**Linux服务面试题**

**你对现在运维工程师的理解和以及对其工作的认识？**

答案：

运维工程师在公司当中责任重大，需要保证时刻为公司及客户提供最高、最快、最稳定、最安全的服务.运维工程师的一个小小的失误，很有可能会对公司及客户造成重大损失，因此运维工程师的工作需要严谨及富有创新精神。

**Nginx出现403，怎么解决？**

答案：

403报错是权限拒绝，查看错误日志找对应的文件，授予相应权限

**HTTP状态码有哪些，分别是什么含义？**

答案：

以下是HTTP状态码的主要分类及其代表的意义：

1xx - 信息性响应（Informational Responses）

100 Continue：请求者应当继续提出请求。服务器已收到部分请求，但希望接收到剩余部分后再处理。

101 Switching Protocols：服务器正在根据客户端的请求切换协议。

2xx - 成功状态码（Successful Status Codes）

200 OK：请求已成功处理，请求所希望的响应头或数据体将随此响应返回。

201 Created：请求已被成功处理，并且创建了一个新的资源。

202 Accepted：请求已被接受处理，但尚未完成。

204 No Content：请求已经成功处理，但没有返回任何内容，通常用于只需要客户端知道请求已成功的情况。

3xx - 重定向状态码（Redirection Status Codes）

301 Moved Permanently：永久性重定向，请求的资源已被分配了新的URL。

302 Found / 302 Moved Temporarily：临时重定向，请求的资源临时移动到了新的URL。

303 See Other：请求应该以GET方法定向获取请求资源的新位置。

307 Temporary Redirect：临时重定向，请求的资源临时位于给定的URL下，与302类似，但要求浏览器保持原始请求的方法不变。

4xx - 客户端错误状态码（Client Error Status Codes）

400 Bad Request：客户端请求语法错误，无法被服务器理解。

401 Unauthorized：请求需要用户身份验证。

403 Forbidden：服务器理解请求但拒绝执行，通常由于权限不足。

404 Not Found：服务器找不到请求的资源。

405 Method Not Allowed：客户端请求的方法不被允许。

5xx - 服务器错误状态码（Server Error Status Codes）

500 Internal Server Error：服务器遇到了未知错误，无法完成请求。

501 Not Implemented：服务器不支持请求的功能，无法完成请求。

502 Bad Gateway：作为网关或代理服务器，从上游服务器接收到无效响应。

503 Service Unavailable：服务器暂时无法处理请求，可能是过载或维护中。

504 Gateway Timeout：作为网关或代理服务器，在规定的时间内未从上游服务器收到响应。

**什么是网站流量？**  
答案：

网站流量（traffic）是指网站的访问量，是用来描述访问一个网站的用户数量以及用户所浏览的网页数量等指标，常用的统计指标包括网站的独立用户数量、总用户数量（含重复访问者）、网页浏览数量、每个用户的页面浏览数量、用户在网站的平均停留时间等。

**什么叫网站灰度发布?**

答案：

网站灰度发布，也称为灰度部署、灰度发布测试、金丝雀发布，是一种软件发布策略，旨在逐步、可控地将新版本的网站功能或更新推送给部分真实用户，以便在全面上线之前对新版本进行验证、收集反馈、评估性能和兼容性，并在必要时对潜在问题进行快速调整和修复，而不影响整个用户群体的正常使用。

**网站访问相应速度慢的原因有哪些？**

答案：

服务器性能问题：网站运行的服务器性能直接影响着网站的响应时间，如果服务器硬件过时、资源分配不合理，或者服务器负载过高，都可能导致网站处理请求的速度变慢。

网络传输问题：数据从服务器传输到用户的浏览器需要经过多个网络节点，任何一个节点的问题都可能引起延迟。

网站设计与编码问题：即使服务器和网络状况良好，网站本身的设计和编码不佳也会导致加载速度变慢。可以使用网站速度测试工具，如Google PageSpeed Insights、GTmetrix等，它们可以提供关于网站性能的详细报告。

数据库查询效率低：动态网站经常需要从数据库中检索信息，如果数据库查询效率低，就会增加页面加载时间。

**nginx实现虚拟主机的三种方式分别是什么？**

答案：

使用不同 IP 地址；

相同 IP 地址，不同端口号；

相同 IP 地址及端口号，不同主机名。

**现在给你三百台服务器，你怎么对他们进行管理？**

答案：

管理3百台服务器的方式：

1）设定跳板机，使用统一账号登录，便于安全与登录的考量。

2）使用 salt、ansiable、puppet 进行系统的统一调度与配置的统一管理。

3）建立简单的服务器的系统、配置、应用的 cmdb 信息管理。便于查阅每台服务器上的各种信息记录。

**LVS、Nginx、HAproxy 有什么区别？工作中你怎么选择？**

答案：

LVS： 是基于四层的转发

HAproxy： 是基于四层和七层的转发，是专业的代理服务器

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

区别：LVS由于是基于四层的转发所以只能做端口的转发，而基于URL的、基于目录的这种转发LVS就做不了。

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

对于超大规模、极高并发的网络环境，优先考虑LVS，其内核级处理能力和无与伦比的性能非常适合。

对于中小型项目或并发量适中的场景，Nginx或HAProxy的性能通常足以满足需求，且功能更为全面。

**你常用的 Nginx 模块，用来做什么?**

答案：

rewrite 模块，实现重写功能

access 模块：来源控制

ssl 模块：安全加密

ngx\_http\_gzip\_module：网络传输压缩模块

ngx\_http\_proxy\_module 模块实现代理

ngx\_http\_upstream\_module 模块实现定义后端服务器列表

ngx\_cache\_purge 实现缓存清除功能

**简述一下Tomcat8005、8009、8080三个端口的含义?**

答案：

8005端口： 这个端口主要用于Tomcat服务器的关闭控制。当您需要停止单独运行的Tomcat实例时，可以通过向此端口发送特定的“SHUTDOWN”命令来优雅地关闭服务器。

8009端口： 这个端口与Tomcat的AJP协议相关联。AJP（Apache JServ Protocol）是一种高效的二进制协议，设计用于在Web服务器（如Apache HTTP Server）与Servlet容器（如Tomcat）之间进行通信。当您希望配置Apache HTTP Server作为前端代理，通过AJP协议将动态内容请求转发给后端的Tomcat服务器处理时，Apache会连接到Tomcat的8009端口。

8080端口： 8080端口是Tomcat默认的HTTP服务端口。

**简述 DNS 进行域名解析的过程？**

答案：

用户要访问 www.baidu.com，会先找本机的 host 文件，再找本地设置的 DNS 服务器，如果也没有的话，就去网络中找根服务器，根服务器反馈结果，说只能提供一级域名服务器.cn，就去找一级域名服务器，一级域名服务器说只能提供二级域名服务器.com.cn，就去找二级域名服务器。

二级域服务器只能提供三级域名服务器.baidu.com.cn，就去找三级域名服务器，三级域名服务器正好有这个网站www.baidu.com，然后发给请求的服务器，保存一份之后，再发给客户端。

**简述一下 A 记录与 NS 记录的区别？**

答案：

A 记录是名称解析的重要记录，它用于将特定的主机名映射到对应主机的 IP 地址上。你可以在 DNS 服务器中手动创建或通过 DNS 客户端动态更新来创建。

NS 记录此记录指定负责此 DNS 区域的权威名称服务器。

A 记录和 NS 记录的区别是，A 记录直接给出目的 IP，NS 记录将 DNS 解析任务交给特定的服务器，NS 记录中记录的 IP 即为该特定服务器的 IP 地址。

NS 记录优先于 A 记录，A 记录优先于 CNAME 记录。

**RabbitMQ 是什么东西？**

答案：

RabbitMQ 也就是消息队列中间件，消息中间件是在消息的传息过程中保存消息的容器，消息中间件再将消息从它的源中到它的目标中标时充当中间人的作用。队列的主要目的是提供路由并保证消息的传递；如果发送消息时接收者不可用，消息队列不会保留消息，直到可以成功地传递为止，当然，消息队列保存消息也是有期限地。

**讲一下Keepalived的工作原理?**

答案：

Keepalived 是一个开源的高可用（High Availability, HA）软件，它结合了VRRP（Virtual Router Redundancy Protocol，虚拟路由冗余协议）和LVS（Linux Virtual Server，Linux虚拟服务器）的功能，旨在为网络服务提供高可用性和负载均衡能力。

**Nginx 如何处理静态内容和动态内容？**

答案:

Nginx 本身是非常擅长处理静态内容的，可以直接将静态文件（如 HTML、CSS、JavaScript、图片等）直接作为 HTTP 响应返回给客户端。对于动态内容，Nginx 通常会配置为反向代理，将请求转发给后端应用服务器（如 Apache、Tomcat 或者应用程序框架），由后端服务器处理动态请求并生成内容。

--------------------------------停止

**LVS有三种负载均衡的模式？**

答案：

1. NAT（Network Address Translation）模式

工作原理：

在NAT模式下，LVS调度器（Director）作为客户端请求的入口，拥有一个公网IP地址（VIP，Virtual IP Address）。当客户端向VIP发起请求时，调度器会修改数据包的IP地址和端口信息，将源IP改为调度器自身的公网IP，将目的IP改为选定的真实服务器（Real Server，RS）的私有IP地址。

RS接收到请求后处理并直接将响应数据包发送回调度器。调度器再将响应数据包的源IP和端口改回客户端的IP和端口，然后将响应发送给客户端。客户端全程感知不到RS的存在，只看到与VIP之间的通信。

优点：

实现简单，只需要在调度器上配置NAT规则。

不要求RS具备公网IP，适用于RS在内网部署的场景。

缺点：

所有进出数据包都需要经过调度器，增加了调度器的网络带宽压力。

RS响应数据包返回调度器时，由于源IP被改为了调度器的IP，RS无法直接根据源IP进行会话保持，可能需要额外的会话保持机制（如基于cookie）。

由于NAT转换，网络监控和诊断可能会变得复杂。

2. DR（Direct Routing）模式

工作原理：

在DR模式下，调度器和RS共享同一VIP。调度器通过修改请求数据包的目标MAC地址，将数据包直接转发给选定的RS，而不是修改IP地址。RS收到请求后，以其自身的IP（VIP）作为源IP直接响应客户端，客户端看到的始终是与VIP的通信。

为了实现这一点，调度器需要在接收到ARP请求（询问VIP对应的MAC地址）时，代替RS响应，提供自己的MAC地址。而RS则需要配置为对VIP的ARP请求保持静默，不响应ARP。

优点：

数据包在调度器与RS之间直接传输，无需进行IP地址转换，减少了网络开销。

RS可以直接响应客户端，利于会话保持和网络监控。

适用于局域网内RS与调度器在同一物理网络且支持MAC地址学习的环境。

缺点：

RS必须与调度器位于同一物理网络，且网络设备（如交换机）需要支持MAC地址漂移（即同一IP对应不同MAC地址的变化）。

需要对RS进行特殊配置，使其不对VIP的ARP请求作出响应。

3. TUN（IP Tunneling）模式

工作原理：

TUN模式下，调度器与RS之间建立IP隧道。客户端请求到达调度器后，调度器封装一个新的IP包，源IP为客户端IP，目的IP为RS的内网IP。RS收到封装后的数据包后，解封并处理原始请求，然后直接将响应数据包发送给客户端，客户端IP地址作为响应的目的IP。

RS无需配置VIP，仅需配置与调度器之间的隧道接口。

优点：

RS无需公网IP，也不必与调度器在同一物理网络，适用于跨广域网部署的RS。

与DR模式类似，RS可以直接响应客户端，便于会话保持和网络监控。

缺点：

需要在调度器与RS之间配置并维护IP隧道，增加了网络配置复杂度。

数据包封装和解封装过程带来一定CPU开销。

总结起来，LVS的三种模式各有优缺点，适用于不同的网络环境和需求。选择哪种模式取决于网络拓扑、服务器位置、带宽要求、会话保持需求等因素。在实际应用中，可以根据具体情况灵活选择或组合使用。

**解释 Nginx 中的 Location 指令。**

答案:

location 指令用于定义 Nginx 如何响应特定的请求URI。location 指令可以匹配请求的URI，并根据不同的匹配结果，应用不同的配置，如代理设置、重写规则或返回不同的内容。

**Nginx负载均衡的算法有哪些？**

答案：

Nginx支持多种负载均衡算法，如：

轮询（Round Robin）

加权轮询（Weighted Round Robin）

最少连接（Least Connections）

IP哈希（IP Hash）

**如何优化 Nginx 的性能？**

答案：

优化 Nginx 的性能可以从多个方面进行：

调整 worker\_processes 和 worker\_connections 来充分利用系统资源。

开启 gzip 压缩来减少网络传输数据量。

使用缓存静态文件，减少对后端服务器的请求。

使用 keepalive 连接来减少 TCP 握手的开销。

优化 SSL，比如启用 SSL session cache。

移除不必要的模块以减少内存消耗。

监控性能并对瓶颈进行分析和优化。

**如何查看 Nginx 的配置错误？**

答案：

可以使用命令 nginx -t 来测试 Nginx 的配置文件是否存在语法错误。如果有错误，Nginx 将不会重新加载配置，并且会在终端上打印出错误信息的位置。

**Nginx 与 Apache 相比有什么优势和劣势？**

答案:

Nginx 的优势在于处理静态文件、高并发连接时性能较好，内存消耗低，配置灵活，并且适合作为负载均衡器和反向代理。Apache 的优势在于模块众多，社区支持强，文档齐全，而且对 .htaccess 文件的支持使得在不具有服务器配置文件写权限的环境下进行配置变得容易。Nginx 的劣势是模块相对较少，而且对动态内容的处理通常需要与后端服务器结合。Apache 的劣势是在高并发环境下性能可能不如 Nginx，并且内存消耗相对较大。

**写出下面服务的常用端口ftp http dns pop3、dhcp、nfs、mysql、samba等服务。**

答案：

20、21/tcp 80、443/tcp 53/tcp udp 110/tcp 67/udp 3306/tcp 445/tcp

**A B 网络是通的，最少列出五种传输文件的服务 ?**

答案：

nfs ,ftp,scp ,rsync,samba,http://

**简述 Apache 两种工作模式，以及它们之间的区别。**

答案：

最主要的两种模式是 prefork 模式与 worker 模式。prefork 每个子进程只有一个线程，

效率高但消耗内存大，是 unix 下默认的模式；worker 模式每个子进程有多个线程，内存消耗低，但一个线程崩溃会牵连其它同子进程的线程。

**nginx反向代理时，如何使后端获取真正的访问来源ip？**

答案：

在location配置段添加以下内容：

proxy\_set\_header Host $http\_host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

**如何进行压力测试？**

答案：

例如：模拟100个用户，对百度首页发起总共10000次请求。

# 测试命令：

ab -n 10000 -c 100 https://www.baidu.com/index.htm

**对于web应用，什么是pv和并发数？**

答案：

对于Web应用来说：

1.PV (Page View):

PV是页面浏览量（Page View）的缩写，它统计的是用户每次访问Web应用中任何一个页面都被记录一次。无论用户是首次访问页面还是刷新页面，只要页面被加载，PV数值就会增加。因此，PV反映了Web应用总的访问流量，它是衡量网站访问热度的重要指标之一。

2.并发数:

并发数是指在同一时间段内，Web应用同时处理请求的能力或者说正在处理的请求数量。在Web服务场景下，它可以指并发连接数，即服务器同时处理的HTTP连接数，或者是并发用户数，即同一时间点上活跃并进行操作的用户数。

具体到Web服务器层面，当多个用户几乎同时向服务器发送请求时，如果这些请求都在服务器端未完成之前到达，则这些请求被认为是“并发”的。并发数的大小直接影响到服务器的压力与系统的整体性能，同时也是评估服务器处理能力和系统架构扩展性的重要依据。

**对于web应用，什么是UV?**

答案：

UV（Unique Visitor）：独立访客，统计1天内访问某站点的用户数。可以统计服务一天的访问日志并根据用户的唯一标识去重得到。响应时间（RT）：响应时间是指系统对请求作出响应的时间，一般取平均响应时间。可以通过Nginx、Apache之类的Web Server得到。

**并发数如何计算？**

答案：

具体的计算公式是：并发连接数 = (PV / 统计时间 \* 页面衍生连接次数 \* http响应时间 \* 因数) / web服务器数量

解释：

页面衍生连接次数: 一个页面请求，会有好几次http连接，如外部的css, js,图片等,这个根据实际情况而定。

http响应时间: 平均一个http请求的响应时间，可以使用1秒或更少。

因数: 峰值流量 和平均流量的倍数，一般使用5 ,最好根据实际情况计算后得出

例子：计算10万PV的并发连接数

(100000PV / 86400秒 \* 50个派生连接数 \* 1秒内响应 \* 5倍峰值) / 1台Web服务器 = 289

**什么叫 CDN？**

答案：

即内容分发网络

其目的是通过在现有的 Internet 中增加一层新的网络架构，将网站的内容发布到

最接近用户的网络边缘，使用户可就近取得所需的内容，提高用户访问网站的速度

**讲述一下 cookie 和 session 的区别**

答案：

Cookie 机制采用的是在客户端保持状态的方案，而 Session 机制采用的是在服务器端保持状态的方案。

Cookie 不是很安全，别人可以分析存放在本地的 Cookie 并进行 Cookie 欺骗,考虑到安全应当使用 Session。

Session 会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能,考虑到减轻服务器性能方面，应当使用 COOKIE。

单个 Cookie 保存的数据不能超过 4K，很多浏览器都限制一个站点最多保存 20 个 Cookie。

Cookie 如果没有设置生存时间，那么关闭浏览器的瞬间，Cookie 就会消失，下一次登陆依旧要输入账号密码，Cookie 默认是存储在硬盘里而不是内存里，如果是设置了生存时间，那么就会保存在内存里，下一次继续使用。

Session 有一个 Session ID，要是服务器能查询的到 ID，就会按这个 ID 号的内容体现数据，如果查询不到就会新建一个 ID，Session ID 是可以用 Cookie 的形式保存的。

**HTTP 一般是无状态的，怎么让它变成有状态的?**

答案：

HTTP 协议跟 IP 协议、UDP 协议一样都是无状态的，HTTP 的无状态意思是每次的请求都是独立的，它的执行情况和结果与前面的请求和之后的请求是无直接关系的，它不会受前面的请求应答情况直接影响，也不会直接影响后面的请求应答情况。补充一下，TCP 是有状态的，它的请求并不独立，它通过包头的一些控制字段来分别包的关系，这里可以自行脑补一下三次握手的图。

那么 HTTP 是无状态的这一点是无法改变的，那么要变得有状态，就需要引入 Cookie 和 Session，通过这两个机制去实现一个有状态的 Web 应用。用一个表达式可以这么理解：Web 应用 = HTTP 协议+ Session、Cookies 等状态机制+其他辅助的机制。

**简述Linux中常见的系统服务，其作用分别是？**

答案：

SSH (Secure Shell)

作用：SSH 是一种加密网络协议，提供安全的远程登录功能，使得用户能够通过网络执行命令、传输文件、管理远程服务器等操作。

Apache HTTP Server

作用：Apache 是一款流行的开源Web服务器，用于托管网站和应用，处理HTTP请求，提供静态网页内容和动态内容生成，支持PHP、Perl等脚本语言。

Nginx

作用：Nginx 也是一个高性能Web服务器和反向代理服务器，擅长处理高并发连接，常用于负载均衡、缓存、静态内容服务，并可作为API gateway或者HTTPS终止代理。

Samba

作用：Samba 实现了SMB/CIFS协议，让Linux系统能与Windows系统无缝集成，实现跨平台的文件共享、打印服务以及其他各种资源共享。

NTP (Network Time Protocol)

作用：NTP服务用于同步系统时钟，确保网络中的所有设备时间保持一致。

SMB/NFS

SMB (Server Message Block)：类似于Samba，提供跨平台的文件共享服务。

NFS (Network File System)：也提供文件共享，但主要用于Unix-like系统间的网络文件系统共享。

DNS (Domain Name System)

作用：DNS服务负责域名解析，将易于记忆的域名转换成IP地址，以便网络设备找到并访问互联网上的其他主机。

FTP（File Transfer Protocol）文件服务器

**简述FTP主要的工作模式?**

**答案：**

FTP有两个主要的工作模式：主动模式（Active Mode）和被动模式（Passive Mode）。

主动模式与被动模式的根本区别在于数据连接的发起方和连接的目标端口。主动模式下，服务器发起数据连接，而在被动模式下，客户端发起数据连接。被动模式主要是为了克服网络环境中的防火墙和NAT问题，因为在这种情况下，服务器不能有效主动地连接到客户端机器上可能被防火墙屏蔽的端口。

**简述FTP两种登录方式？**

答案：

匿名登录 (Anonymous Login)

授权登录 (Authenticated Login)

**请简述JumpServer堡垒机的作用及其主要功能。**

答案：

 JumpServer堡垒机是一种开源的运维安全审计系统，它的核心作用是集中管理对多台服务器的访问权限，提供统一的身份验证、授权、审计和记录功能。主要功能包括但不限于：

双因素或多因素认证，提高身份安全性。

统一入口管理，避免直连服务器，减少攻击面。

访问控制，基于角色的访问控制(RBAC)，精细控制不同用户对目标服务器的操作权限。

操作审计，全程记录运维人员的所有操作行为，形成操作录像和日志。

会话管理，实时监控和控制运维会话，必要时可以强制中断非法或异常会话。

资产管理，对IT资产进行统一管理，包括服务器、网络设备、数据库等。

**在实际运维场景中，JumpServer是如何实现单点登录（SSO）和跳转功能的？**

答案：

JumpServer通过集成了多种认证源（如LDAP、AD、Radius等）实现单点登录，用户只需一次认证就可在受控范围内自由切换至不同目标系统。同时，采用代理转发或隧道技术，如SSH Proxy、RDP Proxy等，当运维人员登录JumpServer后，对其目标服务器的所有操作实际上都经过JumpServer的中介转发，从而实现了跳转功能和细粒度的权限管控。

**---------------Cobbler--------------**

**1.什么是Cobbler，其作用是什么？**

答案：

Cobbler是一个快速网络安装linux的服务，而且在经过调整也可以支持网络安装windows。

使用简单的命令即可完成PXE网络安装环境的配置，同时还可以管理DHCP、DNS、TFTP、RSYNC以及yum仓库、构造系统ISO镜像。 Cobbler支持命令行管理，web界面管理，还提供了API接口，可以方便二次开发使用，Cobbler客户端Koan支持虚拟机安装和操作系统重新安装。

**2.cobbler工作原理？**

答案：

Cobbler是通过将DHCP、TFTP、DNS、HTTP等服务进行集成，创建一个中央管理节点，其可以实现的功能有配置服务，创建存储库，解压缩操作系统媒介，代理或集成一个配置管理系统，控制电源管理等。 Cobbler的最终目的是实现无需进行人工干预即可安装机器。

**3.cobbler由哪些核心对象组成？**

答案：

发行版：表示一个操作系统。它承载了内核和 initrd 的信息，以及内核参数等其他数据。

配置文件：包含一个发行版、一个 kickstart 文件以及可能的存储库，还包含更多特定的内核参数等其他数据。

系统：表示要配给的机器。它包含一个配置文件或一个镜像，还包含 IP 和 MAC 地址、电源管理（地址、凭据、类型）以及更为专业的数据等信息。

存储库：保存一个 yum 或 rsync 存储库的镜像信息。

镜像：可替换一个包含不属于此类别的文件的发行版对象（例如，无法分为内核和 initrd 的对象）

**4.使用Cobbler相比手动或传统PXE安装的主要优势是什么？**

答案:

快速配置：Cobbler简化了配置网络引导环境的过程，只需通过Web界面或命令行就可以方便地管理多个安装源、模板和系统配置。

自动化程度高：它可以自动处理各种应答文件的生成和维护，包括Kickstart配置文件，确保每次安装的一致性和准确性。

批量部署：支持大规模、异构环境下的操作系统部署，可轻松应对成百上千台服务器的快速安装和更新需求。

动态配置：支持动态修改配置并实时生效，比如新增或删除操作系统安装源、调整网络配置等。

集成度高：可与 Puppet、Chef、Ansible 等配置管理工具结合使用，实现从裸机安装到配置管理的全流程自动化。

**5.在Cobbler中，如何添加新的操作系统安装源？**

答案:

在Cobbler中添加新的操作系统安装源通常涉及以下步骤：

1）添加ISO镜像或者指向网络源的URL作为安装源。

2）通过cobbler import命令导入操作系统安装介质，或者在Web管理界面上传ISO并指定相关参数。

3）配置与操作系统关联的kickstart应答文件，定义系统安装的具体配置项。

4）创建一个新的profile，将安装源和相应的kickstart文件关联起来，这样就可以通过Cobbler启动新机器并开始自动安装。

**6.如何利用Cobbler实现多系统选择安装？**

答案:

在Cobbler中，可以通过创建多个profile来实现多系统选择安装。每个profile关联不同的操作系统安装源和应答文件，当客户端通过PXE启动时，Cobbler的pxe菜单会显示可用的操作系统列表，允许用户选择其中的一个进行安装。

**------------Redis----------------**

**你用过哪些缓存中间件？**

答案：

Memcached是一个高性能的分布式内存对象缓存系统，主要用于动态Web应用以减轻数据库负载。它通过将经常访问的数据存储在内存中，以提高数据访问速度。然而，Memcached不支持数据的持久化，如果需要重启或者宕机，可能会导致数据丢失。

MongoDB则是一个介于关系数据库和非关系数据库之间的文档型数据库，具有高度的可扩展性和丰富的查询语言。它的缓存机制主要是通过将热数据存储在内存中来提高查询速度。但是MongoDB的主要功能还是数据存储，其缓存机制相对较弱。

Redis是一个开源的内存数据结构存储系统，支持多种数据结构如字符串、列表、集合、散列和有序集合等。它具有高速读写、支持数据持久化和发布/订阅等功能。并且，通过采用适当的数据淘汰策略，可以保证在内存使用达到上限时仍能正常运行。

**什么是 Redis? 特点？**

答案：

Redis 是完全开源免费的，遵守 BSD 协议，是一个高性能的 key-value 数据库。

Redis 与其他 key - value 缓存产品有以下三个特点：

（1）Redis 支持数据的持久化，可以将内存中的数据保存在磁盘中，重启的时候可以再次加载进行使用。

（2）Redis 不仅仅支持简单的 key-value 类型的数据，同时还提供 list，set，zset，hash 等数据结构的存储。

（3）Redis 支持数据的备份，即 master-slave 模式的数据备份。

**Redis优势？**

答案：

（1）性能极高 – Redis 能读的速度是 110000 次/s,写的速度是 81000 次/s 。

（2）丰富的数据类型 – Redis 支持二进制案例的 Strings, Lists, Hashes, Sets 及Ordered Sets 数据类型操作。

（3）原子 – Redis 的所有操作都是原子性的，意思就是要么成功执行要么失败完全不执行。单个操作是原子性的。多个操作也支持事务，即原子性，通过 MULTI 和 EXEC指令包起来。

（4）丰富的特性 – Redis 还支持 publish/subscribe, 通知, key 过期等等特性。

**Redis 与其他 key-value 存储有什么不同？**

答案：

（1）Redis 有着更为复杂的数据结构并且提供对他们的原子性操作，这是一个不同于其他数据库的进化路径。Redis 的数据类型都是基于基本数据结构的同时对程序员透明，无需进行额外的抽象。

（2）Redis 运行在内存中但是可以持久化到磁盘，所以在对不同数据集进行高速读写时需要权衡内存，因为数据量不能大于硬件内存。在内存数据库方面的另一个优点是，相比在磁盘上相同的复杂的数据结构，在内存中操作起来非常简单，这样 Redis可以做很多内部复杂性很强的事情。同时，在磁盘格式方面他们是紧凑的以追加的方式产生的，因为他们并不需要进行随机访问。

Redis 的数据类型？

Redis 支持五种数据类型：string（字符串），hash（哈希），list（列表），set（集合）及 sortedset：有序集合)。

我们实际项目中比较常用的是 string、hash

使用 Redis 有哪些好处？

（1）速度快，因为数据存在内存中，类似于 HashMap，HashMap 的优势就是查找和操作的时间复杂度都是 O(1)

（2）支持丰富数据类型，支持 string，list，set，Zset，hash 等

（3）支持事务，操作都是原子性，所谓的原子性就是对数据的更改要么全部执行，要么全部不执行

（4）丰富的特性：可用于缓存，消息，按 key 设置过期时间，过期后将会自动删除

Redis 相比 Memcached 有哪些优势？

（1）Memcached 所有的值均是简单的字符串，redis 作为其替代者，支持更为丰富的数据类

（2）Redis 的速度比 Memcached 快很多

（3）Redis 可以持久化其数据

Memcache 与 Redis 的区别都有哪些？

（1）存储方式 Memecache 把数据全部存在内存之中，断电后会挂掉，数据不能超过内存大小。 Redis 有部份存在硬盘上，这样能保证数据的持久性。

（2）数据支持类型 Memcache 对数据类型支持相对简单。 Redis 有复杂的数据类型。

（3）使用底层模型不同 它们之间底层实现方式 以及与客户端之间通信的应用协议不一样。 Redis 直接自己构建了 VM 机制 ，因为一般的系统调用系统函数的话，会浪费一定的时间去移动和请求。

Redis 是单进程单线程的吗？

Redis 是单进程单线程的，redis 利用队列技术将并发访问变为串行访问，消除了传统数据库串行控制的开销。

一个字符串类型的值能存储最大容量是多少？

512M

Redis 的持久化机制是什么？各自的优缺点？

Redis提供两种持久化机制 RDB 和 AOF 机制:

RDB(Redis DataBase)持久化方式：

是指用数据集快照的方式半持久化模式)记录 redis 数据库的所有键值对,在某个时间点将数据写入一个临时文件，持久化结束后，用这个临时文件替换上次持久化的文件，达到数据恢复。

优点：

（1）只有一个文件 dump.rdb，方便持久化。

（2）容灾性好，一个文件可以保存到安全的磁盘。

（3）性能最大化，fork 子进程来完成写操作，让主进程继续处理命令，所以是 IO最大化。使用单独子进程来进行持久化，主进程不会进行任何 IO 操作，保证了 redis的高性能)

（4）相对于数据集大时，比 AOF 的启动效率更高。

缺点：

数据安全性低。RDB 是间隔一段时间进行持久化，如果持久化之间 redis 发生故障，会发生数据丢失。所以这种方式更适合数据要求不严谨的时候

AOF(Append-only file) 持久化方式：

是指所有的命令行记录以 redis 命令请求协议的格式完全持久化存储)保存为 aof 文件。

优点：

（1）数据安全，aof 持久化可以配置 appendfsync 属性，有 always，每进行一次命令操作就记录到 aof 文件中一次。

（2）通过 append 模式写文件，即使中途服务器宕机，可以通过 redis-check-aof工具解决数据一致性问题。

（3）AOF 机制的 rewrite 模式。AOF 文件没被 rewrite 之前（文件过大时会对命令进行合并重写），可以删除其中的某些命令（比如误操作的 flushall）)

缺点：

（1）AOF 文件比 RDB 文件大，且恢复速度慢。

（2）数据集大的时候，比 rdb 启动效率低。

Redis 常见性能优化方案

（1）Master 最好不要写内存快照，如果 Master 写内存快照，save 命令调度 rdbSave函数，会阻塞主线程的工作，当快照比较大时对性能影响是非常大的，会间断性暂停服务

（2）如果数据比较重要，某个 Slave 开启 AOF 备份数据，策略设置为每秒同步一

（3）为了主从复制的速度和连接的稳定性，Master 和 Slave 最好在同一个局域网

（4）尽量避免在压力很大的主库上增加从

（5）主从复制不要用图状结构，用单向链表结构更为稳定，即：Master <- Slave1<- Slave2 <- Slave3…这样的结构方便解决单点故障问题，实现 Slave 对 Master的替换。如果 Master 挂了，可以立刻启用 Slave1 做 Master，其他不变。

Redis如何删除key

del key1 key2 ...

redis 过期键的删除策略？

（1）定时删除:在设置键的过期时间的同时，创建一个定时器 timer。让定时器在键的过期时间来临时，立即执行对键的删除操作。

（2）惰性删除:放任键过期不管，但是每次从键空间中获取键时，都检查取得的键是否过期，如果过期的话，就删除该键;如果没有过期，就返回该键。

（3）定期删除:每隔一段时间程序就对数据库进行一次检查，删除里面的过期键。至于要删除多少过期键，以及要检查多少个数据库，则由算法决定。

为什么 redis 需要把所有数据放到内存中？

Redis 为了达到最快的读写速度将数据都读到内存中，并通过异步的方式将数据写入磁盘。所以 redis 具有快速和数据持久化的特征。如果不将数据放在内存中，磁盘 I/O 速度为严重影响 redis 的性能。在内存越来越便宜的今天，redis 将会越来越受欢迎。如果设置了最大使用的内存，则数据已有记录数达到内存限值后不能继续插入新值。

Redis 的同步机制了解么？

Redis 可以使用主从同步，从从同步。第一次同步时，主节点做一次 bgsave，并同时将后续修改操作记录到内存 buffer，待完成后将 rdb 文件全量同步到复制节点，复制节点接受完成后将 rdb 镜像加载到内存。加载完成后，再通知主节点将期间修改的操作记录同步到复制节点进行重放就完成了同步过程。

是否使用过 Redis 集群，集群的原理是什么？

（1）Redis Sentinal 实现高可用，在 master 宕机时会自动将 slave 提升为master，继续提供服务。

（2）Redis Cluster 高可用和扩展性，在单个 redis 内存不足时，使用 Cluster 进行分片存储。

Redis 如何设置密码及验证密码？

设置密码：config set requirepass 123456

授权密码：auth 123456

说说 Redis 哈希槽的概念？

Redis 集群没有使用一致性 hash,而是引入了哈希槽的概念，Redis 集群有16384 个哈希槽，每个 key 通过 CRC16 校验后对 16384 取模来决定放置哪个槽，集群的每个节点负责一部分 hash 槽

Redis 集群会有写操作丢失吗？为什么？

Redis 并不能保证数据的强一致性，这意味这在实际中集群在特定的条件下可能会丢失写操作

Redis 集群之间是如何复制的？

异步复制

Redis 集群最大节点个数是多少？

16384 个

怎么测试 Redis 的连通性？

使用 ping 命令。

怎么理解 Redis 事务？

（1）事务是一个单独的隔离操作：事务中的所有命令都会序列化、按顺序地执行。事务在执行的过程中，不会被其他客户端发送来的命令请求所打断。

（2）事务是一个原子操作：事务中的命令要么全部被执行，要么全部都不执行。

Redis 如何做内存优化？

尽可能使用散列表（hashes），散列表（是说散列表里面存储的数少）使用的内存非常小，所以你应该尽可能的将你的数据模型抽象到一个散列表里面。比如你的 web 系统中有一个用户对象，不要为这个用户的名称，姓氏，邮箱，密码设置单独的 key,而是应该把这个用户的所有信息存储到一张散列表里面。

都有哪些办法可以降低 Redis 的内存使用情况呢？

如果你使用的是 32 位的 Redis 实例，可以好好利用 Hash,list,sorted set,set等集合类型数据，因为通常情况下很多小的 Key-Value 可以用更紧凑的方式存放到一起。

Redis 主要消耗什么物理资源？

Redis主机消耗内存资源

Redis 的内存用完了会发生什么？

如果达到设置的上限，Redis 的写命令会返回错误信息（但是读命令还可以正常返回。）或者你可以将 Redis 当缓存来使用配置淘汰机制，当 Redis 达到内存上限时会冲刷掉旧的内容。

Redis 最适合的场景？

会话缓存（Session Cache）

最常用的一种使用 Redis 的情景是会话缓存（session cache）。用 Redis 缓存会话比其他存储（如 Memcached）的优势在于：Redis 提供持久化。当维护一个不是严格要求一致性的缓存时，如果用户的购物车信息全部丢失，大部分人都会不高兴的，现在，他们还会这样吗？ 幸运的是，随着 Redis 这些年的改进，很容易找到怎么恰当的使用 Redis 来缓存会话的文档。甚至广为人知的商业平台Magento 也提供 Redis 的插件。

全页缓存（FPC）

除基本的会话 token 之外，Redis 还提供很简便的 FPC 平台。回到一致性问题，即使重启了 Redis 实例，因为有磁盘的持久化，用户也不会看到页面加载速度的下降，这是一个极大改进，类似 PHP 本地 FPC。 再次以 Magento 为例，Magento提供一个插件来使用 Redis 作为全页缓存后端。 此外，对 WordPress 的用户来说，Pantheon 有一个非常好的插件 wp-redis，这个插件能帮助你以最快速度加载你曾浏览过的页面。

队列

Reids 在内存存储引擎领域的一大优点是提供 list 和 set 操作，这使得 Redis能作为一个很好的消息队列平台来使用。Redis 作为队列使用的操作，就类似于本地程序语言（如 Python）对 list 的 push/pop 操作。 如果你快速的在 Google中搜索“Redis queues”，你马上就能找到大量的开源项目，这些项目的目的就是利用 Redis 创建非常好的后端工具，以满足各种队列需求。例如，Celery 有一个后台就是使用 Redis 作为 broker，你可以从这里去查看。

排行榜/计数器

Redis 在内存中对数字进行递增或递减的操作实现的非常好。集合（Set）和有序集合（Sorted Set）也使得我们在执行这些操作的时候变的非常简单，Redis 只是正好提供了这两种数据结构。所以，我们要从排序集合中获取到排名最靠前的 10个用户–我们称之为“user\_scores”，我们只需要像下面一样执行即可： 当然，这是假定你是根据你用户的分数做递增的排序。如果你想返回用户及用户的分数，你需要这样执行： ZRANGE user\_scores 0 10 WITHSCORES Agora Games 就是一个很好的例子，用 Ruby 实现的，它的排行榜就是使用 Redis 来存储数据的，你可以在这里看到。

发布/订阅

最后（但肯定不是最不重要的）是 Redis 的发布/订阅功能。发布/订阅的使用场景确实非常多。我已看见人们在社交网络连接中使用，还可作为基于发布/订阅的脚本触发器，甚至用 Redis 的发布/订阅功能来建立聊天系统！

什么是缓存穿透？

通常的缓存系统，都是按照 key 去缓存查询，若是不存在对应的 value，就应该去后端系统查找（好比DB）。一些恶意的请求会故意查询不存在的 key,请求量很大，就会对后端系统形成很大的压力。这就叫作缓存穿透。

如何避免缓存穿透？

1：对查询结果为空的状况也进行缓存，缓存时间设置短一点，或者该 key 对应的数据 insert 了以后清理缓存。

2：对必定不存在的 key 进行过滤。能够把全部的可能存在的 key 放到一个大的 Bitmap 中，查询时经过该 bitmap 过滤。

redis雪崩了会有哪些原因，怎么解决？

原因：

缓存数据同时过期

大量的缓存数据在同一时间段过期，例如由于误配置或由于某些缓存服务器的故障，缓存数据无法及时更新，那么当有大量请求访问这些数据时，它们将直接落到数据库上，导致数据库请求量骤增，产生压力。

缓存服务器宕机

缓存服务器遇到故障或宕机，所有请求将无法从缓存中获取数据，强迫它们直接访问数据库。由于缓存未能发挥作用，数据库将面临大量请求，导致性能下降，并可能导致数据库崩溃。

解决思路：

设置合理的缓存过期时间

缓存过期时间的设置需要根据业务需求和数据的变化频率来确定。对于不经常变化的数据，可以设置较长的过期时间，以减少对数据库的频繁访问。对于经常变化的数据，可以设置较短的过期时间，确保缓存数据的实时性。

注意的是，缓存过期时间设置过长可能导致数据的实时性降低，而设置过短可能增加缓存失效和数据库压力。因此，需要根据具体应用场景和需求来综合考虑，进行合理的设置。

使用热点数据预加载

预先将热点数据加载到缓存中，并设置较长的过期时间，可以避免在同一时间点大量请求直接访问数据库。可以根据业务需求，在系统启动或低峰期进行预热操作，将热点数据提前加载到缓存中。

热点数据预加载可以提升系统的性能和响应速度，减轻数据库的负载。要注意的是，预加载操作可能会占用系统资源，因此需要合理安排预加载执行的时间和频率，避免对系统正常业务的影响。另外，需要根据实际情况监控和调整预加载策略，以保持缓存数据的实时性和准确性。

缓存数据分布均衡

将缓存数据进行分散存储，可以使用一致性哈希算法或数据分片来将缓存数据分散存储在多个缓存服务器上，避免将所有数据集中存储在同一台缓存服务器上。这样可以提高系统的容错性，避免某个缓存服务器故障导致大量的缓存失效。通过合理的数据分布策略和动态的节点管理，可以确保缓存数据在不同节点之间均衡分布，提高系统的性能和可扩展性。

使用多级缓存架构

使用多级缓存架构可以提高系统的性能和容错性。内存缓存（如Redis）可以提供快速的数据访问能力，而分布式缓存（如Memcached）可以通过多台服务器组成集群，提高系统的可用性。可以根据数据的访问频率和重要程度，将数据存储在不同级别的缓存中。

使用多级缓存架构可以根据数据的访问频率和重要性，将数据存储在不同级别的缓存中，以提高数据访问的速度和稳定性。同时，通过合理的缓存策略和同步机制，保证多级缓存中的数据的实时性和一致性。综合考虑业务需求和技术条件，可以选择适合的多级缓存架构来提升系统的性能和用户体验。

缓存故障转移和降级策略

当缓存服务器发生故障或宕机时，需要有相应的故障转移和降级策略。可以通过监控系统来及时发现缓存故障，并进行自动切换到备份缓存服务器。同时，可以实现降级策略，当缓存失效时，系统可以直接访问数据库，保证系统的可用性。

通过缓存故障转移和降级策略，可以保证系统在缓存不可用或故障的情况下仍然可以正常运行，提高系统的稳定性和容错性。

什么是redis的主从复制 ？

复制是高可用Redis的基础，哨兵和集群都是在复制基础上实现高可用的。复制主要实现了数据的多机备份，以及对于读操作的负载均衡和简单的故障恢复。缺陷：故障恢复无法自动化；写操作无法负载均衡；存储能力受到单机的限制。

Redis的哨兵作用是？

Redis提供了哨兵( Sentinel)机制来实现主从集群的自动故障恢复。

监控: Sentinel会不断检查您的master和slave是否按预期工作

自动故障恢复: 如果master故障，Sentinel会将一个slave提升为master。当故障实例恢复后也以新的master为主

通知: Sentinel充当Redis客户端的服务发现来源，当集群发生故障转移时，会将最新信息推送给Redis的客户端

哨兵的监控状态有哪些？

Sentinel基于心跳机制监测服务状态,每隔1秒向集群的每个实例发送ping命令:

主观下线: 如果某sentinel节 点发现某实例未在规定时间响应，则认为该实例主观下线。

客观下线: 若超过指定数量(quorum) 的sentinel都认为该实例主观下线， 则该实例客观下线。quorum值最好超过Sentinel实例数量的一半。

哨兵master选举的原理？

一旦发现master故障, sentinel需要在salve中选择一个作为新的master,选择依据是这样的:

首先会判断slave节点与master节点断开时间长短，如果超过指定值( down-after-milliseconds \* 10)则会排除该slave节点

然后判断slave节点的slave-priority值, 越小优先级越高,如果是0则永不参与选举如果slave-prority一样,则判断slave节点的offset值,越大说明数据越新，优先级越高。

最后是判断slave节点的运行id大小，越小优先级越高。

redis分片集群的作用？

Redis 分片集群是一种将 Redis 数据库分散到多个节点上的方式，以提供更高的性能和可伸缩性。可以通过并行处理来提高读写吞吐量。分片集群还提供了故障容错和高可用性的功能，即使其中一个节点发生故障，其他节点仍然可以继续工作。

如果redis内存满了怎么处理？

① 通过配置文件配置

通过在Redis安装目录下面的redis.conf配置文件中添加以下配置设置内存大小

redis的配置文件不一定使用的是安装目录下面的redis.conf文件，启动redis服务的时候是可以传一个参数指定redis的配置文件。

② 内存回收

使用redis的时候我们需要给某些键值对设置有效期，redis中可以通过4个独立的命令来给一个键值设置过期时间

（1）expire key ttl ：将key的过期时间设置为 ttl秒

（2）pexpire key ttl ：将key值的过期时间设置为 ttl 毫秒

（3）expireat key timestamp ：将key的过期时间设置为指定的timestamp秒数

（4）pexpireat key timestamp ：将key值的过期时间设置为指定的timestamp毫秒数

③ 过期策略

如果将一个过期的键删除，我们会有三种策略

（1）定时删除：为每一个键设置一个定时器，一旦过期时间到了，则将键删除。这种策略对内存很友好，但是对CPU不友好，因为每一个定时器都会占用一定的CPU资源。

（2）惰性删除：不管键有没有过期都不主动删除，等到每次去获取键时再判断是否过期，如果过期就删除该键，否则返回键对应的值。这种策略对内存不够友好，可能会浪费很多内存。

（3）定期扫描：系统每隔一段时间就定期扫描一次，发现过期的键就进行删除。这种策略相对来说是上面两种策略的折中方案，需要注意的是这个定期的频率要结合实际情况掌控好，使用这种方案有一个缺陷就是可能会出现已经过期的键也被返回。

④ 开启redis持久化

使用 Redis 的持久化功能，将内存中的数据定期地写入磁盘，以释放内存空间。Redis 提供了两种持久化方式：RDB 和 AOF。其中，RDB 持久化会将 Redis 数据库的快照写入磁盘，而 AOF 持久化则会将 Redis 的操作日志写入磁盘。

⑤ 增加服务器内存

条件允许的情况下，可以对服务器内存进行扩容