**1、请解释是否有可能将Nginx的错误替换为502错误、503?**

答： 502 =错误网关 503 =服务器超载 有可能，但是您可以确保fastcgi\_intercept\_errors被设置为ON，并使用错误页面指令。 

**2、nginx和apache的区别？**

1）轻量级，同样起web 服务，比apache 占用更少的内存及资源

 2）抗并发，nginx 处理请求是异步非阻塞的，而apache 则是阻塞型的，在高并发下nginx 能保持低资源低消耗高性能

 3）高度模块化的设计，编写模块相对简单

 4）最核心的区别在于apache是同步多进程模型，一个连接对应一个进程；nginx是异步的，多个连接（万级别）可以对应一个进程

**3、nginx是如何实现高并发的**

一个主进程，多个工作进程，每个工作进程可以处理多个请求，每进来一个request，会有一个worker进程去处理。但不是全程的处理，处理到可能发生阻塞的地方，比如向上游（后端）服务器转发request，并等待请求返回。那么，这个处理的worker继续处理其他请求，而一旦上游服务器返回了，就会触发这个事件，worker才会来接手，这个request才会接着往下走。由于web server的工作性质决定了每个request的大部份生命都是在网络传输中，实际上花费在server机器上的时间片不多。这是几个进程就解决高并发的秘密所在。即@skoo所说的webserver刚好属于网络io密集型应用，不算是计算密集型。

**4、使用“反向代理服务器”的优点是什么?**

反向代理服务器可以隐藏源服务器的存在和特征。它充当互联网云和web服务器之间的中间层。这对于安全方面来说是很好的，特别是当您使用web托管服务时。

**5、请解释Nginx服务器上的Master和Worker进程分别是什么?**

Master进程：读取及评估配置和维持

Worker进程：处理请求

**6、动态资源、静态资源分离**

动态资源、静态资源分离是让动态网站里的动态网页根据一定规则把不变的资源和经常变的资源区分开来，动静资源做好了拆分以后，我们就可以根据静态资源的特点将其做缓存操作，这就是网站静态化处理的核心思路  
    动态资源、静态资源分离简单的概括是：动态文件与静态文件的分离

**7、为什么要做动、静分离？**

在我们的软件开发中，有些请求是需要后台处理的（如：.jsp,.do等等），有些请求是不需要经过后台处理的（如：css、html、jpg、js等等文件）  
    这些不需要经过后台处理的文件称为静态文件，否则动态文件。因此我们后台处理忽略静态文件。这会有人又说那我后台忽略静态文件不就完了吗  
    当然这是可以的，但是这样后台的请求次数就明显增多了。在我们对资源的响应速度有要求的时候，我们应该使用这种动静分离的策略去解决  
    动、静分离将网站静态资源（HTML，JavaScript，CSS，img等文件）与后台应用分开部署，提高用户访问静态代码的速度，降低对后台应用访问  
    这里我们将静态资源放到nginx中，动态资源转发到tomcat服务器中

**8、负载均衡**

负载均衡即是代理服务器将接收的请求均衡的分发到各服务器中  
    负载均衡主要解决网络拥塞问题，提高服务器响应速度，服务就近提供，达到更好的访问质量，减少后台服务器大并发压力

**9、反向代理**

反向代理（Reverse Proxy）方式是指以代理服务器来接受internet上的连接请求，然后将请求，发给内部网络上的服务器  
    并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个反向代理服务器  
    反向代理总结就一句话：代理端代理的是服务端

**10、正向代理**

一个位于客户端和原始服务器(origin server)之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定目标(原始服务器)，然后代理向原始服务器转交请求并将获得的内容返回给客户端。客户端才能使用正向代理  
   正向代理总结就一句话：代理端代理的是客户端

**11、Nginx是如何处理一个请求的呢？**

首先，nginx在启动时，会解析配置文件，得到需要监听的端口与ip地址，然后在nginx的master进程里面  
    先初始化好这个监控的socket，再进行listen  
    然后再fork出多个子进程出来,  子进程会竞争accept新的连接。  
  
此时，客户端就可以向nginx发起连接了。当客户端与nginx进行三次握手，与nginx建立好一个连接后  
  
    此时，某一个子进程会accept成功，然后创建nginx对连接的封装，即ngx\_connection\_t结构体  
    接着，根据事件调用相应的事件处理模块，如http模块与客户端进行数据的交换。  
  
最后，nginx或客户端来主动关掉连接，到此，一个连接就寿终正寝了

**12、为什么不使用多线程？**

Apache: 创建多个进程或线程，而每个进程或线程都会为其分配cpu和内存（线程要比进程小的多，所以worker支持比perfork高的并发），并发过大会榨干服务器资源。

Nginx: 采用单线程来异步非阻塞处理请求（管理员可以配置Nginx主进程的工作进程的数量）(epoll)，不会为每个请求分配cpu和内存资源，节省了大量资源，同时也减少了大量的CPU的上下文切换。所以才使得Nginx支持更高的并发。

OK

**13、为什么要用Nginx？**

优点：  
        跨平台、配置简单  
       非阻塞、高并发连接：处理2-3万并发连接数，官方监测能支持5万并发  
        内存消耗小：开启10个nginx才占150M内存，Nginx采取了分阶段资源分配技术  
        nginx处理静态文件好,耗费内存少  
        内置的健康检查功能：如果有一个服务器宕机，会做一个健康检查，再发送的请求就不会发送到宕机的服务器了。重新将请求提交到其他的节点上。  
        节省宽带：支持GZIP压缩，可以添加浏览器本地缓存  
        稳定性高：宕机的概率非常小  
       master/worker结构：一个master进程，生成一个或者多个worker进程  
        接收用户请求是异步的：浏览器将请求发送到nginx服务器，它先将用户请求全部接收下来，再一次性发送给后端web服务器，极大减轻了web服务器的压力  
        一边接收web服务器的返回数据，一边发送给浏览器客户端  
        网络依赖性比较低，只要ping通就可以负载均衡  
        可以有多台nginx服务器  
        事件驱动：通信机制采用epoll模型

**14、MySQL的主从复制和Nginx负载均衡是自己配置的吗？怎么配置的？**

是.  
开发环境,测试环境是我配置的  
生产环境是运维人员配置  
  
还可以说:备有相关的资料,按照文档可以配上.  
  
具体配置,参见文档