osi 七层模型，tcp 三次握手过程，tcp 连接断开过程，什么情况下 tcp 进入 time_wait?

答：其他问题答案略

什么情况下 tcp 进入 time_wait?

当关闭一个 socket 连接时，主动关闭一端的 socket 将进入 TIME_WAIT 状态，而被动关闭一方则转入 CLOSED 状态。

具体过程如下：

1、 客户端发送 FIN 报文段，进入 FIN_WAIT_1 状态。

2、 服务器端收到 FIN 报文段，发送 ACK 表示确认，进入 CLOSE_WAIT 状态。

3、 客户端收到 FIN 的确认报文段，进入 FIN_WAIT_2 状态。

4、服务器端发送 FIN 报文端，进入 LAST_ACK 状态。

5、 客户端收到 FIN 报文端，发送 FIN 的 ACK，同时进入 TIME_WAIT 状态，启动 TIME_WAIT 定时器，超时时间设为 2MSL。

6、 服务器端收到 FIN 的 ACK，进入 CLOSED 状态。

7、客户端在 2MSL 时间内没收到对端的任何响应，TIME_WAIT 超时，进入 CLOSED 状态。

mysql 的 innodb 如何定位锁问题，mysql 如何减少主从复制延迟？

mysql 的 innodb 如何定位锁问题:

在使用 show engine innodb status 检查引擎状态时，发现了死锁问题

在 5.5 中，information_schema 库中增加了三个关于锁的表（MEMORY 引擎）：

innodb_trx

```
## 当前运行的所有事务 innodb_locks

## 当前出现的锁 innodb_lock_waits

## 锁等待的对应关系
```

mysql 如何减少主从复制延迟:

如果延迟比较大，就先确认以下几个因素：

1. 从库硬件比主库差，导致复制延迟

2. 主从复制单线程，如果主库写并发太大，来不及传送到从库，就会导致延迟。更高版本的 mysql 可以支持多线程复制

3. 慢 SQL 语句过多

4. 网络延迟

5. master 负载主库读写压力大，导致复制延迟，架构的前端要加 buffer 及缓存层

6. slave 负载一般的做法是，使用多台 slave 来分摊读请求，再从这些 slave 中取一台专用的服务器，只作为备份用，不进行其他任何操作.

另外， 2 个可以减少延迟的参数:

–slave-net-timeout=seconds 单位为秒 默认设置为 3600 秒

#参数含义：当 slave 从主数据库读取 log 数据失败后，等待多久重新建立连接并获取数据

–master-connect-retry=seconds 单位为秒 默认设置为 60 秒

#参数含义：当重新建立主从连接时，如果连接建立失败，间隔多久后重试。

通常配置以上 2 个参数可以减少网络问题导致的主从数据同步延迟

MySQL 数据库主从同步延迟解决方案

最简单的减少 slave 同步延时的方案就是在架构上做优化，尽量让主库的 DDL 快速执行。还有就是主库是写，对数据安全性较高，比如 sync_binlog=1，innodb_flush_log_at_trx_commit= 1 之类的设置，而 slave 则不需要这么高的数据安全，完全可以讲 sync_binlog 设置为 0 或者关闭 binlog，innodb_flushlog 也可以设置为 0 来提高 sql 的执行效率。另外就是使用比主库更好的硬件设备作为 slave。

怎么查看当前磁盘的 IO？

1） iostat 可以提供丰富的 **IO** 状态数据。

iostat 是 **sysstat** 工具集的一个工具，需要安装。

```
[root@localhost ~]# iostat -d -k 1 10
Linux 3.10.0-327.el7.x86_64 (localhost.localdomain) 03/23/2017 _x86_64_(2 CPU)
Device:            tps   kB_read/s   kB_wrtn/s   kB_read   kB_wrtn
sda              16.60      597.83       29.44    384048     18909
scd0              0.03        0.10        0.00        66         0
dm-0             15.78      551.54       26.20    354311     16831
dm-1              0.22        1.97        0.00      1268         0
```

参数： **-d** 表示，显示设备（磁盘）使用状态；

-k 某些使用 **block** 为单位的列强制使用 **Kilobytes** 为单位；

1 10 表示，数据显示每隔 **1** 秒刷新一次，共显示 **10** 次。

tps：该设备每秒的传输次数。

kB_read/s：每秒从设备读取的数据量；

kB_wrtn/s：每秒向设备写入的数据量；

kB_read：读取的总数据量；

kB_wrtn：写入 的总数量数据量；这些单位都为 **Kilobytes**。

-x 参数：使用**-x** 参数我们可以获得更多统计信息。

iostat -d -x -k 1 10 #查看设备使用率（**%util**）、响应时间（**%await**）

await：每一个 **IO** 请求的处理的平均时间（单位是微秒）。这里可以理解为 **IO** 的响应时间，一般地系统 **IO** 响应时间应该低于 **5ms**，如果大于 **10ms** 就比较大了。

%util：在统计时间内所有处理 **IO** 时间，除以总共统计时间。例如，如果统计间隔 **1** 秒，该 设备有 **0.8** 秒在处理 **IO**，而 **0.2** 秒闲置，那么该设备的**%util = 0.8/1 = 80%**，所以该参数暗示了设备的繁忙程度。一般地，如果该参数是 **100%**表示设备已经接近满负荷运行了（当然如果是多磁盘，即使**%util** 是 **100%**，因为磁盘的并发能力，所以磁盘使用未必就到了瓶颈）。

2）使用 iotop 命令

要安装 **iotop** 软件包

iotop 命令是一个用来监视磁盘 **I/O** 使用状况的 **top** 类工具。**iotop** 具有与 **top** 相似的 **UI**。**Linux** 下的 **IO** 统计工具如 **iostat**，**nmon** 等大多数是只能统计到 **per** 设备的读写情况，如果你想知道每个进程是如何使用 **IO** 的就比较麻烦，使用 **iotop** 命令可以很方便的查看。

怎么查看当前网络的 **IO**：

iftop 查看网络带宽情况**(**必须从 **epel** 源安装**)**

sar 看看当前网络流量 **,sar -n** DEV**1 999** 表示取样间隔为 **1** 秒,取样 **999** 次

varnish nginx squid 各自缓存的优缺点

要做 cache 服务的话，我们肯定是要选择专业的 cache 服务，优先选择 squid 和 varnish。

Varnish

高性能、开源的反向代理服务器和内存缓存服务器。

优点：

1. 高性能；

2. 多核支持；

3. 支持 0-60 秒的精确缓存时间。

缺点：

1. 不具备自动容错和恢复功能，重启后数据丢失；

2. 在线扩容比较难。

3. 32 位机器上缓存文件大小为最大 2GB；

4. 不支持集群。

应用场景：

并发要求不是很大的小型系统和应用

nginx

1 不支持带参数的动态链接

2Nginx 缓存内部没有缓存过期和清理的任何机制，这些缓存的文件会永久性地保存在机器上，如果要缓存的东西非常多，那就会撑暴整个硬盘空间。

3 只能缓存 200 状态码，因此后端返回 301/302/404 等状态码都不会缓存，假如恰好有一个访问量很大的伪静态链接被删除，那就会不停穿透导致后端承载不小压力

4Nginx 不会自动选择内存或硬盘作为存储介质，一切由配置决定，当然在当前的操作系统里都会有操作系统级的文件缓存机制，所以存在硬盘上也不需要过分担心大并发读取造成的 i o 性能问题。

Squid

Squid，很古老的反向代理软件，拥有传统代理、身份验证、流量管理等高级功能，但是配置太复杂。它算是目前互联网应用得最多的反向缓存代理服务器，工作于各大古老的 cdn 上 squid 的优势在于完整的庞大的 cache 技术资料，和很多的应用生产环境

在浏览器中输入 www.qq.com 域名之后解析的过程？

1、在浏览器中输入 www.qq.com 域名，操作系统会先检查自己本地的 hosts 文件是否有这个网址映射关系，如果有，就先调用这个 IP 地址映射，完成域名解析。

2、如果 hosts 里没有这个域名的映射，则查找本地 DNS 解析器缓存，是否有这个网址映射关系，如果有，直接返回，完成域名解析。

3、如果 hosts 与本地 DNS 解析器缓存都没有相应的网址映射关系，首先会找 TCP/ip 参数中设置的首选 DNS 服务器，在此我们叫它本地 DNS 服务器，此服务器收到查询时，如果要查询的域名，包含在本地配置区域资源中，则返回解析结果给客户机，完成域名解析，此解析具有权威性。

4、如果要查询的域名，不由本地 DNS 服务器区域解析，但该服务器已缓存了此网址映射关系，则调用这个 IP 地址映射，完成域名解析，此解析不具有权威性。

5、如果本地 DNS 服务器本地区域文件与缓存解析都失效，则根据本地 DNS 服务器的设置（是否设置转发器）进行查询，如果未用转发模式，本地 DNS 就把请求发至 13 台根 DNS，根 DNS 服务器收到请求后会判断这个域名(.com)是谁来授权管理，并会返回一个负责该顶级域名服务器的一个 IP。本地 DNS 服务器收到 IP 信息后，将会联系负责.com 域的这台服务器。这台负责.com 域的服务器收到请求后，如果自己无法解析，它就会找一个管理.com 域的下一级 DNS 服务器地址(qq.com)给本地 DNS 服务器。当本地 DNS 服务器收到这个地址后，就会找 qq.com 域服务器，重复上面的动作，进行查询，直至找到 www.qq.com 主机。

6、如果用的是转发模式，此 DNS 服务器就会把请求转发至上一级 DNS 服务器，由上一级服务器进行解析，上一级服务器如果不能解析，或找根 DNS 或把转请求转至上上级，以此循环。不管是本地 DNS 服务器用是是转发，还是根提示，最后都是把结果返回给本地 DNS 服务器，由此 DNS 服务器再返回给客户机。

从客户端到本地 DNS 服务器是属于递归查询，而 DNS 服务器之间就是的交互查询就是迭代查询。

lvs/nginx/haproxy 优缺点是什么

一、Nginx

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。

3、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。

4、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，N

ginx 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。

5、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。

6、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。

7、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。

8、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。

二、LVS

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、应用范围比较广，因为 LVS 工作在 4 层，所以它几乎可以对所有应用做负载均衡，包括 http、数据库、在线聊天室等等。

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。

三、HAProxy

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 URl 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)，表示根据据 cookie(name)来锁定并哈希每一次 TCP 请求。

linux 系统监控命令，查看 cpu 负载内存等情况

top 命令是 Linux 下常用的性能分析工具，比如 cpu、内存的使用，能够实时显示系统中各个进程的资源占用状况，类似于 Windows 的任务管理器。 top 显示系统当前的进程和其他状况,是一个动态显示过程,即可以通过用户按键来不断刷新当前状态.如果在前台执行该命令,它将独占前台,直到用户终止该程序为止.

比较准确的说,top 命令提供了实时的对系统处理器的状态监视.它将显示系统中 CPU 最“敏感”的任务列表.该命令可以按 CPU 使用.内存使用和执行时间对任务进行排序；而且该命令的很多特性都可以通过交互式命令或者在个人定制文件中进行设定。

top 命令参数

d 指定每两次屏幕信息刷新之间的时间间隔。当然用户可以使用 s 交互命令来改变之。

p 通过指定监控进程 ID 来仅仅监控某个进程的状态。

q 该选项将使 top 没有任何延迟的进行刷新。如果调用程序有超级用户权限，那么 top 将以尽可能高的优先级运行。

S 指定累计模式

s 使 top 命令在安全模式中运行。这将去除交互命令所带来的潜在危险。

i 使 top 不显示任何闲置或者僵死进程。

c 显示整个命令行而不只是显示命令名

常用操作

top //每隔 5 秒显式所有进程的资源占用情况

top -d2//每隔 2 秒显式所有进程的资源占用情况

top -c//每隔 5 秒显式进程的资源占用情况，并显示进程的命令行参数(默认只有进程名)

top -p12345-p6789//每隔 5 秒显示 pid 是 12345 和 pid 是 6789 的两个进程的资源占用情况

top -d2-c-p123456//每隔 2 秒显示 pid 是 12345 的进程的资源使用情况，并显式该进程启动的命令行参数

查看网络流量的命令

watch -n 1 "/sbin/ifconfig eth0 | grep bytes"

keepalive 的工作原理和如何做到健康检查

keepalived 是以 VRRP 协议为实现基础的，VRRP 全称 Virtual Router Redundancy Protocol，即虚拟路由冗余协议。

虚拟路由冗余协议，可以认为是实现路由器高可用的协议，即将 N 台提供相同功能的路由器组成一个路由器组，这个组里面有一个 master 和多个 backup，master 上面有一个对外提供服务的 vip（该路由器所在局域网内其他机器的默认路由为该 vip），master 会发组播，当 backu

p 收不到 vrrp 包时就认为 master 宕掉了，这时就需要根据 VRRP 的优先级来选举一个 backup 当 master。这样的话就可以保证路由器的高可用了。

keepalived 主要有三个模块，分别是 core、check 和 vrrp。core 模块为 keepalived 的核心，负责主进程的启动、维护以及全局配置文件的加载和解析。check 负责健康检查，包括常见的各种检查方式。vrrp 模块是来实现 VRRP 协议的。

Keepalived 健康检查方式配置

HTTP_GET|SSL_GET
HTTP_GET | SSL_GET
{
url {

path /# HTTP/SSL 检查的 url 可以是多个

digest <STRING> # HTTP/SSL 检查后的摘要信息用工具 genhash 生成

status_code 200# HTTP/SSL 检查返回的状态码

}

connect_port 80 # 连接端口

bindto<IPADD>

connect_timeout 3 # 连接超时时间

nb_get_retry 3 # 重连次数

delay_before_retry 2 #连接间隔时间

}

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。

lvs 的三种模式

一、NAT 模式（VS-NAT）

原理：就是把客户端发来的数据包的 IP 头的目的地址，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 处理完成后把数据交给经过负载均衡器,负载均衡器再把数据包的原 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。

服务器开不了机怎么解决一步步的排查

硬件有无报警灯提示

主面板液晶面板有没有提示什么报错信息，例如 raid 错误的提示信息

先排除硬件还是软件问题

Linux 系统中病毒怎么解决

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：

linux 服务器流量剧增,用 iftop 查看有连接外网的情况。netstat 连接的外网 ip 和端口。

#top 命令找到 cpu 使用率高的进程，一般病毒文件命名都比较乱

#可以用 ps aux 查看是否有不明进程，找出病毒文件的位置

#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疑文件等

chkconfig --list | grep 3:on

服务器启动级别是 3 的，检查一下了开机启动项，没有特别明显的服务。然后检查了一下开机启动的一个文件，more /etc/rc.local

发现一个病毒文件你删了他又自动创建怎么解决

ps axu 一个个排查，方法是查看可疑的用户和系统相似而又不是的进程找出进程可疑。

杀掉所有与病毒相关的进程，然后删掉病毒这个可执行文件，最后删除病毒创建的文件

日志文件很大，怎么把他们切分

针对这些日志按每或每周进行分割，例如只保留一周的数据，用 logrotate 来实现日志的轮替。

或者编写日志文件大小监控脚本，定期检查该日志文件的大小，接近设定大小时，进行轮换。

如果日志文件存在并且很大，可以用 Linux 下的 split 进行文件分割：

模式一：指定分割后文件行数

Split:按指定的行数截断文件格式: split [-n] file [name]参数说明：-n: 指定截断的每一文件的长度，

不指定缺省为 1000 行 file: 要截断的文件 name： 截断后产生的文件的文件名的开头字母，不

指定，缺省为 x，即截断后产生的文件的文件名为 xaa,xab....直到 xzz

模式二：指定分割后文件大小

命令：split -b 10m server.log server_part_

其中 server.log 是要分割的文件，server_part_是分割文件的前缀。

对二进制文件我们同样也可以按文件大小来分隔

tcp/ip 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、ASCll、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）：

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

请写出 apache2.X 版本的两种工作模式，以及各自工作原理。如何查看 apache 当前所支持的模块，并且查看是工作在哪种模式下？

prefork(多进程，每个子进程产生一个线程)和 worker（多进程多线程，每个进程生成多个线程)

其主要工作方式是：当 Apache 服务器启动后，mpm_prefork 模块会预先创建多个子进程(默认为 5 个)，每个子进程只有一个线程，当接收到客户端的请求后，mpm_prefork 模块再将请求转交给子进程处理，并且每个子进程同时只能用于处理单个请求。如果当前的请求数将超过预先创建的子进程数时，mpm_prefork 模块就会创建新的子进程来处理额外的请求。Apache 总是试图保持一些备用的或者是空闲的子进程用于迎接即将到来的请求。这样客户端的请求就不需要在接收后等候子进程的产生。

由于在 mpm_prefork 模块中，每个请求对应一个子进程，因此其占用的系统资源相对其他两种模块而言较多。不过 mpm_prefork 模块的优点在于它的每个子进程都会独立处理对应的单个请求，这样，如果其中一个请求出现问题就不会影响到其他请求。Prefork 在效率上要比 Worker 要高，但是内存使用大得多不擅长处理高并发的场景。

Worker 模式(多线程多进程)：

和 prefork 模式相比，worker 使用了多进程和多线程的混合模式，worker 模式也同样会先预派生一些子进程，然后每个子进程创建一些线程，同时包括一个监听线程，每个请求过来会被分配到一个线程来服务。线程比起进程会更轻量，因为线程是通过共享父进程的内存空间，因此，内存的占用会减少一些，在高并发的场景下会比 prefork 有更多可用的线程，表现会更优秀一些；另外，如果一个线程出现了问题也会导致同一进程下的线程出现问题，如果是多个线程出现问题，也只是影响 Apache 的一部分，而不是全部。

总的来说，prefork 方式速度要稍高于 worker，然而它需要的 cpu 和 memory 资源也稍多于 woker。

可以通过命令 httpd -M 或 httpd -l 可以查看 apache 当前的模块

可以通过命令 httpd -V 查看是工作在哪种模式下

你使用过监控软件吗？说说其特点

使用 nagios 对服务器进行监控，其特点侧重于对检测项的状态监控，主要通过 nrpe 实现对远程主机的监控，但也可以通过 snmp 对设备（如路由器、交换机）进行监控，可实时实现手机短信、电子邮件、MSN、飞信报警。

使用 cacti 对服务器进行监控,其特点侧重性能和流量监控并通过图表显示，主要通过 snmp 协议收集监测项数据，可实时实现手机短信、电子邮件、

使用 zabbix 对服务器进行监控，zabbix 是完全开源的工具，整合了 cacti 和 nagios 等特性。

zabbix 可以对主机的性能监控、网络设备性能监控、数据库、FTP 等通用协议监控、多种告警方式、详细的报表图表绘制

支持自动发现网络设备和服务器（可以通过配置自动发现服务器规则来实现）

支持分布式，能集中展示、管理分布式的监控点，扩展性强

可以自己开发完善各类监控（根据相关接口编写程序实现），编写插件容易，可以自定义监控项，报警级别的设置。

数据收集，支持 snmp(包括 trapping and polling)，IPMI，JMX，SSH，TELNET；

如何查看二进制文件的内容

我们一般通过 hexdump 命令 来查看二进制文件的内容。

hexdump -C XXX(文件名) -C 是参数 不同的参数有不同的意义

-C 是比较规范的 十六进制和 ASCII 码显示

-c 是单字节字符显示

-b 单字节八进制显示

-o 是双字节八进制显示

-d 是双字节十进制显示

-x 是双字节十六进制显示

ps aux 中的 VSZ 代表什么意思，RSS 代表什么意思

VSZ:虚拟内存集,进程占用的虚拟内存空间

RSS:物理内存集,进程战用实际物理内存空间

检测并修复/dev/hda5 用什么命令

fsck 用来检查和维护不一致的文件系统。若系统掉电或磁盘发生问题，可利用 fsck 命令对文件系统进行检查

Linux 系统的开机启动顺序

加载 BIOS->

读取 MBR->Boot Loader->

加载内核->

用户层 init 一句 inittab 文件来设定系统运行的等级(一般 3 或者 5，3 是多用户命令行，5 是界面)->

init 进程执行 rc.syninit->

启动内核模块->

执行不同级别运行的脚本程序->

执行/etc/rc.d/rc.local(本地运行服务)->

执行/bin/login,就可以登录了。

符号链接与硬链接的区别

我们可以把符号链接，也就是软连接 当做是 windows 系统里的 快捷方式。

硬链接 就好像是 又复制了一份.

ln 3.txt 4.txt 这是硬链接，相当于复制，不可以跨分区，但修改 3,4 会跟着变，若删除 3,4 不受任何影响。

ln -s 3.txt 4.txt 这是软连接，相当于快捷方式。修改 4,3 也会跟着变，若删除 3,4 就坏掉了。不可以用了。

FTP 的主动模式和被动模式

FTP 协议有两种工作方式：PORT 方式和 PASV 方式，中文意思为主动式和被动式。

PORT（主动）方式的连接过程是：客户端向服务器的 FTP 端口（默认是 21）发送连接请 求，服务器接受连接，建立一条命令链路。当需要传送数据时，客户端在命令链路上用 PORT 命令告诉服务器：“我打开了 XX 端口，你过来连接我”。于是服务器从 20 端口向客户端的 XX 端口发送连接请求，建立一条数据链路来传送数据。

PASV（被动）方式的连接过程是：客户端向服务器的 FTP 端口（默认是 21）发送连接请 求，服务器接受连接，建立一条命令链路。当需要传送数据时，服务器在命令链路上用 PASV 命令告诉客户端：“我打开了 XX 端口，你过来连接我”。于是客户端向服务器的 XX 端口 发送连接请求，建立一条数据链路来传送数据。

从上面可以看出，两种方式的命令链路连接方法是一样的，而数据链路的建立方法就完 全不同。

docker 命令

容器生命周期管理 – docker [run|start|stop|restart|kill|rm|pause|unpause]

容器操作运维 – docker [ps|inspect|top|attach|events|logs|wait|export|port]

容器 rootfs 命令 – docker [commit|cp|diff]

镜像仓库 – docker [login|pull|push|search]

本地镜像管理 – docker [images|rmi|tag|build|history|save|import]

其他命令 – docker [info|version]

docker 怎样实现容器件的独立

1)pid namespace

不同用户的进程就是通过 pid namespace 隔离开的，且不同 namespace 中可以有相同 pid。所有的 LXC 进程在 docker 中的父进程为 docker 进程，每个 lxc 进程具有不同的 namespace 。

2) net namespace

有了 pid namespace, 每个 namespace 中的 pid 能够相互隔离 ,但是网络端口还是共享 host 的端口。网络隔离是通过 net namespace 实现的，每个 net namespace 有独立的 network devices, IP addresses, IP routing tables, /proc/net 目录。这样每个 container 的网络就能隔离开来。docker 默认采用 veth 的方式将 container 中的虚拟网卡同 host 上的一个 docker bridge: docker0 连接在一起。

3) ipc namespace

container 中进程交互还是采用 linux 常见的进程间交互方法 (interprocess communication - IPC),包括常见的信号量、消息队列和共享内存。container 的进程间交互实际上还是 host 上具有相同 pid namespace 中的进程间交互。

4) mnt namespace

类似 chroot，将一个进程放到一个特定的目录执行。mnt namespace 允许不同 namespace 的进程看到的文件结构不同，这样每个 namespace 中的进程所看到的文件目录就被隔离开了。在 container 里头，看到的文件系统，就是一个完整的 linux 系统，有/etc、/lib 等，通过 chroot 实现。

5) uts namespace

UTS("UNIX Time-sharing System") namespace 允许每个 container 拥有独立的 hostname 和 domain name, 使其在网络上可以被视作一个独立的节点而非 Host 上的一个进程。

6) user namespace

每个 container 可以有不同的 user 和 group id, 也就是说可以在 container 内部用 container 内部的用户执行程序而非 Host 上的用户。

有了以上 6 种 namespace 从进程、网络、IPC、文件系统、UTS 和用户角度的隔离，一个 container 就可以对外展现出一个独立计算机的能力，并且不同 container 从 OS 层面实现了隔离。然

而不同 namespace 之间资源还是相互竞争的，仍然需要类似 ulimit 来管理每个 container 所能使用的资源 - -cgroup。

cgroups（Control groups）实现了对资源的配额和度量。

Linux 如何挂载 windows 下的共享目录

mount.cifs //IP 地址/server /mnt/server -o user=administrator,password=123456

linux 下的 server 需要自己手动建一个 后面的 user 与 pass 是 windows 主机的账号和密码 注意空格 和逗号

LNMP 的工作流程：

当 LNMP 工作的时候，首先是用户通过浏览器输入域名请求 Nginx Web 服务，如果是请求的是静态的资源，则由 Nginx 解析返回给用户。

如果是动态的资源，那么久通过 Fast CGI 接口发送给 PHP 引擎服务(Fast CGI 进程 php-fpm)进行解析。

如果这个动态的请求要读取数据库，那么 PHP 就会继续向后请求 MySQL 数据库，读取需要的数据。

最终通过 Nginx 服务把获取的数据返回给用户，这就是 LNMP 的基本流程。

企业选用 MySQL 作为数据库的优点：

1.性能卓越，服务稳定，很少出现异常宕机。

2.开放源代码并且没有版权的限制，自主传播，使用成本低。

3.历史悠久，社区及用户非常活跃，遇到问题很快可以获取帮助。

4.软件体积小安装简单，并且易于维护，安装及维护的成本低。

5.支持多种操作系统，提供 API 接口。

6.品牌效应，使得企业无需考虑就直接使用。

LNMP 环境搭建问题：

当安装 LNMP 一体化环境的时候 MySQL 数据库要装在 Nginx 所在的服务器上，

如果 MySQL 和 Nginx 不在同一台机器上 ,那么 Nginx 服务器上的 MySQL 数据库软件只要解压移动安装目录中就行。

不需要对 MySQL 进行初始化配置。

在 PHP5.3 以上的版本中，Nginx 服务器上安装了 MySQL 软件，只需要在编译 PHP 的时候指定相关参数即可。

编译参数：--with-mysql=mysqlnd

表示在编译的时候会调用内置的 MySQL 的库。

什么是 FCGI：

FastCGI 是一个可伸缩的、高速的在 HTTP 服务器的动态脚本语言间通信的接口(在 Linux 下，FastCGI 就是 socket，这个 socket 可以是文件 socket 或 IPsocket)。

FastCGI 采用 C/S 架构 ,它可以将 HTTP 服务器和脚本服务器分开 ,同时还可以在脚本解析服务器上启动一个或多个服务器来解析守护进程。

当 HTTP 服务器遇到动态程序的时候，可以将其直接交付给 FastCGI 进程来执行，然后将得到的结果返回给浏览器。这种方式可以让 HTTP 服务器专一的处理静态的请求。

这会很高的提高整个应用系统的性能。

FastCGI 的重要特点：

1.HTTP 服务器和动态脚本语言间通信的接口或工具。

2.可以把动态语言解析或 HTTP 服务器分离开。

3.Nginx、Apache、Lighttpd，以及多数动态语言都支持 FastCGI。

4.PHP 动态语言方式采用 C/S 结构 ,分为客户端(HTTP 服务器)和服务器端(动态语言解析服务器)。

5.PHP 动态语言服务器端可以启动多个 FastCGI 的守护进程。

6.HTTP 服务器通过 FastCGI 客户端和动态语言 FastCGI 服务器端通信。

Nginx FCGI 运行原理：

Nginx 不支持对外部动态程序的直接调用或者解析。所有的外部程序(包括 PHP)必须通过 FastCGI 接口来调用。

FastCGI 接口在 Linux 下是一个 socket，为了调用 CGI 程序，还需要一个 FastCGI 的 wrapper(可以理解为用户启动另一个程序的程序)，这个 wrapper 绑定在某个固定的 socket 上。

当 Nginx 将 CGI 请求发送给这个 socket 的时候，通过 FastCGI 接口，wrapper 接受到请求，然后派生出一个新的线程，这个线程调用解释器或外部的程序处理脚本来读取返回的数据。

然后 wrapper 再将返回的数据通过 FastCGI 接口，沿着固定的 socket 传递给 Nginx，最后 Nginx 将返回的数据发送给客户端。

FastCGI 的主要优点就是把动态的语言和 HTTP 服务器分离开来，使 Nginx 专门处理静态的请求，动态的请求直接使用 PHP/PHP-FPM 服务器专门处理。

灰度发布如何实现？

仔细考虑一下灰度发布系统要达到哪些目的，基本就能有答案了。需要注意的是，客户端应用（无论 PC 端还是移动端）的灰度发布要比 Web 应用的灰度发布更为复杂，因为应用运行在用户持有的终端上，数据采集和回滚都更为困难（但可采集的数据类型要更加丰富）。

注：本人缺乏移动客户端产品的经验，下述内容可能不适用于移动客户端产品。

我所理解的灰度发布系统，主要任务是从产品用户群中按照一定策略选取部分用户，让他们先行体验新版本的应用，通过收集这部分用户对新版本应用的显式反馈（论坛、微博）或隐

式反馈（应用自身统计数据），对新版本应用的功能、性能、稳定性等指标进行评判，进而决定继续放大新版本投放范围直至全量升级或回滚至老版本。

从上述描述可以得出灰度发布系统需要具备的一些要素：

用户标识

用于区分用户，辅助数据统计，保证灰度发布过程中用户体验的连贯性（避免用户在新旧版本中跳变，匿名 Web 应用比较容易有这个问题）。匿名 Web 应用可采用 IP、Cookie 等，需登录的应用可直接采用应用的帐号体系。

目标用户选取策略

即选取哪些用户先行体验新版本，是强制升级还是让用户自主选择等。可考虑的因素很多，包括但不限于地理位置、用户终端特性（如分辨率、性能）、用户自身特点（性别、年龄、忠诚度等）。对于细微修改（如文案、少量控件位置调整）可直接强制升级，对于类似新浪微博改版这样的大型升级，应让用户自主选择，最好能够提供让用户自主回滚至旧版本的渠道。

对于客户端应用，可以考虑类似 Chrome 的多 channel 升级策略，让用户自主选择采用 stable、beta、unstable channel 的版本。在用户有明确预期的情况下自行承担试用风险。

数据反馈

用户数据反馈：在得到用户允许的前提下，收集用户的使用新版本应用的情况。如客户端性能、客户端稳定性、使用次数、使用频率等。用于与旧版本进行对比，决策后续是继续扩大新版本投放范围还是回滚。

服务端数据反馈：新版本服务端性能、服务端稳定性等，作用与用户数据反馈类似。

新版本回滚策略

当新版本灰度发布表现不佳时，应回滚至旧版本。对于纯粹的 Web 应用而言，回滚相对简单。主要难点在于用户数据的无缝切换。对于客户端应用，如果期待用户自行卸载新版本另行安装旧版本，成本和流失率都太高。可以考虑通过快速另行发布新版本，利用升级来“回滚”，覆盖上次灰度发布的修改。

对于移动客户端，新版本发布成本较高，需要 Appstore、Market 审核。本人没有移动客户端产品的经验，不太确定移动客户端产品如何处理灰度发布及回滚。但尽量将客户端打造成 Web App，会更有利于升级和回滚。（不过苹果对纯 Web App 类的 App 有较强的限制，好像已经不允许在 Appstore 上发布这类应用了？）

新版本公关运营支持

对于改版级别的大型升级，需要配合公关运营支持，用于及时处理用户在微博、博客等渠道给出的“显式反馈”。对比通过隐式数据反馈得到的结论后，综合考虑应对策略。

TCP/IP 原理说一下？TCP 有哪几个状态，分别是什么意思？

以 tcp/ip 协议为核心,分五层。tcp 工作在第 4 层，主要有 tcp 和 udp 协议。其中 tcp 是可靠协议，udp 是不可靠协议。 tcp 传输之前，需要建立连接，通过三次握手实现。

TCP 三次握手状态：首先是 closed 状态，当发起连接后，进入 Listen 状态，当三次握手之后，进入 EST 状态。三次握手中间还有一个临时状态:SYN_SENT。SYN_SENT 当应用程序发送 ack 之后，进入 EST 状态,如果没有发送，就关闭 closed.

有个客户说访问不到你们的网站，但是你们自己测试内网和外网访问都没问题。你会怎么排查并解决客户的问题？

我们自己测了都没问题，只是这个客户访问有问题，那肯定是要先联系到这个客户，能远程最好，问一下客户的网络是不是正常的，访问其它的网站有没有问题（比如京东、百度什么的）。如果访问其它网站有问题，那叫客户解决本身网络问题。如果访问其它网站都没问题，用 ping 和 nslookup 解析一下我们的网站是不是正常的，让客户用 IP 来访问我们的网站是否可行，如果 IP 访问没问题，那就是客户的 DNS 服务器有问题或者 DNS 服务器解析不到我们的网站。还有一种可能就是跨运营商访问的问题，比如我们的服务器用的是北方联通、而客户用的是南方移动，就也有可能突然在某个时间段访问不到，这种情况在庞大的中国网络环境中经常发生（一般是靠 CDN 解决）。还有可能就是我们的网站没有 SSL 证书，在公网是使用的是 http 协议，这种情况有可能就是没有用 https 协议网站被运营商劫持了。

redhat 6.X 版本系统 和 centos 7.X 版本有啥区别？

桌面系统（6/GNOE2.x、7/GNOME3.x）、
文件系统（6/ext4、7/xfs）、
内核版本（6/2.6x、7/3.10x）、
防火墙（6/iptables、7/firewalld）、
默认数据库（6/mysql、7/mariadb）、
启动服务（6/service 启动、7/systemctl 启动）、
网卡（6/eth0、7/ens192）等。

你会用什么方法查看某个应用服务的流量使用情况？

如果是单一应用的服务器，只需要用 iftop、sar 等工具统计网卡流量就可以。如果服务器跑了多个应用，可以使用 nethogs 工具实现，它的特别之处在于可以显示每个进程的带宽占用情况，这样可以更直观获取网络使用情况。

说一下你们公司怎么版本发布的（代码怎么发布的）？

发布：jenkins 配置好代码路径（SVN 或 GIT），然后拉代码，打 tag。需要编译就编译，编译之后推送到发布服务器（jenkins 里面可以调脚本），然后从分发服务器往下分发到业务服务器上。

elk 中的 logstash 是怎么收集日志的，在客户端的 logstash 配置文件主要有哪些内容？

input、output 两大块配置；input 中指定日志（type、path）等，output 指定日志输出的目标（host、port）等。

ansible 你用过它的哪些模块，ansbile 同时分发多台服务器的过程很慢（它是逐台分发的），你想过怎么解决吗？

用过 ansible 的（copy file yum ping command shell）等模块；ansible 默认只会创建 5 个进程，所以一次任务只能同时控制 5 台机器执行.那如果你有大量的机器需要控制,或者你希望减少进程数,那你可以采取异步执行.ansible 的模块可以把 task 放进后台,然后轮询它.这使得在一定进程数下能让大量需要的机器同时运作起来.

nginx 有哪几种调度算法，解释一下 ip　hash 和轮询有啥不一样？

常用的有 3 种调度算法（轮询、ip hash、权重）。
轮询：upstream 按照轮询（默认）方式进行负载，每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器 down 掉，能自动剔除。
ip hash：每个请求按访问 ip 的 hash 结果分配，这样每个访客固定访问一个后端服务器，可以解决 session 的问题。
权重：指定轮询几率，权重（weight）和访问比率成正比，用于后端服务器性能不均的情况。

nginx 你用到了哪些模块，在 proxy 模块中你配置过哪些参数？

用到过（负载均衡 upstream、反向代理 proxy_pass、location、rewrite 等）。
proxy 模块中配置过:proxy_set_header、proxy_connect_timeout、proxy_send_timeout、proxy_buffer_*

说一下 iptables 的原理，有哪些表、哪些链？怎么修改默认策略全部为 DROP?

iptables 是工作在 TCP/IP 的 2、3、4 层。你要说它的原理也不是几话能概括的，当主机收到一个数据包后，数据包先在内核空间中处理，若发现目的地址是自身，则传到用户空间中交给对应的应用程序处理，若发现目的不是自身，则会将包丢弃或进行转发。

4 张表（raw 表、mangle 表、net 表、filter 表）

5 条链（INPUT 链、OUTPUT 链、PORWARD 链、PREROUTING 链、POSTROUTING 链）。

全部设置为 DROP：

#iptables -P INPUT DROP

#iptables -P OUTPUT DROP

#iptables -P FORWARD DROP

小结：iptables 远不止这几句话就能描述清楚的，也不是随便在网上趴些资料就能学好的，需要自己用起来，经过大量的实验和实战才能熟悉它，iptables 真的很考验运维人员的技术水平，大家一定要用心学好这个 iptables。

nginx 中 rewrite 有哪几个 flag 标志位（last、break、redirect、permanent），说一下都什么意思？

last ： 相当于 Apache 的[L]标记，表示完成当前的 rewrite 规则

break ： 停止执行当前虚拟主机的后续 rewrite 指令集

redirect ： 返回 302 临时重定向，地址栏会显示跳转后的地址

permanent ： 返回 301 永久重定向，地址栏会显示跳转后的地址

301 和 302 不能简单的只返回状态码，还必须有重定向的 URL，这就是 return 指令无法返回 301,302 的原因了。这里 last 和 break 区别有点难以理解：

last 一般写在 server 和 if 中，而 break 一般使用在 location 中

last 不终止重写后的 url 匹配，即新的 url 会再从 server 走一遍匹配流程，而 break 终止重写后匹配

break 和 last 都能组织继续执行后面的 rewrite 指令

你在 shell 脚本中用过哪些语法，case 语法会用到哪些地方？

一般会用到 if 语句、for 语句、while 语句、case 语句以及 function 函数的定义；case 语句为多选择语句，可以用 case 语句匹配一个值与一个模式，如果匹配成功，执行相匹配的命令。最典型的 case 语法会用到启动服务脚本的处理。

linux 系统中你会用到什么命令查看硬件使用状态信息？

这个命令就很多了，

比如：lscpu(查看 cpu 信息)、

free -m（查看内存信息）、

df -h（查看硬盘分区信息）、

top（还可以动态查看 cpu、内存使用情况的信息）、

/proc/目录下也可以查看很多硬件信息。

我要过滤一段文本(test.txt)中第二列的内容？如果这段文件有很多特殊符号，比如用:（冒号）怎么过滤它的第二段？如果我要过滤这段文本中，其中有一行只有 7 个符如何实现？

```
awk '{print $2}'     tset.txt
awk -F':'    '{print $2}'     tset.txt
```

比如开发想找你查看 tomcat 日志，但是 catalia.out 特别大，你不可能用 vi 打开去看，你会怎么查看？如果你用 grep -i"error" 过滤只是包含 error 的行，我想同时过滤 error 上面和下面的行如何实现？

```
grep -i "error" catalia.out
grep -C 1    -i "error" catalia.out
```

参数-C：是匹配前后的行，后面 1 是匹配前后各 1 行

zabbix 如何修改其中监控的一台服务器中内存阈值信息，比如正常内存使用到了 80%报警，我想修改为 60%报警？

正常来说，一般会把监控的服务器统一加入到一个模板中，修改模板的其是某一项的监控项参数和告警阈值后，加入模板中的所有主机都会同步。如果单独想修改其中某一台服务器内存告警阈值，需要进入这台主机，单独创建一个告警 Triggers，关联这台主机监控内存的项，配置好告警的阈值为 60%即可实现。其实，zabbix 一切都为图形化操作，如果没有接触过 zabbix 的朋友，可能听起来不太清楚。

mysql 主从复制原理说一下？

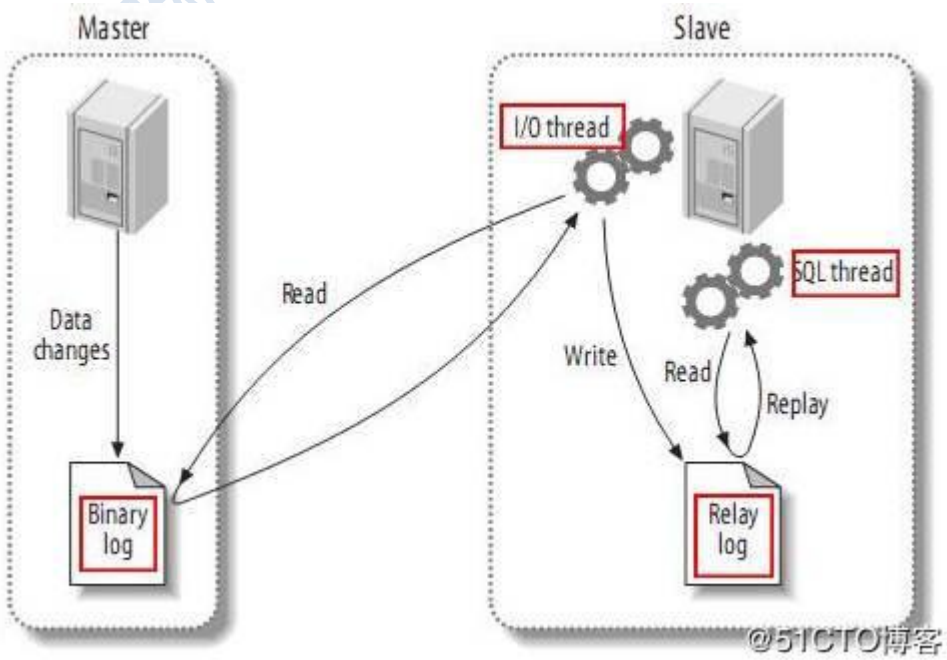
mysql 支持三种复制类型（基于语句的复制、基于行的复制、混合类开进的复制）。

如果你记不住太多内容，可以简单说明一下原理：

(1) master 将改变记录到二进制日志(binary log)中（这些记录叫做二进制日志事件，binary log events）；

(2) slave 将 master 的 binary log events 拷贝到它的中继日志(relay log)；

(3) slave 重做中继日志中的事件，将改变反映它自己的数据。



如果你能详细记住它的原理，可以这么回答：
该过程的第一部分就是 master 记录二进制日志。在每个事务更新数据完成之前，master 在二日志记录这些改变。MySQL 将事务串行的写入二进制日志，即使事务中的语句都是交叉执行的。在事件写入二进制日志完成后，master 通知存储引擎提交事务。
下一步就是 slave 将 master 的 binary log 拷贝到它自己的中继日志。首先，slave 开始一个工作线程——I/O 线程。I/O 线程在 master 上打开一个普通的连接，然后开始 binlog dump process。Binlog dump process 从 master 的二进制日志中读取事件，如果已经跟上 master，它会睡眠并等待 master 产生新的事件。I/O 线程将这些事件写入中继日志。
SQL slave thread（SQL 从线程）处理该过程的最后一步。SQL 线程从中继日志读取事件，并重放其中的事件而更新 slave 的数据，使其与 master 中的数据一致。只要该线程与 I/O 线程保持一致，中继日志通常会位于 OS 的缓存中，所以中继日志的开销很小。

用什么命令可以查看上一次服务器启动的时间、上一次谁登录过服务器？

last 命令查看上次服务器启动时间。
w 命令　查看登录。

redis 集群原理说一下，正常情况下 mysql 有多个库，redis 也有多个库，我怎么进入 redis 集群中的第 2 个库？还有，我想查看以 BOSS 开头的值？redis 持久化是如何实现（一种是 RDS、一种是 AOF），说一下他们有啥不一样？

这个 redis 原理的问题又问到了，看样子很多面试官都很关心这个 redis，在上一篇文章笔者的一次面试也有这个面试问题。
【集群原理】：其实它的原理不是三两句话能说明白的，redis 3.0 版本之前是不支持集群的，官方推荐最大的节点数量为 1000，至少需要 3(Master)+3(Slave)才能建立集群，是无中心的分布式存储架构，可以在多个节点之间进行数据共享，解决了 Redis 高可用、可扩展等问题。集群可以将数据自动切分(split)到多个节点，当集群中的某一个节点故障时，redis 还可以继续处理客户端的请求。
【切库】：单机情况下用 select 2 可以切换第 2 个库，select 1 可以切换第 1 个库。但是集群环境下不支持 select。
可参考 https://yq.aliyun.com/articles/69349
【redis 持久化】：持久化通俗来讲就是将内存中的数据写入硬盘中，redis 提供了两种持久化的功能（RDB、AOF），默认使用 RDB 的方式。
RDB：全量写入持久化，而 RDB 持久化也分两种（SAVE、BGSAVE）。
SAVE 是阻塞式的 RDB 持久化，当执行这个命令时 redis 的主进程把内存里的数据库状态写入到 RDB 文件（即上面的 dump.rdb）中，直到该文件创建完毕的这段时间内 redis 将不能处理任何命令请求。
　　BGSAVE 属于非阻塞式的持久化，它会创建一个子进程专门去把内存中的数据库状态写入 RDB 文件里，同时主进程还可以处理来自客户端的命令请求。但子进程基本是复制的父进程，这等于两个相同大小的 redis 进程在系统上运行，会造成内存使用率的大幅增加。
AOF：与 RDB 的保存整个 redis 数据库状态不同，AOF 的持久化是通过命令追加、文件写入和文件同步三个步骤实现的。AOF 是通过保存对 redis 服务端的写命令（如 set、sadd、rpush）来记录数据库状态的，即保存你对 redis 数据库的写操作。
为了大家能够更好的理解 redis 持久化，笔者建议大家可以看下这两篇文章会比较好理解：
https://www.cnblogs.com/Fairy-02-11/p/6182478.html
http://blog.csdn.net/mishifangxiangdefeng/article/details/48977269

你在工作的过程中，遇到过你映像最深的是什么故障问题，你又是如何解决？

这个问题主要也是考你排查故障的思路及用到的相关命令工具，其每个人在工作中都会遇到各种各样的问题（不管是网络问题、应用配置问题、还是 APP 打开慢/网站打开慢）等等。你只要记住一个你映像最为深刻、最为典型的故障就行。笔者也遇到过各种问题，我在这里就是写出来，怕误导了大家。

在 linux 服务器上，不管是用 rz -y 命令还是 tftp 工具上传，我把本地的一个文件上传到服务器完成后，服务器上还是什么都没有，这有可能是什么问题？

根据这种现象有可能是：服务器磁盘满了；文件格式破坏了；或者你用的是普通用户上传，正好上传的目录没有权限；还有可能就是你上传的文件大小超出了该目录空间的范围。

你在工作中都写过什么脚本？

这个问题的回答别把话说得太大了，要结合实际情况来回答。写过 mysql、redis、mongodb 等数据库备份的脚本；服务器文件备份的脚本；日常代码发布的脚本；之前用 nagios 的时候写过一些 nagios 插件的脚本。

rsync+inotify 是实现文件实时同步的，加什么参数才能实现实时同步，--delete 参数又是什么意思？

rsync 是远程同步工具、inotify 是一种强大的异步文件系统系统监控机制。通过 inotifywait 中的 -m 参数可以实现“始终保持事件监听状态”。rsync 中的-delete 参数是指“ 删除那些 DST 中 SRC 没有的文件”。

在 linux 系统中，一般都会有 swap 内存，你觉得使用 swap 内存有什么好处，在什么情况下 swap 内存才会被使用？你觉得在生产环境中要不要用 swap 内存？

好处：在内存不够用的时候，将部分内存上的数据交换到 swap 空间上，以便让系统不会因为内存不够用而导致 oom 或者更致命的情况出现。

什么情况下会用 swap：当系统的物理内存不够用的时候，就需要将物理内存中的一部分空间释放出来，以供当前运行的程序使用。那些被释放的空间可能来自一些很长时间没有什么操作的程序，这些被释放的空间被临时保存到 swap 空间中，等到那些程序要运行时，再从 swap 中恢复保存的数据到内存中。这样，系统总是在物理内存不够时，才进行 swap 交换。

怎么查看两台服务器之间的网络是不是正常的，服务器是禁 ping 的？

不能用 ping，那可以用 telnet 对方服务器的端口、或者互相访问对方打开的服务。

比如我访问百度网站，有什么方法可以跟踪经过了哪些网络节点？

这个太简单了吧，干运维必备的网络排查技能。用 tracert 命令就可以跟踪，主要是查询本机到另一个主机经过的路由跳数及数据延迟情况。然后你也可以把具体跟踪后输出的信息也说出来，你能说出来都是为你加分的。

如果你们公司的网站访问很慢，你会如何排查？

其实这种问题都没有具体答案，只是看你回答的内容与面试官契合度有多高，能不能说到他想要的点上，主要是看你排查问题的思路。我是这么说的：问清楚反应的人哪个服务应用或者页面调取哪个接

口慢，叫他把页面或相关的 URL 发给你，首先，最直观的分析就是用浏览器按 F12，看下是哪一块的内容过慢（DNS 解析、网络加载、大图片、还是某个文件内容等），如果有，就对症下药去解决（图片慢就优化图片、网络慢就查看内网情况等）。其次，看后端服务的日志，其实大多数的问题看相关日志是最有效分析，最好用 tail -f 跟踪一下日志，当然你也要点击测试来访问接口日志才会打出来。最后，排除 sql，，找到 sql 去 mysql 执行一下，看看时间是否很久，如果很久，就要优化 SQL 问题了，expain 一下 SQL 看看索引情况啥的，针对性优化。数据量太大的能分表就分表，能分库就分库。如果 SQL 没啥问题，那可能就是写的逻辑代码的问题了，一行行审代码，找到耗时的地方改造，优化逻辑。

给你一套环境，你会如何设计高可用、高并发的架构？

如果这套环境是部署在云端(比如阿里云)，你就不用去考虑硬件设计的问题。可直接上阿里云的 SLB +ECS+RDS 这套标准的高可用、高并发的架构。对外服务直接上 SLB 负载均衡技术，由阿里的 SLB 分发到后端的 ECS 主机；ECS 主机部署多台，应用拆分在不同的 ECS 主机上，尽量细分服务。数据库用 RDS 高可用版本（一主一备的经典高可用架构）、或者用 RDS 金融版（一主两备的三节点架构）。在结合阿里其它的服务就完全 OK，业务量上来了，主机不够用了，直横向扩容 ECS 主机搞定。

如果这套环境托管在 IDC，那么你就要从硬件、软件（应用服务）双面去考虑了。硬件要达到高可用、高并发公司必须买多套网络硬件设备（比如负载设备 F5、防火墙、核心层交换、接入层交换）都必须要冗余，由其是在网络设计上，设备之间都必须有双线连接。设备如果都是跑的单机，其中一个设备挂了，你整个网络都瘫痪了，就谈不上高可用、高并发了。其次在是考虑应用服务了，对外服务我会采用成熟的开源方案 LVS+Keepalived 或者 Nginx+Keepalived，缓存层可以考虑 redis 集群及 Mongodb 集群，中间件等其它服务可以用 kafka、zookeeper，图片存储可以用 fastDFS 或 MFS，如果数据量大、又非常多，那么可采用 hadoop 这一套方案。后端数据库可采用 “主从+MHA”。这样一套环境下来是绝对满足高可用、高并发的架构。

你会使用哪些虚拟化技术？

vmware vsphere 及 kvm，我用得比较多的是 vmware vsphere 虚拟化，几本上生产环境都用的 vmware vsphere，kvm 我是用在测试环境中使用。vmware 是属于原生架构虚拟化技术，也就是可直接在硬件上运行。kvm 属于寄居架构的虚拟化技术，它是依托在系统之上运行。vmware vcenter 管理上比较方便，图形管理界面功能很强大，稳定性强，一般比较适合企业使用。KVM 管理界面稍差点，需要管理人员花费点时间学习它的维护管理技术。

假如有人反应，调取后端接口时特别慢，你会如何排查？

其实这种问题都没有具体答案，只是看你回答的内容与面试官契合度有多高，能不能说到他想要的点上，主要是看你排查问题的思路。我是这么说的：问清楚反应的人哪个服务应用或者页面调取哪个接口慢，叫他把页面或相关的 URL 发给你，首先，最直观的分析就是用浏览器按 F12，看下是哪一块的内容过慢（DNS 解析、网络加载、大图片、还是某个文件内容等），如果有，就对症下药去解决（图片慢就优化图片、网络慢就查看内网情况等）。其次，看后端服务的日志，其实大多数的问题看相关日志是最有效分析，最好用 tail -f 跟踪一下日志，当然你也要点击测试来访问接口日志才会打出来。最后，排除 sql，，找到 sql 去 mysql 执行一下，看看时间是否很久，如果很久，就要优化 SQL 问题了，expain 一下 SQL 看看索引情况啥的，针对性优化。数据量太大的能分表就分表，能分库就分库。如果 SQL 没啥问题，那可能就是写的逻辑代码的问题了，一行行审代码，找到耗时的地方改造，优化逻辑。

说一下用过哪些监控系统？

笔者曾经用过 zabbix、nagios、 cacit 等。但是在这次面试中只说用过 zabbix 和 nagios。说完了之后，面试官就让我说一下这两个监控有啥区别：

从 web 功能及画图来讲：

Nagios 简单直观，报警与数据都在同一页面， 红色即为问题项。Nagios web 端不要做任何配置。Nagios 需要额外安装插件，且插件画图不够美观。

Zabbix 监控数据与报警是分开的，查看问题项需要看触发器，查看数据在最新数据查看。而且 zabbix 有很多其它配置项， zabbix 携带画图功能，且能手动把多个监控项集在一个图中展示。

从监控服务来讲：

Nagios 自带的监控项很少。对一些变动的如多个分区、多个网卡进行监控时需要手动配置。

Zabbix 自带了很多监控内容，感觉 zabbix 一开始就为你做了很多事，特别是对多个分区、多个网卡等自动发现并进行监控时，那一瞬间很惊喜，很省心的感觉。

从批量配置和报警来讲：

Nagios 对于批量监控主机，需要用脚本在 server 端新增 host，并拷贝 service 文件。 Nagios 用脚本来修改所有主机的 services 文件，加入新增服务。

Zabbix 在 server 端配置自动注册规则，配置好规则后，后续新增 client 端不需要对 server 端进行操作。 Zabbix 只需手动在模板中新增一监控项即可。

总体来讲：

Nagios 要花很多时间写插件，Zabbix 要花很多时间探索功能。

Nagios 更易上手，Nagios 两天弄会，Zabbix 两周弄会。

Zabbix 画图功能比 Nagios 更强大

Zabbix 对于批量监控与服务更改，操作更简洁；Nagios 如果写好自动化脚本后，也很简单，问题在于写自动化脚本很费神。

edis 集群的原理，redis 分片是怎么实现的，你们公司 redis 用在了哪些环境？

reids 集群原理：

其实它的原理不是三两句话能说明白的，redis 3.0 版本之前是不支持集群的，官方推荐最大的节点数量为 1000，至少需要 3(Master)+3(Slave)才能建立集群，是无中心的分布式存储架构，可以在多个节点之间进行数据共享，解决了 Redis 高可用、可扩展等问题。集群可以将数据自动切分(split)到多个节点，当集群中的某一个节点故障时，redis 还可以继续处理客户端的请求。

redis 分片：

分片(partitioning)就是将你的数据拆分到多个 Redis 实例的过程，这样每个实例将只包含所有键的子集。当数据量大的时候，把数据分散存入多个数据库中，减少单节点的连接压力，实现海量数据存储。分片部署方式一般分为以下三种：

（1）在客户端做分片；这种方式在客户端确定要连接的 redis 实例，然后直接访问相应的 redis 实例；

（2）在代理中做分片；这种方式中，客户端并不直接访问 redis 实例，它也不知道自己要访问的具体是哪个 redis 实例，而是由代理转发请求和结果；其工作过程为：客户端先将请求发送给代理，代理通过分片算法确定要访问的是哪个 redis 实例，然后将请求发送给相应的 redis 实例，redis 实例将结果返回给代理，代理最后将结果返回给客户端。

（3）在 redis 服务器端做分片；这种方式被称为“查询路由”，在这种方式中客户端随机选择一个 redis 实例发送请求，如果所请求的内容不再当前 redis 实例中它会负责将请求转交给正确的 redis 实例，也有的实现中，redis 实例不会转发请求，而是将正确 redis 的信息发给客户端，由客户端再去向正确的 redis 实例发送请求。

redis 用在了哪些环境：

java、php 环境用到了 redis，主要缓存有登录用户信息数据、设备详情数据、会员签到数据等

LVS 有哪些负载均衡技术和调度算法，和 Nginx 负载有啥区别？

LVS 是 Liunx 虚拟服务器的简称，利用 LVS 提供的负载均衡技术和 linux 操作系统可实现高性能、高可用的服务器集群，一般 LVS 都是位于整个集群系统的最前端，由一台或者多台负载调度器（Director Server）组成，分发给应用服务器（Real Server）。它是工作在 4 层（也就是 TCP/IP 中的传输层），LVS 是基于 IP 负载均衡技术的 IPVS 模块来实现的，IPVS 实现负载均衡机制有三种，分别是 NAT、TUN 和 DR，详述如下：

- **VS/NAT：** 即（Virtual Server via Network Address Translation）

也就是网络地址翻译技术实现虚拟服务器，当用户请求到达调度器时，调度器将请求报文的目标地址（即虚拟 IP 地址）改写成选定的 Real Server 地址，同时报文的目标端口也改成选定的 Real Server 的相应端口，最后将报文请求发送到选定的 Real Server。在服务器端得到数据后，Real Server 返回数据给用户时，需要再次经过负载调度器将报文的源地址和源端口改成虚拟 IP 地址和相应端口，然后把数据发送给用户，完成整个负载调度过程。

可以看出，在 NAT 方式下，用户请求和响应报文都必须经过 Director Server 地址重写，当用户请求越来越多时，调度器的处理能力将称为瓶颈。

- **VS/TUN ：**即（Virtual Server via IP Tunneling）

也就是 IP 隧道技术实现虚拟服务器。它的连接调度和管理与 VS/NAT 方式一样，只是它的报文转发方法不同，VS/TUN 方式中，调度器采用 IP 隧道技术将用户请求转发到某个 Real Server，而这个 Real Server 将直接响应用户的请求，不再经过前端调度器，此外，对 Real Server 的地域位置没有要求，可以和 Director Server 位于同一个网段，也可以是独立的一个网络。因此，在 TUN 方式中，调度器将只处理用户的报文请求，集群系统的吞吐量大大提高。

- VS/DR： 即（Virtual Server via Direct Routing）

也就是用直接路由技术实现虚拟服务器。它的连接调度和管理与 VS/NAT 和 VS/TUN 中的一样，但它的报文转发方法又有不同，VS/DR 通过改写请求报文的 MAC 地址，将请求发送到 Real Server，而 Real Server 将响应直接返回给客户，免去了 VS/TUN 中的 IP 隧道开销。这种方式是三种负载调度机制中性能最高最好的,但是必须要求 Director Server 与 Real Server 都有一块网卡连在同一物理网段上。

回答负载调度算法，IPVS 实现在八种负载调度算法，我们常用的有四种调度算法（轮叫调度、加权轮叫调度、最少链接调度、加权最少链接调度）。一般说了这四种就够了，也不会需要你详细解释这四种算法的。你只要把上面 3 种负载均衡技术讲明白面试官就对这道问题很满意了。接下来你在简单说下与 nginx 的区别：

LVS 的优点：

抗负载能力强、工作在第 4 层仅作分发之用，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；无流量，同时保证了均衡器 IO 的性能不会受到大流量的影响；

工作稳定，自身有完整的双机热备方案，如 LVS+Keepalived 和 LVS+Heartbeat；

应用范围比较广，可以对所有应用做负载均衡；

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。

LVS 的缺点：

软件本身不支持正则处理，不能做动静分离，这就凸显了 Nginx/HAProxy+Keepalived 的优势。

如果网站应用比较庞大，LVS/DR+Keepalived 就比较复杂了，特别是后面有 Windows Server 应用的机器，实施及配置还有维护过程就比较麻烦，相对而言，Nginx/HAProxy+Keepalived 就简单一点

Nginx 的优点：

工作在 OSI 第 7 层，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。它的正则比 HAProxy 更为强大和灵活；

Nginx 对网络的依赖非常小，理论上能 ping 通就就能进行负载功能，这个也是它的优势所在；

Nginx 安装和配置比较简单，测试起来比较方便；

可以承担高的负载压力且稳定，一般能支撑超过几万次的并发量；

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；

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 现在也是非常流行的 web 环境，大有和 LAMP 环境分庭抗礼之势，Nginx 在处理静态页面、特别是抗高并发方面相对 apache 有优势；

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，有需求的朋友可以考虑用其作为反向代理加速器；

Nginx 的缺点：

Nginx 不支持 url 来检测。

Nginx 仅能支持 http 和 Email，这个它的弱势。

Nginx 的 Session 的保持，Cookie 的引导能力相对欠缺。

你会怎么统计当前访问的 IP，并排序？

统计用户的访问 IP，用 awk 结合 uniq、sort 过滤 access.log 日志就能统计并排序好。一般这么回答就够了，当然你还可以说出其它方式来统计，这都是你的加分项。

cpu 单核和多核有啥区别？

很少有面试官会问这样的问题，即然问到了，也要老实回答。还好笔者之前了解过 CPU，我是这么说的：双核 CPU 就是能处理多份任务，顺序排成队列来处理。单核 CPU 一次处理一份任务，轮流处理每个程序任务。双核的优势不是频率，而是对付同时处理多件事情。单核同时只能干一件事，比如你同时在后台 BT 下载，前台一边看电影一边拷贝文件一边 QQ。

监控用什么实现的？

现在公司的业务都跑在阿里云上，我们首选的监控就是用阿里云监控，阿里云监控自带了 ECS、RDS 等服务的监控模板，可结合自定义报警规则来触发监控项。

上家公司的业务是托管在 IDC，用的是 zabbix 监控方案，zabbix 图形界面丰富，也自带很多监控模板，特别是多个分区、多个网卡等自动发现并进行监控做得非常不错，不过需要在每台客户机（被监控端）安装 zabbix agent。

Tomcat 工作模式？

Tomcat 是一个 JSP/Servlet 容器。其作为 Servlet 容器，有三种工作模式：独立的 Servlet 容器、进程内的 Servlet 容器和进程外的 Servlet 容器。

进入 Tomcat 的请求可以根据 Tomcat 的工作模式分为如下两类：

Tomcat 作为应用程序服务器：请求来自于前端的 web 服务器，这可能是 Apache, IIS, Nginx 等；

Tomcat 作为独立服务器：请求来自于 web 浏览器；

你是怎么备份数据的，包括数据库备份？

在生产环境下，不管是应用数据、还是数据库数据首先在部署的时候就会有主从架构、或者集群，这本身就是属于数据的热备份；其实考虑冷备份，用专门一台服务器做为备份服务器，比如可以用 rsync+inotify 配合计划任务来实现数据的冷备份，如果是发版的包备份，正常情况下有台发布服务器，每次发版都会保存好发版的包。