是在inittab配置里定义的。最后一步就是用户登录系统，5级别的话可以通过图形界面登录，3级别只能通过命令行登录，开启了ssh服务的话，它们都可以通过ssh进行远程登录。