**1. 说一下公司多少台服务器，是什么架构**

**答案：有7台服务器，lnmp+nginx负载+keepalived，其中2台keepalived+2台nginx/php-fpm+2台mysql（一主一从）+NFS（上面兼着跑监控、备份）**

**2. Nginx有哪些优化参数**

**答案：**

**A.** **worker\_processes 8;   nginx 进程数，建议按照cpu 数目来指定，一般为它的倍数 (如,2个四核的cpu计为8)。**

**B.** **worker\_cpu\_affinity 00000001 0000001000000100 00001000 00010000 00100000 01000000 10000000;   为每个进程分配cpu，上例中将8 个进程分配到8 个cpu，当然可以写多个，或者将一个进程分配到多个cpu。**

**C.** **worker\_rlimit\_nofile 65535; 这个指令是指当一个nginx进程打开的最多文件描述符数目，理论值应该是最多打开文件数与进程数相除，现在在linux2.6内核下开启文件打开数为65535，worker\_rlimit\_nofile就相应应该填写65535。这是因为nginx调度时分配请求到进程并不是那么的均衡，所以假如填写10240，总并发量达到3-4万时就有进程可能超过10240了，这时会返回502错误。**

**D.** **useepoll;**

**使用epoll的I/O模型**

**E.** **Worker\_connections 65535;  每个进程允许的最多连接数， 理论上每台nginx服务器的最大连接数为worker\_processes\*worker\_connections。**

**F.** **keepalive\_timeout 60;   keepalive超时时间。**

**G.** **client\_header\_buffer\_size 4k;客户端请求头部的缓冲区大小，这个可以根据你的系统分页大小来设置，一般一个请求头的大小不会超过1k，不过由于一般系统分页都要大于1k，所以这里设置为分页大小。**

**H.** **open\_file\_cachemax=65535 inactive=60s;这个将为打开文件指定缓存，默认是没有启用的，max指定缓存数量，建议和打开文件数一致，inactive是指经过多长时间文件没被请求后删除缓存。**

**I.** **Open\_file\_cache\_valid  80s;这个是指多长时间检查一次缓存的有效信息。**

**J.** **open\_file\_cache\_min\_uses  1;   open\_file\_cache 指令中的inactive参数时间内文件的最少使用次数，如果超过这个数字，文件描述符一直是在缓存中打开的，如上例，如果有一个文件在inactive时间内一次没被使用，它将被移除。**

**3. 提高性能和并发数，需要优化哪些内核参数**

**答案：**

**net.ipv4.tcp\_max\_tw\_buckets = 6000  //timewait的数量，默认是180000。**

**net.ipv4.ip\_local\_port\_range = 1024 65000  //允许系统打开的端口范围。**

**net.ipv4.tcp\_tw\_reuse = 1 //允许将TIME-WAIT sockets 重新用于新的TCP 连接。**

**net.ipv4.tcp\_syncookies = 1 //开启SYN Cookies，当出现SYN 等待队列溢出时，启用cookies 来处理。**

**net.ipv4.tcp\_max\_orphans = 262144 //系统中最多有多少个TCP套接字不被关联到任何一个用户文件句柄上。如果超过这个数字，孤儿连接将即刻被复位并打印出警告信息。这个限制仅仅是为了防止简单的DoS攻击，不能过分依靠它或者人为地减小这个值，更应该增加这个值(如果增加了内存之后)。**

**net.ipv4.tcp\_max\_syn\_backlog = 262144  //记录的那些尚未收到客户端确认信息的连接请求的最大值。对于有128M内存的系统而言，缺省值是1024，小内存的系统则是128。**

**net.ipv4.tcp\_synack\_retries = 1  //为了打开对端的连接，内核需要发送一个SYN 并附带一个回应前面一个SYN的ACK。也就是所谓三次握手中的第二次握手。这个设置决定了内核放弃连接之前发送SYN+ACK 包的数量。**

**net.ipv4.tcp\_syn\_retries = 1  //在内核放弃建立连接之前发送SYN 包的数量。**

**net.ipv4.tcp\_keepalive\_time = 30 //当keepalive 起用的时候，TCP 发送keepalive 消息的频度。缺省是2 小时。**

**4. Nginx和lvs负载均衡比较**

**1)lvs的特点：**

**A. 抗负载能力强，因为lvs工作方式的逻辑是非常之简单，而且工作在网络4层仅做请求分发之用，没有流量，所以在效率上基本不需要太过考虑。在我手里的 lvs，仅仅出过一次问题：在并发最高的一小段时间内均衡器出现丢包现象，据分析为网络问题，即网卡或linux2.4内核的承载能力已到上限，内存和 cpu方面基本无消耗。**

**B. 配置性低，这通常是一大劣势，但同时也是一大优势，因为没有太多可配置的选项，所以除了增减服务器，并不需要经常去触碰它，大大减少了人为出错的几率。**

**C. 工作稳定，因为其本身抗负载能力很强，所以稳定性高也是顺理成章，另外各种lvs都有完整的双机热备方案，所以一点不用担心均衡器本身会出什么问题，节点出现故障的话，lvs会自动判别，所以系统整体是非常稳定的。**

**D. 无流量，上面已经有所提及了。lvs仅仅分发请求，而流量并不从它本身出去，所以可以利用它这点来做一些线路分流之用。没有流量同时也保住了均衡器的IO性能不会受到大流量的影响。**

**E. 基本上能支持所有应用，因为lvs工作在4层，所以它可以对几乎所有应用做负载均衡，包括http、数据库、聊天室等等。**

**另：lvs也不是完全能判别节点故障的，譬如在wlc分配方式下，集群里有一个节点没有配置VIP，会使整个集群不能使用，这时使用wrr分配方式则会丢掉一台机。目前这个问题还在进一步测试中。所以，用lvs也得多多当心为妙。**

**2) Nginx的特点是：**

**A. 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。**

**B. Nginx对网络稳定性的依赖非常小，理论上能ping通就就能进行负载功能，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；**

**C. 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。**

**D. 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。**

**E. 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。**

**F. 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。**

**G. 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，可以考虑用其作为反向代理加速器。**

**H. 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。**

**I. 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。**

**J. Nginx新版本已经支持代理tcp各种协议，不再仅仅局限在代理http、https以及email。**

**5. zabbix监控哪些项目，模板是不是自己写的，触发报警有哪些，阀值都是多少**

**答案：**

**监控了CPU使用、系统负载、内存剩余、磁盘使用百分比、mysql主从、mysql队列数量、网站访问量、网卡流量、web状态码，有自己写的自定义监控脚本。触发报警的有系统负载（当高于20报警）、磁盘使用百分比（高于90%报警）、mysql主从是否正常（不正常告警）、mysql队列数量（高于400报警）、网卡流量（高于100M报警）等等。**

**6. mysql两种引擎的区别，innodb原理，主从原理**

**1）myisam引擎和innodb引擎，两者的区别是：**

**MyISAM不支持事务，而InnoDB支持。InnoDB的AUTOCOMMIT默认是打开的，即每条SQL语句会默认被封装成一个事务，自动提交，这样会影响速度，所以最好是把多条SQL语句显示放在begin和commit之间，组成一个事务去提交。**

**InnoDB支持数据行锁定，MyISAM不支持行锁定，只支持锁定整个表。即MyISAM同一个表上的读锁和写锁是互斥的，MyISAM并发读写时如果等待队列中既有读请求又有写请求，默认写请求的优先级高，即使读请求先到，所以MyISAM不适合于有大量查询和修改并存的情况，那样查询进程会长时间阻塞。因为MyISAM是锁表，所以某项读操作比较耗时会使其他写进程饿死。**

**InnoDB支持外键，MyISAM不支持。**

**InnoDB的主键范围更大，最大是MyISAM的2倍。**

**InnoDB不支持全文索引，而MyISAM支持。全文索引是指对char、varchar和text中的每个词（停用词除外）建立倒排序索引。MyISAM的全文索引其实没啥用，因为它不支持中文分词，必须由使用者分词后加入空格再写到数据表里，而且少于4个汉字的词会和停用词一样被忽略掉。**

**MyISAM支持GIS数据，InnoDB不支持。即MyISAM支持以下空间数据对象：Point,Line,Polygon,Surface等。**

**没有where的count(\*)使用MyISAM要比InnoDB快得多。因为MyISAM内置了一个计数器，count(\*)时它直接从计数器中读，而InnoDB必须扫描全表。所以在InnoDB上执行count(\*)时一般要伴随where，且where中要包含主键以外的索引列。为什么这里特别强调“主键以外”？因为InnoDB中primary index是和raw data存放在一起的，而secondary index则是单独存放，然后有个指针指向primary key。所以只是count(\*)的话使用secondary index扫描更快，而primary key则主要在扫描索引同时要返回raw data时的作用较大。**

**2）innodb原理太复杂了，给个链接大家简单看看吧，真想不通出题的人咋想的，这么复杂的原理怎么可能一句两句写得完。http://blog.sina.com.cn/s/blog\_aed82f6f010188kw.html**

**3）主从原理看前面的笔试题吧（16套12）。**

**7. mysql用的哪个版本，和5.1比增加了哪些特性**

**答：目前使用5.6。大家看下版本功能列表吧，如果面试的时候被问到，最起码可以说出至少5点。**

**【5.1 2008 5.1.63GA 】**

**1 Sun收购后的版本**

**2 分区表**

**3 行复制.**

**4 BerkeleyDB移出**

**5 Federated were deprecated**

**6 Oracle推出InnoDB plugin版本**

**7 Event**

**8 Plugin API**

**9 Pluggable storage engine API**

**10 日志表**

**11 加强information\_schema**

**12 mysqlslap**

**13 慢查询阀值可设置为毫秒**

**【5.5 2010 5.5.25GA 】**

**1 Oracle收购后的版本**

**2 InnoDB变为默认引擘**

**3 加入performance\_schema库,监控MySQL服务器运行时的性能**

**4 半同步复制**

**5 多缓冲池实例**

**6 Supplementary Unicode characters**

**7 多核利用率提高**

**8 加强SHOW ENGINE INNODB STATUS输出**

**9 分区支持新增RANGE COLUMNS,LIST COLUMNS**

**10 ALTER TABLE … TRUNCATE PARTITION**

**11 TO\_SECONDS() function**

**12 编译采用cmake**

**13 多个回滚段（Multiple Rollback Segments）**

**14 提高了默认线程并发数,设置innodb\_thread\_concurrency**

**15 后台输入/输出线程控制,设置innodb\_read\_io\_threads和innodb\_write\_io\_threads**

**16 主线程输入/输出速率控制,设置innodb\_io\_capacity**

**17 可以关闭适应性散列索引功能**

**18 可以关闭InnoDB的插入缓冲**

**19 恢复组提交（Restored Group Commit）**

**20 快速恢复重启**

**21 MySQL 5.5为基于Linux和Windows的数据库系统提供了真实的异步输入/输出支持。如果新的默认设置与主机输入/输出子系统不兼容,要使用该功能,需要在Linux系统中安装libaio用户空间库,并配置innodb\_use\_native\_aio选项参数（该参数可以关闭）**

**22 使用innodb\_change\_buffering配置参数来控制删除缓冲和既有插入缓冲功能,默认在所有的系统中都使用**

**23 每一个缓冲池实例都有一个单独的刷新（Flush）列表互斥,从而进一步减少了争用**

**24 设置innodb\_purge\_threads配置参数来选择清除操作是否使用单独线程**

**25 禁止会话执行另一个会话中,未完成事务处理正在使用的表格中的DDL语句,能够保证事务处理的串行化**

**26 Linux系统中,在高事务处理频率和连接负载的情况下,MySQL5.5在处理只读（Read Only）操作时的性能和可扩展性是MySQL5.1的200%,处理读/写操作时的性能和可扩展性是MySQL5.1的370%**

**27 复制Heartbeat**

**28 relay\_log\_recovery,自动丢弃自己未处理的中继日志（Relay Log）,然后从源主机服务器恢复挂起的事务处理**

**29 根据服务器过滤项复制**

**30 SLAVE\_TYPE\_CONVERSION,提供了一种精确的数据类型转换机制**

**31 建立快速索引,添加（Add）或删除（Drop）索引,而无需复制整个目标表中的底层数据**

**32 数据压缩,ROW\_FORMAT和KEY\_BLOCK\_SIZE参数,按照1K,2K,4K,8K或标准16K字节压缩等级存储数据页。**

**33 ROW\_FORMAT=DYNAMIC或ROW\_FORMAT=COMPRESSED创建表,这提高了大对象的存储效率**

**34 存储对象（包括存储过程,函数,触发器）内部使用SIGNAL/RESIGNAL语法,用于项目的前期研发和调试,然后产生错误条件,并使用该条件调用主机应用程序中的专用错误处理程序**

**35 表和索引RANG和LIST分区范围扩展到了非整数列和日期,并增加了在多个列上分区的能力，COLUMNS关键字支持这些选项**

**【5.6 目前无GA版，但已经正式冻结准备发送GA版 5.6.5m8 】**

**1 查询优化：Index Condition Pushdown,Multi-Range Read,File Sort Optimization**

**2 加强performance\_schema库**

**3 Dump/Restore buffer pool for fast start up**

**4 UUID唯一标识服务器,内部事务序列产生全局唯一GTID**

**5 自动回滚复制到一半的事务并重做整个事务**

**6 row格式Binlog中只记必需字段**

**7 延时复制**

**8 远程Binlog备份**

**9 row格式复制记录SQL语句**

**10 kernel mutex splits into different mutexes**

**11 InnoDB 全文搜索**

**12 Online add index,Add / Drop Foreign Key,Add / Drop Column,Rename Table,Rename Column**

**13 Transportable Tablespaces**

**14 Direct Access to InnoDB via Memcached**

**15 innodb压缩增强**

**16 improved accuracy of InnoDB index statistics, and consistency across MySQL restarts**

**17 NoSQL Interface via memcached**

**18 分区增强：Explicit Partition Selection–SELECT \* FROM employees PARTITION (p0, p2),Import / Export for Partitioned Tables-ALTER TABLE e EXCHANGE PARTITION p0 WITH TABLE e2**

**19 多线程复制,数据库级别**

**20 Replication Checksums**

**21 Hash Join**

**22 Read-Only事务区分**

**23 5.6.6峰值最高可以达到16w的QPS，比原版的5.1提升4倍(使用flash设备)**

**8. myisam优化参数有哪些，innodb优化参数有哪些**

**答：**

**Myisam**

**max\_connections = 151**

**#同时处理最大连接数，推荐设置最大连接数是上限连接数的80%左右**

**sort\_buffer\_size = 2M**

**#查询排序时缓冲区大小，只对order by和group by起作用，可增大此值为16M**

**query\_cache\_limit = 1M**

**#查询缓存限制，只有1M以下查询结果才会被缓存，以免结果数据较大把缓存池覆盖**

**query\_cache\_size = 16M**

**#查看缓冲区大小，用于缓存SELECT查询结果，下一次有同样SELECT查询将直接从缓存池返回结果，可适当成倍增加此值**

**open\_files\_limit = 1024**

**#打开文件数限制，如果show global status like 'open\_files'查看的值等于或者大于open\_files\_limit值时，程序会无法连接数据库或卡死**

**Innodb：**

**innodb\_buffer\_pool\_size = 128M**

**#索引和数据缓冲区大小，一般设置物理内存的60%-70%**

**innodb\_buffer\_pool\_instances = 1**

**#缓冲池实例个数，推荐设置4个或8个**

**innodb\_flush\_log\_at\_trx\_commit = 1**

**#关键参数，0代表大约每秒写入到日志并同步到磁盘，数据库故障会丢失1秒左右事务数据。1为每执行一条SQL后写入到日志并同步到磁盘，I/O开销大，执行完SQL要等待日志读写，效率低。2代表只把日志写入到系统缓存区，再每秒同步到磁盘，效率很高，如果服务器故障，才会丢失事务数据。对数据安全性要求不是很高的推荐设置2，性能高，修改后效果明显。**

**innodb\_file\_per\_table = OFF**

**#默认是共享表空间，共享表空间idbdata文件不断增大，影响一定的I/O性能。推荐开启独立表空间模式，每个表的索引和数据都存在自己独立的表空间中，可以实现单表在不同数据库中移动。**

**innodb\_log\_buffer\_size = 8M**

**#日志缓冲区大小，由于日志最长每秒钟刷新一次，所以一般不用超过16M**

9. mysql怎么优化存储

A. 使用索引

B. 使用explain分析查询

C. 调整mysql内部配置

**(1)、改变索引缓冲区长度（key\_buffer）；推荐设置整个系统内存的25%。**

**(2)、改变表长(read\_buffer\_size)**

**(3)、设定打开表的数目的大小(table\_cache)**

**(4)、对缓查询设定一个时间限制(long\_query\_time)**

10. php优化参数有哪些，fastcgi设置是多少，动态还是静态

答：

**A. php.ini安全配置，禁掉危险的函数**

**disable\_functions（exec,system,passthru,error\_log,ini\_alter,dl,openlog,syslog,readlink,  　　symlink,link,leak,fsockopen,proc\_open,popepassthru,chroot,scandir,  　　chgrp,chown,escapeshellcmd,escapeshellarg,shell\_exec,proc\_get\_status,popen）**

**B. php.ini配置日志**

**display\_errors =   默认Off** **[On]  打开错误日志**

**error\_log=/usr/local/php/logs/php\_errors.log**

**error\_reporting=E\_ALL & ~E\_NOTICE**

**C. Php.ini中定义open\_basedir = /data/www:/tmp    白名单目录**

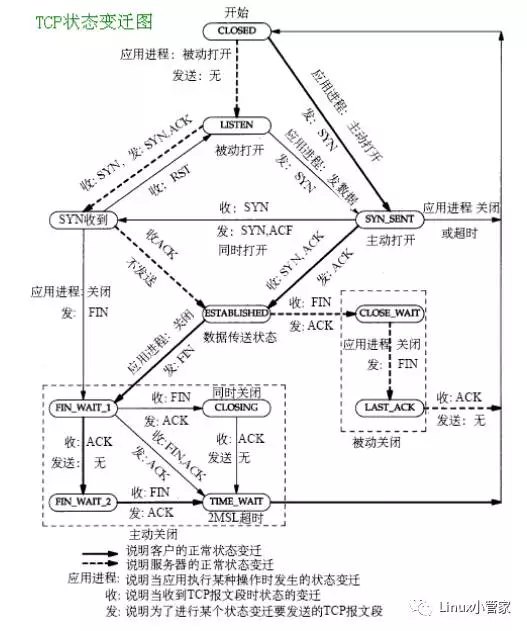
**D. Php-fpm.conf中配置慢执行日志**

**E. Php-fpm.conf定义max\_children**

**F. 问题中的fastcgi指的就是这个fpm配置，最大进程数（max\_children)为300，动态（dynamic）**

11. TCP有哪些了解，TCP连接状态中“TIME\_WAIT”是什么意思，影响什么

答：**关于tcp有点复杂，直接上图吧，更直观**



状态描述：

**CLOSED:** 这个没什么好说的了，表示初始状态。

**LISTEN:** 这个也是非常容易理解的一个状态，表示服务器端的某个SOCKET处于监听状态，可以接受连接了。

**SYN\_RCVD:** 这个状态表示接受到了SYN报文，在正常情况下，这个状态是服务器端的SOCKET在建立TCP连接时的三次握手会话过程中的一个中间状态，很短暂，基本 上用netstat你是很难看到这种状态的，除非你特意写了一个客户端测试程序，故意将三次TCP握手过程中最后一个ACK报文不予发送。因此这种状态时，当收到客户端的ACK报文后，它会进入到ESTABLISHED状态。

**SYN\_SENT:** 这个状态与SYN\_RCVD遥想呼应，当客户端SOCKET执行CONNECT连接时，它首先发送SYN报文，因此也随即它会进入到了SYN\_SENT状 态，并等待服务端的发送三次握手中的第2个报文。SYN\_SENT状态表示客户端已发送SYN报文。

**ESTABLISHED：**这个容易理解了，表示连接已经建立了。

**FIN\_WAIT\_1:** 这个状态要好好解释一下，其实FIN\_WAIT\_1和FIN\_WAIT\_2状态的真正含义都是表示等待对方的FIN报文。而这两种状态的区别 是：FIN\_WAIT\_1状态实际上是当SOCKET在ESTABLISHED状态时，它想主动关闭连接，向对方发送了FIN报文，此时该SOCKET即 进入到FIN\_WAIT\_1状态。而当对方回应ACK报文后，则进入到FIN\_WAIT\_2状态，当然在实际的正常情况下，无论对方何种情况下，都应该马 上回应ACK报文，所以FIN\_WAIT\_1状态一般是比较难见到的，而FIN\_WAIT\_2状态还有时常常可以用netstat看到。

**FIN\_WAIT\_2：**上面已经详细解释了这种状态，实际上FIN\_WAIT\_2状态下的SOCKET，表示半连接，也即有一方要求close连接，但另外还告诉对方，我暂时还有点数据需要传送给你，稍后再关闭连接。

TIME\_WAIT: 表示收到了对方的FIN报文，并发送出了ACK报文，就等2MSL后即可回到CLOSED可用状态了。如果FIN\_WAIT\_1状态下，收到了对方同时带 FIN标志和ACK标志的报文时，可以直接进入到TIME\_WAIT状态，而无须经过FIN\_WAIT\_2状态。

**CLOSING:** 这种状态比较特殊，实际情况中应该是很少见，属于一种比较罕见的例外状态。正常情况下，当你发送FIN报文后，按理来说是应该先收到（或同时收到）对方的 ACK报文，再收到对方的FIN报文。但是CLOSING状态表示你发送FIN报文后，并没有收到对方的ACK报文，反而却也收到了对方的FIN报文。什 么情况下会出现此种情况呢？其实细想一下，也不难得出结论：那就是如果双方几乎在同时close一个SOCKET的话，那么就出现了双方同时发送FIN报 文的情况，也即会出现CLOSING状态，表示双方都正在关闭SOCKET连接。

**CLOSE\_WAIT:** 这种状态的含义其实是表示在等待关闭。怎么理解呢？当对方close一个SOCKET后发送FIN报文给自己，你系统毫无疑问地会回应一个ACK报文给对 方，此时则进入到CLOSE\_WAIT状态。接下来呢，实际上你真正需要考虑的事情是察看你是否还有数据发送给对方，如果没有的话，那么你也就可以 close这个SOCKET，发送FIN报文给对方，也即关闭连接。所以你在CLOSE\_WAIT状态下，需要完成的事情是等待你去关闭连接。

**LAST\_ACK:** 这个状态还是比较容易好理解的，它是被动关闭一方在发送FIN报文后，最后等待对方的ACK报文。当收到ACK报文后，也即可以进入到CLOSED可用状态了。

12. 网站出现500,502,400,403,404都是什么意思，怎么排查和解决

答案：

**500：**服务器内部错误，因为服务器上的程序写的有问题，需要打开错误日志，查看日志，分析错误信息。

**502：**网关错误，服务器作为网关或代理，从上游服务器收到无效响应。Nginx出现最多，出现502要么是nginx配置的不对，要么是php-fpm资源不够，可以分析php-fpm的慢执行日志，优化php-fpm的执行速度。

**400：**错误请求，服务器不理解请求的语法。这可能是用户发起的请求不合理，需要检查客户端的请求。

**403：**服务器拒绝请求。检查服务器配置，是不是对客户端做了限制。

**404：**未找到请求的资源。检查服务器上是否存在请求的资源，看是否是配置问题。

13. 从运维角度讲一下怎么预防cc攻击和ddos攻击

答案：**先来说一下什么是cc攻击和ddos攻击。**

1） CC主要是用来攻击页面的。大家都有这样的经历，就是在访问论坛时，如果这个论坛比较大，访问的人比较多，打开页面的速度会比较慢，访问的人越多，论坛的页面越多，数据库就越大，被访问的频率也越高，占用的系统资源也就相当可观。

一个静态页面不需要服务器多少资源，甚至可以说直接从内存中读出来发给你就可以了，但是论坛就不一样了，我看一个帖子，系统需要到数据库中判断我是否有读帖子的权限，如果有，就读出帖子里面的内容，显示出来——这里至少访问了2次数据库，如果数据库的数据容量有200MB大小，系统很可能就要在这200MB大小的数据空间搜索一遍，这需要多少的CPU资源和时间？如果我是查找一个关键字，那么时间更加可观，因为前面的搜索可以限定在一个很小的范围内，比如用户权限只查用户表，帖子内容只查帖子表，而且查到就可以马上停止查询，而搜索肯定会对所有的数据进行一次判断，消耗的时间是相当的大。

CC就是充分利用了这个特点，模拟多个用户（多少线程就是多少用户）不停的进行访问（访问那些需要大量数据操作，就是需要大量CPU时间的页面）.这一点用一个一般的性能测试软件就可以做到大量模拟用户并发。

防御CC攻击可以通过多种方法，比如，可以分析攻击的请求头信息，分析它的特点，然后针对这些请求做一些限制。也可以分析请求的ip，利用iptables来限制ip。将网站做成静态页面，也可以有效降低服务器资源使用。另外，还可以限制连接数量，修改最大超时时间等。

2） ddos攻击的方式有很多种，最基本的ddos攻击就是利用合理的服务请求来占用过多的服务资源，从而使合法用户无法得到服务的响应。单一的ddos攻击一般是采用一对一方式的，当攻击目标CPU速度低、内存小或者网络带宽小等等各项指标不高的性能，它的效果是明显的。随着计算机与网络技术的发展，计算机的处理能力迅速增长，内存大大增加，同时也出现了千兆级别的网络，这使得DoS攻击的困难程度加大了-目标对恶意攻击包的"消化能力"加强了不少。这时候分布式的拒绝服务攻击手段（ddos）就应运而生了。ddos就是利用更多的傀儡机（肉鸡）来发起进攻，以比从前更大的规模来进攻受害者。

简单的ddos，比如cc，我们可以通过限定ip来解决攻击。但有时候攻击量很大，甚至可以把机房的网络攻击瘫痪，这时候只能临时在上层网络把目标IP封掉，这样牺牲单个ip而保全大局。也可以接入第三方的防ddos攻击的cdn。