

一、云计算基础

1、Tcp的三次握手和四次分手的详细过程(最好描述出一系列的序号认证)?

如何判断三次握手是否成功？为什么连接的时候是三次握手，关闭的时候却是四次握手？为什么不能用两次握手进行连接？如果已经建立了连接，但是客户端突然出现故障了怎么办？你了解那几款抓包工具？说说怎么使用的？

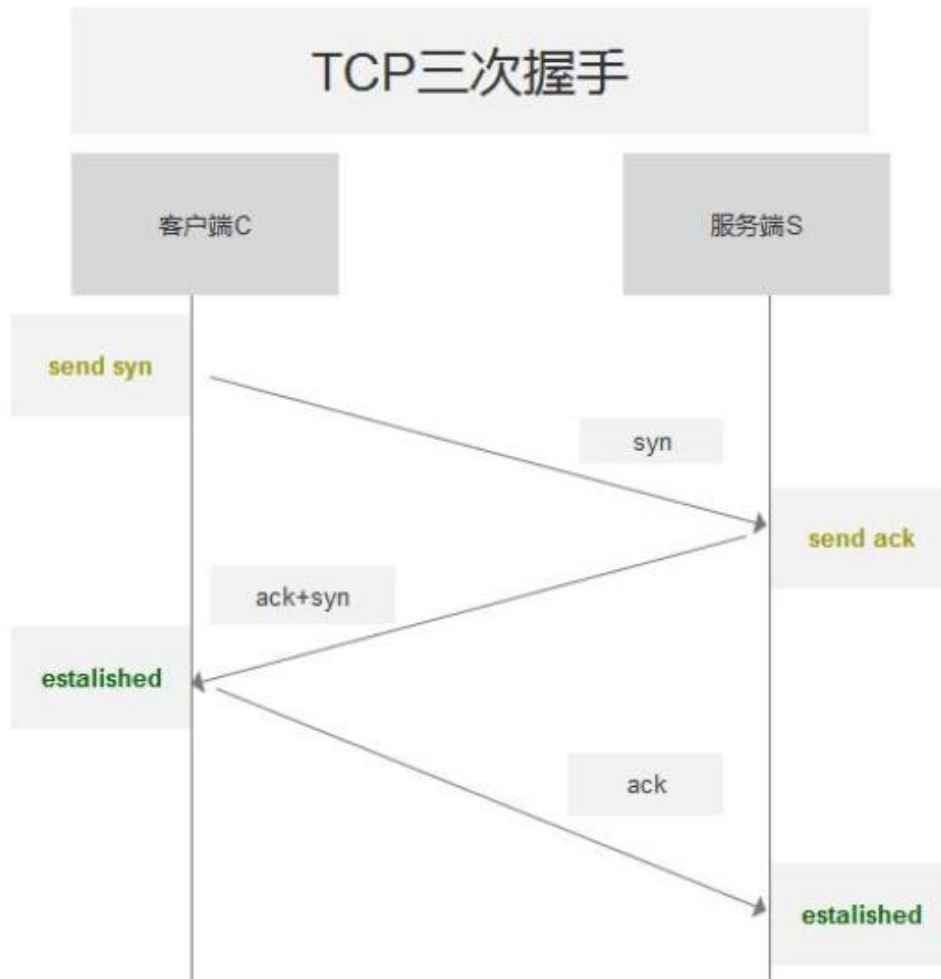
三次握手：

第一次握手：建立连接时,客户端发送syn包($\text{syn}=\text{j}$)到服务器,并进入SYN_SEND状态,等待服务器确认：（SYN：同步序列编号(Synchronize Sequence Numbers)）

第二次握手：服务器收到syn包,必须确认客户的SYN ($\text{ack}=\text{j}+1$) ,同时自己也发送一个SYN包 ($\text{syn}=\text{k}$) ,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ACK($\text{ack}=\text{k}+1$) ,此包发送完毕,客户端和服务器进入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。

注意：第三次握手是为了防止已经失效的连接请求报文段突然又传到服务端，因而产生错误。



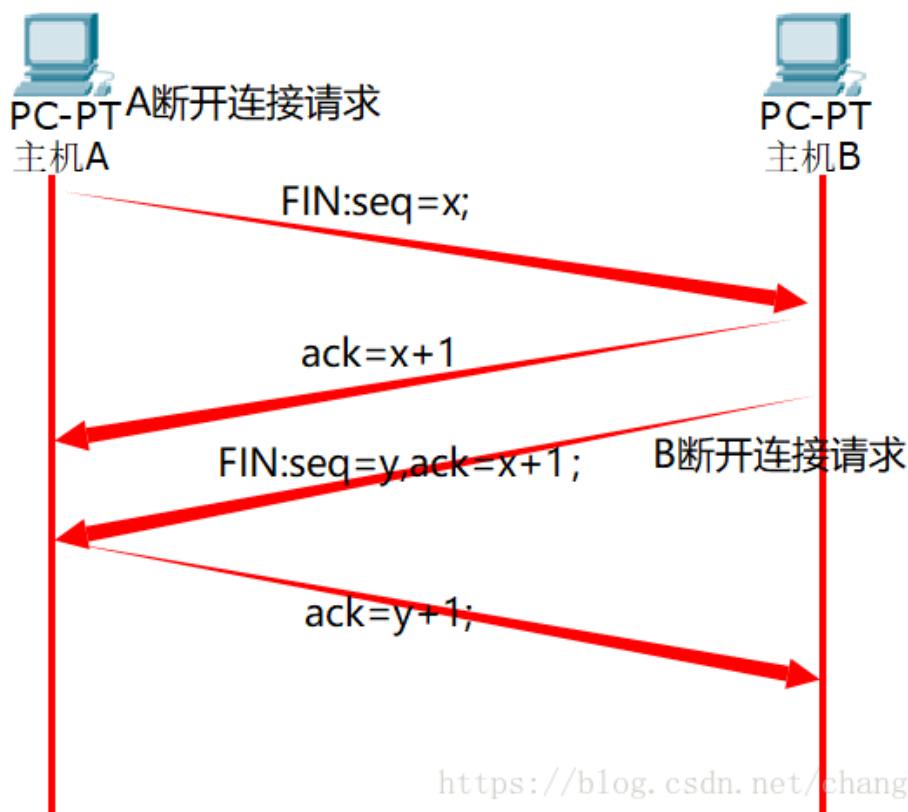
四次分手：

(1) 客户端发送断开tcp连接请求的报文，其中报文中包含seq序列号，是由发送端随机生成的，并且还将报文中的fin字段设置为1，表示需要断开tcp连接。(fin=1, seq=x, x由客户端随机生成)

(2) 服务端会回复客户端发送的tcp断开请求报文，其包含seq序列号，是由回复端随机生成的，而且会产生ack字段，ack字段数值是在客户端发过来的seq序列号基础上加1进行回复，以便客户端收到信息时，知晓自己的tcp断开请求已经得到验证(fin=1, ack=x+1, seq=y, y由服务端随机生成)

(3) 服务端在回复完客户端的tcp断开请求后，不会马上进行tcp连接的断开，服务端会先确保断开前，所有传输到a的数据是否已经传输完毕，一旦确认传输数据完毕，就会将回复报文的fin字段置1，并且生产随机seq序列。(fin=1, ack=x+1, seq=z, z由服务端随机生成)

(4) 客户端收到服务端的tcp断开请求后，会回复服务端的断开请求，包含随机生成的seq字段和ack字段，ack字段会在服务端的tcp断开请求的seq的基础上加1，从而完成服务端请求的验证回复。(fin=1, ack=z+1, seq=h, h为客户端随机生成)至此tcp断开的四次挥手过程完毕。



判断三次握手成功：

使用抓包工具捕获

为什么连接的时候是三次握手，关闭的时候却是四次握手？

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。故需要四次握手。

为什么不能用两次握手进行连接？

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。

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秒钟发送一次。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。

你了解那几款抓包工具？说说怎么使用的？

tcpdump: tcpdump可以将网络中传送的数据包完全截获下来提供分析。它支持针对网络层、协议、主机、网络或端口的过滤，并提供and、or、not等逻辑语句来帮助你去掉无用的信息。

tcpflow实际上也是一个抓包工具，tcpflow与tcpdump不同的是它是以流为单位显示数据内容，而tcpdump以包为单位显示数据。分析HTTP数据，使用tcpflow会更便捷。

https://blog.csdn.net/u013332124/article/details/84112926?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.channel_param&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.channel_param (抓包工具了解)

2、Raid0、Raid1、Raid5的区别？（raid0、raid1、raid5为冗余磁盘阵列）

raid0:条带卷，利用率100%，相对读写速率最快，相对安全性差。数据随机存入到阵列中的一个磁盘上。同时从2块磁盘读数据；读速度与raid1相差不多；

raid1:镜像卷，使用率50%，相对读写速率一般，相对安全性高。最少2块磁盘组成，数据同时存入到两块磁盘上。同时从2块磁盘读数据；写速度会比raid0慢；

raid5:带奇偶校验的镜像卷，相对读写速率较快，相对安全性高可以添加热备磁盘作为冗余。

3、TCP和UDP的区别

TCP: 传输控制协议（TCP, Transmission Control Protocol）是一种面向连接的、可靠的、基于字节的传输层通信协议。TCP旨在适应支持多网络应用的分层协议层次结构。 连接到不同但互连的计算机通信网络的主计算机中的成对进程之间依靠TCP提供可靠的通信服务。

UDP: Internet协议集支持一个无连接的传输层协议，该协议称为用户数据报协议（UDP, User Datagram Protocol），提供面向事务的简单不可靠信息传送服务。UDP 为应用程序提供了一种无需建立连接就可以发送封装的 IP 数据包的方法。

1. TCP要建立连接，UDP不需要

2. TCP可靠，UDP不可靠，是因为有确认包的原因

3. TCP效率低，UDP效率高，一是因为连接的问题，二是因为确认包的问题，但主要的还是因为确认包的问题

4. TCP适合用在较大的数据量，因为UDP的字节一旦超过512，就极易丢失

4、七层协议 (osi模型)，并简述你认为在当今网络中比较重要的部分？（爱云校笔试）

物理层、数据链路层、网络层、传输层、会话层、表示层、应用层

物理层: 传输介质/比特流;

数据链路层: MAC地址/局域网/, 分段目标, 局部地址、邮递员

网络层: IP地址/寻址/路由, 总目标, 全球地址; 具有的协议: IP/互联网协议/寻址、ICMP/网络消息管理协议/测试

传输层: 分段/重组/端口号, 传输效率; 具有的协议: TCP/传输控制协议/可靠、UDP/用户数据报协议/不可靠

会话层: 会话/全双工/半双工, 身份信息, 电话/对讲机

表示层: 格式/压缩/加密, 快速传递, 安全传递

应用层: 应用程序/原始数据/; 具有的协议: HTTP/超文本传输协议/网站、SSH/远程连接协议/远程控制、FTP/文件传输协议、SMTP/简单邮件传输协议/Email、DNS/域名服务/www

解释:

物理层: 将数据转换为可通过物理介质传送的电子信号 相当于邮局中的搬运工人。

数据链路层：决定访问网络介质的方式。在此层将数据分帧，并处理流控制。本层指定拓扑结构并提供硬件寻址，相当于邮局中的装拆箱工人。

网络层：使用权重数据路由经过大型网络 相当于邮局中的排序工人。

传输层：提供端到端的可靠连接 相当于公司中跑邮局的送信职员。

会话层：允许用户使用简单易记的名称建立连接 相当于公司中收寄信、写信封与拆信封的秘书。

表示层：协商数据交换格式 相当公司中简报老板、替老板写信的助理。

应用层：用户的应用程序和网络之间的接口。

重要的部分可以根据每层功能解释，自圆其说即可

5、逻辑卷的优点，及如何制作？

优点：

随意扩张大小，缩减大小，快照备份

1. 文件系统可以跨多个磁盘，因此文件系统大小不会受物理磁盘的限制。
2. 可以在系统运行的状态下动态的扩展文件系统的大小。
3. 可以增加新的磁盘到LVM的存储池中。
4. 可以以镜像的方式冗余重要的数据到多个物理磁盘。
5. 可以方便的导出整个卷组到另外一台机器。

缺点：

1. 在从卷组中移除一个磁盘的时候必须使用**reducevg**命令（这个命令要求**root**权限，并且不允许在快照卷组中使用）。
2. 当卷组中的一个磁盘损坏时，整个卷组都会受到影响。
3. 因为加入了额外的操作，存储性能受到影响。

使用他的时候，需要先创建物理卷(PV)，然后把物理卷合成卷组(VG)，然后在卷组中创建逻辑卷(LV)。

1. 将物理磁盘，转换成物理卷-PV

pvcreate 准备的物理磁盘

例：**pvcreate /dev/sdc**

查看PV[root@server0 ~]# **pvscan**

PV /dev/vdd lvm2 [2.00 GiB]

Total: 1 [2.00 GiB] / in use: 0 [0] / in no VG: 1 [2.00 GiB]

[root@server0 ~]# **pvs**

PV VG Fmt Attr PSize PFree

/dev/vdd lvm2 a-- 2.00g 2.00g

[root@server0 ~]# **pvdisplay**

2. 创建卷组-VG

格式：**vgcreate** 卷组名 准备的物理磁盘（卷组名是自己起的）

例：**vgcreate vg1 /dev/sdc**

查看VG

[root@server0 ~]# **vg** 或

[root@server0 ~]# **vgscan** 或

[root@server0 ~]# **vgdisplay**

3、创建LV

格式：**lvcreate -L** 大小 **-n** 逻辑卷名 从你哪个卷组拿的卷组名

[root@server0 ~]# **lvcreate -L 200M -n lv2 vg1**

指定大小，单位M，G，200M后的M是单位兆

4. 创建文件系统并挂载

[root@server0 ~]# **mkfs.ext4 /dev/vg1/lv2**（一直写到逻辑卷的名字）

5. 创建挂载点

[root@server0 ~]# **mkdir /mnt/lv2**

6. 挂载

[root@server0 ~]# **mount /dev/vg1/lv2 /mnt/lv2**

二、扩大VG **vgextend**

步骤1，创建PV。

[root@server0 ~]# **pvcreeate /dev/sdd1**

步骤2: 扩展VG, 同时包含方法1.

```
[root@server0 ~]# vgextend vg1 /dev/sdd1  
Volume group "vg1" successfully extended
```

三、LV扩容

1. 查看VG空间。用vgs

2. 扩容LV。lvextend -L +200M /dev/vg1/lv2

增加200M空间, 给lv2

更多内容: <https://blog.csdn.net/yy150122/article/details/107009293>

6、如何设置磁盘自动挂载?

在/etc/fstab中写上挂载的命令

7、CPU利用率和CPU负载均衡区别

CPU利用率: 显示的是程序在运行期间实时占用的CPU百分比

CPU负载: 显示的是一段时间内正在使用和等待使用CPU的平均任务数。CPU利用率高, 并不意味着负载就一定大。

举例来说: 如果我有一个程序它需要一直使用CPU的运算功能, 那么此时CPU的使用率可能达到100%, 但是CPU的工作负载则是趋近于“1”, 因为CPU仅负责一个工作嘛! 如果同时执行这样的程序两个呢? CPU的使用率还是100%, 但是工作负载则变成2了。所以也就是说, 当CPU的工作负载越大, 代表CPU必须要在不同的工作之间进行频繁的工作切换。

举例说明:

网上有篇文章举了一个有趣比喻, 拿打电话来说明两者的区别, 我按自己的理解阐述一下。

某公用电话亭, 有一个人在打电话, 四个人在等待, 每人限定使用电话一分钟, 若有人一分钟之内没有打完电话, 只能挂掉电话去排队, 等待下一轮。电话在这里就相当于CPU, 而正在或等待打电话的人就相当于任务数。

在电话亭使用过程中, 肯定会有人打完电话走掉, 有人没有打完电话而选择重新排队, 更会有新增的人在这儿排队, 这个人数的变化就相当于任务数的增减。为了统计平均负载情况, 我们5分钟统计一次人数, 并在第1、5、15分钟的时候对统计情况取平均值, 从而形成第1、5、15分钟的平均负载。

有的人拿起电话就打, 一直打完1分钟, 而有的人可能前三十秒在找电话号码, 或者在犹豫要不要打, 后三十秒才真正在打电话。如果把电话看作CPU, 人数看作任务, 我们就说前一个人(任务)的CPU利用率高, 后一个人(任务)的CPU利用率低。

当然, CPU并不会在前三十秒工作, 后三十秒歇着, 只是说, 有的程序涉及到大量的计算, 所以CPU利用率就高, 而有的程序牵涉到计算的部分很少, CPU利用率自然就低。但无论CPU的利用率是高是低, 跟后面有多少任务在排队没有必然关系。

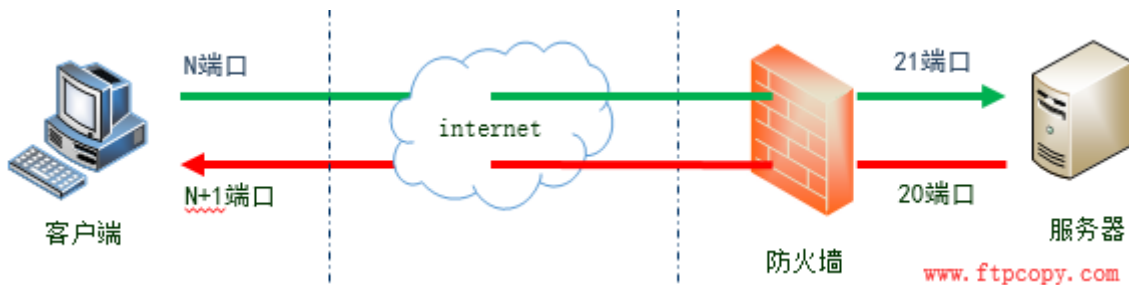
8、ftp的主动和被动模式

1. 在主动模式下, 客户端会开启N和N+1两个端口, N为客户端的命令端口, N+1为客户端的数据端口。

第一步, 客户端使用端口N连接FTP服务器的命令端口21, 建立`控制连接`并告诉服务器我这边开启了数据端口N+1。

第二步, 在`控制连接`建立成功后, 服务器会使用数据端口20, 主动连接客户端的N+1端口以建立`数据连接`。这就是FTP主动模式的连接过程。

我们可以看到, 在这条红色的`数据连接`建立的过程中, 服务器是主动的连接客户端的, 所以称这种模式为主动模式。

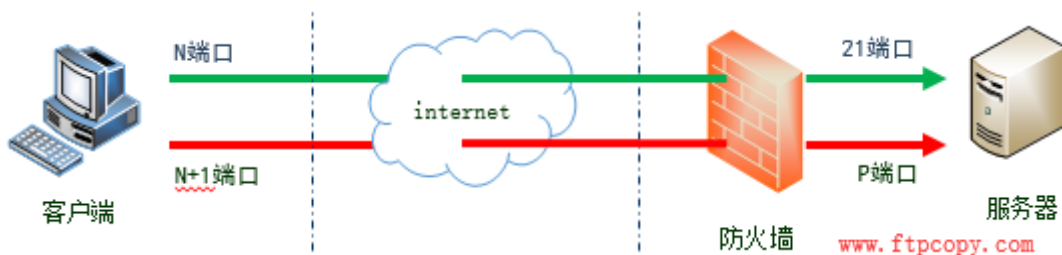


被动模式：

第一步，客户端的命令端口N主动连接服务器命令端口21，并发送PASV命令，告诉服务器用“被动模式”，`控制连接`建立成功后，服务器开启一个数据端口P，通过PORT命令将P端口告诉客户端。

第二步，客户端的数据端口N+1去连接服务器的数据端口P，建立`数据连接`。

我们可以看到，在这条红色的`数据连接`建立的过程中，服务器是被动的等待客户端来连接的，所以称这种模式为被动模式。



9、网络不稳定如何排查？

提示：抓包工具

- 1、网络硬件问题检查。（机率较低）
- 2、检查网卡能否正常工作。（较高、主要表现为人为配置错误）
- 3、检查局域网之间联机是否正常。（非常高）
- 4、检查DNS是否设定正确。（较低）
- 5、服务是否正常打开。（低）
- 6、检查访问权限是否打开。（较高）

1、`lsmod | grep ip`

查看相关的网卡模块是否已加载

2、`ifconfig -a`

能使用该命令查找到对应网卡配置信息，则说明网卡驱动程序正常

3、使用ping命令、依次ping自己、ping局域网主机、ping网关

ping自己异常，问题：服务异常、网卡配置未生效

ping局域网主机异常，问题：配置文件有误、网卡配置未生效、网线损坏

ping网关异常，问题：配置文件有误、网卡配置未生效

4、当前3步还不能正常上网的话。所有route查看默认路由表。

处理方法：删除不必要的路由信息，并保证默认路由是从对应网关地址出去的。

5、临时停止iptables服务、SELinux服务、NetworkManager服务

6、如能上网但访问域名有异常时，那将需要检查/etc/hosts、/etc/resolv.conf两个配置

7、假如以上6步检查完毕之后，还发现不能上网。有如下可能。

7.1、主机MAC地址被路由器禁止上网

7.2、外网服务异常。如宽带账号欠费、光纤被挖断等物理攻击。

10、编译安装和yum安装的区别？ yum与rpm安装的区别？

编译安装和yum安装的区别：

yum安装目录不集中，但基本遵循Linux文件夹的作用去划分文件，比如配置文件通常在/etc下。yum安装模块定制不能自行解决，安装的版本也比较低；而源码编译安装模块可自定义，但是编译时间长。

yum与rpm安装的区别：
yum解决依赖，rpm不解决依赖

11、进程与线程的区别

进程：进程（**Process**）是计算机中的程序关于某数据集合上的一次运行活动，是系统进行资源分配和调度的基本单位，是操作系统结构的基础。

线程：线程（**thread**）是操作系统能够进行运算调度的最小单位。它被包含在进程之中，是进程中的实际运作单位。

区别：

（1）根本区别：进程是操作系统资源分配的基本单位，而线程是处理器任务调度和执行的基本单位

（2）资源开销：每个进程都有独立的代码和数据空间（程序上下文），程序之间的切换会有较大的开销；线程可以看做轻量级的进程，同一类线程共享代码和数据空间，每个线程都有自己独立的运行栈和程序计数器（PC），线程之间切换的开销小。

（3）包含关系：如果一个进程内有多条线程，则执行过程不是一条线的，而是多条线（线程）共同完成的；线程是进程的一部分，所以线程也被称为轻权进程或者轻量级进程。

（4）内存分配：同一进程的线程共享本进程的地址空间和资源，而进程之间的地址空间和资源是相互独立的

（5）影响关系：一个进程崩溃后，在保护模式下不会对其他进程产生影响，但是一个线程崩溃整个进程都死掉。所以多进程要比多线程健壮。

（6）执行过程：每个独立的进程有程序运行的入口、顺序执行序列和程序出口。但是线程不能独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制，两者均可并发执行

12、什么是非对称加密？什么是对称加密？

非对称加密算法需要两个密钥：公开密钥（**publickey**:简称公钥）和私有密钥（**privatekey**:简称私钥）。公钥与私钥是一对，如果用公钥对数据进行加密，只有用对应的私钥才能解密。因为加密和解密使用的是两个不同的密钥，所以这种算法叫作非对称加密算法。

非对称加密算法实现机密信息交换的基本过程是：甲方生成一对密钥并将公钥公开，需要向甲方发送信息的其他角色(乙方)使用该密钥(甲方的公钥)对机密信息进行加密后再发送给甲方；甲方再用自己私钥对加密后的信息进行解密。甲方想要回复乙方时正好相反，使用乙方的公钥对数据进行加密，同理，乙方使用自己的私钥来进行解密。

对称加密：采用单钥密码系统的加密方法，同一个密钥可以同时用作信息的加密和解密

13、如何解决服务器卡的问题？

（1）防火墙问题：检查防火墙，确定该IP是否有攻击，有攻击的情况下可以针对攻击模式加上相对应的防护进行规则。

（2）CPU占用率问题：进服务器看下，CPU占用率是否偏高可以打开任务管理器，查看CPU占用率是否正常，看下CPU占用高的程序是否是正常的系统或客户已知的程序。

（3）硬盘读取速度问题：使用HDTune测试硬盘速度，如果硬盘读写速度慢(平均速度在30M/秒以下)，可以考虑让机房运维拆开机箱，看下是否因机箱内散热风扇震动影响硬盘速度。如果不是机箱内风扇震动造成的，则考虑更换硬盘，或将硬盘内数据转移或备份，将硬盘格式化测试。

（4）服务器硬件内存不足问题

14、网卡的配置文件在什么位置？

/etc/sysconfig/network-scripts/ifcfg-ens33

15、什么是正向解析？什么是反向解析？

正向解析是将域名映射为IP地址；
反向解析是将IP地址映射为域名。

16、如何查找虚拟机上所有以.txt结尾的文件？

```
find / --name "*.txt"
```

17、如何制作swap分区？

1、准备分区：使用fdisk命令创建分区，fdisk会将分区ID默认使用83(linux文件系统)，划分分区后，按t将类型设置为82(swap)。

```
[root@localhost ~]# fdisk /dev/sde
```

2、格式化：使用“mkswap 设备名称”，格式化分区为swap格式

```
[root@localhost ~]# mkswap /dev/sde1
```

3、挂载：使用“swapon 设备名称”，来启动swap

```
[root@localhost ~]# swapon /dev/sde1
```

4、验证：使用“free -m”命令即可

问题解决后，取消swap的增加，使用“swapoff 设备名称”

18、服务器如何安装centos？

可以用docker镜像

VMware

kvm安装

19、一台服务器安装几块磁盘？一块磁盘多大？

6块 1T

20、测试环境和线上环境一样吗？

测试环境：一般是克隆一份生产环境的配置是开发环境到生产环境的过度环境。测试环境的分支一般是develop分支，部署到公司私有的服务器或者局域网服务器上，主要用于测试是否存在bug，一般会让用户和其他人看到，并且测试环境会尽量与生产环境相似。

生产环境：生产环境是指正式提供对外服务的，一般会关掉错误报告，打开错误日志，是最重要的环境。部署分支一般为master分支。

21、虚拟机多少台？都装了什么服务？

200-250台，每台开发应用不一样，但是基础环境差不多

22、CPU利用率若超过500%怎么办？

方法一：

第一步：使用

top命令，然后按shift+p按照CPU排序

找到占用CPU过高的进程的pid

第二步：使用

top -H -p [进程id]

找到进程中消耗资源的线程的id

第三步：使用

echo 'obase=16;[线程id]' | bc或者printf "%x\n" [线程id]

将线程id转换为16进制(字母要小写)

bc是linux的计算器命令

第四步: 执行

```
jstack [进程id] |grep -A 10 [线程id的16进制]"
```

查看线程状态信息

方法二:

第一步: 使用

top命令, 然后按shift+p按照CPU排序

找到占用CPU过高的进程

第二步: 使用

```
ps -mp pid -o THREAD,tid,time | sort -rn
```

获取线程信息, 并找到占用CPU高的线程

第三步: 使用

```
echo 'obase=16;[线程id]' | bc或者printf "%x\n" [线程id]
```

将需要的线程ID转换为16进制格式

第四步: 使用

```
jstack pid |grep tid -A 30 [线程id的16进制]
```

打印线程的堆栈信息

参考: <https://www.cnblogs.com/gucb/p/11229702.html>

23、删除某个文件, 若空间没有释放怎么办?

空间没有释放可能是因为进程在占用, 若占用的进程比较多可以使用echo "" > 文件名, 将文件内容覆盖掉, 若占用的进程比较少, 可以考虑停掉进程再将其删除。

24、如何删除一个10G的日志文件? (爱云校面试)

```
echo "" > 日志文件名 //用此将文件内容覆盖, 方法同上
```

25、提权方法有哪些?

(1) sudo提权: 允许一个用户以超级用户或者其它用户的角色运行一个已授权的命令。

命令格式:

```
sudo -l #查看自己的授权信息
```

```
sudo [特权命令] #执行特权命令
```

```
sudo [-u 用户名] 特权命令 #指定用户执行特权命令
```

sudo提权为临时提权, su提权为永久提权

(2) 内核提权: Linux 内核在具有 setgid 权限的目录中创建文件时处理不当, 导致可以创建具有 setgid 权限的空文件, 随后通过巧妙利用系统调用可以随意更改文件内容, 从而构造出具有 setgid 权限的可执行文件, 实现越权。

(3) SUID提权: 具有这种权限的文件会在其执行时, 使调用者暂时获得该文件拥有者的权限。如果拥有SUID权限, 那么就可以利用系统中的二进制文件和工具来进行root提权。

(4) sudoer配置文件错误提权: 有的时候, 普通用户经常要执行某个命令, 但是经常需要sudo输入密码, 我们就可以通过配置/etc/sudoers文件来实现普通用户某个命令权限的提升, 但是如果一旦是给了用户写入的root权限, 比如vi, 那么这个普通用户一旦被入侵, 就可以通过vi来提权。

(5) 定时任务提权

(6) 密码复用提权

(7) 第三方服务提权

26、centos6与centos7的区别? centos7及centos6的启动流程?

区别:

- (1) 启动引导: 6设置200M就行, 7的boot设置为最少1024M。boot内存大点, 是为了后续升级需要。
[CentOS6] GRUB Legacy (+efibootmgr)
[CentOS7] GRUB2
- (2) 启动技术: 6启动采用Upstart技术sysvinit, 7是用systemd技术, 并行启动
[CentOS6] Upstart
[CentOS7] systemd
- (3) 文件系统
[CentOS6] ext4
[CentOS7] xfs
ext4 分别支持1EB (1,048,576TB, 1EB=1024PB, 1PB=1024TB) 的文件系统, 以及 16TB (4K block size) 的文件。可支持无限数量的子目录。
xfs是一个64位文件系统, 最大支持8EB减1字节的单个文件系统, 实际部署时取决于宿主操作系统的最大块限制。对于一个32位Linux系统, 文件和文件系统的大小会被限制在16TB。
- (4) 内核版本: 内核支持资源调优和分配 (cgroup)
[CentOS6] 2.6.x-x
[CentOS7] 3.10.x-x
- (5) 桌面系统
[CentOS6] GNOME 2.x
[CentOS7] GNOME 3.x (GNOME Shell)
- (6) 嵌套虚拟化技术 (支持了在虚拟机上运行更快)
[CentOS6] 不支持
[CentOS7] 支持
- (7) USB扩展
[CentOS6] USB2.0
[CentOS7] USB3.0
- (8) 防火墙
[CentOS6] iptables
[CentOS7] firewallld
- (9) 默认数据库
[CentOS6] MySQL
[CentOS7] MariaDB
- (10) 文件结构
[CentOS6] /bin, /sbin, /lib, and /lib64在/下
[CentOS7] /bin, /sbin, /lib, and /lib64移到/usr下
- (11) 主机名
[CentOS6] /etc/sysconfig/network
[CentOS7] /etc/hostname
/etc/sysconfig/network在7上此文件已经弃用
修改主机名用命令即可 hostnamectl set-hostname xxxx
- (12) 修改时间
[CentOS6]
\$vim /etc/sysconfig/clock
ZONE="Asia/Tokyo"
UTC=false
\$ sudo ln -s /usr/share/zoneinfo/Asia/Tokyo /etc/localtime

[CentOS7]
\$ timedatectl set-timezone Asia/Tokyo
\$ timedatectl status
- (13) 修改地区
[CentOS6]
\$ vim /etc/sysconfig/i18n LANG="ja_JP.utf8"
\$ /etc/sysconfig/i18n
\$ locale

[CentOS7]

```
$ localectl set-locale LANG=ja_JP.utf8
$ localectl status
```

(14) 服务相关

启动停止:

```
[CentOS6]
$ service xxxxx start
$ service xxxxx stop
$ service sshd restart/status/reload

[CentOS7]
$ systemctl start xxxxx
$ systemctl stop xxxxx
$ systemctl restart/status/reload sshd
```

自启动:

```
[CentOS6]
$ chkconfig xxxxx on/off

[CentOS7]
$ systemctl enable xxxxx
$ systemctl disable xxxxx
```

服务一览:

```
[CentOS6]
$ chkconfig --list

[CentOS7]
$ systemctl list-unit-files
$ systemctl --type service
```

强制停止:

```
[CentOS6]$ kill -9 <PID>
[CentOS7]$ systemctl kill --signal=9 sshd
```

(15) 网络相关

网络信息:

```
[CentOS6]
$ netstat
$ netstat -I
$ netstat -n
```

```
[CentOS7]
$ ip n
$ ip -s l
$ ss
```

IP地址MAC地址:

```
[CentOS6]
$ ifconfig -a

[CentOS7]
$ ip address show
$ ip addr
```

路由:

```
[CentOS6]
$ route -n
$ route -A inet6 -n

[CentOS7]
$ ip route show
$ ip -6 route show
```

(16) 重启关闭

关闭:

```
[CentOS6]
$ shutdown -h now
```

```
[CentOS7]
$ poweroff
$ systemctl poweroff
```

重启:

```
[CentOS6]
$ reboot
$ shutdown -r now
```

```
[CentOS7]
$ reboot
$ systemctl reboot
```

单用户模式:

```
[CentOS6]$ init s
[CentOS7]$ systemctl rescue
```

启动模式:

```
[CentOS6]
[GUICUI]
$ vim /etc/inittab
id:3:initdefault:
[CUIGUI]
$ startx
```

```
[CentOS7]
[GUICUI]
$ systemctl isolate multi-user.target
[CUIGUI]
$systemctl isolate graphical.target
```

总结:

ipconfig 变成了 ip addr

service iptables restart 变成了 systemctl restart firewallld

chkconfig iptables off 变成了 systemctl disable firewallld

CentOS 6.x基本启动过程

- (1) 服务器加电,加载BIOS信息, BIOS进行系统检测
- (2) 加载启动引导程序(grub), 由grub加载系统内核
- (3) 系统内核重新自检,并加载硬件驱动
- (4) 由内核启动系统第一个进程/sbin/init
由/sbin/init进程调用/etc/init/rcs.conf,进行系统初始化配置
由/etc/init/rcs conf调用/etc/inittab,确定系统的默认运行级别
- (5) 确定默认运行级别后,调用/etc/init/rc.conf配置文件>
运行相应的运行级别目录/etc/rc[0-6]. d/中的脚本
在启动登录界面之前,执行/etc/rc .d/rc.local中的程序

CentOS 7.x基本启动过程

- (1) 服务器加电,加载BIOS信息, BIOS进行系统检测
- (2) 加载启动引导程序(grub2)
- (3) 由grub2加载系统内核,内核重新自检
由grub2加载inintamfs虚拟文件系统
- (4) 内核初始化,以加载动态模块的形式加载部分硬件的驱动
内核启动系统的第一个进程,也就是systemd

27、划分子网标准? 主机位, 网络位的运算?

IP地址分类：

A类（1~126）

127：回环地址：我

B类（128~191）

C类（192~223）

D类（224~239）组播

E类（240~255）科研

私有IP分类

A类：10.0.0.0~10.255.255.255

B类：172.16.0.0~172.31.255.255

C类：192.168.0.0~192.168.255.255

传统子网划分：

IP地址结构=网络号+主机号

每一类IP有默认的网络号

类别	网络号	主机号	子网掩码
A类	前8位	32-8=24位	255.0.0.0
B类	前16位	32-16=16位	255.255.0.0
C类	前24位	32-24=8位	255.255.255.0

主机位、网络位计算：

将子网掩码转换为二进制，为1的表示网络位，为0的表示主机位。

28、vlan如何划分？

VLAN（Virtual Local Area Network）又称虚拟局域网，是指在交换局域网的基础上，采用网络管理软件构建的可跨越不同网段、不同网络的端到端的逻辑网络。

（1）基于端口的VLAN划分

这种划分是把一个或多个交换机上的几个端口划分一个逻辑组，这是最简单、最有效的划分方法。该方法只需网络管理员对网络设备的交换端口进行重新分配即可，不用考虑该端口所连接的设备。

（2）基于MAC地址的VLAN划分

MAC地址其实就是指网卡的标识符，每一块网卡的MAC地址都是惟一且固化在网卡上的。MAC地址由12位16进制数表示，前8位为厂商标识，后4位为网卡标识。网络管理员可按MAC地址把一些站点划分为一个逻辑子网。

（3）基于路由的VLAN划分

路由协议工作在网络层，相应的工作设备有路由器和路由交换机（即三层交换机）。该方式允许一个VLAN跨越多个交换机，或一个端口位于多个VLAN中。

29、软硬链接的区别？

由于硬链接是有着相同inode号仅文件名不同的文件，因此硬链接存在以下几点特性：

- （1）文件有相同的inode及 data block；
- （2）只能对已存在的文件进行创建；
- （3）不能交叉文件系统进行硬链接的创建；
- （4）不能对目录进行创建，只可对文件创建；
- （5）删除一个硬链接文件并不影响其他有相同inode号的文件；
- （6）删除源文件链接仍可用。

软链接的创建与使用没有类似硬链接的诸多限制：

- （1）软链接有自己的文件属性及权限等；
- （2）可对不存在的文件或目录创建软链接；
- （3）软链接可交叉文件系统；
- （4）软链接可对文件或目录创建；
- （5）创建软链接时，链接计数 i_nlink 不会增加；

(6) 删除软链接并不影响被指向的文件，但若被指向的原文件被删除，则相关软连接不可用被称为死链接（即 `dangling link`，若被指向路径文件被重新创建，死链接可恢复为正常的软链接）。

30、服务器比较卡怎么办？如何调优？Linux操作系统如何调优？

服务器较卡：

- (1) 服务器的配置低或者老旧（这种情况是设备性能所致，只能更新或升级配置解决）；
- (2) 应用较多，数据过多，造成服务器资源紧张，这种情况可以将一些不用的应用删除以及及时处理数据，也可以适当增加带宽。

Linux操作系统调优：

性能：内存和交换分区的使用策略，调整文件句柄数，优化内核参数等

安全：ssh用私钥方式连接，常用服务端口的修改等

- 1、不用root用户，添加普通用户，通过sudo授权管理
- 2、更改默认的远程连接ssh默认端口及禁止root用户远程连接
- 3、定时自动更新服务器时间
- 4、配置yum更新源，从国内更新源现在安装rpm包
- 5、关闭selinux和iptables。
- 6、调整文件描述符的数量。进程文件的打开都会消耗描述符。
- 7、定时自动清理/var/spool/clientmqueue/目录垃圾文件。防止inodes节点被占满。
- 8、精简开机自启动服务（crond sshd network rsyslog）。
- 9、linux内核参数优化/etc/sysctl.conf,执行sysctl -p 生效。
- 10、更改字符集，支持中文，但建议还是使用英文字符集，防止乱码问题，不要使用中文。
- 11、锁定关键系统文件。
- 12、清空/etc/issue，去除系统及内核版本登陆前的显示。

31、top命令右上角参数的意思及其他参数意思？（飞哥+爱云校面试）

使用top可以查看当前服务器的负载

第一行：

当前时间、系统启动时间、当前系统登录用户数目、平均负载（1分钟,5分钟,15分钟）。

平均负载（load average），一般对于单个cpu来说，负载在0~1.00之间是正常的，超过1.00须引起注意。在多核cpu中，系统平均负载不应该高于cpu核心的总数。

第二行：

进程总数、运行进程数、休眠进程数、终止进程数、僵死进程数。

第三行：

%us用户空间占用cpu百分比；

%sy内核空间占用cpu百分比；

%ni用户进程空间内改变过优先级的进程占用cpu百分比；

%id空闲cpu百分比，反映一个系统cpu的闲忙程度。越大越空闲；

%wa等待输入输出（I/O）的cpu百分比；

%hi指的是cpu处理硬件中断的时间；

%si值的是cpu处理软件中断的时间；

%st用于有虚拟cpu的情况，用来指示被虚拟机偷掉的cpu时间。

第四行：

total总的物理内存；

used使用物理内存大小；

free空闲物理内存；

buffers用于内核缓存的内存大小

第五行：

total总的交换空间大小；

used已经使用交换空间大小；

free空间交换空间大小；

cached缓冲的交换空间大小

buffers于**cached**区别：**buffers**指的是块设备的读写缓冲区，**cached**指的是文件系统本身的页面缓存。他们都是Linux系统底层的机制，为了加速对磁盘的访问。

第六行：

PID 进程号

USER 运行用户

PR

优先级，**PR(Priority)**所代表的值有什么含义？它其实就是进程调度器分配给进程的时间片长度，单位是时钟个数，那么一个时钟需要多长时间呢？这

跟CPU的主频以及操作系统平台有关，比如linux上一般为10ms，那么PR值为15则表示这个进程的时间片为150ms。

NI 任务nice值

VIRT 进程使用的虚拟内存总量，单位kb。**VIRT=SWAP+RES**

RES 物理内存用量

SHR 共享内存用量

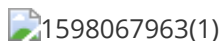
S 该进程的状态。其中**S**代表休眠状态；**D**代表不可中断的休眠状态；**R**代表运行状态；**Z**代表僵死状态；**T**代表停止或跟踪状态

%CPU 该进程自最近一次刷新以来所占用的CPU时间和总时间的百分比

%MEM 该进程占用的物理内存占总内存的百分比

TIME+ 累计cpu占用时间

COMMAND 该进程的命令名称，如果一行显示不下，则会进行截取。内存中的进程会有一个完整的命令行



32、服务器开不了机怎么办？

(1) 挂载：可能是挂载有问题导致无法开机，解决：进入救援模式，将分区挂载/etc/fstab文件里的内容改改；

(2) 内核：可能是内核存在问题导致无法开机，解决：修改启动参数、驱动；

(3) 文件系统损坏：异常断电导致文件系统损坏，只能通过修复。

33、Linux操作系统有哪些文本处理方式？

文件浏览

(1) **cat**查看文件内容

(2) **more**以翻页形式查看文件内容（只能向下翻页）

(3) **less**以翻页形式查看文件内容(可以上下翻页)

(4) **head**查看文件的开始10行（或指定行数）

(5) **tail**查看文件的结束10行（或指定行数）

(6) 基于关键字进行搜索：**grep**

(7) **sed**: **sed [-hnv] [-e<script>] [-f<script文件>] [文本文件]**

动作说明：

a：新增，**a**的后面可以接字符串，而这些字符串会在新的一行出现(目前的下一行)～

c：取代，**c**的后面可以接字符串，这些字符串可以取代 **n1,n2**之间的行！

d：删除，因为是删除啊，所以 **d** 后面通常不接任何咚咚；

i：插入，**i**的后面可以接字符串，而这些字符串会在新的一行出现(目前的上一行)；

p：打印，亦即将某个选择的数据印出。通常 **p** 会与参数 **sed -n** 一起运行～

s：取代，可以直接进行取代的工作，通常这个**s**的动作可以搭配正规表示法！例如

1,20s/old/new/g

(8) **awk**:

AWK是一种处理文本文件的语言，是一个强大的文本分析工具。可以是来自标准输入，也可以通过管道符获取标准输入，**awk**可以在命令行上直接编辑命令进行操作，也可以编写成**awk**程序来进行更为复杂的运用。

(9) 文本统计：**wc**，用以统计文本信息。

-l：只统计行数

- w: 只统计单词
- c: 只统计字节数
- m: 只统计字符数

(10) 文本排序: **sort**, 用以对文本内容进行排序。

- r: 进行倒序排序
- n: 基于数字进行排序
- f: 忽略大小写
- u: 删除重复行
- t c: 使用c作为分隔符分割为列进行排序
- k x: 当进行基于指定字符分割为列的排序时, 指定基于哪个列排序

34、如何截取磁盘根分区使用情况？如何查看磁盘分区的信息？磁盘分区如何扩容？

```
df -Th |awk 'NR==1,NR==2{print}'
```

查看磁盘分区信息: `lsblk`

- 1、添加磁盘
- 2、创建分区
- 3、创建文件系统
- 4、挂载mount
- 5、查看挂载信息

35、如何截取cpu使用率？

```
top -n 1 | grep %Cpu
```

36、内存缓冲与缓存有什么区别？

缓冲区是一块特定的内存区域。开辟缓冲区的目的是通过缓解应用程序上下层之间的性能差异, 提高系统的性能。在日常生活中, 缓冲的一个典型应用是漏斗

缓冲最常用的场景就是提高I/O的速度。为此, JDK内不少I/O组件都提供了缓冲功能。

缓存的主要作用是暂存数据处理结果, 并提供下次访问使用每次打开一个网页, IE会自动创建一份该网页文字和图像的缓存文件(一个临时副本)。当再次打开该页时, IE会检查网站服务器上该页的变化。如果页面变化了, IE从网络上重新下载新的网页。如果该页面没有变化, IE就从内存或硬盘上使用缓存中的临时复本来显示它。

缓冲(buffer)和缓存(cache)的区别:

缓存(cache)是在读取硬盘中的数据时, 把最常用的数据保存在内存的缓存区中, 再次读取该数据时, 就不去硬盘中读取了, 而在缓存中读取。

缓冲(buffer)是在向硬盘写入数据时, 先把数据放入缓冲区, 然后再一起向硬盘写入, 把分散的写操作集中进行, 减少磁盘碎片和硬盘的反复寻道, 从而提高系统性能。

简单来说, 缓存(cache)是用来加速数据从硬盘中"读取"的, 而缓冲(buffer)是用来加速数据"写入"硬盘的。

37、如何实时查看日志？

```
tail -f 日志路径及名称
```

38、你通过什么方式安装操作系统？

- 1、配置服务器硬件raid
 - 2、进入BIOS修改开启启动项，选择从网络启动
 - 3、将内网网口PXE选项开启
 - 4、通过DHCD服务向企业内部一体化（一体化装机系统，自己搭建的）发送安装请求。
- 总结：公司安装Linux操作系统可以使用网络安装、U盘安装、光驱安装

39、vim几种模式？

- （1）命令模式：命令模式是vim打开文件后默认进入的模式，这个模式不能插入字符，但可以设定vim的工作方式
- （2）插入模式：可对文件进行编辑
- （3）可视化模式：可视模式是选中一块区域进行操作，包括删除，替换，复制，粘贴，改变大小写等

40、cpu配置？

28核56线程 或 32核64线程

41、文件系统损坏如何修复？

- （1）出错的时候如果告诉你是哪一块硬盘的分区有问题，比如是/dev/hda3
使用命令：`#fsck -y /dev/hda3`
结束后，reboot。这样就OK了！
- （2）如果你不知道时哪个地方出了问题。（常用此种方法）
可以直接`#fsck`
在随后的多个确认对话框中输入：`y`
结束后，reboot。就ok了。
然后再执行以下`mount -a`看一下，文件系统是否已经修复了。
修复好以后重启一下系统。
`mount -a` 可以查看哪个文件系统坏了
`fsck -y /dev/sda6` 修复文件系统。

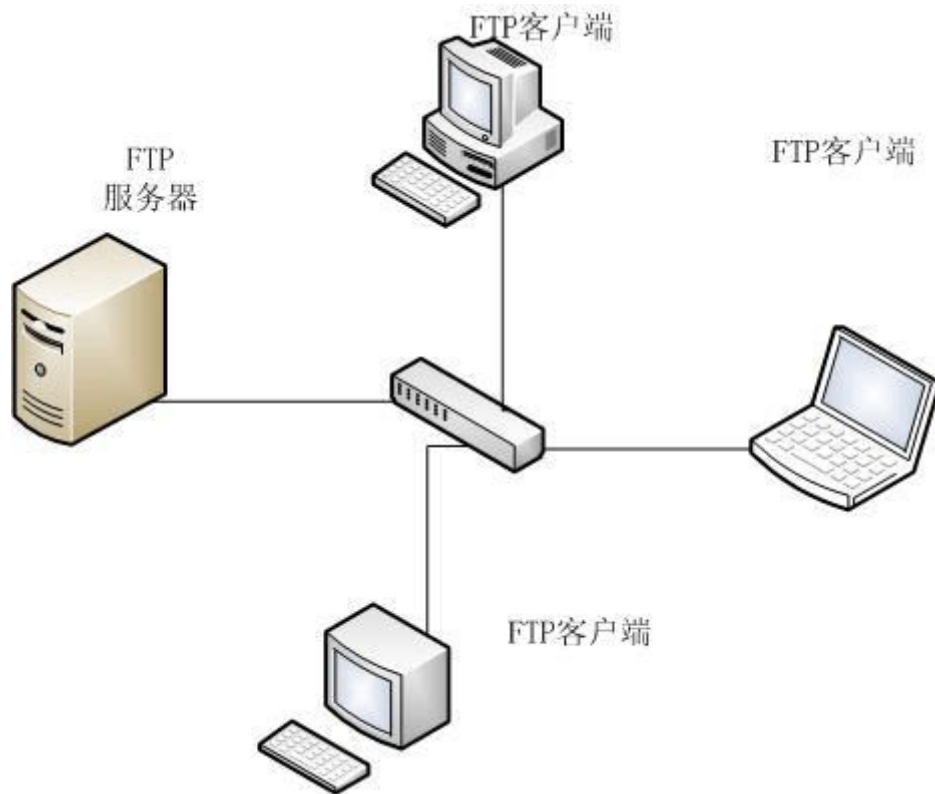
42、swap分区的作用

swap分区通常被称为交换分区，这是一块特殊的硬盘空间，即当实际内存不够用的时候，操作系统会从内存中取出一部分暂时不用的数据，放在交换分区中，从而为当前运行的程序腾出足够的内存空间。

43、共享文件怎么做（ftp、nfs怎么搭建）？实时共享文件怎么做？

ftp、nfs可以提供文件共享服务
ftp:

- （1）在server端安装vsftpd，在client端安装lftp
- （2）在server端启动上传功能(启动上传文件的能力: `anon_upload_enable=YES`，启用创建目录的能力: `anon_mkdir_write_enable=YES`)，创建上传目录
- （3）在client端使用lftp命令登录服务器，使用get下载文件



nfs:

- (1) 在server端安装nfs-utils, 并在/etc/exports写入/webdata 192.168.122.0/24(rw) , 其中/webdata 为共享文件的目录192.168.122.0/24(rw)为允许查看的网段, 启动nfs-server
- (2) 在client端安装nfs-utils、httpd, 并使用“showmount -e nfs的ip”查看nfs服务器可用目录, 用“mount -t nfs nfs的ip:发布目录”进行手动挂载
- (3) 访问server的ip即可查看共享文件

44、一个盘可以创建多少分区？磁盘的分区方式及区别？磁盘的格式？

硬盘分区有三种，主磁盘分区、扩展磁盘分区、逻辑分区。

一个硬盘主分区至少有1个，最多4个，扩展分区可以没有，最多一个。且主分区+扩展分区总共不能超过4个。逻辑分区可以有若干个。

磁盘分区方式及区别：

MBR:

MBR支持最大的磁盘容量是 <2TB。设计时分配4个分区

如果希望超过4个分区，需放弃主分区，改为扩展分区和逻辑分区。

GPT:

GPT 支持大于2T的硬盘，支持128个分区

格式:

windows下磁盘格式主要有FAT16、FAT32、NTFS 等，最新格式为exFAT，不同的磁盘格式有不同的特性。FAT格式基本上已经不再使用。linux下的格式为ext系列，ext4，ext3等。

45、为什么硬盘实际使用的大小小于硬盘本身大小？实际使用率是多少？

一个文件所占用的空间包括文件本身以及相关的一些碎片及读写操作等等，所以实际上文件的大小一般情况下来说都是小于所占用的空间，这个跟你文件系统没有太大的关系，几乎所有的系统都是这样情况
实际使用率70%以下

46、虚拟内存和物理内存的区别？

虚拟内存集VSZ：进程占用的虚拟内存空间
物理内存集RSS：进程占用实际物理内存空间

47、磁盘报错 “no space left on device” 但是df -h 查看磁盘空间没满，请问有哪些原因导致？怎么处理？（爱云校笔试）

磁盘报错“No space left on device”，当使用df -h 查询文件属性信息时，实际上是查的磁盘的block数量，从下面看磁盘没有满，说明有可能是磁盘的inode满了。 解决办法： 使用df -i 查看是否是inode满了， 进入inode查看100%的目录，删除无用的文件即可。

48、/var/log/messages 日志出现kernel: nf_conntrack:table full,dropping packet 请问是什么原因导致的，如何解决？（爱云校笔试）

原因：服务器访问量大，内核netfilter模块conntrack相关参数配置不合理，导致新连接被drop掉。
解决：

- 1、关闭防火墙
- 2、调整内核参数：调大nf_conntrack_buckets和nf_conntrack_max，调小超时时间。

49、查看服务的运行时间？（爱云校面试）

用ps -ef

```
[root@5201351 ~]# ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root          1        0  0 06:50 ?          00:00:02 /sbin/init
root          2        0  0 06:50 ?          00:00:00 [kthreadd]
root          3        2  0 06:50 ?          00:00:00 [migration/0]
root          4        2  0 06:50 ?          00:00:00 [ksoftirqd/0]
```

UID	PID	PPID	C	STIME	TTY	CMD
用户ID	进程的ID	父进程ID	进程占用CPU的百分比	进程启动的时间	该进程在那个终端上运行。若与终端无关，则显示? 若为pts/0等，则表示由网络连接主机进程。	命令的名称和参数

```
[root@5201351 ~]# ps aux
USER          PID  %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1    0.0  0.1 19224  1540 ?        Ss   06:50   0:02 /sbin/init
root          2    0.0  0.0     0     0 ?        S    06:50   0:00 [kthreadd]
root          3    0.0  0.0     0     0 ?        S    06:50   0:00 [migration/0]
root          4    0.0  0.0     0     0 ?        S    06:50   0:00 [ksoftirqd/0]
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
用户名	同上	进程占用的CPU百分比	占用内存的百分比	该进程使用的虚拟内存量 (KB)	该进程占用的固定内存量 (KB) (驻留中页的数量)	同上	进程的状态	同上	该进程实际使用CPU运行的时间	同上

50、云服务器交换分区怎么做？（爱云校面试）

- (1) 首先创建一个文件用于swap分区，下面命令可以创建一个4G的文件
`dd if=/dev/zero of=/data/swap bs=512 count=8388616`
`bs*count`即为容量，如果需要2G的swap可以使用
`dd if=/dev/zero of=/data/swap bs=1k count=2048000`
- (2) 将文件设置为swap分区
`mkswap /data/swap`
- (3) 启用swap分区
`swapon /data/swap`
- (4) 添加指令到fstab文件中这样系统引导时会自动启动
`echo "/data/swap swap swap defaults 0 0" >> /etc/fstab`
- (5) 查看是否生效可用`free -m`
如果需要关闭swap分区可以依次执行如下命令
`swapoff /data/swap`
`swapoff -a >/dev/null`
`rm -rf /data/swap`

51、linux系统中一般会有swap内存，你觉得使用swap内存有什么好处，swap内存会被使用吗？生产中要不要使用swap内存？

swap好处是防止内存溢出。OOM（out of memory），当内存不够时会使用。
生产环境中需要使用swap内存。

52、crontab计划任务中最小单位是什么？简述crontab的执行过程（爱云校笔试）

crontab 定时任务的最小时间单位只能是分，但可以在执行命令前通过sleep来实现秒级
例如：

```
* * * * * /root/shell/a.sh
* * * * * sleep 30;/root/shell/a.sh
```

这样就可以实现30秒触发一次a.sh了 如果要1秒一次 那就写60行 sleep从1到59

53、计划任务五列代表的意思？写过什么计划任务？写过的计划任务没有生效？

分时日月周；
定时备份数据库的数据、定时重启tomcat；
执行的脚本权限不够。

54、如何取8：00到8：30时间段内的日志？（日志时间格式：[2020-07-20T08：56：47.860z]）（爱云校笔试）

```
sed -n '/2020-07-20T08: 00/,/2020-07-20T08: 30/'p /日志文件 > /导出到的文件地址
```

55、tcp段头最小长度是多少字节

IP+TCP头都是20字节，加起来40字节，不过需要填充一些数据，达到64字节，否则，网络设备会认为这个数据包是碎片而丢弃。

56、DNS解析过程

例如客户端解析 `www.126.com`

1. 客户端查询自己的缓存（包含hosts中的记录），如果没有将查询发送/etc/resolv.conf中的DNS服务器
2. 如果本地DNS服务器对于请求的信息具有权威性，会将（权威答案）发送到客户端。
3. 否则（不具有权威性），如果DNS服务器在其缓存中有请求信息，则将（非权威答案）发送到客户端
4. 如果缓存中没有该查询信息，DNS服务器将搜索权威DNS服务器以查找信息：
 - a. 从根区域开始，按照DNS层次结构向下搜索，直至对于信息具有权威的名称服务器，为客户端获答案
DNS服务器将信息传递给客户端，并在自己的缓存中保留一个副本，以备以后查找。
 - b. 转发到其它DNS服务器

57、什么是DNS?

域名系统（英文：Domain Name System，缩写：DNS）是互联网的一项服务。它作为将域名和IP地址相互映射的一个分布式数据库，能够使人更方便地访问互联网。DNS使用TCP和UDP端口53。

域名系统是Internet上解决网上机器命名的一种系统。就像拜访朋友要先知道别人家怎么走一样，Internet上当一台主机要访问另外一台主机时，必须首先获知其地址，TCP/IP中的IP地址是由四段以“.”分开的数字组成，记起来总是不如名字那么方便，所以，就采用了域名系统来管理名字和IP的对应关系。

dns有两种情况，一种是区域传输，一种是域名解析

- 1、区域传输时，一个区中主dns服务器从自己本身机器的数据文件中读取dns数据信息，而辅助dns服务器则从主dns服务器中读取该区的dns数据信息，传输协议是tcp
- 2、域名解析时，首选的通讯协议是udp，使用udp传输，不经过三次握手，这样dns服务器负载更低，响应更快。但是当域名解析反馈长度超过512字符，将不能使用udp协议进行解析，此时必须使用tcp

58、服务器装虚拟机吗？用什么装（KVM）？

服务器需要装虚拟机

kvm虚拟化平台软件一般装在操作系统为Centos上面

需要下载以下几个软件：

```
qemu-kvm libvirt virt-manager
```

kvm负责cpu虚拟化+内存虚拟化，实现了cpu和内存的虚拟化，但kvm不能模拟其他设备；

qemu是模拟IO设备（网卡，磁盘），kvm加上qemu之后就能实现真正意义上服务器虚拟化。

因为用到了上面两个东西，所以一般都称之为qemu-kvm。

libvirt则是调用kvm虚拟化技术的接口用于管理的，用libvirt管理方便，直接用qemu-kvm的接口太繁琐。

virt-manager是用于管理KVM虚拟环境的主要工具

1、图型安装

```
# virt-manager之后就是点点点
```

2、可以通过模板镜像+配置文件 方式安装虚拟机

（1）拷贝模板镜像和配置文件

```
# cp /var/lib/libvirt/images/vm2.img /var/lib/libvirt/images/vm3.img
```

```
# cp /etc/libvirt/qemu/vm2.xml /etc/libvirt/qemu/vm3.xml
```

（2）修改配置文件

不可重复：名字、uuid、mac地址

可选：内存、磁盘、cpu核数

59、如何更改ssh默认端口号？

修改/etc/ssh/sshd_config

Port 22 改为 Port 10022

连接时需要指定端口：ssh root@192.168.11.201 -p 10022

60、jvm分为哪三个区？

新生代、老年代、持久代

所有通过new创建的对象内存都在堆中分配，其大小可以通过-Xmx和-Xms来控制。

堆被划分为新生代和老年代，新生代又被进一步划分为Eden和Survivor区，Survivor由FromSpace和ToSpace组成

新生代：新建的对象都是用新生代分配内存，Eden空间不足的时候，会把存活的对象转移到Survivor中，新生代大小可以由-Xmn来控制，也可以用-XX:SurvivorRatio来控制Eden和Survivor的比例。

老年代：用于存放新生代中经过多次垃圾回收仍然存活的对象。

持久代：用于存放静态文件，如今Java类、方法等。

有的虚拟机并没有持久代，JAVA8 开始持久代也已经被彻底删除了，取代它的是另一个内存区域也被称为元空间。

61、抓包工具怎么用？

tcpdump采用命令行方式对接口的数据包进行筛选抓取，其丰富特性表现在灵活的表达式上。

```
tcpdump [ -DenNqvX ] [ -c count ] [ -F file ] [ -i interface ] [ -r file ]  
[ -s snaplen ] [ -w file ] [ expression ]
```

抓包选项：

-c：指定要抓取的包数量。注意，是最终要获取这么多个包。例如，指定"-c 10"将获取10个包，但可能已经处理了100个包，只不过只有10个包是满足条件的包。

-i interface：指定tcpdump需要监听的接口。若未指定该选项，将从系统接口列表中搜寻编号最小的已配置好的接口(不包括loopback接口，要抓取loopback接口使用tcpdump -i lo)，一旦找到第一个符合条件的接口，搜寻马上结束。可以使用'any'关键字表示所有网络接口。

-n：对地址以数字方式显式，否则显式为主机名，也就是说-n选项不做主机名解析。

-N：不打印出host的域名部分。例如tcpdump将会打印'nic'而不是'nic.ddn.mil'。

-P：指定要抓取的包是流入还是流出的包。可以给定的值为"in"、"out"和"inout"，默认为"inout"。

-s len：设置tcpdump的数据包抓取长度为len，如果不设置默认将会是65535字节。对于要抓取的数据包较大时，长度设置不够可能会产生包截断，若出现包截断，输出行中会出现"[|proto]"的标志(proto实际会显示为协议名)。但是抓取len越长，包的处理时间越长，并且会减少tcpdump可缓存的数据包的数量，从而会导致数据包的丢失，所以在能抓取我们想要的包的前提下，抓取长度越小越好。

输出选项：

-e：输出的每行中都将包括数据链路层头部信息，例如源MAC和目标MAC。

-q：快速打印输出。即打印很少的协议相关信息，从而输出行都比较简短。

-X：输出包的头部数据，会以16进制和ASCII两种方式同时输出。

-XX：输出包的头部数据，会以16进制和ASCII两种方式同时输出，更详细。

-v：当分析和打印的时候，产生详细的输出。

-vv：产生比-v更详细的输出。

-vvv：产生比-vv更详细的输出。

其他功能性选项：

-D：列出可用于抓包的接口。将会列出接口的数值编号和接口名，它们都可以用于"-i"后。

-F：从文件中读取抓包的表达式。若使用该选项，则命令行中给定的其他表达式都将失效。

-w：将抓包数据输出到文件中而不是标准输出。可以同时配合"-G time"选项使得输出文件每time秒就自动切换到另一个文件。可通过"-r"选项载入这些文件以进行分析和打印。

-r：从给定的数据包文件中读取数据。使用 "-" 表示从标准输入中读取。

62、什么叫 CDN 服务，CDN服务有什么作用？目前哪些厂商提供CDN服务？Vpn是用来干什么的？

CDN是构建在网络之上的内容分发网络，依靠部署在各地的边缘服务器，通过中心平台的负载均衡、内容分发、调度等功能模块，使用户就近获取所需内容，降低网络拥塞，提高用户访问响应速度和命中率。CDN的关键技术主要有内容存储和分发技术。

作用：可以将其理解为离您很近的服务器，可以从上面获取完整的原始数据，它将定期与原始内容服务器同步，以确保用户可以获取的最新内容。

目前市场上的CDN厂商主要分为3大类：

- 1、创新型CDN厂商：以云帆CDN为代表，通过与流量矿石合作，众包CDN共享创新技术，重新定义CDN服务并提供专注的技术服务。
- 2、传统类CDN厂商：这类厂商具有较长时间的发展沉淀，主要包括蓝xun、某宿科技等，通过企业自身建设的服务器资源提供CDN服务。
- 3、云服务厂商：以阿里腾讯、百度为代表的综合类服务提供商，CDN服务是其众多云计算服务中的一项。

VPN就是在INTER网上虚拟出来一个局域网，还可以使用用户验证，这样公司想用的内部资源，但是外出或者非本地用户就可以通过VPN来访问公司的局域网，更加安全

63、堡垒机(Jumpserver)是用来干什么的？

即在一个特定的网络环境下，为了保障网络和数据不受来自外部和内部用户的入侵和破坏，而运用各种技术手段监控和记录运维人员对网络内的服务器、网络设备、安全设备、数据库等设备的操作行为，以便集中报警、及时处理及审计定责。

64、你新安装完操作系统后都做什么？做安全加固吗？

1. 防火墙只开放对外的服务端口
2. 禁止root远程登录
3. 修改/etc/passwd访问权限及属性
4. 修改SSHD的服务端口
5. 禁止不用的服务和应用
6. 经常检查系统日志
7. 执行初始化脚本，实现杀毒软件的安装及ntp时间对时、开机挂载磁盘等
8. 修改yum源

65、如何判断Linux服务器是否被入侵，或者在公司中被黑了怎么办？

1. 首先检查当前有谁在登录：w
2. 检查谁曾经登录过：last
3. 回顾历史命令：~/.bash_history/
4. 检查哪些进程在消耗CPU，并消耗非常厉害：ps aux | less
5. 检查所有系统进程
6. 以上都检查完后，你要只允许从你的IP地址登录ssh

66、DHCP是什么？

动态主机配置协议，主要目的是为了方便我们主机IP地址的配置，如果网络中存在大量主机时，可部署DHVP服务器分配地址给主机

67、什么是丢包？

网络丢包是在网络较差的情况下，由于数据包的传输不可能百分之百的能够完成，从而造成在数据的传输中出现空洞，造成丢包

68、服务默认的端口（爱云校笔试）

服务	端口	版本号
FTP	21	
SSH	22	
Telnet	23	
Nginx	80	1.16.1
Apache	80	2.4.6
Zookeeper	2181	3.4.10
Mysql	3306	5.7
Mariadb	3306	5.5.65
Kibana	5601	6.5或7.8
logstash	5601	2.1
Redis	6379	4.0.9
API server/k8s	6443	1.17.4
mycat	8066 9066	1.6
Tomcat	8005（本身端口） 8080（客户端） 8009（http）	7.0
jenkins	8080	2.122
php	9000	2.4
Kafka	9092	2.11
Head插件	9100	
ElasticSearch	9200 9300	6.5或7.8
zabbix	10051（server） 10050（client）	3.2
gluster	24007	
ansible	ssh	2.3
git	ssh	1.8
maven		3.53
kibana		7.8

69、如果运行中的服务器的某一分区出现readonly，导致进程无法写这个分区（比如nginx进程无法写日志文件到此分区，手动测试touch文件到此分区也显示：cannot touch 'xxxxx': read-only file system），该怎么办？

解决：磁盘read-only的原因一般有两种，一种是没有正常关机导致，还有一种是硬盘故障导致。如果是/分区，这种情况只能下线报修磁盘了，如果是其他分区，则可以尝试三步解决此问题：

- 1、先取消挂载分区 `umount`
- 2、再`fsck -y /dev/sdb1`
- 3、最后挂载此分区，检查是否可以正常读写。如果仍旧不可以正常读写，请保修磁盘。

70、说说更换磁盘的详细过程，及报修流程

- 1、确定磁盘故障后，确定磁盘的型号例如ST3300657 300G（容量） 10k RPM（转速）SATA（SAS）（接口类型）
- 2、找到硬盘后除了新盘，旧盘一定要做磁盘检测（找一台旧服务器，一般在公司）并且清除数据，让硬盘变成ready状态。才可以发送（快递）到idc机房，机房值守收到硬盘后，工程师要给现场派发工单。
更换服务器名称为xxx，哪个机房，机柜，id=?
例如：请更换磁盘：亦庄4M3A机房，k3列 服务器D161234 ID=2 300G 型号：st3300657ss 10k rpm sata，现场更换完后，旧的磁盘单独存放，定期发送给相关部门进行销毁

71、在保服务器怎么更换硬件？

首先确定机器在保，然后给厂商打售后电话，提供设备SN/PN号。（设备的唯一标识）售后客服和你确定后，联系厂家工程师上门维护，更换备件，此操作由厂家操作，机房值守陪同。工程师验证。坏的设备由厂家人拿走。提前给厂家人员办理进入机房的授权手续。如果用的云服务器，直接提工单，联系云供应商。

72、为什么服务器有磁盘空间但是却创建不了文件？

因为inode节点被占满，想要创建新的文件，需要删除垃圾文件及没用的日志文件，终止一些服务的进程，释放空间。

创建不了文件，touch失败的原因：a磁盘满了 b磁盘坏了 faild c inode节点被占满 d磁盘坏道过多。

73、说说你接触的服务器都是什么样的？

我接触过HP的服务器，一般是2-4U，最多可以有24块盘，基本都是2.5寸的。磁盘序列号从B1-B24，型号一般为HP 380G5-HP380G9 有一些比较老
dell服务器 一般是2U的，大部分只有6块盘 磁盘容量为300G 500G 750G 1T 2T
全都用过 型号为dell r710 dell r720
还有比较多的国产，例如lenovo, inspur, h3c, 但是质量都不太好，有点瑕疵
瑕疵的解释：服务器不稳定，经常异常重启（1查内存2查系统）售后保障不好，有时候新买的服务器会出现开不开机的现象，一般故障率为200台坏2台。还有bios版本和硬件不兼容等等硬件兼容问题，还有网卡识别问题，有些网卡不识别百兆。

74、给你一台新买的服务器（物理机），你要怎么做？

申请服务器上线-->申请ip地址（资源分配）[机器的交换机端口，物理机柜的位置，ip地址]-->确定网络结构（内？外？）-->现场机房上架-->连线-->网络组划分网络-->现场手做raid-->安装操作系统-->确定服务器账户添加-->登陆-->执行业务初始化脚本-->更新cmdb信息-->搭建应用服务-->安装环境-->测试

75、cookie和session的区别

http是一种无状态的连接，客户端每次读取**web**页面时，服务都会认为这是一次新的会话，但有时候我们又需要持久保持某些信息，比如登陆时的用户名，密码，用户上一次连接时的信息等。这些信息就由**cookie**和**session**保存

cookie实际上是一小段文本信息，客户端请求服务器，如果服务器需要记录该用户状态，就使用**response**向客户端浏览器颁发一个**cookie**，客户端浏览器会把**cookie**保存起来，当浏览器再次请求访问该网站时，浏览器会把请求的网站连同该**cookie**一同提交给服务器，服务器检查**cookie**，以此来辨认用户状态。

简单来说，**cookie**的工作原理可总结如下：

- 1、**client**连接**server**
- 2、**client**生成**cookie**（有效期），再次访问时携带**cookie**
- 3、**server**根据**cookie**的信息识别用户身份

session是服务器端使用的一种记录客户端状态的机制，使用上比**cookie**简单一些，同一个客户端每次和服务端交互时，不需要每次都传回所有的**cookie**值，而是只要传回一个**id**，这个**id**是客户端第一次访问服务器的时候生成的，而且每个客户端是唯一的，这样每个客户端就有了一个唯一的**id**，客户端只要传回这个**id**就行了，这个通常是**name**为**sessionid**的一个**cookie**。**session**依据这个**id**来识别是否为同一个用户（只认**id**不认人）。

76、域名申请流程

找机器-->搭环境-->基础配置-->和开发沟通-->要包-->测试-->域名申请-->域名备案-->域名绑定（耗时一周最少） 万网-->阿里云进行域名申请及绑定

77、telnet和ssh有什么区别

telnet：不安全，没有对传输数据进行加密，容易被监听，还有遭受中间人攻击，**telnet**不能压缩传输数据，所以传输慢。

ssh：对数据进行了加密，安全性高，**ssh**传输数据经过压缩，所以传输速度比较快。

78、冷备份和热备份的不同点以及各自的优点

热备份针对归档模式的数据库，在数据库仍旧处于工作状态时进行备份。而冷备份指数据库关闭后进行备份，适用于所有数据库。

不同点：

热备份：备份时数据库仍旧处于运行状态

冷备份：备份时数据处于关闭状态

优点：

热备份：在备份时，数据库仍可以使用并且可以将数据库恢复到任意一个时间点

冷备份：它的备份和恢复操作相当简单，并且冷备份可以工作在非归档模式下，数据库性能会比归档模式稍好

79、如果接到客服反馈：A项目www.lanxinA.cn无法打开，作为一名运维人员需要怎么操作？

- 1、先把服务切走，保证公司网站能正常对外提供服务
- 2、查看网络是否正常
- 3、本地网络速率正常时，再检查网站是否正常，域名是否能被正常解析到ip

使用故障诊断命令定位故障点。

命令分析：**ping**：检测IP或域名的连通性

dig/nslookup：查看DNS解析情况

tracert：显示从访问者到网站的路由连接状态，如果有节点无法连接，只需针对该故障点进行修复便可快速恢复网络。

- 4、检查服务器本身是否故障

- 5、协调开发检查设计是否异常

80、什么是磁盘碎片

其实磁盘碎片应该称为文件碎片，磁盘碎片指的是硬盘读写过程中产生的不连续文件。硬盘上非连续写入的档案会产生磁盘碎片，磁盘碎片会加长硬盘的寻道时间，影响系统效能。比如虚拟内存使用了硬盘，硬盘上便会产生磁盘碎片。

81、什么是内存碎片

内存碎片分为：内部碎片和外部碎片。

内部碎片就是已经被分配出去（能明确指出属于哪个进程）却不能被利用的内存空间；

外部碎片指的是还没有被分配出去（不属于任何进程），但由于太小了无法分配给申请内存空间的新进程的内存空闲区域。

82、怎么开启路由转发

修改/etc/sysctl.conf文件，让包转发功能在系统启动时自动生效：

```
net.ipv4.ip_forward = 1
```

83、怎么查看文件句柄数

```
ulimit -a
```

84、如何实现Nginx代理的节点访问 日志记录客户的IP而不是代理的IP?

使用proxy反向代理模块中的proxy_set_header参数

```
proxy_set_header X-Forwarded-For $remote_addr;
```

85、如何修改内核参数?

在/etc/sysctl.conf中 修改 然后sysctl -p生效

86、一个存储高清视频的磁盘,大小为4TB,如何优化磁盘IO?

通过适当的调整nr_requests 参数可以大幅提升磁盘的吞吐量

缺点就是你要牺牲一定的内存修改默认请求队列

87、文件描述符

file descriptors ,FD, 文件描述符，进程使用文件描述符来管理打开的文件

FD是访问文件的标识，即链接文件

- 0是键盘只读，标准输入
- 1,2是终端可以理解是屏幕，1为标准输出，2为标准错误
- 3+是文件，可读可写

#####

二、数据库

1、关系型数据库都有哪些？非关系型数据库都有哪些？

关系型数据库：

mysql、mariadb	oracle公司
SQL server	微软
oracle	oracle
db2	IBM
sqlite3	centos默认安装，没有进程

非关系型数据库：

redis
mongodb
memcached

2、关系型数据库和非关系型数据库的区别？（为什么两种都要使用？）

- （1）查询速度：关系型数据库将数据存放在硬盘中，非关系型数据库将数据存放在内存中，因此非关系型数据库的查询速度较快；
- （2）存储数据的格式：关系型数据库只支持基础类型，而Nosql的存储格式是key，value形式、文档形式、图片形式等，所以可以存储基础类型以及对对象或者是集合等各种格式；
- （3）扩展性：关系型数据库有类似join这样的多表查询机制的限制导致扩展很艰难，非关系型数据库有高扩展性，可做集群，可随意扩展节点且数据不需要提前保持一致；
- （4）数据一致性：非关系型数据库一般强调的是数据最终一致性，关系型数据库强调数据的强一致性，从非关系型数据库中读到的有可能还是处于一个中间态的数据；
- （5）持久存储：非关系型数据库的数据存放在内存重启操作系统会导致数据丢失，关系型数据库的数据存放在硬盘重启操作系统数据不会丢失。

3、Mysql数据库引擎以及区别？

数据库存储引擎是数据库底层软件组件，数据库管理系统使用数据库引擎进行创建、查询、更新和删除数据等操作。不同的存储引擎提供不同的存储机制、索引技巧、锁定水平等功能，使用不同的存储引擎还可以获得特定的功能。

现在许多数据库管理系统都支持多种不同的存储引擎。MySQL的核心就是存储引擎。

在MySQL中，不需要在整个服务器中使用同一种存储引擎，针对具体的要求，可以对每一个表使用不同的存储引擎。

MySQL 5.7 支持的存储引擎有 InnoDB、MyISAM、Memory、MRG_MYISAM、Archive、Federated、CSV、BLACKHOLE 、PERFORMANCE_SCHEMA。

可以使用`SHOW ENGINES`语句查看系统所支持的引擎类型，Support 列的值表示某种引擎是否能使用，`YES`表示可以使用，`NO`表示不能使用，`DEFAULT`表示该引擎为当前默认的存储引擎。

InnoDB事务型数据库的首选引擎，支持事务安全表（ACID），支持行锁定和外键。MySQL5.5.5之后，InnoDB作为默认存储引擎。

MyISAM是基于ISAM的存储引擎，并对其进行扩展，是在web、数据仓储和其他应用环境下最常使用的存储引擎之一。MyISAM拥有较高的插入、查询速度，但不支持事务。是mysql5.1前的默认存储引擎

MEMORY存储引擎将表中的数据存储在内存中，为查询和引用其他数据提供快速访问。

可以根据以下的原则来选择 MySQL 存储引擎：

- 如果要提供提交、回滚和恢复的事务安全（ACID 兼容）能力，并要求实现并发控制，InnoDB 是一个很好的选择。
- 如果数据表主要用来插入和查询记录，则MyISAM引擎提供较高的处理效率。
- 如果只是临时存放数据，数据量不大，并且不需要较高的数据安全性，可以选择将数据保存在内存的MEMORY引擎中，MySQL中使用该引擎作为临时表，存放查询的中间结果。
- 如果只有INSERT和SELECT 操作，可以选择Archive引擎，Archive存储引擎支持高并发的插入操作，但是本身并不是事务安全的。Archive 存储引擎非常适合存储归档数据，如记录日志信息可以使用Archive 引擎。

功能	MyISAM	MEMORY	InnoDB	Archive
存储限制	256TB	RAM	64TB	None
支持事务	No	No	Yes	No
支持全文索引	Yes	No	No	No
支持树索引	Yes	Yes	Yes	No
支持哈希索引	No	Yes	No	No
支持数据缓存	No	N/A	Yes	No
支持外键	No	No	Yes	No

4、Mysql数据库事务的特性？

数据库事务必须具备ACID特性，ACID分别是Atomic原子性，Consistency一致性，Isolation隔离性，Durability持久性。

- (1)原子性：在一个事物中所有的操作要么都成功，要么都失败。
- (2)一致性：在事务执行前后，数据库的一致性约束没有被破坏。ACID中的一致性包含实体完整性约束不被破坏，完整性包含实体完整性（主属性不为空）、参照完整性（外键必须存在原表中）、用户自定义的完整性。
- (3)隔离性：隔离性是指两个事物之间互不干扰。实现事物隔离性主要有两种方式：枷锁、多版本控制。
- (4)持久性：事物对数据库所做的更改会持久的保存在数据库中，不会被回滚。持久性需要考虑到事物在执行过程中可能出现的各种异常，并对异常做出相应的处理。

5、Mysql数据库索引的概念、作用及应用？

1.索引的概念：

索引是一种特殊的文件，包含着对数据表中所有的记录的引用指针，数据库索引好比是一本书前面的目录，能加快数据库的查询速度，数据库索引就是为了提高表的搜索效率而对某些字段中的值建立的目录。

2.索引的作用：

建立索引的目的是加快对表中记录的查找或排序，为表设置索引要付出代价：一是增加了数据库的存储空间，二是在插入和修改数据时要花费更多的时间（因为索引也会随之改变）。

3.索引的应用

- (1)设置合适的索引之后，数据库利用各种快速的定位技术，可以大大加快数据的查询速度，这也是创建索引的最主要的原因。
- (2)当表很大的时候，或者查询涉及到多个表时，使用索引可以使查询速度快上千倍。
- (3)可以降低数据库的IO成本，并且索引还可以降低数据库的排序成本。
- (4)通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。
- (5)在使用分组和排序子句进行数据查询时，可以显著减少查询中分组和排序的时间。

6、Mysql有关权限的表格都有哪些？

user：用户账号、全局权限
db：库级别权限
host：废弃
tables_priv：表级别权限
columns_priv：列级别权限
procs_priv：存储过程和存储函数相关的权限
proxies_priv：代理用户权限

7、Mysql有哪些数据类型？

int: 整数
bigint: 长整数
float: 小数
char: 字符（删除尾部空格）
varchar: 长字符（保留尾部空格）
time: 时间 xx:xx:xx
date: 日期 xxxx-xx-xx
datetime: 日期时间 xxxx-xx-xx xx:xx:xx

8、Mysql数据备份的几种方式，以及数据恢复的几种方式？(以及你上家公司采用哪种备份和恢复方式)，mysql备份时需要注意什么？

逻辑备份：备份的是建表、建库、插入等操作所执行SQL语句，适用于中小型数据库，效率相对较低

恢复：使用输入重定向的方式，将备份文件恢复到数据库中的指定库。

bin-log日志恢复数据（开启bin-log日志，找到要恢复的sql语句的起始位置、结束位置）

物理备份：直接复制数据库文件，适用于大型数据库环境，不受存储引擎的限制，但不能恢复到不同的MySQL版本

完整备份：每次都把所有数据（不管自第一次备份以来有没有修改过），进行一次完整的复制，备份后会清除文件的存档属性，方便日后增量备份或者差异备份进行版本比较。特点：占用空间大，备份速度慢，但恢复时一次恢复到位，恢复速度快。

增量备份：每次备份上一次备份到现在产生的新数据。特点：因每次仅备份自上一次备份（注意是上一次，不是第一次）以来有变化的文件，所以备份体积小，备份速度快，但是恢复的时候，需要按备份时间顺序，逐个备份版本进行恢复，恢复时间长

差异备份：只备份跟完整备份不一样的。特点：占用空间比增量备份大，比完整备份小，恢复时仅需要恢复第一个完整版本和最后一次的差异版本，恢复速度介于完整备份和增量备份之间。

恢复：

增量备份恢复流程

1. 停止数据库
2. 清理环境
3. 依次重演回滚redo log——> 恢复数据
4. 修改权限
5. 启动数据库
6. binlog恢复

差异备份恢复流程

1. 停止数据库
2. 清理环境
3. 重演回滚redo log（周一，某次差异）——> 恢复数据
4. 修改权限
5. 启动数据库
6. binlog恢复

注意事项:

1. 从二级复制服务器上备份。
2. 在进行备份期间停止复制，以避免在数据依赖和外键约束上出现不一致。
3. 彻底停止MySQL，从数据库文件进行备份。
4. 如果使用 `MySQL dump` 进行备份，请同时备份二进制日志文件，确保复制没有中断。
5. 不要信任LVM 快照，这很可能产生数据不一致，将来会给你带来麻烦。
6. 如果数据是与其他表隔离的，为了更容易进行单表恢复，以表为单位导出数据。
7. 当使用`mysqldump`时请使用`opt`。
8. 在备份之前检查和优化表。
9. 为了更快的进行导入，在导入时临时禁用外键约束和唯一性检测。
10. 在每一次备份后计算数据库，表以及索引的尺寸，以便监控数据尺寸的增长。
11. 通过自动调度脚本监控复制实例的错误和延迟。
12. 定期执行备份，定期测试你的备份。

9、Mysql主从复制的原理？以及方式？主从复制失效该如何处理？判断主从延迟的方法？从库同步慢的原因？从库同步数据比较慢该如何处理？

原理:

主服务器上面的任何修改都会保存在二进制日志（`Bin-log`日志）里面。

从服务器上面启动一个I/O线程，连接到主服务器上面请求读取二进制（`Bin-log`）日志，然后把读取到的二进制日志写到本地的`Realy-log`（中继日志）里面。

从服务器上面同时开启一个SQL线程，读取`Realy-log`（中继日志），如果发现有更新立即把更新的内容在本机的数据库上面执行一遍。

方式:

MySQL可以通过两种方式配置主从复制，一种是通过二进制日志（`binary log`）的方式；另一种是通过GTID（全局事务ID）方式，不过GTID方式仍然依赖MySQL的`binary log`。

`binary log`的核心是事件（`event`）。基本原理是：主服务器把所有对数据库的操作（例如`update`、`delete`，`create`等）作为一个事件，当有事件产生，就把它们写入到对应的`binary log`中，每个事件都对应一个位置（可以理解这个位置就是事件的编号）。然后从服务器读取主服务器中的日志文件来获取主服务产生的事件（把这些日志保存到从服务器本地的`relay-log`中），从而把主服务的数据库操作在从服务器中重复执行一次，达到数据复制的目的。由于一切操作都是基于`binary log`，因此主服务器必须开启`log-bin`选项，另外，主从服务器都必须分配一个唯一的`server-id`。

GTID实际上是对传统基于`binary log`的复制进行了增强。在`binary log`复制方式中，我们必须手动跟踪主服务器的日志名称和位置；但在GTID工作方式下面，我们无需跟踪这两个值，取而代之的是由MySQL自动跟踪它们，并使用GTID来标记哪些事务已经被处理，哪些还没有被处理。GTID的一个优点是：相同GTID的事务不会被重复处理，好处是可以最大限度地确保数据的一致性。

GTID由两部分构成：一部分是服务器的UUID，另一部分是事务ID

失效处理:

进入slave服务器的数据库，输入命令“`show slave status\G`”

发现`Slave_SQL_Running: No`

解决方法一:

- （1）程序可能在slave上进行了写操作
- （2）也可能是slave机器重启后，事务回滚造成的。

一般是事务回滚造成的:

解决办法:

```
mysql> slave stop;
mysql> set GLOBAL SQL_SLAVE_SKIP_COUNTER=1;
mysql> slave start;
```

解决办法二:

- （1）首先停掉Slave服务: `slave stop`
- （2）到主服务器上查看主机状态:
- （3）记录File和Position对应的值
- （4）进入master

```
mysql> show master status;
(5) 然后到slave服务器上执行手动同步:
mysql> change master to
> master_host='master_ip',
> master_user='user',
> master_password='pwd',
> master_port=3306,
> master_log_file=localhost-bin.000094',
> master_log_pos=33622483 ;
mysql> slave start;
```

注意：手动同步需要停止master的写操作！

判断延迟的方法：

通过show slave status进行查看，比如可以看看Seconds_Behind_Master参数的值来判断，是否有发生主从延时。

NULL-表示io_thread或是sql_thread有任何一个发生故障，也就是该线程的Running状态是No,而非Yes.

0-该值为零，是我们极为渴望看到的情况，表示主从复制状态正常

同步较慢原因：

主要是因为服务器压力大、从库太多

- 1、从库太多导致复制延迟 优化：建议从库数量3-5个为宜
- 2、从库的硬件比主库差 优化：提升硬件性能
- 3、慢SQL语句过多 优化：SQL语句执行时间太长，需要优化SQL，包括建立索引或者采用分库分表等。
- 4、主从复制的设计问题
优化：主从复制单线程，可以通过多线程IO方案解决；另外mysql5.6.3支持多线程的IO复制。
- 5、主从之间的网络延迟 优化：尽量采用短的链路，提升端口的带宽
- 6、主库读写压力大 优化：前端加buffer和缓存。主从延迟不同步

同步较慢处理：

(1)架构方面

- 1.业务的持久化层的实现采用分库架构，mysql服务可平行扩展，分散压力。
- 2.单个库读写分离，一主多从，主写从读，分散压力。这样从库压力比主库高，保护主库。
- 3.服务的基础架构在业务和mysql之间加入memcached或者redis的cache层。降低mysql的读压力。
- 4.不同业务的mysql物理上放在不同机器，分散压力。
- 5.使用比主库更好的硬件设备作为slave，mysql压力小，延迟自然会变小。

(2)硬件方面

- 1.采用好服务器，比如4u比2u性能明显好，2u比1u性能明显好。
- 2.存储用ssd或者盘阵或者san，提升随机写的性能。
- 3.主从间保证处在同一个交换机下面，并且是万兆环境。

(3)mysql主从同步加速

- 1、sync_binlog在slave端设置为0
- 2、logs-slave-updates 从服务器从主服务器接收到的更新不记入它的二进制日志。
- 3、直接禁用slave端的binlog
- 4、slave端，如果使用的存储引擎是innodb，innodb_flush_log_at_trx_commit =2

10、Mysql的binlog日志如何开启，以及有什么作用？

```
[root@mysql-1 ~]# vim /etc/my.cnf
server-id = 1
log-bin = /var/log/mysql/mysql-bin.log
[root@mysql-1 ~]# mkdir /var/log/mysql
[root@mysql-1 ~]# chown mysql:mysql /var/log/mysql
[root@mysql-1 ~]# systemctl restart mysqld
```

11、MySQL数据库cpu飙升到500%的话，该怎么处理？

- (1) 多实例的服务器，先top查看是那一个进程，哪个端口占用CPU多；
- (2) 如果是mysqld造成的，进库用show processeslist查看是否由于大量并发，锁引起的负载问题；
- (3) 否则，查看慢查询，找出执行时间长的sql；explain分析sql是否走索引，sql优化；
- (4) 也有可能是每个sql消耗资源并不多，但是突然之间，有大量的session连进来导致cpu飙升，这种情况就需要跟应用一起来分析为何连接数会激增，再做出相应的调整，比如说限制连接数等
- (5) 再查看是否缓存失效引起，需要查看buffer命中率；

12、Mysql数据损坏，如何修复？

(1) myisamchk

使用 myisamchk 必须暂时停止MySQL服务器。例如，我们要检修discuz数据库。执行以下操作：

```
# systemctl stop mysqld (停止 MySQL );  
# myisamchk -r /数据库文件的绝对路径/*MYI  
# systemctl start mysqld
```

myisamchk 会自动检查并修复数据表中的索引错误。

(2) mysqlcheck

使用 mysqlcheck 无需停止MySQL，可以进行热修复。操作步骤如下：

```
# mysqlcheck -r discuz.*  
# systemctl stop mysqld  
# myisamchk -r /数据库文件的绝对路径/*MYI  
# systemctl start mysqld
```

myisamchk 会自动检查并修复数据表中的索引错误。

注意：无论是 myisamchk 还是 mysqlcheck ，一般情况下不要使用 -f 强制修复，-f 参数会在遇到一般修复无法成功的时候删除部分出错数据以尝试修复。所以，不到万不得已不要使用 -f。

13、Mysql做主从为什么写server_id？

server_id用于标识该语句最初是从哪个server写入的

(1) 每一个同步中的slave在master上都对应一个master线程，该线程就是通过slave的server-id来标识的；每个slave在master端最多有一个master线程，如果两个slave的server-id 相同，则后一个连接成功时，前一个将被踢掉。 这里至少有这么一种考虑：

slave主动连接master之后，如果slave上面执行了slave stop；则连接断开，但是master上对应的线程并没有退出；当slave start之后，master不能再创建一个线程而保留原来的线程，那样同步就可能有问题；

(2) 在mysql做主主同步时，多个主需要构成一个环状，但是同步的时候有要保证一条数据不会陷入死循环，这里就是靠server-id来实现的

14、如何保证数据库数据的安全？

- 1、避免从互联网访问MySQL数据库，确保特定主机才拥有访问特权
- 2、禁用或限制远程访问
- 3、定期备份数据库
- 4、设置root用户的口令并改变其登录名
- 5、移除测试(test)数据库
- 6、禁用LOCALINFILE
- 7、移除匿名账户和废弃的账户
- 8、降低系统特权
- 9、降低用户的数据库特权
- 10、移除和禁用.mysql_history文件
- 11、保持数据库为最新稳定版本，因为攻击者可以利用上一个版本的已知漏洞来访问企业的数据库。
- 12、启用日志

15、mysql如何新建用户？

```
# 指定任何ip的mjj用户登录
create use 'mjj'@'%' identified by '123';
```

16、如何在数据库中杀掉某条sql语句？

首先登录mysql，然后使用：`show processlist;` 查看当前mysql中各个线程状态。
然后`kill +id`号杀死即可

17、innodb的优点

- (1) 支持事务(事务是指逻辑上的一组操作,组成这组操作的各个单元,要么全成功,要么全失败)
- (2) 行级锁定(更新时一般是锁定当前行):通过索引实现,全表扫描仍然会是锁定整个表,注意间隙锁的影响.
- (3) 读写阻塞与事务隔离级别相关.
- (4) 具有非常高效的缓存特性,能缓存索引,也能缓存数据.
- (5) 整个表和主键以Cluster方式存储,组成一颗平衡树.
- (6) 所有Secondary Index 都会保存主键信息.
- (7) 支持分区,表空间.类似于Oracle数据库.
- (8) 支持外键约束,不支持全文索引,5.5之前支持,后面不再支持.
- (9) 和MyISAM相比,InnoDB对于硬件资源要求比较高.

18、什么样的数据存到关系型数据库，什么样的数据存到非关系型数据库？

非关系型数据库的优势：

- (1) 性能：NOSQL是基于键值对的，可以想象成表中的主键和值的对应关系，而且不需要经过SQL层的解析，所以性能非常高
- (2) 可扩展性：同样也是因为基于键值对，数据之间没有耦合性，所以非常容易水平扩展。
- (3) 使用场景：日志、埋点、论坛、博客等

关系型数据库的优势：

- (1) 复杂查询：可以用SQL语句方便的在一个表以及多个表之间做非常复杂的数据查询
- (2) 事务支持：使得对于安全性能很高的数据访问要求得以实现。
- (3) 使用场景：所有有逻辑关系的数据存储

19、Oracle和mysql区别？

- (1) 对事务的提交
MySQL默认是自动提交。
Oracle默认不自动提交，需要用户手动提交，需要再写commit指令或点击commit按钮
- (2) 分页查询
MySQL是直接在SQL语句中写"`select... from ...where...limit x, y`",有limit就可以实现分页。
Oracle则是需要用到伪列ROWNUM和嵌套查询
- (3) 事务隔离级别
MySQL是read committed的隔离级别，而Oracle是repeatable read的隔离级别，但二者都支持serializable串行化事务隔离级别。MySQL没有类似Oracle的构造多版本数据块的机制，只支持read committed的隔离级别。一个session读取数据时，其他session不能更改数据，但可以在表最后插入数据。session更新数据时，要加上排它锁，其他session无法访问数据。
- (4) 对事务的支持
MySQL在innodb存储引擎的行级锁的情况下才可支持事务
Oracle则完全支持事务
- (5) 保存数据的持久性
MySQL是在数据库更新或者重启，则会丢失数据。
Oracle把提交的sql操作写入了在线联机日志文件中，保持到了磁盘上，可以随时恢复。
- (6) 并发性

MySQL: 以表级锁为主, 对资源锁定的力度很大, 如果一个**session**对一个表加锁时间过长, 会让其他**session**无法更新此表中的数据。

Oracle: 使用行级锁, 对资源锁定的力度要小很多, 只是锁定**sql**需要的资源, 并且加锁是在数据库中的数据行上, 不依赖与索引。所以**Oracle**对并发性的支持要好很多。

(7) 逻辑备份

MySQL: 逻辑备份时要锁定数据, 才能保证备份的数据是一致的, 影响业务正常的**dml**使用。

Oracle: 逻辑备份时不锁定数据, 且备份的数据是一致的。

(8) 复制

MySQL: 复制服务器配置简单, 但主库出问题时, 从库有可能丢失一定的数据。且需要手工切换从库到主库。

Oracle: 既有推或拉式的传统数据复制, 也有**dataguard**的双机或多机容灾机制, 主库出现问题时, 可以自动切换备库到主库, 但配置管理较复杂。

(9) 性能诊断

MySQL的诊断调优方法较少, 主要有慢查询日志。

Oracle有各种成熟的性能诊断调优工具, 能实现很多自动分析、诊断功能。比如**awr**、**addm**、**sqltrace**、**tkproof**

(10) 权限与安全

MySQL的用户与主机有关, 感觉没有什么意义, 另外更容易被仿冒主机及**ip**有可乘之机。

Oracle的权限与安全概念比较传统, 中规中矩。

(11) 分区表和分区索引

MySQL的分区表还不太成熟稳定。

Oracle的分区表和分区索引功能很成熟, 可以提高用户访问**db**的体验。

(12) 管理工具

MySQL管理工具较少, 在**linux**下的管理工具的安装有时要安装额外的包, 有一定复杂性。

Oracle有多种成熟的命令行、图形界面、**web**管理工具, 还有很多第三方的管理工具, 管理极其方便高效。

(13) 最重要的区别

MySQL是轻量型数据库, 并且免费, 没有服务恢复数据。

Oracle是重量型数据库, 收费, **Oracle**公司对**Oracle**数据库有任何服务。

20、数据库如何优化?

- 1、优化索引、**sql**语句、分析慢查询
- 2、设计表的时候严格根据数据的设计规范来设计数据库
- 3、使用缓存, 把经常访问到的数据而且不需要变化的数据放到缓存中
- 4、使用固态硬盘
- 5、采用**MySQL**内部自带的表分区技术, 把数据分层到不同的文件中, 能够提高磁盘的读写效率
- 6、垂直分表, 把一些不经常用到的数据放到一个表中, 节约磁盘的**I/O**
- 7、主从读写分离, 采取主从复制把数据库的读操作和写操作分离出来
- 8、数据库分表分机器 (数据特别大的)
- 9、选择合适的表引擎, 对参数上的优化

21、慢查询日志是做什么的?

MySQL的慢查询日志是**MySQL**提供的一种日志记录, 它用来记录在**MySQL**中响应时间超过阈值的语句, 具体指运行时间超过**long_query_time**值的**SQL**, 则会被记录到慢查询日志中。**long_query_time**的默认值为**10**, 意思是运行**10s**以上的语句。默认情况下, **Mysql**数据库并不启动慢查询日志, 需要我们手动来设置这个参数, 当然, 如果不是调优需要的话, 一般不建议启动该参数, 因为开启慢查询日志会或多或少带来一定的性能影响。慢查询日志支持将日志记录写入文件, 也支持将日志记录写入数据库表。

注意: 开启慢查询会带来**CPU**损耗与日志记录的**IO**开销, 所以我们要间断性的打开慢查询日志来查看**Mysql**运行状态。

22、bin-log日志是记录什么的?

(1) 什么是binlog

binlog日志用于记录所有更新了数据或者已经潜在更新了数据（例如，没有匹配任何行的一个DELETE）的所有语句。语句以“事件”的形式保存，它描述数据更改。

(2) binlog作用

因为有了数据更新的binlog，所以可以用于实时备份，用于master/slave主从复制。

23、mysql读写分离的部署？

- 1、在mycat机器上面做解析，解析后端的数据库服务器
- 2、上传jdk的包，写java的环境变量，并source 让它生效
- 3、下载mycat的包，并解压
- 4、修改mycat的配置文件server.xml
- 5、修改mycat的schema.xml
- 6、启动mycat，并检查
- 7、在真实的后端数据库上给用户授权
- 8、使用mysql的客户端登录 mycat机器的8066端口
- 9、登录成功之后，检测

24、mysql有没有去中心化的集群？

有：Mysql-galera集群

Galera是一个MySQL(也支持MariaDB, Percona)的同步多主集群软件。

所有节点可以同时读写数据库

自动成员资格控制，失败节点从集群中删除

新节点加入数据自动复制

真正的并行复制，行级

25、Mysql数据恢复失败，如何处理？

首先在恢复之前就应该做足准备工作，避免恢复的时候出错。比如说备份之后的有效性检查、权限检查、空间检查等。如果万一报错，再根据报错的提示来进行相应的调整。或者回滚、删掉数据，重新恢复

26、mysql的主键？

mysql设置主键的代码是PRIMARY KEY

27、mysql中的char和varchar区别？

一个固定一个不固定，char固定了长度，长度不够会填充，varchar不会
char的长度不可变，存储英文字符一个字节，汉字两个字节；varchar的长度可变，都是两个字节

28、列举三种表连接算法以及各自高性能的场景

三种连接算法：嵌套循环连接、合并连接、Hash连接

嵌套循环连接：通常在小数据量并且语句比较简单的场景中使用

合并连接：在SQL数据库中，如果查询优化器，发现要连接的两张对象表，在连接上都已经排序并包含索引，那么优化器将会极大

可能选择“合并”连接策略。条件是：两个表都是排序的，并且两个表连接条件中至少有一个等号连接，查询分析器会去选择合并连接

Hash连接：当我们尝试将两张数据量较大，没有排序和索引的两张表进行连接时，SQLServer的查询优化器会尝试使用HashJoin

29、对于数据库备份正确的做法是什么？

同时使用多个备份设备，使得备份可以同时写入所有设备。同样，也可以同时从多个设备还原备份。

使用数据库备份、差异数据库备份和事务日志备份的组合，使得将数据库恢复到故障点所用的备份数量减到最少

使用文件和文件组备份以及事务日志备份，使得可以只备份或还原那些包含相关数据的文件，而不用备份整个数据库。

使用快照备份将备份和还原时间减到最少。

30、MySQL—主多从，主库宕机，如何合理切换到从库,其他的从库又该如何处理？

- 1、首先应该停业务
- 2、所有从库停掉IO线程
- 3、所有从库执行show slave status\G;
对比所有从库的Relay_Master_Log_File,Exec_Master_Log_Pos两个参数的值，数值最大的作为新的主库，如果所有从库的值都相同，都没有落后于主库，选择哪个从库都可以做为新主库。
- 4、将选好的那个从库设置为主库，并修改程序中的ip为新设置的主库的IP
- 5、创建其他从库的复制账号、其他从库指向新主库

31、误操作drop语句导致数据库数据被破坏，请给出恢复思想及实际大体步骤。

所有数据恢复的基础都在于备份，必须要有完整的备份，否则恢复无从谈起，误操作导致的数据库破坏需要使用增量恢复的方法进行恢复数据库，具体步骤如下：

- 1) 查看备份与binlog文件
- 2) 刷新并备份binlog文件
`mysqladmin -uroot -pmysql123 -s /data/mysql.sock flush-logs`
- 3) 将binlog文件恢复成sql语句
`mysqlbinlog --no-defaults mysql-bin.000061 mysql-bin.000062 >bin.sql`
- 4) 将其中误操作的语句删除（就是drop的动作）
- 5) 解压全备文件，恢复全备文件
`gzip -d mysql_backup_2016-10-12.sql.gz`
`mysql -uroot -pmysql123 -s/data/3306/mysql.sock < mysql_backup_2016-10-12.sql`
如果有对表的操作，恢复数据时需要接表名
- 6) 恢复误操作前的binlog文件记录的sql语句
`mysql -uroot -pmysql123 -s/data/3306/mysql.sock < bin.sql`

最后登陆数据库，查看数据是否恢复成功，如果有确定的误操作时间，就直接恢复这段时间的数据即可。

32、分析器在MongoDB中的作用是什么？

mongodb 中包括了一个可以显示数据库中每个操作性能特点的数据分析器，通过这个分析器可以找到比预期慢的查询（或操作）；利用这一信息：比如可以确定是否添加索引

33、MongoDB名称空间(namespace)是什么？

MongoDB存储BSON对象在丛集(collection)中。数据库名字和丛集名字以句点连结起来叫做名字空间(namespace)。

...

34、MongoDB如果用户移除对象的属性，该属性是否从存储层中删除？

是的，用户移除属性然后对象会重新保存(re-save())。

35、MongoDB能否使用日志特征进行安全备份？

可以

36、MongoDB支持存储过程吗? 如果支持的话, 怎么用?

MongoDB支持存储过程, 它是javascript写的, 保存在db.system.js表中。

37、为什么mongodb的数据文件比实际使用的要大?

MongoDB采用的预分配空间的方式来防止文件碎片。

38、MongoDB的getLastError的作用

getError的安全写入

Mongodb的写操作默认是没有任何返回值的, 这减少了写操作的等待时间, 也就是说, 不管有没有写入到磁盘或者有没有遇到错误, 它都不会报错。但一般我们是不放心这么做的, 这时候就调用getError命令, 得到返回值。

getError 是Mongodb的一个命令, 它好像是取得最后一个error, 但其实它是Mongodb的一种客户端阻塞方式, 用这个命令来获得写操作是否成功的信息。getError有几个参数:j, w, fsync。

在多线程模式下读写Mongodb的时候, 如果这些读写操作是有逻辑顺序的, 那么这时候也有必要调用getError命令, 用以确保上个操作执行完下个操作才能执行, 因为两次执行的连接有可能是不同的。在大多数情况下, 我们都会使用连接池去连接mongodb, 所以这是需要注意的。

39、MongoDB分片(sharding) 和复制(replication) 是怎样工作的?

每一个分片(shard)是一个分区数据的逻辑集合。分片可能由单一服务器或者集群组成, 我们推荐为每一个分片(shard)使用集群。

40、请介绍下mongodb数据结构?

Collections: 在mongodb中叫做集合, 是文档的集合。无模式, 可以存储各种各样的文档。类似mysql中的表。

在关系型数据库中, 关系数据库的每一张表就是一个关系模型的映射, 每张表的字段就是对应的实体的属性和主外键的集合, 每个字段需要提前定义。

Document: 这里的user集合(“表”)有一个document (document可以理解为mysql中的记录)。文档是mongodb保存数据的基本单元。数据的存储结构为BSON格式, 也就是我们开始添加的文档, key value键值对类型。

文档中保存到数据类型可以为: null、boolean、String、Object、32位整数、64位整数、64位浮点数、日期、正则表达式、js代码、二进制数据、数组、内嵌文档、最大值、最小值、未定义类型。

GridFS: 因为bson对象的大小有限制, 不适合存储大型文件, GridFS文件系统为大型文件提供了存储的方案, GridFS下的fs保存的是图片、视屏等大文件。

无论是bson对象还是GridFS中存储的大文件, 我们发现当添加一个文档的时候, 会自动的添加_id, 不同的是图片添加后会自动的加上_id, chunksize, md5, length, aliases等, 这些属性是我们上传完图片后, mongodb分析后自动添加的, 系统自动保存。

41、mysql锁表是什么?

MySQL的两种表锁, 一种是lock table的表锁, 一种是元数据锁 metadata Lock。

第一种表级别的锁, 一个线程执行lock table操作, 比如说你执行读操作, 别的线程想写操作就不行了, 但是他也要读呢? 没问题啊, 反正你不影响我就好了。同理写操作, 你这时候读肯定就不行了。什么时候可以正常的读写, 那就是执行unlock table以后就可以了。

第二种元数据锁, metadata Lock也叫MDL。这个其实是我们使用中最常见的一个锁表。这个MDL是一个隐式锁, 当你访问一个表它会自动加一个MDL, 来保证你数据的读写的正确。举个例子, 人家在访问, 这时候一个字段正在修改甚至删除或者增加, 肯定会影响对写的结果, 这时候就把这表锁上。

同样我们在修改字段时，有的时候会被锁表很大的概率是MDL导致的，而同样的操作可能有的时候就不会锁表，那是因为你在执行修改的时候，没有人遍历你所修改的表。而实际情况中，你的程序可能在跑，在访问这个表，别人不知情的情况下也回去访问读写这个表，结果就可能导致表被锁，尽管你只做了一个操作，或者访问，或者修改。

处理：

步骤1 查看锁的表

```
show OPEN TABLES where In_use > 0;
```

显示的结果就是被锁的表，有的表会关联别的表会导致多个表被锁（其实你可以越过这个步骤，因为第二步你能知道那些操作锁了。）

步骤2 查找进程

```
show processlist;
```

步骤3 找到锁表的进程

就是那个锁表的进程，记住这个进程号，就是第一个id字段。

步骤4 kill掉这个进程

```
kill xxxxx
```

（类似于我们Linux那种杀死进程那种。）

步骤5 回头看一看还有没有被锁的表

```
show OPEN TABLES where In_use > 0;
```

发现没有异常的锁表进程就OK。

42、物理备份

n、mysql做三主三从产生的数据量有多少？

n+1、公司一个月能产生多少数据？

三、shell

1、你写过什么shell脚本，都来实现什么功能的？说一下大概的编写思路？

监控，定时服务，自动批量部署，数据库定时备份

比如说通过监控tomcat的端口是否开启，来监控tomcat的web服务是否正常运行

写一个脚本，每5分钟执行一次获取当前服务器的基本情况，如内存的使用率、cpu负载、磁盘使用率等，并保存到数据库或者文件里

监控mysql主从复制，备份，

编写脚本，清理一个持续增长的文件，以日为单位形成新的压缩文件，并删除30天前的日志压缩文件

2、如何用awk、sed命令查看nginx日志，统计某天的访问量？

```
grep"07/Apr/2017:0[4-5]" access.log | awk '{print $1}' | sort | uniq -c| sort -nr | wc -l
```

3、awk与sed的区别？

awk是一种程序语言，对文档资料的处理具有很强的功能。**awk**擅长从格式化报文或从一个大的文本文件中抽取数据。

awk的命令格式为：

```
awk [-F filed-separator] "commands" input-file(s)
```

awk将一行文字按分隔符分为多个域，依次记为\$0, \$1, \$2 . . . \$n
代表所有域值。因此**awk**更适合于以域为单位来处理文件。

sed是一个精简的、非交互式的编辑器，它能执行与编辑**vi**相同的编辑任务。

sed的命令格式为：

```
sed [options] 'command' file(s)
```

作为编辑器，当然少不了插入（a/、i/）、删除（d）、查找替换（s）等命令。

4、\$系列问题？

\$0 当前脚本的文件名

\$n 传递给脚本或函数的参数。**n** 是一个数字，表示第几个参数。例如，第一个参数是**1**，第二个参数是**2**。

\$# 传递给脚本或函数的参数个数。

\$* 传递给脚本或函数的所有参数。

@ 传递给脚本或函数的所有参数。被双引号（" "）包含时，与 \$* 稍有不同，下面将会讲到。

\$? 上个命令的退出状态，或函数的返回值。

\$\$ 当前Shell进程ID。对于 Shell 脚本，就是这些脚本所在的进程ID。

5、写一个脚本查找最后创建时间是3天前，后缀是*.log的文件并删除。

```
#!/bin/bash
find / -mtime +3 -type f -name '*.log' -exec rm -f {} \
```

6、写一个脚本进行nginx日志统计，得到访问ip最多的前10个(nginx日志路径： /home/nin:/og/default/access log)

```
awk '{print $1}' /var/log/nginx/access.log | sort | uniq -c | sort -nr -k1 |
head -n 10
```

#####

四、nginx

1、请列举Nginx的一些特性？

Nginx (engine x) 是一个高性能的HTTP和反向代理web服务器，同时也提供了**IMAP/POP3/SMTP**服务。
特性：

（1）作为 **web 服务器**：可以作为**HTTP** 代理服务器和反向代理服务器，支持通过缓存加速访问，可以完成简单的负载均衡和容错，支持包过滤功能，支持**SSL**等，相比 **Apache**, **Nginx** 使用更少的资源，支持更多的并发连接，体现更高的效率，这点使 **Nginx** 尤其受到虚拟主机提供商的欢迎。能够支持高达 **50,000** 个并发连接数的响应，感谢 **Nginx** 为我们选择了 **epoll and kqueue** 作为开发模型。

（2）作为负载均衡服务器：可以进行自定义配置，支持虚拟主机，支持**URL**重定向，支持网络监控，支持流媒体传输等。**Nginx** 既可以在内部直接支持 **Rails** 和 **PHP**，也可以支持作为 **HTTP**代理服务器 对外进行服务。**Nginx** 用 **C** 编写，不论是系统资源开销还是 **CPU** 使用效率都比 **Perlbal** 要好的多。

（3）作为邮件代理服务器：**Nginx** 同时也是一个非常优秀的邮件代理服务器（最早开发这个产品的目的之一也是作为邮件代理服务器），他支持**IMAP/POP3** 代理服务功能，支持内部**SMTP**代理服务功能。

2、请列举Nginx和Apache 之间的不同点？

（1）内核和功能上的比较

特性	Nginx	Apache
请求管理	事件驱动模型使用异步套接字处理，减少了内存和CPU开销	同步套接字、进程和线程每个请求都要使用一个单独的进程或线程，使用同步套接字
设计语言	C	C、C++
可移植性	多平台	多平台
诞生时间	2002	1994

(2) 一般功能比较

功能	Nginx	Apache
HTTPS支持	作为模块支持	作为模块支持
虚拟主机	原生支持	原生支持
CGI支持	仅支持FastCGI	支持CGI和FastCGI
系统模块	静态模块系统	动态模块系统

注意：

CGI（Common Gateway Interface）全称是“通用网关接口”，是一种让客户端（web浏览器）与web服务器（nginx等）程序进行通信（数据传输）的协议。

FastCGI（Fast Common Gateway Interface）全称是“快速通用网关接口”是通用网关接口（CGI）的增强版本，由CGI发展改进而来，主要用来提高CGI程序性能，类似于CGI，FastCGI也是一种让交互程序与web服务器通信的协议

(3) Nginx 相对 apache 的优点

- （1）轻量级，同样起web 服务比Apache 占用更少的内存及资源
- （2）静态处理，Nginx 静态处理性能比 Apache 高3倍以上
- （3）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。在Apache+PHP（prefork）模式下，如果PHP处理慢或者前端压力很大的情况下，很容易出Apache进程数飙升，从而拒绝服务的现象。
- （4）高度模块化的设计，编写模块相对简单
- （5）社区活跃，各种高性能模块出品迅速

3、请解释Nginx如何处理HTTP请求？

nginx会根据过来的http请求头里的Host字段里的值来判断使用哪个server{}

如果请求头里没有Host字段，或者Host字段里的值，和Nginx配置文件里的server{}里的{server_name}都不匹配，则使用第一个server{}，来处理这个请求。

如果请求头里的Host字段里的值和Nginx配置文件里的某个server{}里的{server_name}，匹配上了，则使用这个server{}，来处理这个请求。

4、nginx访问日志的各个参数

\$remote_addr #记录访问网站的客户端地址
\$remote_user #远程客户端用户名
\$time_local #记录访问时间与时区
\$request #用户的http请求起始行信息
\$status #http状态码，记录请求返回的状态码，例如：200、301、404等
\$body_bytes_sent #服务器发送给客户端的响应body字节数
\$http_referer #记录此次请求是从哪个连接访问过来的，可以根据该参数进行防盗链设置。
\$http_user_agent #记录客户端访问信息，例如：浏览器、手机客户端等
\$http_x_forwarded_for #当前端有代理服务器时，设置web节点记录客户端地址的配置，此参数生效的前提是代理服务器也要进行相关的x_forwarded_for设置

变量	含义
\$bytes_sent	发送给客户端的总字节数
\$body_bytes_sent	发送给客户端的字节数，不包括响应头的大小
\$connection	连接序列号
\$connection_requests	当前通过连接发出的请求数量
\$msec	日志写入时间，单位为秒，精度是毫秒
\$pipe	如果请求是通过http流水线发送，则其值为"p"，否则为"."
\$request_length	请求长度（包括请求行，请求头和请求体）
\$request_time	请求处理时长，单位为秒，精度为毫秒，从读入客户端的第一个字节开始，直到把最后一个字符发送给客户端进行日志写入为止
\$status	响应状态码
\$time_iso8601	标准格式的本地时间,形如"2017-05-24T18:31:27+08:00"
\$time_local	通用日志格式下的本地时间，如"24/May/2017:18:31:27 +0800"
\$http_referer	请求的referer地址。
\$http_user_agent	客户端浏览器信息。
\$remote_addr	客户端IP

5、反向代理与正向代理的区别？为什么使用反向代理？

正向代理：正向代理的过程隐藏了真实的请求客户端，服务器不知道真实的客户端是谁，客户端请求的服务都被代理服务器代替请求。proxy和client同属一个LAN，对server透明

反向代理：反向代理的过程隐藏了真实的服务器，客户不知道真正提供服务的人是谁，客户端请求的服务都被代理服务器处理。反向代理代理的是响应方，也就是服务端。proxy和server同属一个LAN，对client透明。

使用反向代理的原因：

将服务端隐藏，可以防止服务器被入侵或者恶意攻击

6、Nginx如何记录真实客户端的ip地址？

将配置文件中的log_format模块打开，使用\$remote_addr，\$http_x_forwarded_for变量可以记录客户端真实ip，并且在server中添加以下在location中的内容：

```
limit_req_zone $binary_remote_addr zone=mylimit:10m rate=1r/s;
upstream myweb {
    server 192.168.62.157:80 weight=1 max_fails=1 fail_timeout=1;
}
server {
    listen 80;
    server_name localhost;
    location /login {
        limit_req zone=mylimit;
        proxy_pass http://myweb;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

7、nginx防盗链怎么做的？

- 1、区分不正常的用户；方法：HTTP Referer是Header的一部分，当浏览器向web服务器发送请求的时候，一般会带上Referer，告诉服务器我是从哪个页面链接过来的，服务器借此可以获得一些信息用于处理。但HTTP Referer头信息是可以通程序来伪装生成的，所以通过Referer信息防盗链并非100%可靠，但是，它能够限制大部分的盗链情况。
- 2、nginx防止网站资源被盗用模块：ngx_http_referer_module
- 3、通过查看访问日志确定盗链机的ip，在防盗链机上的nginx配置文件中设置将盗链机的访问拒绝掉即可

8、nginx做七层怎么配置？

所谓四层就是基于IP+端口的负载均衡，通过虚拟IP+端口接收请求，然后再分配到真实的服务器；七层通过虚拟的URL或主机名接收请求，然后再分配到真实的服务器七层就是基于URL等应用层信息的负载均衡。

9、Nginx 是如何实现高并发的？

简单来讲，就是异步，非阻塞，使用了epoll和大量的底层代码优化。
稍微详细一点展开的话，就是nginx的特殊进程模型和事件模型的设计。

进程模型

nginx采用一个master进程，多个worker进程的模式。

1. master进程主要负责收集、分发请求。当一个请求过来时，master拉起一个worker进程负责处理这个请求。
2. master进程也要负责监控worker的状态，保证高可靠性
3. worker进程一般设置为跟cpu核心数一致。nginx的worker进程跟apache不一样。apache的进程在同一时间只能处理一个请求，所以它会开很多个进程，几百甚至几千个。而nginx的worker进程在同一时间可以处理额请求数只受内存限制，因此可以处理多个请求。

事件模型

nginx是异步非阻塞的。

每进来一个request，会有一个worker进程去处理。但不是全程的处理，处理到什么程度呢？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不会这么傻等着，他会在发送完请求后，注册一个事件：“如果upstream返回了，告诉我一声，我再接着干”。于是他就休息去了。此时，如果再有request 进来，他就可以很快再按这种方式处理。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。

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。

nginx 的异步非阻塞工作方式正是利用了这点等待的时间。在需要等待的时候，这些进程就空闲出来待命了。因此表现为少数几个进程就解决了大量的并发问题。

nginx是如何利用的呢，简单来说：同样的4个进程，如果采用一个进程负责一个request的方式，那么，同时进来4个request之后，每个进程就负责其中一个，直至会话关闭。期间，如果有第5个request进来了。就无法及时反应了，因为4个进程都没干完活呢，因此，一般有个调度进程，每当新进来了一个request，就新开个进程来处理。

10、为什么 Nginx 不使用多线程？

Nginx 要保证它的高可用、高可靠性，如果Nginx使用了多线程的时候，由于线程之间是共享同一个地址空间的，当某一个第三方模块引发了一个地址空间导致的断错时（eg：地址越界），会导致整个Nginx全部挂掉；当采用多进程来实现时，往往不会出现这个问题。

11、为什么要做动、静分离？

Nginx是当下最热的web容器，网站优化的重点在于静态化网站，网站静态化的关键点则是动静分离，动静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，我们则根据静态资源的特点将其做缓存操作。

让静态的资源只走静态资源服务器，动态的走动态的服务器

Nginx的静态处理能力很强，但是动态处理能力不足，因此，在企业中常用动静分离技术。

对于静态资源比如图片，js，css等文件，我们则在反向代理服务器nginx中进行缓存。这样浏览器在请求一个静态资源时，代理服务器nginx就可以直接处理，无需将请求转发给后端服务器tomcat。

若用户请求的动态文件，比如servlet,jsp则转发给Tomcat服务器处理，从而实现动静分离。这也是反向代理服务器的一个重要的作用。

12、Nginx 如何开启压缩？开启压缩之后有什么作用？

默认情况下，Nginx的gzip压缩是关闭的，gzip压缩功能就是可以让你节省不少带宽，但是会增加服务器CPU的开销，Nginx默认只对text/html进行压缩，如果要对html之外的内容进行压缩传输，我们需要手动来调。

```
gzip                on;          #开启gzip
gzip_min_length     10k;        # 设置允许压缩的页面最小字节数
gzip_buffers        4 16k;      #设置缓冲区大小
gzip_http_version   1.1;
gzip_comp_level      6;          #压缩级别官网建议是6
gzip_vary            on;
gzip_types           text/plain text/css application/javascript application/json
application/xml text/xml image/png image/gif image/jpeg; #压缩的类型
```

注意：

1、如果不指定类型，Nginx仍然不会压缩

2、gzip_http_version的设置，它的默认值是1.1，就是说对HTTP/1.1协议的请求才会进行gzip压缩
如果我们使用了proxy_pass进行反向代理，那么nginx和后端的upstream server之间是用HTTP/1.0协议通信的

13、为什么nginx采用多进程结构，而不使用多线程呢？

因为线程之间共享一个地址空间，如果某一个三方模块引发错误，会导致nginx挂掉，如果采用多进程就不会。

14、http常见访问码？（爱云校面试）

200（ok）一切正常

301（永久重定向）

302（临时重定向）

304：请求的资源并没有被修改过

400: 请求出现错误, 比如请求头不对等。
401: 没有提供认证信息。请求的时候没有带上 **Token** 等。
402: 为以后需要所保留的状态码。
403: 请求的资源不允许访问。就是说没有权限。
404 (未找到) 服务器找不到请求的网页。
410 (已删除) 如果请求的资源已永久删除, 服务器就会返回此响应。
500: 服务器错误。
501: 请求还没有被实现。
502 (错误网关) 服务器作为网关或代理, 从上游服务器收到无效响应。
503: 服务暂时不可用。服务器正好在更新代码重启。
504 (网关超时) 服务器作为网关或代理, 但是没有及时从上游服务器收到请求。
505: 请求的 **HTTP** 版本不支持。

15、apache和nginx工作模式, nginx如何调优?

apache工作模式

prefork (线程), **worker** (进程), **event** (事件模式)

nginx工作模式

master-worker模式

单线程模式

nginx调优:

- 1、查看cpu
 - 2、nginx事件处理模型
 - 3、开启高效传输模式
 - 4、连接超时时间
 - 5、**fastcgi**调优
 - 6、**gzip**调优
 - 7、**expires**缓存调优
 - 8、防盗链
 - 9、内核参数优化
 - 10、关于系统链接数的优化
- 详情可见班级打印面试题P71

16、nginx端口如何跳转?

rewrite地址重写

例:

```
# http://www.testpm.com/a/1.html ==> http://www.testpm.com/b/2.html
```

```
server {  
    listen      80;  
    server_name www.testpm.com;  
  
    location /a {  
        root /html;  
        index 1.html index.htm;  
        rewrite .* /b/2.html permanent;  
    }  
  
    location /b {  
        root /html;  
        index 2.html index.htm;  
    }  
}
```

17、nginx301（永久重定向）和302（临时重定向）区别？

301是永久重定向
302是临时重定向

18、nginx如何限流？

Nginx 提供两种限流方式：一是控制速率，二是控制并发连接数。

正常限流

ngx_http_limit_req_module模块提供限制请求处理速率能力，使用了漏桶算法(leaky bucket)。下面例子使用 ngx_limit_req_zone 和 limit_req 两个指令，限制单个IP的请求处理速率

处理突发流量

如果有时正常流量突然增大，超出的请求将被拒绝，无法处理突发流量，可以结合burst参数使用来解决该问题

限制连接数

ngx_http_limit_conn_module提供了限制连接数的能力，利用limit_conn_zone和limit_conn两个指令即可。

19、nginx有哪几种调度算法

- （1）轮询（默认）：可以通过weight指定轮询的权重，权重越大，被调度的次数越多
- （2）ip_hash：可以实现会话保持，将同一客户的IP调度到同样后端服务器，可以解决session的问题，不能使用weight
- （3）fair：可以根据请求页面的大小和加载时间长短进行调度，使用第三方的upstream_fair模块
- （4）url_hash：按请求的url的hash进行调度，从而使每个url定向到同一服务器，使用第三方的url_hash模块

五、tomcat

1、Tomcat的默认端口是多少，默认发布目录在哪？怎么修改？

8080

tomcat默认的项目发布目录是/webapp/ROOT

在server.xml文件中，有一段如下：

```
<Connector port="8080" protocol="HTTP/1.1"
            connectionTimeout="20000"
            redirectPort="8443" />
```

将port="8080"改为其它的就可以了。

2、tomcat 如何优化？具体操作？（爱云校面试）

可以对其进行内存优化、并发优化、缓存优化、io优化、开启线程池和组件优化

1、内存优化：有富余物理内存的情况，加大tomcat使用的jvm的内存

2、服务器资源

服务器所能提供CPU、内存、硬盘的性能对处理能力有决定性影响。

（1）对于高并发情况下会有大量的运算，那么CPU的速度会直接影响到处理速度。

（2）内存存在大量数据处理的情况下，将会有较大的内存容量需求，可以用-Xmx -Xms -XX:MaxPermSize等参数对内存不同功能块进行划分。

(3) 硬盘主要问题就是读写性能，当大量文件进行读写时，磁盘极容易成为性能瓶颈。最好的办法是利用缓存。

3、利用缓存和压缩（缓存优化）

对于静态页面最好是能够缓存起来，这样就不必每次从磁盘上读。这里我们采用了Nginx作为缓存服务器，将图片、css、js文件都进行了缓存，有效的减少了后端tomcat的访问。

另外，为了加快网络传输速度，开启gzip压缩也是必不可少的。但考虑到tomcat已经需要处理很多东西了，所以把这个压缩的工作就交给前端的Nginx来完成。

除了文本可以用gzip压缩，其实很多图片也可以用图像处理工具预先进行压缩，找到一个平衡点可以让画质损失很小而文件可以减小很多。

4、采用集群

单个服务器性能总是有限的，最好的办法自然是实现横向扩展，那么组建tomcat集群是有效提升性能的手段。我们还是采用了Nginx来作为请求分流的服务器，后端多个tomcat共享session来协同工作。

5、优化tomcat参数

需要修改conf/server.xml文件，主要是优化连接配置，关闭客户端dns查询。

六、负载均衡、高可用

1、四层与七层负载均衡的区别？四七层负载均衡都有什么？

	四层负载均衡	七层负载均衡
基于	基于IP+Port的	基于URL或主机IP等。
类似于	路由器	代理服务器
复杂度	低	高
性能	高；无需解析内容	中；需要算法识别 URL和 HTTP head 等信息
安全性	低，	高，
额外功能	无	会话保持，图片压缩，等

总结：从上面的对比看来四层负载与七层负载最大的区别就是效率与功能的区别。四层负载架构设计比较简单，无需解析具体的消息内容，在网络吞吐量及处理能力上会相对比较高，而七层负载均衡的优势则体现在功能多，控制灵活强大。在具体业务架构设计时，使用七层负载或者四层负载还得根据具体的情况综合考虑。

四层负载均衡：LVS、Nginx（需要换http模块）、Haproxy

七层负载均衡：Nginx、Haproxy(配置文件比较麻烦，语法不是很好写，功能没有Nginx多)

2、LVS工作模式？DR和NAT的区别？

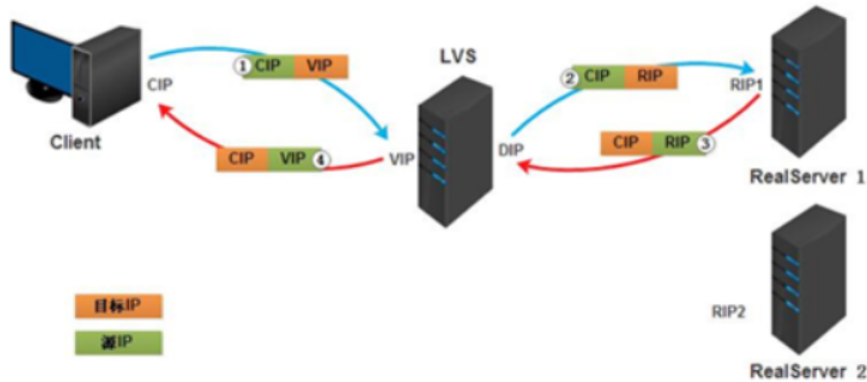
NAT：网络地址转换模式，进站/出站的数据流量经过分发器(IP负载均衡，他修改的是IP地址)，利用三层功能

DR：直接路由模式，只有进站的数据流量经过分发器(数据链路层负载均衡，因为他修改的是目的mac地址)，利用二层功能mac地址

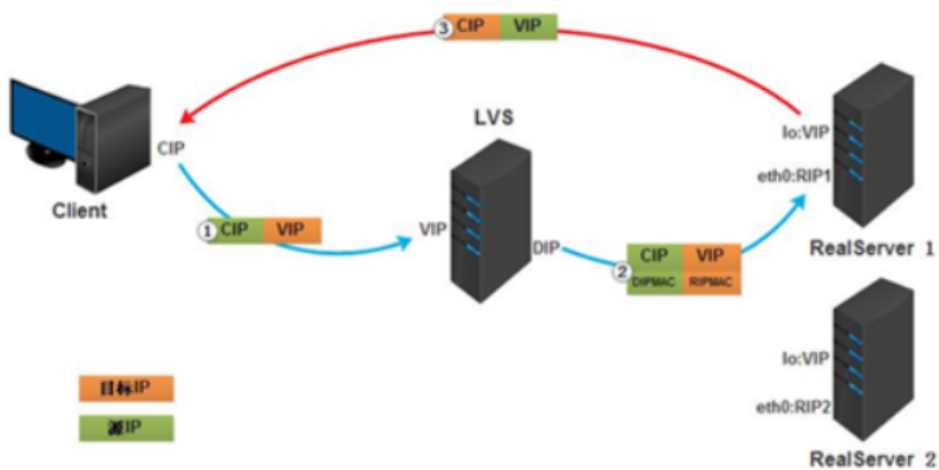
TUN：隧道模式，只有进站的数据流量经过分发器

full-nat:双向转换:通过请求报文的源地址为DIP，目标为RIP来实现转发：对于响应报文而言，修改源地址为VIP，目标地址为CIP来实现转发

LVS-NAT



LVS-DR



3、云服务器如何做高可用？物理服务器如何做高可用？

云服务器可以直接买高可用

参考: <https://www.cnblogs.com/onetwo/p/6007059.html>

物理服务器用keepalived

4、keepalived工作原理？keepalived会出现什么问题？

keepalived是以vrrp协议为基础实现的，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backup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模块是实现VRRP协议的。

脑裂问题：

用systemctl stop keepalived停不掉问题（可以去配置文件里改）

主备机器之间通信出了问题，就会发生脑裂

1. 因心跳线坏了
2. 高可用服务器对之间心跳线链路发生故障，导致无法正常通信
3. 因网卡及相关驱动坏了
4. 因心跳间连接的设备故障
5. 开启了iptables防火墙阻挡了心跳信息传输

解决方案：

1. 设置仲裁机制

当心跳线完全断开时，2个节点都各自ping一下参考ip，不通表明断点就出在本端，那就主动放弃竞争，让能够ping通的去接管

2. 同时使用串行电缆和以太网电缆连接。同时用两条心跳线路，这样一条坏了，依然还能传送心跳信息

vip不飘逸问题：

去查看设置的优先级，优先级设置应该差50

5、LVS、Nginx、HAProxy 有什么区别、优缺点？ 分别适合什么应用场景？

LVS：是基于四层的转发，通过LVS提供的负载均衡技术实现一个高性能、高可用的服务器群集，它具有良好可靠性、可扩展性和可操作性。从而以低廉的成本实现最优的服务性能。

HAProxy：是基于四层和七层的转发，是专业的代理服务器，是一款高性能的负载均衡软件。

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，也可以做四层（需要换http模块）

各自优缺点

LVS

优点：

- （1）高并发连接：**LVS**基于内核工作，有超强的承载能力和并发处理能力。单台**LVS**负载均衡器，可支持上百万并发连接。
- （2）稳定性强：是工作在网络4层之上仅作分发之用，这个特点也决定了它在负载均衡软件里的性能最强，稳定性最好，对内存和cpu资源消耗极低。
- （3）成本低廉：硬件负载均衡器少则十几万，多则几十万上百万，**LVS**只需一台服务器和就能免费部署使用，性价比极高。
- （4）配置简单：**LVS**配置非常简单，仅需几行命令即可完成配置，也可写成脚本进行管理。
- （5）支持多种算法：支持多种调算法，可根据业务场景灵活调配进行使用
- （6）支持多种工作模型：可根据业务场景，使用不同的工作模式来解决生产环境请求处理问题。
- （7）应用范围广：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、DNS、ftp服务等

缺点：

工作在4层，不支持7层规则修改，机制过于庞大，不适合小规模应用。

Nginx

优点：

- （1）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；
- （2）**Nginx**对网络的依赖非常小，理论上能ping通就能进行负载功能，这个也是它的优势所在；
- （3）**Nginx**安装和配置比较简单，测试起来比较方便；
- （4）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；
- （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；
- （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。

缺点：

- （1）**Nginx**不支持url来检测。
- （2）**Nginx**仅能支持http和Email，这个它的弱势。
- （3）**Nginx**的Session的保持，Cookie的引导能力相对欠缺。

HAProxy的优点:

- (1) HAProxy是支持虚拟主机的,可以工作在4、7层(支持多网段);
- (2) 能够补充Nginx的一些缺点比如Session的保持, Cookie的引导等工作;
- (3) 支持url检测后端的服务器;
- (4) 它跟LVS一样, 本身仅仅就只是一款负载均衡软件; 单纯从效率上来讲HAProxy更会比Nginx有更出色的负载均衡速度, 在并发处理上也是优于Nginx的;
- (5) HAProxy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, 不过在后端的MySQL slaves数量超过10台时性能不如LVS;
- (6) HAProxy的算法较多, 达到8种;

四种常用算法:

- 1.roundrobin: 轮询, 轮流分配到后端服务器;
- 2.static-rr: 根据后端服务器性能分配;
- 3.leastconn: 最小连接者优先处理;
- 4.source: 根据请求源IP, 与Nginx的IP_Hash类似。

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配置简单, 所以中小型企业推荐使用Haproxy

七、ansible自动化运维

1、自动化运维工具有什么? 及有什么模块?

1、Puppet特点

Puppet是早期的Linux自动化运维工具, 是一种Linux、WINDOWS、UNIX平台的集中配置管理系统, 到现在已经非常成熟, 可以批量管理远程服务器, 模块丰富, 配置复杂, 基于Ruby语言编写。是最典型的C/S结构, 需要安装服务端和客户端。采用C/S星状的结构, 所有的客户端和一个或者多个服务器交互, 每个客户端周期地(默认半个小时)向服务器发送请求, 获得最新的配置信息, 保证和配置信息同步。

2、SaltStack特点

和Puppet一样, 也是C/S模式, 需要安装服务端和客户端, 基于Python编写, 加入了MQ消息同步, 可以使执行命令和执行结果高效返回, 但其执行过程需要等待客户端全部返回, 如果客户端没有及时返回或者没有响应的话, 可能会导致部分机器没有执行结果。

3、Ansible特点

ansible是一种自动化运维工具, 基于python开发的, 并且基于模块化工作, 本身没有批量部署的能力, 真正具有批量部署的是ansible所运行的模块, ansible只是提供一种框架。不需要在远程主机上安装client/agents, 只需要在一台普通的服务器上运行即可, 因为它们是基于ssh来和远程主机通讯的

模块是Ansible执行的最小单位, 可以由Python编写, 也可以是Shell编写, 也可以是由其他语言编写。

1、远程复制备份模块: copy

- src=: 指定源文件路径 #这个路径是控制节点的路径
- dest=: 目标地址(拷贝到哪里) #这个路径是被控制节点的路径
- owner: 指定属主
- group: 指定属组
- mode: 指定权限, 可以以数字指定比如0644
- backup: 在覆盖之前将原文件备份, 备份文件包含时间信息。有两个选项: yes|no

2、软件包管理 yum模块

- name: 要安装软件的名字
- state: 状态
- latest: 表安装, 最新的
- removed: 表卸载
- absent: 用于remove安装包, 也指卸载

3、服务管理 service模块

- name: 要操作的软件的名字

state: 状态
started: 启动
stopped: 停止
restarted: 重启
enabled=yes: 设置为开机自启, 可与**state**一起使用
enabled=no: 设置开机不自启, 可与**state**一起使用

4、文件模块 **file**模块
owner:修改属主
group:修改属组
mode:修改权限
path=:要修改文件的路径 #这个路径是被控制节点的路径
recurse: 递归的设置文件的属性, 只对目录有效
 yes:表示使用递归设置
state:
 touch:创建一个新的空文件, 文件已存在仍创建文件
 directory:创建一个新的目录, 当目录存在时不会进行修改
 absent: 删除某个文件
 还有**link,hard**

5、收集信息模块**setup**
 ansible webserver1 -m setup #收集webserver1组的所有信息
 ansible webserver1 -m setup -a 'filter=ansible_all_ipv4_addresses' #只查询
 ipv4的地址, filter:过滤

6、**group**模块
 name: 必须参数, 用于指定组名称。
 state: 用于指定组的状态, 两个值可选, **present**, **absent**, 默认为**present**表示创建, 设置为**absent**表示删除组。
 gid: 用于指定组的**gid**。如果不指定为随机
 system:如果是**yes**为系统组。--可选

2、使用ansible写过的剧本有哪些?

ansible剧本类似于**shell**脚本 但是他有自己的语法规范

ansible剧本作用介绍

- (1) 可以实现服务自动部署
- (2) 可以提高运维工作效率
- (3) 可以较少运维工作问题(报错)
- (4) 可以节省公司运维成本

3、ansible有什么优势?

- (1) **ansible**比较适合做“一次性”的工作, 例如, 系统部署、应用发布、打补丁等
- (2) **ansible**基于**Python**开发, 集合了众多运维工具的优点, 实现了批量系统配置、批量程序部署、批量运行命令等
- (3) 不需要在被管控主机上安装任何客户端; 无服务器端, 使用时直接运行命令即可。
- (4) 无需代理不依赖**PKI** (无需**ssl**)
- (5) 可使用任何编程语言写模块

4、ansible基于哪些协议?

ssh

基于 **Python paramiko**开发, 分布式, 无需客户端, 轻量级, 配置语法使用**YAML**及**Jinja2**模板语言, 可以自动化部署**APP**; 自动化管理配置项; 自动化的持续交互; 自动化的(**AWS**)云服务管理。

paramiko是一个纯**Python**实现的**ssh**协议库。因此**fabric**和**ansible**还有一个共同点就是。不需要在远程主机上安装**client/agents**, 因为它们是基于**ssh**来和远程主机通讯的

八、zabbix监控

1、zabbix优缺点？

Zabbix 是一个企业级的、开源的、分布式的监控套件，被用来监控IT基础设施的可用性和性能。**Zabbix**可以监控网络和服务的监控状况，利用灵活的告警机制，允许用户对事件发送基于邮件、短信、微信和钉钉等告警方式，这样可以保证快速的对问题作出相应。支持主动和被动两种方式。所有的**Zabbix**报告都可以通过配置参数在WEB前端进行访问。**web**前端将帮助你在任何区域都能够迅速获得你的网络及服务状况。**zabbix**架构不仅支持小型组织，还支持大规模的公司的部署。

zabbix优点：

- (1) 开源,无软件成本投入
- (2) **Server** 对设备性能要求低
- (3) 支持设备多,自带多种监控模板
- (4) 支持分布式集中管理,有自动发现功能,可以实现自动化监控
- (5) 当监控的 **item** 比较多,服务器队列比较大时可以采用被动状态,被监控客户端主动从**server**端去下载需要监控的**item**然后取数据上传到**server**端。这种方式对服务器的负载比较小。
- (6) **Api**的支持,方便与其他系统结合

zabbix缺点：

需在被监控主机上安装 **agent**,所有数据都存在数据库里，产生的数据很大,瓶颈主要在数据库。

2、Zabbix监控的流程？还了解其他监控软件吗？

Zabbix监控过程是这样的：安装在主机上的**zabbix_agent**负责监控主机（具体的监控任务是由**agent**端的**Item**（监控项）来完成的），并收集数据，然后将数据发送到**zabbix server**端。如果是分布式系统，需要监控的机器较多，为了减轻**server**端的压力，可能中间还会再搭建一个**proxy**端，用来暂时接收监控数据，然后将数据转发到**server**端。**Server**端将数据存储到数据库中，**zabbix web**再将数据在前端以图表或者文字的形式展现出来。

目前使用比较多的服务器监控软件有这三款：**zabbix**、**cacti**、**nagios**

zabbix

zabbix是一个基于WEB界面的提供分布式系统监视以及网络监视功能的企业级的开源解决方案。能监视各种网络参数，保证服务器系统的安全运营；并提供灵活的通知机制以让系统管理员快速定位/解决存在的各种问题。

cacti

Cacti是一套基于PHP,MySQL,SNMP及RRDTool开发的网络流量监测图形分析工具。

nagios

Nagios是一款开源的免费网络监视工具，能有效监控**Windows**、**Linux**和**Unix**的主机状态，交换机路由器等网络设备，打印机等。在系统或服务状态异常时发出邮件或短信报警第一时间通知网站运维人员，在状态恢复后发出正常的邮件或短信通知。

3、如何用zabbix监控mysql主从复制是否失效？

在MySQL的从上查看从的运行状态是通过**Slave_IO_Running**线程和**Slave_SQL_Running**线程是否**yes**，通过命令“**show slave status\G;**”即可查看

<https://www.cnblogs.com/yanjieli/p/10996843.html>代码链接

4、zabbix中的功能组件有什么？

- 1) **zabbix server**: 负责接收agent发送的报告信息的核心组件, 所有配置、统计数据及操作数据都由它组织进行;
- 2) **database storage**(数据库存储): 专用于存储所有配置信息, 以及由zabbix收集的数据;
- 3) **web interface**(网络接口): zabbix的GUI接口; 通常与 **Server** 运行在同一台主机上;
- 4) **proxy**(代理): 可选组件, 常用于监控节点很多的分布式环境中, 代理server收集部分数据转发到server, 可以减轻server的压力;
- 5) **agent**: 部署在被监控的主机上, 负责收集主机本地数据如cpu、内存、数据库等数据发往server端或proxy端;

5、分布式监控与普通监控相比, 有什么优点?

由于zabbix是分布式监控, 可以回答zabbix的优点

6、分布式监控怎么部署?

采用zabbix proxy做为监控方案, 在每个节点部署zabbix proxy, 由zabbix proxy收集agentd数据, 然后将采集到的数据主动推送给zabbix server, zabbix server将数据存入数据库, 并在WEB前端显示。

7、设置报警邮件没有收到的原因?

报警类型选错了 没有权限 用户组没有更新

8、zabbix自定义监控监控过什么? (爱云校面试)

```
redis:
Redis.Status --检测Redis运行状态, 返回整数
Redis_conn --检测Redis成功连接数, 返回整数
Redis_rss_mem --检测Redis系统分配内存, 返回整数
Redis_lua_mem --检测Redis引擎消耗内存, 返回整数
Redis_cpu_sys --检测Redis主程序核心CPU消耗率, 返回整数
Redis_cpu_user --检测Redis主程序用户CPU消耗率, 返回整数
Redis_cpu_sys_cline --检测Redis后台核心CPU消耗率, 返回整数
Redis_cpu_user_cline --检测Redis后台用户CPU消耗率, 返回整数
Redis_keys_num --检测库键值数, 返回整数
Redis_loding --检测Redis持久化文件状态, 返回整数
```

nginx:
连接数

```
tcp:
tcp[TIMEWAIT] --检测TCP的驻留数, 返回整数
tcp[ESTAB] --检测tcp的连接数、返回整数
tcp[LISTEN] --检测TCP的监听数, 返回整数
tcp[TIMEWAIT] --检测TCP的驻留数, 返回整数
tcp[ESTAB] --检测tcp的连接数、返回整数
tcp[LISTEN] --检测TCP的监听数, 返回整数
```

系统:

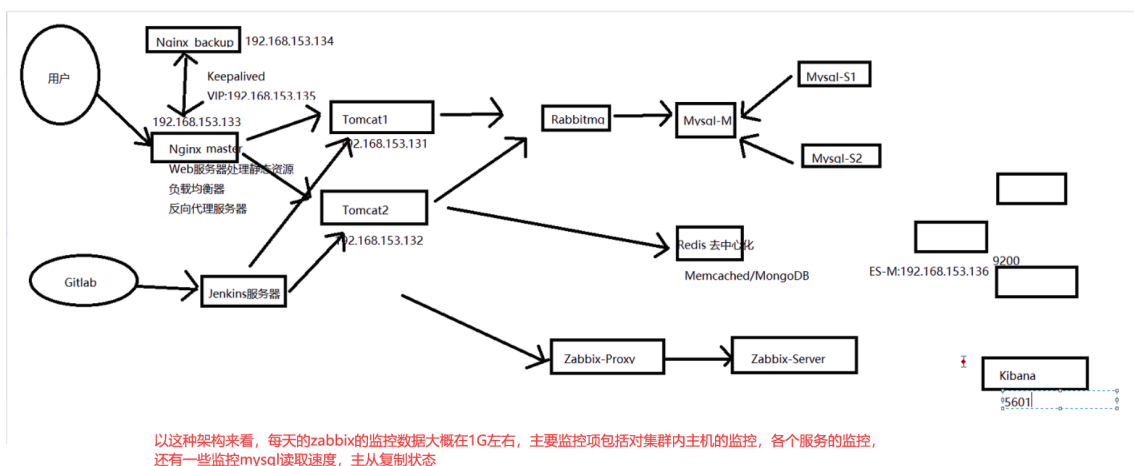
```
agent.ping 检测客户端可达性、返回nothing表示不可达。1表示可达
system.cpu.load --检测cpu负载。返回浮点数
system.cpu.util -- 检测cpu使用率。返回浮点数
vfs.dev.read -- 检测硬盘读取数据, 返回是sps.ops.bps浮点类型, 需要定义1024倍
vfs.dev.write -- 检测硬盘写入数据。返回是sps.ops.bps浮点类型, 需要定义1024倍
net.if.out[br0] --检测网卡流速、流出方向, 时间间隔为60S
net-if-in[br0] --检测网卡流速, 流入方向 (单位: 字节) 时间间隔60S
```


proc.num[] 目前系统中的进程总数，时间间隔60s
proc.num[,run] 目前正在运行的进程总数，时间间隔60s
system.cpu.switches --cpu的进程上下文切换，单位sps，表示每秒采样次数，api中参数history需指定为3
system.cpu.intr --cpu中断数量、api中参数history需指定为3
system.cpu.load[percpu,avg1] --cpu每分钟的负载值，按照核数做平均值(Processor load (1 min average per core)), api中参数history需指定为0
system.cpu.load[percpu,avg5] --cpu每5分钟的负载值，按照核数做平均值(Processor load (5 min average per core)), api中参数history需指定为0
system.cpu.load[percpu,avg15] --cpu每5分钟的负载值，按照核数做平均值(Processor load (15 min average per core)), api中参数history需指定为0
ram.info[Cached] --检测内存的缓存使用量、返回整数，需要定义1024倍
ram.info[MemFree] --检测内存的空余量，返回整数，需要定义1024倍
ram.info[Buffers] --检测内存的使用量，返回整数，需要定义1024倍

9、zabbix监控数据库读写速度？

server端开启Zabbix官方提供的监控mysql的模板Template App MySQL，由于并不能完全监控到指定内容，所以要结合shell脚本，更改agentd配置文件，添加用户参数
重启agentd端，在server端的web界面就可以看到
需要在数据库里创建一个有只读权限的用户

10、zabbix监控一天能产生多少数据量？



11、zabbix界面出现字符乱码如何处理？

linux直接更改字体
windows解决方案:Win+R打开运行，输入fonts，回车进入windows字体目录，找到微软雅黑-常规字体，复制出来将文件名修改为msyh.ttf，注意后缀ttf，将msyh.ttf上传到服务器zabbix字体目录中：/usr/share/zabbix/fonts/
,查看字体配置#grepFONT_NAME/usr/share/zabbix/include/defines.inc.php-n,执行快捷替换：
sed-i"s/graphfont/msyh/g"/usr/share/zabbix/include/defines.inc.php
确认是否替换成功:grepFONT_NAME/usr/share/zabbix/include/defines.inc.php-n
45:define('ZBX_GRAPH_FONT_NAME','msyh');//fontfilename
93:define('ZBX_FONT_NAME','msyh');
字体配置修改成功后，刷新图形界面即可看到图形字体显示正常了。

九、git

1、git与svn有什么区别？

1. GIT是分布式的没有中心代码库，所有机器之间的地位同等（每台机器上都有相同的代码），SVN不是，是集中式的，有中心代码库，其他都是客户端
2. GIT把内容按元数据方式存储，而SVN是按文件
3. GIT分支和SVN的分支不同，git可以通过简单的命令切换各个分支，可以用简单点的命令合并分支；而svn必须手动的合并，并且经常容易忽略一些分支
4. GIT没有一个全局的版本号，而SVN有
5. GIT的内容完整性要优于SVN

2、github与gitlab区别？

github 是公开的，但收费
gitlab 是内部代码仓库，免费

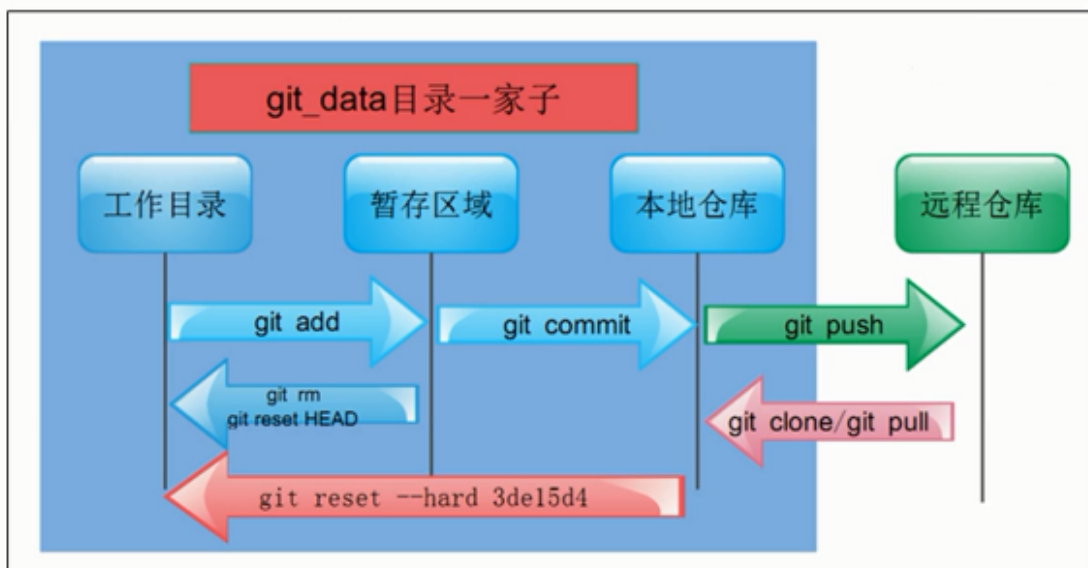
3、用什么中心代码仓库？代码有问题怎么办？谁负责推送代码？

gitlab,找开发

4、git常用的操作命令？

git add # 将工作区的修改提交到暂存区
git commit # 将暂存区的修改提交到当前分支
git push # 将本地代码更新到远程分支上
git reset # 回退到某一个版本
git stash # 保存某次修改
git pull # 从远程更新代码
git reflog # 查看历史命令
git status # 查看当前仓库的状态
git diff # 查看修改
git log # 查看提交历史
git revert # 回退某个修改

常用操作示意图



5、git是分布式的，svn是集中式的，分布和集中的区别？

集中式开发：

集中式开发是将项目集中存放在中央服务器中，在工作的时候，大家只在自己电脑上操作，从同一个地方下载最新版本，然后开始工作，做完的工作再提交给中央服务器保存。这种方式需要联网，现在云开发就是这样的处理方式。

优点：

减少了硬件和软件成本，硬件不用说了，现在流行盒子，一个小盒子只要连上中央服务器即可，以前都是一个个主机箱，那成本大多了。如果用到工具软件需要收费，只需买一套正版就OK了。

缺点：

1. 如果网络出现异常或者很卡，直接影响工作效率。如果是中央服务器挂了，那就集体喝茶去了。
2. 还有一种情况，各自电脑中操作的所有软件工具，都存放在一个中央服务器上（现在流行叫云服务器），只需要用各自电脑登陆连接到云服务器上（一般服务器都是用linux），比如用ps工具，大家其实用的是云服务器中的同一个ps 软件，在使用率高的情况下，ps会出现异常，当用ps筛选颜色的时候，已经混乱，无法正常选择颜色。
3. 安全度不高，重要的东西都放在一个中央服务器中，如果被黑，那损失就大了。

分布式开发：

只要提供一台电脑作为版本集中存放的服务器就够了，但这个服务器的作用仅仅是用来方便“交换”大家的修改，没有它也一样干活，只是交换修改不方便而已。而每一台电脑有各自独立的开发环境，不需要联网，本地直接运行，相对集中式安全系数高很多。

6、代码仓库中代码出现问题时回滚的方法？

已经提交了不合适的修改到版本库时，想要撤销本次提交，使用版本回退，不过前提是没有推送到远程库

查看现在的版本：`git log`

查看消失的版本：`git reflog`

回到上一个版本：`git reset --hard HEAD^` //HEAD^^回到为上上个版本

回到指定的版本：`git reset --hard 版本号`

7、gitlab支持版本回退吗？

支持

1. 查看提交记录，获得版本号

`git log`

2. 本地回退到相应的版本

注意使用 `--hard` 参数会抛弃当前工作区的修改

使用 `--soft` 参数的话会回退到之前的版本，但是保留当前工作区的修改，可以重新提交

`git reset --hard <版本号>`

3. 为了覆盖掉远端的版本信息，使远端的仓库也回退到相应的版本，需要加上参数`--force`

`git push origin <分支名> --force`

十、jenkins

1、Jenkins的工作流程原理(或者说CI/CD怎么做)？如果构建失败，原因有哪些？发布成功之后，功能有问题，如何处理？

工作流程图：

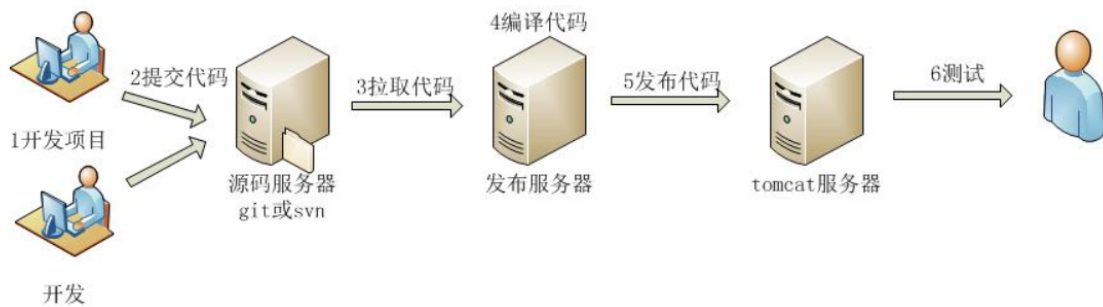
测试环境中：

1. 开发者会将代码上传到版本库中。
2. jenkins通过配置版本库的连接地址，获取到源代码。
3. jenkins获取到源代码之后通过参数化构建(或者触发器)开始编译打包。
4. jenkins通过调用maven（Ant或者Gradle）命令实现编译打包过程。
5. 生成的war包通过ssh插件上传到远程tomcat服务器中通过shell脚本自动发布项目。

生产环境：

测试环境将项目测试没问题后，将项目推送到线上正式环境。

1. 可以选择手动。
2. 也可以通过调用脚本推送过去。



构建失败的原因：

- 1、检查jenkins是否能连接。
- 2、检查代码在本地是否能编译通过。 如果代码编译不过即为代码本身问题，可以排除是jenkins的问题。
- 3、检查编译脚本是否正确。

发布成功后功能有问题的解决方法：仓库里回滚代码，或让开发重新推送代码重新发布

2、jenkins中CI/CD什么意思？

持续集成/持续发布

3、jenkins的脚本是放在后台服务器？jenkins中遇到什么问题？

是放在后台服务器

问题：

1. 缺少插件HTML Publisher plugin（构建后操作，以HTML格式输出构建报告）。

可到【系统管理】-【插件管理】-【可选插件】中搜索HTML Publisher plugin，选择直接安装

2. jenkins无法识别mvn指令

解决思路如下：

- (1) 确认服务器中是否安装了maven；
- (2) 确认shell中是否加载了配置文件。

maven确认安装，而我的shell脚本当中，第一行source /etc/profile也已经加载了配置文件，这是因为：jenkins默认情况下执行shell脚本是使用非登录方式，执行shell脚本时不会去加载/etc/profile。解决方法：

- 1) 修改环境变量
vi /etc/profile
PATH=\$JAVA_HOME/bin:/data/maven/bin:\$PATH
source /etc/profile （使得环境变量立即生效）
which mvn （查看是否能找到mvn命令）
- 2) 杀掉进程，重启jenkins 服务
ps -ef | grep jenkins
kill -9 进程号
- 3) 启动服务

```
/data/jenkins/bin/catalina.sh start
```

4、jenkins拉取代码时如何实现免密拉取？

配置公钥私钥

在jenkins中,系统设置-->全局工具配置里面进行配置git

下载git安装包,安装到/usr/local/git目录,如果不知道git的安装目录可以: `whereis git` 就可以得到路径。

然后在新建任务的时候在源码管理进行设置

添加账号密码

设置好之后,就是到服务器上面生成密钥了,

以阿里云服务器为例:

```
cd /root/.ssh/
```

如果目录存在就说明之前操作过,等下覆盖就是了;

```
ssh-keygen -t rsa
```

一直回车直到生成密钥;

生成的公钥在id_rsa.pub里面,复制出来添加git的公钥里面去就可以了!

然后可以到服务器里面的某个目录下进行测试,

先git init ,然后 git clone 项目的git地址

免密成功的话就会直接clone项目代码了

5、jenkins把war包发布过去之后还要做什么？

检查后端是否部署成功

十一、非关系型数据库

1、redis持久化方式？（飞哥+爱云校面试）

redis提供了两种持久化的方式，分别是RDB（Redis DataBase）和AOF（Append Only File）。

RDB: 是在不同的时间点，将redis存储的数据生成快照并存储到磁盘等介质上；

特点：

1. 周期性

2. 不影响数据写入 #RDB会启动子进程，备份所有数据。当前进程，继续提供数据的读写。当备份完成，才替换老的备份文件。

3. 高效 #一次性还原所有数据

4. 完整性较差 #故障点到上一次备份，之间的数据无法恢复。

AOF: 则是换了一个角度来实现持久化，那就是将redis执行过的所有写指令记录下来，在下次redis重新启动时，只要把这些写指令从前到后再重复执行一遍，就可以实现数据恢复了。

特点：

1. 实时性

2. 完整性较好

3. 体积大 #记录数据的指令，删除数据的指令都会被记录下来。

2、Redis相比较其他缓存数据库(例:mongodb、memcache)有什么优势？

(1) MongoDB源码是C++语言，redis源码是C语言。

(2) MongoDB文件存储是BSON格式类似JSON，或自定义的二进制格式。

(3) MongoDB性能都很依赖于内存的大小，MongoDB有丰富的数据表达、索引；最类似于关系数据库，支持丰富的查询语言，redis数据丰富，较少的IO，这方面mongodb优势明显。

(4) MongoDB不支持事务，靠客户端自身保证，redis支持事务，不过比较弱，仅能保证事务中的操作按顺序执行，这方面redis由于mongodb。

(5) mongodb对海量数据的访问效率提升，redis较小数据量的性能及运算，这方面MongoDB性能优于redis。mongodb有mapreduce功能，提供数据分析，redis没有，这方面mongodb优于redis。

redis具有以下特点：

- 1.丰富的数据结构：String,list,set,hash等数据结构的存储
- 2.支持持久化
- 3.支持事务
- 4.支持主从

redis和memcache比较

- 1).Redis不仅仅支持简单的k/v类型的数据,同时还提供了list(列表),set(字典),zset(集合),hash等数据结构的存储
- 2).Redis支持master-slave(主-从)模式应用
- 3).Redis支持数据的持久化

3、redis工作模式？各模式的使用场景、原理、特点及工作方式？

工作模式：

主从、哨兵、去中心化（Redis-Cluster集群）

主从：

使用场景：

主从结构，一是为了纯粹的冗余备份，二是为了提升读性能，比如很消耗性能的SORT就可以由从服务器来承担。

原理：

从服务器连接主服务器，发送SYNC命令；

主服务器接收到SYNC命令后，开始执行BGSAVE命令生成RDB文件并使用缓冲区记录此后执行的所有写命令；

主服务器BGSAVE执行完后，向所有从服务器发送快照文件，并在发送期间继续记录被执行的写命令；

从服务器收到快照文件后丢弃所有旧数据，载入收到的快照；

主服务器快照发送完毕后开始向从服务器发送缓冲区中的写命令；

从服务器完成对快照的载入，开始接收命令请求，并执行来自主服务器缓冲区的写命令；（从服务器初始化完成）

主服务器每执行一个写命令就会向从服务器发送相同的写命令，从服务器接收并执行收到的写命令（从服务器初始化完成后的操作）



主从：

优点：

- (1) 支持主从复制，主机会自动将数据同步到从机，可以进行读写分离，从而提高服务器性能
- (2) 一个Master可以同步多个Slaves
- (3) Slave同样可以接受其它Slaves的连接和同步请求，这样可以有效的分载Master的同步压力。因此我们可以将Redis的Replication架构视为图结构
- (4) Master Server是以非阻塞的方式为Slaves提供服务。所以在Master-Slave同步期间，客户端仍然可以提交查询或修改请求；Slave Server同样是以非阻塞的方式完成数据同步。在同步期间，如果有客户端提交查询请求，Redis则返回同步之前的数据。
- (5) 为了分载Master的读操作压力，slave服务器可以为客户端提供只读操作的服务，写服务仍然必须由Master来完成。即便如此，系统的伸缩性还是得到了很大的提高
- (6) Master可以将数据保存操作交给Slaves完成，从而避免了在Master中要有独立的进程来完成此操作

缺点：

- (1) Redis不具备自动容错和恢复功能，主机从机的宕机都会导致前端部分读写请求失败，需要等待机器重启或者手动切换前端的IP才能恢复。
- (2) 主机宕机，宕机前有部分数据未能及时同步到从机，切换IP后还会引入数据不一致的问题，降低了系统的可用性。
- (3) Redis较难支持在线扩容，在集群容量达到上限时在线扩容会变得很复杂。

工作方式：

如果设置了一个从服务器，在连接时它发送了一个SYNC命令，不管它是第一次连接还是再次连接都没有关系。然后主服务器开始后台存储，并且开始缓存新连接进来的修改数据的命令。当后台存储完成后，主服务器把数据文件发送到从服务器，从服务器将其保存在磁盘上，然后加载到内存中。然后主服务器把刚才缓存的命令发送到从服务器。这是作为命令流来完成的，并且和Redis协议本身格式相同。

你可以通过telnet自己尝试一下。在Redis服务器工作时连接到Redis端口，发送SYNC命令，会看到一个批量的传输，并且主服务器接收的每一个命令都会通过telnet会话重新发送一遍。

当主从服务器之间的连接由于某些原因断开时，从服务器可以自动进行重连接。当有多个从服务器同时请求同步时，主服务器只进行一个后台存储。

当连接断开又重新连上之后，一般都会进行一个完整的重新同步，但是从Redis 2.8开始，只重新同步一部分也可以。

哨兵：

使用场景：

redis集群需要监控主服务器和从服务器是否正常运行及为防止**master**出现故障而使整个集群瘫痪。

优点：

- (1) 哨兵模式是基于主从模式的，所有主从的优点，哨兵模式都具有。
- (2) 主从可以自动切换，系统更健壮，可用性更高。

缺点：

- (1) **Redis**较难支持在线扩容，在集群容量达到上限时在线扩容会变得很复杂。
- (2) 如果是从节点下线了，**sentinel**是不会对其进行故障转移的，连接从节点的客户端也无法获取到新的可用从节点

工作方式：

- (1) 每个**Sentinel**以每秒钟一次的频率向它所知的**Master**，**Slave**以及其他 **Sentinel** 实例发送一个 **PING** 命令
- (2) 如果一个实例 (**instance**) 距离最后一次有效回复 **PING** 命令的时间超过 **down-after-milliseconds** 选项所指定的值，则这个实例会被 **Sentinel** 标记为主观下线。
- (3) 如果一个**Master**被标记为主观下线，则正在监视这个**Master**的所有 **Sentinel** 要以每秒一次的频率确认**Master**的确进入了主观下线状态。
- (4) 当有足够数量的 **Sentinel** (大于等于配置文件指定的值) 在指定的时间范围内确认**Master**的确进入了主观下线状态， 则**Master**会被标记为客观下线。

去中心化 (**Redis-Cluster**集群)：

概念：

- (1) 由多个**Redis**服务器组成的分布式网络服务集群；
- (2) 集群之中有多个**Master**主节点，每一个主节点都可读可写；
- (3) 节点之间会互相通信，两两相连；
- (4) **Redis**集群无中心节点。

优点：

- (1) 当集群中的某个主节点下线时，集群中的其他在线主节点会注意到这一点，并对已下线的主节点进行故障转移。
- (2) 在集群里面，故障转移是由集群中其他在线的主节点负责进行的，所以集群不必另外使用**Redis Sentinel**。
- (3) 可扩展性：可线性扩展到 1000 多个节点，节点可动态添加或删除。
- (4) 高可用性：部分节点不可用时，集群仍可用。通过增加 **Slave** 做**standby** 数据副本，能够实现故障自动 **failover**，节点之间通过**gossip** 协议交换状态信息，用投票机制完成 **Slave**到**Master**的角色提升。

缺点：

- (1) 节点会因为某些原因发生阻塞。
- (2) 数据通过异步复制，不保证数据的强一致性。
- (3) 多个业务使用同一套集群时，无法根据统计区分冷热数据，资源隔离性较差，容易出现相互影响的情况。

工作方式：

在**redis**的每一个节点上，都有这么两个东西，一个是插槽 (**slot**)，它的取值范围是：0-16383。还有一个就是**cluster**，可以理解为是一个集群管理的插件。当我们的存取的**key**到达的时候，**redis**会根据**crc16**的算法得出一个结果，然后把结果对 16384 求余数，这样每个 **key** 都会对应一个编号在 0-16383 之间的哈希槽，通过这个值，去找到对应的插槽所对应的节点，然后直接自动跳转到这个对应的节点上进行存取操作。

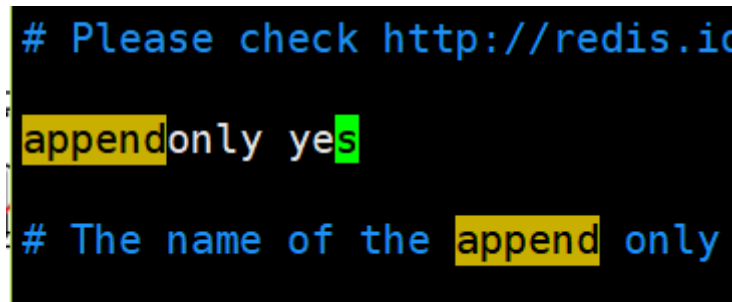
为了保证高可用，**redis-cluster**集群引入了主从模式，一个主节点对应一个或者多个从节点，当主节点宕机的时候，就会启用从节点。当其它主节点**ping**一个主节点**A**时，如果半数以上的主节点与**A**通信超时，那么认为主节点**A**宕机了。如果主节点**A**和它的从节点**A1**都宕机了，那么该集群就无法再提供服务了。

4、redis支持的并发量？

15万

5、redis如何开启持久化？

RDB默认开启
AOF默认关闭，开启如下：
[root@redis-master src]# cd ..
[root@redis-master redis]# vim redis.conf
修改如下：



```
# Please check http://redis.io
appendonly yes
# The name of the append only
```

6、redis数据备份怎么做？其中数据的异地备份怎么做？（爱云校面试）

Redis所有数据都是保存在内存中，Redis数据备份可以定期的通过异步方式保存到磁盘上，该方式称为半持久化模式，如果每一次数据变化都写入aof文件里面，则称为全持久化模式。同时还可以基于Redis主从复制实现Redis备份与恢复。

所谓异地备份，就是将服务器端的数据文件，拷贝到本地或者其他地方，以防止服务器崩溃丢失数据文件。

十二、iptables、firewalld防火墙

1、防火墙Iptables和Firewalld的区别？

1、firewalld可以动态修改单条规则，动态管理规则集，允许更新规则而不破坏现有会话和连接；而iptables，在修改了规则后必须得全部刷新才可以生效。
2、firewalld使用区域和服务而不是链式规则；而iptables基于四表五链。
3、firewalld默认是拒绝的，需要设置以后才能放行；而iptables默认是允许的，需要拒绝的才去限制。
4、firewalld自身并不具备防火墙的功能，而是和iptables一样需要通过内核的netfilter来实现。也就是说，firewalld和iptables一样，它们的作用都用于维护规则，而真正使用规则干活的是内核的netfilter。只不过firewalld和iptables的结果以及使用方法不一样！
firewalld是iptables的一个封装，可以让你更容易地管理iptables规则。它并不是iptables的替代品，虽然iptables命令仍可用于firewalld，但建议firewalld时仅使用firewalld命令。

2、iptables四表五链

四表:

raw -----追踪数据包, -----此表用处较少, 可以忽略不计
mangle --- 给数据包打标记, 做标记
nat -----网络地址转换即来源与目的的IP地址和port的转换。
filter ---做过滤的, 防火墙里面用的最多的表。用的最多

五链:

PREROUTING ---进路由之前数据包, 修改目标地址
INPUT -----就是过滤进来的数据包 (输入)
FORWARD -----转发
OUTPUT -----发出去的数据包
POSTROUTING --路由之后数据包, 修改源地址

3、firewalld有几个分区?

8个分区

trusted : 允许所有流量通过
home/internal: 仅允许ssh数据通过
work: 仅允许ssh,ipp-client,dhcpv6-client数据通过
public: 默认区域, 仅允许ssh,dhcpv6-client数据通过
external: 仅允许ssh数据通过, 通过该区域的数据将会伪装 (SNAT/DNAT)
dmz: 仅允许ssh数据通过
block: 任何传入的网络数据包都将被阻止。拒绝所有流量
drop: 拒绝所有流量, 没有返回回应消息

4、iptables、firewalld如何打开80端口?

```
# iptables -I INPUT -p tcp --dport 80 -j ACCEPT
```

常用参数:

-I: 在链的开头 (或指定序号) 插入一条规则
-A: 在链的末尾追加一条规则
-p: 协议; tcp用的最多, ssh用的这个协议、udp、icmp在ping的时候用的协议
-j: 控制类型; ACCEPT允许, REJECT拒绝, DROP丢弃
--dport: 目标端口
--sport: 源端口
-s: 源ip地址

```
# firewall-cmd --permanent --zone=public --add-port=80/tcp
```

```
# firewall-cmd --reload
```

常用参数:

--permanent: 永久生效的配置参数、资源、端口以及服务等信息

1、域zone相关的命令

--get-default-zone: 查询默认的区域名称
--set-default-zone=<区域名称>: 设置默认的区域
--get-active-zones: 显示当前正在使用的区域与网卡名称
--get-zones: 显示总共可用的区域

2、services管理的命令

--add-service=<服务名> --zone=<区域>: 设置指定区域允许该服务的流量
--remove-service=<服务名> --zone=<区域>: 设置指定区域不再允许该服务的流量

3、Port相关命令

--add-port=<端口号/协议> --zone=<区域>: 设置指定区域允许该端口的流量
--remove-port=<端口号/协议> --zone=<区域>: 设置指定区域不再允许该端口的流量

4、source管理的命令

--add-source=<ip地址/子网掩码> --zone=<区域>: 设置指定ip地址允许访问
--remove-source=<ip地址/子网掩码> --zone=<区域>: 设置指定ip地址不允许访问

5、查看所有规则的命令

--list-all --zone=<区域>: 显示指定区域的网卡配置参数、资源、端口以及服务等信息
--reload: 让“永久生效”的配置规则立即生效，并覆盖当前的配置规则

5、iptables的优势?

可以配置有状态的防火墙。有状态的防火墙能够指定并记住为发送或接收数据包所建立的连接状态。防火墙可以从数据包的连接跟踪状态获得该信息。在决定新的数据包过滤时，防火墙所使用的这些状态信息可以增加其效率和速度。有四种有效状态，分别为：**ESTABLISHED**（已建立的连接）、**INVALID**(非法或无法识别)、**NEW**(已经或将启动新的连接)和**RELATED**(正在启动新连接)。另一个优点：用户可以完全自己控制防火墙配置和数据包过滤，也可以定制自己的规则来满足特定的需求，从而允许想要的网络流量进入

6、iptables、firewalld分别基于哪一层?

iptables网络层 firewalld应用层

目前市面上比较常见的有3、4层的防火墙，叫网络层的防火墙，还有7层的防火墙，其实是代理层的网关。

对于TCP/IP的七层模型来讲，我们知道第三层是网络层，三层的防火墙会在这层对源地址和目标地址进行检测。但是对于七层的防火墙，不管你源端口或者目标端口，源地址或者目标地址是什么，都将对你所有的东西进行检查。所以，对于设计原理来讲，七层防火墙更加安全，但是这却带来了效率更低。所以市面上通常的防火墙方案，都是两者结合的。而又由于我们都需要从防火墙所控制的这个口来访问，所以防火墙的工作效率就成了用户能够访问数据多少的一个最重要的控制，配置的不好甚至有可能成为流量的瓶颈。

7、有IP恶意刷流量怎么办?

将对应ip禁掉 #iptables -t filter -A INPUT -s ip -p tcp --dport 80 -j REJECT
或者 #iptables -t filter -A INPUT -s ip -p tcp --dport 80 -j DROP

十三、ELK

1、zookeeper的作用，kafka与rabbitmq的区别？（飞哥+爱云校面试题）

zookeeper作用:

为分布式系统提供一致性服务，提供的功能包括：配置维护、分布式同步等。kafka的运行依赖Zookeeper，可以用来协调Kafka的各个broker，不仅可以实现broker的负载均衡，而且当增加了broker或者某个broker故障了，Zookeeper将会通知生产者和消费者，这样可以保证整个系统正常运转。

kafka与rabbitmq的区别:

1. 应用场景方面

RabbitMQ: 用于实时的，对可靠性要求较高的消息传递上。

kafka: 用于处于活跃的流式数据，大数据量的数据处理上。

2. 架构模型方面

RabbitMQ: 以broker为中心，有消息的确认机制

kafka: 以consumer为中心，无消息的确认机制

3. 吞吐量方面

RabbitMQ: 支持消息的可靠的传递，支持事务，不支持批量操作，存储可以采用内存或硬盘，吞吐量小。

kafka: 内部采用消息的批量处理，数据的存储和获取是本地磁盘顺序批量操作，消息处理的效率高，吞吐量高。

4. 集群负载均衡方面

RabbitMQ: 本身不支持负载均衡，需要loadbalancer的支持

kafka: 采用zookeeper对集群中的broker, consumer进行管理，可以注册topic到zookeeper上，通过zookeeper的协调机制，producer保存对应的topic的broker信息，可以随机或者轮询发送到broker上，producer可以基于语义指定分片，消息发送到broker的某个分片上。

2、ELK怎么工作的？公司需要用几台？

需要被收集日志的机器上安装logstash，由logstash负责收集日志，logstash将收集的日志推送给elasticsearch，由elasticsearch负责搜索、分析日志，最后交由kibana来展示

小公司三四台

收集日志

| 搜索，分析。

展示

需要被收集日志的机器上安装logstash，----->elasticSearch -----kibana -----客户机

3、ES单点故障怎么解决？

大体可以从以下几个方面来消除单点故障：

一个网站，从基础的硬件层，到操作系统层，到数据库层，到应用程序层，再到网络层，都有可能产生单点故障。如果要有效地消除单点故障，最重要的一点是设计的时候要尽量避免引入单点，随着架构的变化，定期审查系统潜在的单点也是有必要的。

磁盘镜像(Disk Mirroring)

为了避免磁盘驱动器发生故障而丢失数据，便增设了磁盘镜像功能。为实现该功能，须在同一磁盘控制器下，再增设一个完全相同的磁盘驱动器。当采用磁盘镜像方式时，在每次向主磁盘写入数据后，都需要将数据再写到备份磁盘上，使两个磁盘上具有完全相同的位像图。把备份磁盘看作是主磁盘的一面镜子。当主磁盘驱动器发生故障时，由于有备份磁盘的存在，在进行切换后，使主机仍能正常工作。磁盘镜像虽然实现了容错功能，却使磁盘的利用率降至50%，也未能使服务器的磁盘I/O速度得到提高。

独立磁盘冗余阵列 (RAID, redundant array of independent disks)

是把相同的数据存储在多个硬盘的不同的地方。通过把数据放在多个硬盘上，输入输出操作能以平衡的方式交叠，改良性能。因为多个硬盘增加了平均故障间隔时间 (MTBF)，储存冗余数据也增加了容错。

磁盘阵列其样式有三种，一是外接式磁盘阵列柜、二是内接式磁盘阵列卡，三是利用软件来仿真。

外接式磁盘阵列柜最常被使用大型服务器上，具有可热交换 (Hot Swap) 的特性，不过这类产品的价格都很贵。

内接式磁盘阵列卡，因为价格便宜，但需要较高的安装技术，适合技术人员使用操作。硬件阵列能够提供在线扩容、动态修改阵列级别、自动数据恢复、驱动器漫游、超高速缓冲等功能。它能提供性能、数据保护、可靠性、可用性和可管理性的解决方案。阵列卡专用的处理单元来进行操作。

利用软件仿真的方式，是指通过网络操作系统自身提供的磁盘管理功能将连接的普通SCSI卡上的多块硬盘配置成逻辑盘，组成阵列。软件阵列可以提供数据冗余功能，但是磁盘子系统的性能会有所降低，有的降低幅度还比较大，达30%左右。因此会拖累机器的速度，不适合大数据流量的服务器。

4、ELK为什么要收集日志？收集什么日志？收集日志失败的原因？

收集日志的原因：

- 1、方便问题排查；
- 2、可以实现监控和预警；
- 3、关联事件：多个数据源产生的日志进行联动分析，通过某种分析算法，就能够解决一些问题。
- 4、可以实现数据分析。

收集的日志：

数据库、nginx、tomcat等

失败的原因：

收集日志的服务还没有产生日志

5、filebeat与logstash的区别？

常见的日志采集工具有Logstash、Filebeat、Fluentd、Logagent、rsyslog等等

logstash 和**filebeat**都具有日志收集功能，**filebeat**更轻量，占用资源更少，**logstash**能过滤分析日志。一般结构都是**filebeat**采集日志，然后发送到消息队列，**redis**，**kafka**。然后**logstash**去获取，利用**filter**功能过滤分析，然后存储到**elasticsearch**中。**Logstash**是一个开源数据收集引擎，具有实时管道功能。**Logstash**可以动态地将来自不同数据源的数据统一起来，并将数据标准化到你所选择的目的地。

优势

Logstash主要的特点就是灵活性，主要因为它有很多插件，详细的文档以及直白的配置格式让它在多种场景下应用。我们基本上可以在网上找到很多资源，几乎可以处理任何问题。

劣势

Logstash致命的问题是它的性能以及资源消耗(默认的堆大小是**1GB**)。尽管它的性能在近几年已经有很大提升，与它的替代者们相比还是要慢很多的。另一个问题是它目前不支持缓存，目前的典型替代方案是将**Redis**或**kafka**作为中心缓冲池。

典型应用场景

因为**Logstash**自身的灵活性以及网络上丰富的资料，**Logstash**适用于原型验证阶段使用，或者解析非常复杂的时候。在不考虑服务器资源的情况下，如果服务器的性能足够好，我们也可以为每台服务器安装**Logstash**。我们也不需要使用缓冲，因为文件自身就有缓冲的行为，而 **Logstash** 也会记住上次处理的位置。如果服务器性能较差，并不推荐为每个服务器安装**Logstash**，这样就需要一个轻量的日志传输工具，将数据从服务器端经由一个或多个**Logstash**中心服务器传输到 **Elasticsearch**。

Filebeat

作为 **Beats** 家族的一员，**Filebeat**是一个轻量级的日志传输工具，它的存在正弥补了 **Logstash** 的缺点：**Filebeat** 作为一个轻量级的日志传输工具可以将日志推送到中心 **Logstash**。在版本 **5.x** 中，**Elasticsearch** 具有解析的能力(像 **Logstash** 过滤器)– **Ingest**。这也就意味着可以将数据直接用 **Filebeat** 推送到 **Elasticsearch**，并让 **Elasticsearch** 既做解析的事情，又做存储的事情。也不需要使用缓冲，因为 **Filebeat** 也会和 **Logstash** 一样记住上次读取的偏移，如果需要缓冲(例如，不希望将日志服务器的文件系统填满)，可以使用 **Redis/kafka**，因为 **Filebeat** 可以与它们进行通信。

优势

Filebeat只是一个二进制文件没有任何依赖。它占用资源极少，尽管它还十分年轻，正是因为它简单，所以几乎没有什么可以出错的地方，所以它的可靠性还是很高的。它为我们提供了很多可以调节的点，例如：它以何种方式搜索新的文件，以及当文件有一段时间没有发生变化时，何时选择关闭文件句柄。

劣势

Filebeat的应用范围十分有限，所以在某些场景下我们会碰到问题。例如，如果使用**Logstash**作为下游管道，我们同样会遇到性能问题。正因为如此，**Filebeat**的范围在扩大。开始时，它只能将日志发送到**Logstash**和**Elasticsearch**，而现在它可以将日志发送给**kafka**和**Redis**，在**5.x**版本中，它还具备过滤的能力。

典型应用场景

Filebeat在解决某些特定的问题时：日志存于文件，我们希望将日志直接传输存储到**Elasticsearch**。这仅在我们只是抓去(**grep**)它们或日志是存于**JSON**格式(**Filebeat**可以解析**JSON**)。或者如果打算使用**Elasticsearch**的**Ingest**功能对日志进行解析和丰富。

将日志发送到**kafka/Redis**。所以另外一个传输工具(例如，**Logstash** 或自定义的 **kafka** 消费者)可以进一步丰富和转发。这里假设选择的下游传输工具能够满足我们对功能和性能的要求。

6、ES集群中数据是如何存储的？怎么从logstash中读取？

ES是基于磁盘可用空间做数据分配的，在所有节点磁盘空间足够的情况下，**ES**会平均分配数据到**n**台机器通过**output**读取，**output**负责将数据输出到指定位置

7、EFK的工作流程？

Filebeat用于日志采集

Logstash对日志数据进行过滤和格式化（转成**JSON**格式），然后传给**Elasticsearch**

Elasticsearch对进行存储、建搜索的索引

kibana提供前端的页面再进行搜索和图表可视化，它是调用**Elasticsearch**的接口返回的数据进行可视化。

8、ES中的数据可以删吗？怎么删除？

可以

删除数据分为两种：一种是删除索引（数据和表结构同时删除，作用同MySQL中DROP TABLE "表名"），另一种是删除数据（不删除表结构，作用同MySQL中Delete语句）。

一：删除索引：

删除单个索引可以使用命令：Delete 索引名称

删除多个索引可以使用命令：DELETE 索引1，索引2

删除以某名开头的所有索引文件（配置文件中禁止后此方式不能使用）：Delete 索引名称*

删除全部索引命令：DELETE /_all 或DELETE /*（配置文件中禁止后此方式不能使用）

注意事项：对数据安全来说，能够使用单个命令来删除所有的数据可能会带来很可怕的后果，所以，为了避免大量删除，可以在elasticsearch.yml 配置文件中（或者动态配置中）修改

action.destructive_requires_name: true

设置之后只限于使用特定名称来删除索引，使用_all 或者通配符来删除索引无效（上述中说明配置文件中禁止后此方式不能使用）

二：删除数据：

1.根据主键删除数据：Delete 索引名称/文档名称/主键编号

2.根据匹配条件删除数据（注意请求方式是Post）

POST 索引名称/文档名称/_delete_by_query

```
{ "query": { "term": { "_id": "100000100" } } }
```

如果你想根据条件来删除你的数据，则在Query查询体中组织你的条件就可以了。

9、kibana要创建什么才能读取ES中的数据？

索引（用head插件看ES中的索引）

10、为什么要收集日志？

日志数据在以下几方面具有非常重要的作用：

- 数据查找：通过检索日志信息，定位相应的 bug ，找出解决方案
- 服务诊断：通过对日志信息进行统计、分析，了解服务器的负荷和服务运行状态
- 数据分析：可以做进一步的数据分析，比如根据请求中的课程 id ，找出 TOP10 用户感兴趣课程。

十四、RabbitMQ中间件

1、消息队列有什么作用？Rabbitmq有什么优势？

消息队列是一种应用程序对应用程序的通信方法。应用程序通过读写出入队列的消息（针对应用程序的数据）来通信，而无需专用连接来连接它们。消息传递指的是程序之间通过在消息中发送数据进行通信

Rabbitmq优势：

（1）解耦，比如说系统A会交给系统B去处理一些事情，通过将A，B中间加入消息队列，A将要处理的事情交给消息队列，B的输入来源于与消息队列

（2）有序性。先来先处理，比如一个系统处理某件事需要很长一段时间，但是在处理这件事情时候，有其他人也发出了请求，可以把请求放在消息队里，一个一个来处理

（3）消息路由：按照不同的规则，将队列中消息发送到不同的其他队列中

（4）异步处理：处理一件事情，需要甲先做A，然后做乙丙丁分别处理B C D，B C D这三件事情在A之后，但是相互之间没有关联。此时甲处理A1之后，把事件发送到消息队列里边，乙丙丁接受到事件之后分别处理B1 C1 D1。

（5）另外还具有可靠性、扩展性、高可用性、多种协议、多语言客户端、管理界面、插件机制

2、中间件有哪些？

RabbitMQ、Kafka、RocketMQ、ActiveMQ

RabbitMQ是使用Erlang语言开发的开源消息队列系统，基于AMQP协议来实现。AMQP的主要特征是面向消息、队列、路由（包括点对点和发布/订阅）、可靠性、安全。AMQP协议更多用在企业系统内对数据一致性、稳定性和可靠性要求很高的场景，对性能和吞吐量的要求还在其次。

Kafka是LinkedIn开源的分布式发布-订阅消息系统，目前归属于Apache顶级项目。Kafka主要特点是追求高吞吐量，一开始的目的就是用于日志收集和传输。0.8版本开始支持复制，不支持事务，对消息的重复、丢失、错误没有严格要求，适合产生大量日志数据的互联网服务的数据收集业务。

RocketMQ是阿里开源的消息中间件，它是纯Java开发，具有高吞吐量、高可用性、适合大规模分布式系统应用的特点。它对消息的可靠传输及事务性做了优化，目前在阿里集团被广泛应用于交易、充值、消息推送、日志流式处理、binglog分发等场景。

RabbitMQ比Kafka可靠，Kafka更适合IO高吞吐的处理，一般应用在大数据日志处理或对实时性（少量延迟），可靠性（少量丢数据）要求稍低的场景使用，比如ELK日志收集。

3、rabbitmq的工作模式及区别？

工作模式：

单节点、普通模式(默认的集群模式)、镜像

镜像模式(把需要的队列做成镜像队列，存在于多个节点，属于RabbitMQ的HA(高可用)方案，在对业务可靠性要求较高的场合中比较适合)。要实现镜像模式，需要先搭建出普通集群模式，在这个模式的基础上再配置镜像模式以实现高可用。

十五、docker容器

1、dockerfile中常用的命令是什么？RUN语句可以无限次用吗？为什么？（爱云校面试）

- (1) FROM：制作image时依据的基本image
- (2) RUN：制作image时执行的命令，一般在Dockerfile中多次出现
- (3) CMD：启动docker时执行的命令，在Dockerfile中只出现一次
- (4) ENV：设置环境变量
- (5) COPY：制作image时，将文件系统中的文件复制到Docker镜像中
- (6) WORKDIR：设置工作目录
- (7) EXPOSE：设置向外暴露的端口
- (8) VOLUME：设置容器与外界映射的目录

Dockerfile中的RUN是镜像创建阶段使用的命令，而docker run则是使用镜像启动容器阶段使用的命令。尽量减少一个Dockerfile中的RUN命令的个数。

RUN命令在构建时会创建一个新层，如非特殊的需要，建议一个Dockerfile在需要使用RUN命令的时候尽可能的只用一个RUN命令，将多条RUN命令进行合并可以有效降低构建的镜像的层数，从而可以降低大小。

2、Docker相比较传统虚拟机，有什么优点？

- (1) 表面区别：容器占用体积小，虚拟机占用体积大
- (2) 隔离性：容器提供了基于进程的隔离，而虚拟机提供了资源的完全隔离。容器的隔离性没有虚拟机的好
- (3) 启动速度：虚拟机可能需要一分钟来启动，而容器只需要一秒钟或更短。
- (4) 容器使用宿主操作系统的内核，而虚拟机使用独立的内核。Docker 的局限性之一是，它只能用在64位的操作系统上。
- (5) 本质区别：容器是被隔离的进程

3、docker的优势

1、交付物标准化

Docker的标准化交付物称为"镜像"，它包含了应用程序及其所依赖的运行环境，大大简化了应用交付的模式。

2、应用隔离

Docker可以隔离不同应用程序之间的相互影响，但是比虚拟机开销更小。总之，容器技术部署速度快，开发、测试更敏捷；提高系统利用率，降低资源成本。

3、一次构建，多次交付

类似于集装箱的"一次装箱，多次运输"，**Docker**镜像可以做到"一次构建，多次交付"。

Docker的度量：

Docker是利用容器来实现的一种轻量级的虚拟技术，从而在保证隔离性的同时达到节省资源的目的。

Docker的可移植性可以让它一次建立，到处运行。**Docker**的度量可以从以下四个方面进行：

1) 隔离性：通过内核的命名空间来实现的，将容器的进程、网络、消息、文件系统和主机名进行隔离。

2) 可度量性：**Docker**主要通过**cgroups**控制组来控制资源的度量和分配。

3) 移植性：**Docker**利用**AUFS**来实现对容器的快速更新。

AUFS是一种支持将不同目录挂载到同一个虚拟文件系统下的文件系统，支持对每个目录的读写权限管理。**AUFS**具有层

的概念，每一次修改都是在已有的只写层进行增量修改，修改的内容将形成新的文件层，不影响原有的层。

4) 安全性：安全性可以分为容器内部之间的安全性；容器与托管主机之间的安全性。

容器内部之间的安全性主要是通过命名空间和**cgroups**来保证的。

容器与托管主机之间的安全性主要是通过内核能力机制的控制，可以防止**Docker**非法入侵托管主机。

Docker容器使用**AUFS**作为文件系统，有如下优势：

1) 节省存储空间：多个容器可以共享同一个基础镜像存储。

2) 快速部署

3) 升级方便：升级一个基础镜像即可影响到所有基于它的容器。需要注意已经在运行的**docker**容器不受影响

4、Docker三大核心组件及三大组成要素

核心组件：镜像、仓库、容器

组成要素：名称空间、资源限制、文件系统

5、docker隔离方式

1. pid名字空间

不同用户的进程就是通过 **pid** 名字空间隔离开的，且不同名字空间中可以有相同 **pid**。所有的 **LXC** 进程在 **Docker**中的父进程为**Docker**进程，每个 **LXC** 进程具有不同的名字空间。同时由于允许嵌套，因此可以很方便的实现嵌套的 **Docker** 容器。

2. net名字空间 ----做网络接口隔离的

有了**pid** 名字空间，每个名字空间中的 **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** 名字空间实现的，每个 **net** 名字空间有独立的网络设备，**IP** 地址，路由表，**/proc/net** 目录。这样每个容器的网络就能隔离开来。

3. ipc名字空间

容器中进程交互还是采用了 **Linux** 常见的进程间交互方法(**interprocess communication - IPC**)， **包括信号量、消息队列和共享内存、**socket**、管道等。**

linux系统里面**ipc**通信有几种方式

socket:网络进程间的通信

管道:本地进程间的通信: **echo hello | grep e**

信号: **kill -9 PID** 这种我们叫信号量级，也是本地进程间的通信

共享内存:每个操作系统里面共享内存多大，是物理内存的一半

消息队列

4. mnt名字空间

mnt 名字空间允许不同名字空间的进程看到的文件结构不同，这样每个名字空间中的进程所看到的文件目录就被隔离开了。

5. uts名字空间

UTS("UNIX Time-sharing System") 名字空间允许每个容器拥有独立的 **hostname** 和 **domain name**，使其在网络上可以被视作一个独立的节点而非主机上的一个进程。

6. user名字空间

每个容器可以有不同的用户和组 **id**，也就是说可以在容器内用容器内部的用户执行程序而非主机上的用户。

6、docker如何解决不同节点的通信？

1、使用编排工具

2、使用GRE：通用路由协议封装

隧道技术(Tunneling)是一种通过使用互联网络的基础设施在网络之间传递数据的方式。使用隧道传递的数据(或负载)可以是不同协议的数据帧或包。

7、docker网络有什么？

docker安装后，默认会创建三种网络类型，bridge、host和none**

1、bridge:网络桥接相当于vmware的nat模式

默认情况下启动、创建容器都是用该模式，所以每次docker容器重启时会按照顺序获取对应ip地址。

2、none: 无指定网络，相当于vmware的host-only

启动容器时，可以通过--network=none,docker容器不会分配局域网ip

3、host: 主机网络

docker容器和主机共用一个ip地址。

使用host网络创建容器：

```
# docker run -it --name testnginx2 --net host 98ebf73ab
```

```
# netstat -lntp | grep 80
```

```
tcp6          0          0 :::80          :::*           LISTEN
```

```
3237/docker-proxy
```

浏览器访问宿主ip地址

4、固定ip

十六、kvm

1、KVM网络有什么？

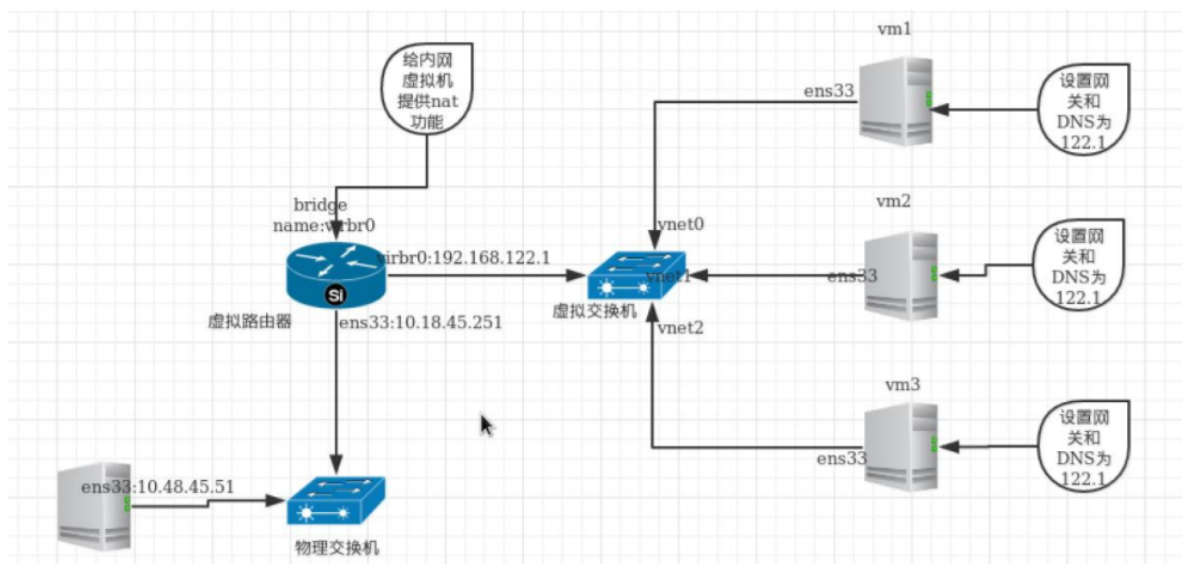
1. 隔离模式：虚拟机之间组建网络，该模式无法与宿主机通信，无法与其他网络通信，相当于虚拟机只是连接到一台交换机上。

2. 路由模式：相当于虚拟机连接到一台路由器上，由路由器(物理网卡)，统一转发，但是不会改变源地址。

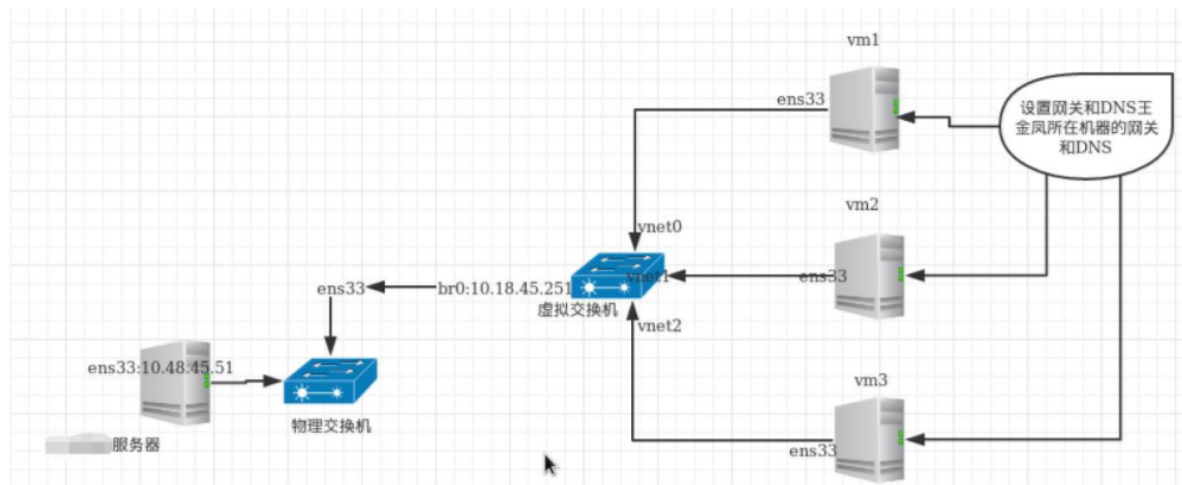
3. NAT模式：在路由模式中，会出现虚拟机可以访问其他主机，但是其他主机的报文无法到达虚拟机，而NAT模式则将源地址转换为路由器(物理网卡)地址，这样其他主机也知道报文来自那个主机，在docker环境中经常被使用。

4. 桥接模式：在宿主机中创建一张虚拟网卡作为宿主机的网卡，而物理网卡则作为交换机。

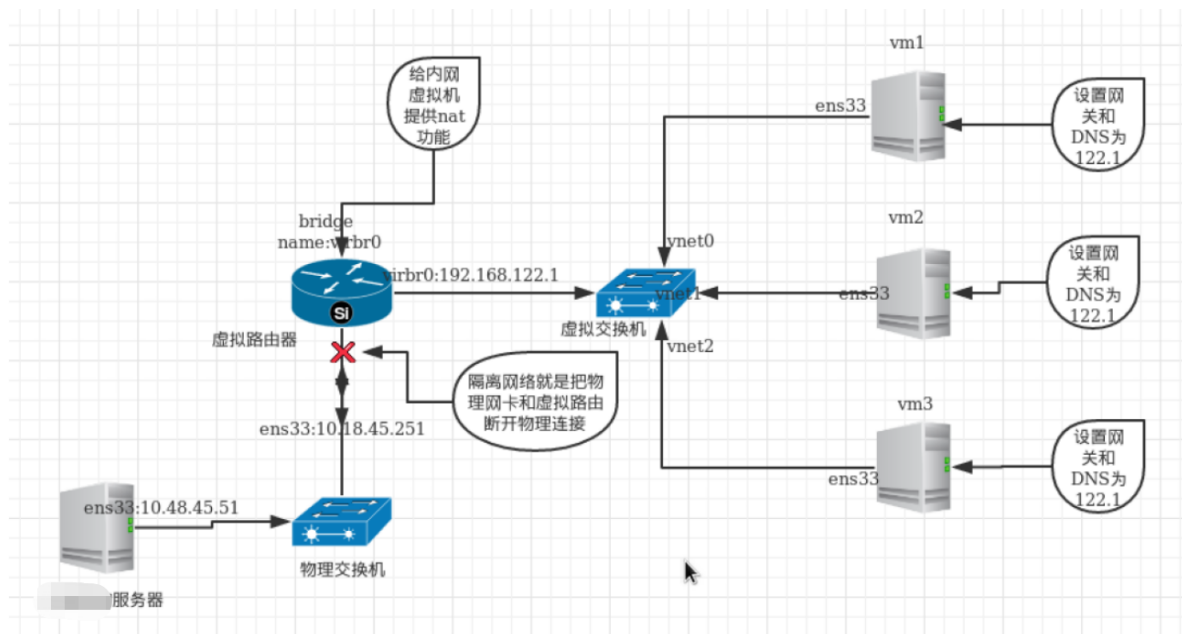
nat网络



桥接网络：



隔离网络：



2、kvm磁盘镜像文件格式raw与qcow2区别？

raw: 原始格式, 性能最好 直接占用你一开始给多少 系统就占多少 不支持快照
qcow: 先去网上了解一下**cow**(写时拷贝copy on write), 性能远不能和**raw**相比, 所以很快夭折了, 所以出现了**qcow2**(性能低下 早就被抛弃)
qcow2 性能上还是不如**raw**, 但是**raw**不支持快照, **qcow2**支持快照。
raw立刻分配空间, 不管你有没有用到那么多空间
qcow2只是承诺给你分配空间, 但是只有当你需要用空间的时候, 才会给你空间。最多只给你承诺空间的大小, 避免空间浪费

十七、ceph、gluster分布式存储

1、分布式存储系统的特性

1、可扩展

分布式存储系统可以扩展到几百台甚至几千台的集群规模, 而且随着集群规模的增长, 系统整体性能表现为线性增长。分布式存储的水平扩展有以下几个特性:

- 1) 节点扩展后, 旧数据会自动迁移到新节点, 实现负载均衡, 避免单点过热的情况出现;
- 2) 水平扩展只需要将新节点和原有集群连接到同一网络, 整个过程不会对业务造成影响;
- 3) 当节点被添加到集群, 集群系统的整体容量和性能也随之线性扩展, 此后新节点的资源就会被管理平台接管, 被用于分配或者回收。

2、低成本

分布式存储系统的自动容错、自动负载均衡机制使其可以构建在普通的PC机之上。另外, 线性扩展能力也使得增加、减少机器非常方便, 可以实现自动运维。

3、高性能

无论是针对整个集群还是单台服务器, 都要求分布式存储系统具备高性能。

4、易用

分布式存储系统需要能够提供易用的对外接口, 另外, 也要求具备完善的监控、运维工具, 并能够与其他系统集成。

5、易管理

可通过一个简单的WEB界面就可以对整个系统进行配置管理, 运维简便, 极低的管理成本。

分布式存储系统的挑战主要在于数据、状态信息的持久化, 要求在自动迁移、自动容错、并发读写的过程中保证数据的一致性。分布式存储涉及的技术主要来自两个领域: 分布式系统以及数据库。

2、文件系统有哪些? 分布式文件系统有哪些?

文件系统:

(1) **Ext3**: 是一款日志文件系统, 能够在系统异常宕机时避免文件系统资料丢失, 并能自动修复数据的不一致与错误。然而, 当硬盘容量较大时, 所需的修复时间也会很长, 而且也不能百分之百地保证资料不会丢失。它会把整个磁盘的每个写入动作的细节都预先记录下来, 以便在发生异常宕机后能回溯追踪到被中断的部分, 然后尝试进行修复。

(2) **Ext4**: **Ext3**的改进版本, 是**RHEL 6**系统中的默认文件管理系统, 它支持的存储容量高达**1EB(1073741824GB)**, 且能够有无限多的子目录。另外, **Ext4**文件系统能够批量分配**block**块, 从而极大地提高了读写效率。

(3) **XFS**: 是一种高性能的日志文件系统, 而且是**RHEL 7**中默认的文件管理系统, 它的优势在发生意外宕机后尤其明显, 可以快速地恢复可能被破坏的文件, 而且强大的日志功能只用花费极低的计算和存储性能。并且它最大可支持的存储容量为**18EB**, 这几乎满足了所有需求。

ext4与**xfs**比较:

初始化模式下, **ext4**性能并没有比**xfs**来得高
随机读写模式下, **ext4**性能比**xfs**将近高一倍
其他测试模式中, **ext4**和**xfs**性能相当

分布式文件系统:

(1) **HDFS**: **HDFS** (**Hadoop Distributed File System**) 是一个分布式文件系统,是**hadoop**生态系统的-一个重要组成部分,是**hadoop**中的的存储组件.**HDFS**是一个高度容错性的系统, **HDFS**能提供高吞吐量的数据访问,非常适合大规模数据集上的应用。

HDFS的优点:

1. 高容错性
数据自动保存多个副本
副本丢失后,自动恢复
2. 良好的数据访问机制
一次写入、多次读取,保证数据一致性
3. 适合大数据文件的存储
TB、 甚至**PB**级数据
扩展能力很强

HDFS的缺点:

1. 低延迟数据访问
难以应付毫秒级以下的-应用
2. 海量小文件存取
占用**NameNode**大量内存
3. 一个文件只能有一个写入者
仅支持**append**(追加)

(2) **GlusterFS**: 是一种全对称的开源分布式文件系统,所谓全对称是指**GlusterFS**采用弹性哈希算法,没有中心节点,所有节点全部平等。**GlusterFS**配置方便,稳定性好,可轻松达到**PB**级容量,数千个节点,2011年被红帽收购。基本类型: 条带, 复制, 哈希。

分布卷: 存储数据时, 将文件随机存储到各台**glusterfs**机器上。

优点: 存储数据时, 读取速度快

缺点: 一个**brick**坏掉, 文件就会丢失

复制卷: 存储数据时, 所有文件分别存储到每台**glusterfs**机器上。

优点: 对文件进行的多次备份, 一个**brick**坏掉, 文件不会丢失, 其他机器的**brick**上面有备份

缺点: 占用资源

条带卷: 存数据时, 一个文件分开存到每台**glusterfs**机器上

优点: 对大文件, 读写速度快

缺点: 一个**brick**坏掉, 文件就会坏掉

3、GPFS和HDFS有什么区别?

GPFS和**Hadoop**的**HDFS**系统对比, 它设计用于在商用硬件上存储类似或更大的数据

HDFS还将文件分割成块, 并将它们存储在不同的文件系统节点内。

HDFS对磁盘可靠性的依赖并不高, 它可以在不同的节点内存储块的副本。保存单一副本块的一个节点出现故障可以再复制该组其它有效块内的副本。相较而言, 虽然**GPFS**支持故障节点恢复, 但它是一个更严重的事件, 它可能包括数据(暂时性)丢失的高风险。

GPFS支持完整的**Posix**文件系统语义。**HDFS**和**GFS**(谷歌文件系统)并不支持完整的**Posix**语义。

GPFS跨文件系统分布它的目录索引和其它元数据。相反, **Hadoop**将它们保留在主要和次要**Namenode**中, 大型服务器必须在**RAM**内存储所有的索引信息。

GPFS将文件分割成小块。**Hadoop HDFS**喜欢**64MB**甚至更多的块, 因为这降低了**Namenode**的存储需求。小块或很多小的文件会快速填充文件系统的索引, 因此限制了文件系统的大小。

4、公有云对象存储

Amazon类似产品就是S3;

微软类似产品Azure Blob;

阿里云对象存储服务 (Object Storage Service, 简称 OSS), 是阿里云提供的海量、安全、低成本、高可靠的云存储服务

存储类型 (Storage Class)

OSS 提供标准、低频访问、归档三种存储类型, 其中标准存储类型提供高可靠、高可用、高性能的对象存储服务, 能够支持频繁的数据访问; 低频访问存储类型适合长期保存不经常访问的数据 (平均每月访问频率1到2次), 存储单价低于标准类型; 归档存储类型适合需要长期保存 (建议半年以上) 的归档数据, 在三种存储类型中单价最低。

应用场景

1、图片和音视频等应用的海量存储

OSS可用于图片、音视频、日志等海量文件的存储。

2、云端数据处理

上传文件到OSS后, 可以配合媒体处理服务和图片处理服务进行云端的数据处理。

3、网页或者移动应用的静态和动态资源分离

利用海量互联网带宽, OSS可以实现海量数据的互联网并发下载。

5、ceph的优势

Ceph主要设计的初衷是变成一个可避免单节点故障的分布式文件系统, PB级别的扩展能力, 而且是一种开源自由软件, 许多超融合分布式文件系统都是基于Ceph开发的。

Ceph是一个统一的分布式存储系统, 设计初衷是提供较好的性能、可靠性和可扩展性。

优势:

高扩展性: 使用普通x86服务器, 支持10~1000台服务器, 支持TB到EB级的扩展。

高可靠性: 没有单点故障, 多数据副本, 自动管理, 自动修复。

高性能: 数据分布均衡。

可用于对象存储, 块设备存储和文件系统存储

6、Ceph的基本组件

1、Osd

用于集群中所有数据与对象的存储。处理集群数据的复制、恢复、回填、再均衡。并向其他osd守护进程发送心跳, 然后向Mon提供一些监控信息。

当Ceph存储集群设定数据有两个副本时 (一共存两份), 则至少需要两个OSD守护进程即两个OSD节点, 集群才能达到active+clean状态。

2、MDS(可选)

为Ceph文件系统提供元数据计算、缓存与同步 (也就是说, Ceph 块设备和 Ceph 对象存储不使用MDS)。在ceph中, 元数据也是存储在osd节点中的, mds类似于元数据的代理缓存服务器。MDS进程并不是必须的进程, 只有需要使用CEPHFS时, 才需要配置MDS节点。

3、Monitor

监控整个集群的状态, 维护集群的cluster MAP二进制表, 保证集群数据的一致性。ClusterMAP描述了对象块存储的物理位置, 以及一个将设备聚合到物理位置的桶列表。

4、Manager (ceph-mgr)

用于收集ceph集群状态、运行指标, 比如存储利用率、当前性能指标和系统负载。对外提供ceph dashboard (ceph ui) 和 resetful api。Manager组件开启高可用时, 至少2个

十八、k8s

1、一键部署k8s的工具?

kubeadm

2、列举k8s的一些组件, 说明其作用?


```
(master):  
    api server 是k8s得入口  
    scheduler 为新建立得pod进行节点选择得，负责集群得资源调度  
    controller 负责执行各种控制器  
(node):  
    kubelet 负责管控容器  
    proxy 负责为pod创建代理服务器  
    etcd 数据库
```

3、create与apply的区别

create创建的应用如果需要修改yaml文件，必须先指定yaml文件删除，再创建新的pod。
如果是apply创建的应用可以直接修改yaml文件，继续apply创建，不用先删掉。

4、k8s常用操作命令？

```
kubectl get nodes #查看集群信息  
kubectl delete node node1 #删除节点  
kubectl get node kub-k8s-node1 -o yaml #-o 以yaml形式输出详细信息，.查看某一个节点(节点名称可以用空格隔开写多个)  
kubectl describe node node1 #查看node的详细信息，也可以查看pod的信息  
kubectl get service -n kube-system #查看service的信息，-n:namespace名称空间  
kubectl get pods --all-namespaces #get podspod/po都可以，查看所有名称空间内的资源  
kubectl cluster-info #查看主节点  
kubectl api-versions #查看api  
kubectl apply -f namespace.yaml #创建资源  
kubectl get namespace #查看资源  
kubectl get namespace ns-monitor #查看某一个namespace  
kubectl describe namespace ns-monitor #查看某个namespace的详细信息  
kubectl delete -f namespace.yaml #删除名称空间  
kubectl get pods -n default#查看pod，-n 查看名称空间  
kubectl get pods -o wide #查看pod运行在哪台机器上  
kubectl exec -it website /bin/bash #进入Pod对应的容器内部  
kubectl delete pod pod名1 pod名2 #单个或多个删除pod
```

十九、python

1、Python的类模块？

模块分为三类：内置模块；第三方；自定义；

二十、安全

1、服务器中挖矿病毒怎么解决？

解决方法：

找到病毒文件并将其删除；中毒之后的机器会出现CPU、内存使用率会比较高，不断向外发包等异常情况。

排查方法：

Linux服务器流量剧增，使用iftop查看是否有外网连接的情况

netstat 查看连接外网的ip和端口是否有问题

top找到CPU使用率高的进程，一般病毒文件命名会比较乱

ps aux 查看是否有不明进程，找出病毒文件的位置

rm -f 删除病毒文件

检查：

对计划任务、开启启动项和病毒文件目录有无其他可疑文件

chkconfig --list|grep 3:on

开机自启动文件：more /etc/rc.local

2、如何防御CC攻击？什么是CC攻击？CC攻击危害是什么？

CC是一种利用肉鸡模仿用户大量访问你网站，从而占用你IIS的一种攻击方式，如果规模较小，可以通过重启服务器的方式解决，如果攻击量较大，需要做一些安全策略来过滤伪装用户的肉鸡，甚至可以通过输入验证码的方式来避免非正常用户的访问。

解决方法：

1) 开启防火墙，过滤掉访问次数最多的IP地址

2) 拒绝代理服务器访问你的服务器

拒绝代理服务器访问的方法：

代理服务器有固定的IP地址，将这些IP地址都加到防火墙规则下，全都drop掉

危害：

大量的流量不断冲击你的服务器，会让你的负载及压力越来越大，直到服务器崩溃宕机。

3、Linux服务器被入侵的症状以及怎么做？

被入侵后的服务器会消耗非常高的资源，尤其是CPU等，可以用此机器挖矿、发送垃圾邮件、消耗宽带发动DNS攻击

表现：服务器变慢，跑的网页的话（运行着web用服务）网站打开缓慢

二十一、工作经验问题

1、在之前的运维工作中你遇到过哪些运维故障？是怎么解决的？

1、问题描述：服务器tomcat重启之后，网站很长一段时间无法访问，然后过几分钟就可以访问了。（admin启动成功后api启动日志不更新；端口8005启动慢；启动后发现日志卡在这里启动事件卡在这里十几分钟）使用命令netstat -tulnp | grep java发现是8005端口的问题，起来很慢，日志里面又没有什么报错信息。

原因：Tomcat启动过程很慢，JVM上的随机数与熵池策略耗时：session引起的随机数问题导致的。熵池中随机数耗尽，随机数产生器会手机来自设备驱动器和其它源的环境噪声数据，并放入熵池中。产生器会评估熵池中的噪声数据的数量。当熵池为空时，这个噪声数据的收集是比较花时间的。这就意味着，Tomcat在生产环境中使用熵池时，会被阻塞较长的时间。

解决方案： 修改\$JAVA_HOME/jre/lib/security/java.security文件中securerandom.source配置项：

```
#securerandom.source=file:/dev/urandom 改为
securerandom.source=file:/dev/./urandom
```

最后重启tomcat发现速度快了。

2、Tomcat每周六重启失败

问题背景：生产环境上面的tomcat每周六重启，但是重启失败

原因：经过调查发现是因为一个tomcat产生了多个java进程，所以执行自带脚本shutdown.sh的时候没有把tomcat进程完全杀死。

解决方案：将杀死tomcat进程脚本放在重启脚本的前面执行

```
592**6/app/bin/kill.sh>>/app/log/backup/kill.log2&1
```

```
013**6/app/bin/restart_tomcat.sh>>/app/log/backup/restart.log2&1
```

2、公司的架构,服务器的数量,之前公司运维部的人员数量,之前工作流程

服务器数量五六十台,运维人数2人,运维人员对公司互联网业务所依赖的基础设施、基础服务、线上业务进行稳定性加强，进行日常巡检发现服务可能存在的隐患，对整体架构进行优化以屏蔽常见的运行故障，多数据中接入提高业务的容灾能力。通过监控、日志分析等技术手段，及时发现和响应服务故障，减少服务中断的时间，使公司的互联网业务符合预期的可用性要求，持续稳定地为用户提供务。

时间及大小问题：

1、jenkins一次构建大约需要半小时

2、物理备份中一次完备大约20多分钟，前提数据量小

3、打卡导出sql语句一天2000条，四五个月大约200M

4、zabbix一天产生多少监控数据？1000个监控项90天大约五六个G

5、nginx一天的访问量：PV，UV

1) .根据访问IP统计UV

```
awk '{print $1}' access.log|sort | uniq -c |wc -l
```

2) .统计访问URL统计PV

```
awk '{print $7}' access.log|wc -l
```

3) .查询访问最频繁的URL

```
awk '{print $7}' access.log|sort | uniq -c |sort -n -k 1 -r|more
```

4) .查询访问最频繁的IP

```
awk '{print $1}' access.log|sort | uniq -c |sort -n -k 1 -r|more
```

5) .根据时间段统计查看日志

```
cat access.log| sed -n '/14\Mar\2018:21/,/14\Mar\2018:22/p'|more
```

也可用zabbix进行监控

补充：

1、ubuntu的内核比centos内核更稳定

2、搭建kafka集群需要解析，若没有解析则成不了

3、消息队列缓解数据库的写压力