**0：查看上次服务器启动时间：w**

**查看登录系统：last 登录失败：lastb**

**查看跟踪经过了哪些网络节点：tracert**

**查看端口的网络服务运行状况：netstat、ss**

**查看对应的服务名/文件：lsof -i :80**

**查看两台服务器之间的网络是不是正常的，服务器是禁ping的可以用：telnet**

**查看内存：vmstat、free、sar -r（虚拟内存：vmstat）**

**查看CPU负载：uptime、sar -u、iostat**

**查看CPU信息：lscpu**

**[查看磁盘使用情况：sar -d](https://blog.csdn.net/liyongbing1122/article/details/89517282" \l "7.%E6%9F%A5%E7%9C%8B%E7%A3%81%E7%9B%98%E4%BD%BF%E7%94%A8%E6%83%85%E5%86%B5" \t "/root/文档\x/_self)、**

**查看磁盘：iostat、lsblk、df**

**查看流量信息：ifconfig、iftop、sar 多个应用使用nethogs**

**查看文件挂载：df -h**

**查看二进制文件的内容：hexdump**

**修复文件系统：fsck**

**修改内核参数：/etc/sysctl.conf，刷新：sysctl -p**

**查看主板设备信息：dmidecode**

**查看路由信息：ip route show**

**查看默认路由：netstat -nr**

**查看路由表：route -n**

**查看当前系统加载的模块情况的命令是：lsmod**

**查看服务器各接口卡情况的命令是：lspci**

**查看进程：ps、pstree、top（P按cpu M按内存排序）**

**ps -aux（显示进程信息非常详细)**

**ps -elf (显示进程父进程信息)**

**检索进程：pgrep**

**杀死进程（-9强杀）kill：后台任务编号、PID**

**killall：进程名**

**pkill：查找条件**

**find：精确查找（-type -name -size -user -mtime）**

**rsync:数据同步**

**下载：rsync -av --delete root@IP:远程目录 本地目录**

**上传：rsync -av --delete 本地目录 root@IP:远程目录**

**inotify:对目标文件持续监控（-m）**

**-q：减少屏幕输出信息**

**du -sh:统计文件的占用空间**

**tar -zcvf 归档**

**tar -xf 释放**

**-z、-j、-J：调用 .gz、.bz2、.xz 格式工具进行处理**

**-t：显示归档中的文件清单**

**-C：指定释放路径**

**使用--exclude选项在打包时可用排除特定的目录**

**iostat参数说明**

**%user   用户空间的CPU使用**

**%nice   改变过优先级的进程的CPU使用率**

**%system   内核空间的CPU使用率**

**%iowait   CPU等待IO的百分比**

**%steal   虚拟机的虚拟机CPU使用的CPU**

**%idle   空闲的CPU**

**0.1：**工作日常内容描述：

负责公司日常服务器的故障处理，应用调整和优化，检查服务器系统日志，编写常用的Shell脚本和部署一些工作需要的环境，监控网站服务器，数据库等运行状态

你之前运维都做哪些工作？

1看监控中心，处理警告问题：比如磁盘分区超过80%、某台机器负载长时间偏高（超过20）、某台nginx服务器偶尔出现502状态码

2编写数据库备份脚本、监控脚本（web服务、系统状态）、系统安全检测脚本、日志收集脚本、日常代码发布的脚本

3根据开发或者运营同事的需求，部署或者配置环境

4故障的处理：网站访问慢或访问不了、服务器宕机、某个服务宕掉、ddos攻击、cc攻击、磁盘损坏、存储损坏、磁盘空间撑满、mysql主从不同步、mysql查询慢等。

5关注一线资讯，学习新技能：开源中国、51cto、CSDN、博客园

谈谈你对运维工程师的理解以及对工作的认识？

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快、最稳定、最安全的服务。在日常工作中，需要做好监控和事故预案，在平时，运维不仅仅要给研发、测试提供协助，还要自发性地去研究和分析服务器上的一些性能指标和日志，从而给研发同事提供一些数据和线索来优化业务

0.2：linux上进程有5种状态:

1. 运行(正在运行或在运行队列中等待)

2. 中断(休眠中, 受阻, 在等待某个条件的形成或接受到信号)

3. 不可中断(收到信号不唤醒和不可运行, 进程必须等待直到有中断发生)

4. 僵死(进程已终止, 但进程描述符存在, 直到父进程调用wait4()系统调用后释放)

5. 停止(进程收到SIGSTOP, SIGSTP, SIGTIN, SIGTOU信号后停止运行运行)

0.3：描述Linux运行级别0-6的各自含义

答：0：关机模式

1：单用户模式<==破解root密码

2：无网络支持的多用户模式

3：有网络支持的多用户模式（文本模式，工作中最常用的模式）

4：保留，未使用

5：有网络支持的X-windows支持多用户模式（桌面）

6: 重新引导系统，即重启

描述Liunx系统从开机到登陆界面的启动过程

答：BIOS 自检

启动 GRUB 2 (内核引导程序)

加载内核

执行 systemd 进程

初始化系统环境

执行/bin/login 程序

0.4：缓存雪崩：是指在某一个时间段，缓存集中过期失效，大量的请求都跳过缓存直接到数据库中 解决：不设置超时时间，避免过期；在失效时间上加随机的过期时间，避免集中过期；有相应的限流降级策略

1. 缓存击穿：是指大量的请求集中对一个热点key进行访问，当这个key失效的瞬间，持续的大并发直接落到数据库上，就在这个key的点上击穿了缓存 解决：加互斥锁，避免大量请求同时打库；设置key不过期，更新时直接覆盖
2. 缓存穿透：指缓存和数据库中都没有数据，而用户不断发起请求，这会给服务器带来很大的压力 解决：把空对象存到缓存中，并设置一个较短的过期时间；接口层增加校验；利用布隆过滤器

1：**熟悉LINUX系统常用命令，管理维护服务器；**

**ls pwd cd Mkdir 创建目录 Touch 创建文件**

**Find 查找文件 Grep 查找字符 Mv 移动**

**Vim 编辑 yum或 dnf 装包**

**2：熟悉SHELL脚本编写，实现自动化批量部署；**

**监控脚本(监控系统、监控服务、监控硬件信息、监控性能、安全监控等)**

**系统初始化脚本(创建目录,创建账户,安装软件包,设置权限,修改内核参数等)**

**一键部署(源码安装脚本)**

**备份脚本(自动备份数据库,备份网站数据,备份日志,备份配置文件等)**

**日志分析脚本(分析日志数据,汇总并统计相关信息,如 PV、UV 等)**

**3：Nginx优化**

**为每个进程分配cpu**

**优化单个进程允许的最大连接数**

**配置连接超时时间**

**采用epoll事件模型**

**优化gzip压缩**

**配置防盗链**

**内核参数优化**

**限制上传文件的大小**

**Tomcat优化**

**内存优化**

**并发优化**

**缓存优化**

**IO优化**

**开启线程池**

**组件优化APR**

mySql优化

**Mysql怎么优化？**

**Sql语句优化**

**数据结构的优化**

**服务运行参数的优化**

**服务器硬件优化**

Mysql 性能优化:就是通过合理安排资源,调整系统参数使 MYSQL 运行更快,更节省资源。

MYSQL 性能优化包括:查询速度优化,更新速度优化,mysql 服务器优化等等,原则是减少系统的瓶颈,

减少资源的占用,增加系统反应的速度。

0.锁定表:可以维护数据的完整性.

1.使用外键:保证数据的关联性.

2.使用索引:索引是提高数据库性能的常用方法,它可以令数据库服务器以比没有索引快得多的速度检索特定的行,尤其是在查询语句当中包含有 MAX(),MIN()和 ORDERBY 这些命令的时候,性能提高

4：**Zabbix监控原理：**

**Agentd安装在被监控的主机上，Agent负责定期收集客户端本地各项数据，并发送至Zabbix Server端，Zabbix Server收到数据，将数据存储到数据库中，用户基于Zabbix WEB可以看到数据在前端展现图像。当Zabbix监控某个具体的项目，改项目会设置一个触发器阈值，当被监控的指标超过该触发器设定的阈值，会进行一些必要的动作，动作包括：发送信息（邮件、微信、短信）、发送命令（SHELL 命令、Reboot、Restart、Install等）**

**。Zabbix监控什么？**

**监控主要监控哪些项**

**硬件资源(如 CPU、内存、硬盘、网络设备)**

**流量(如连通性、接口流量)**

**服务状态(web 服务、数据库服务、容器)**

**Zabbix图形界面丰富，自带很多监控模板**

**监控了CPU使用率、内存剩余、磁盘使用空间、网卡流量、web服务、mysql主从、访问日志等基础项目，也可以自定义监控项目，通过shell、python等脚本实现。可以通过邮件、短信、微信等方式告警触发报警的有系统负载（当高于20报警）、磁盘使用百分比（高于90%报警）、mysql主从是否正常（不正常告警）、mysql队列数量（高于400报警）、网卡流量（高于100M报警）等等。**

**5：熟悉MYSQL数据库的数据备份，主从同步，读写分离；**

**你是怎么备份数据的，包括数据库备份？**

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

**mysql主从原理**

**master服务器讲sql数据库改写的记录，放入bin-log日志当中，****slave通过IO线程获取master上边的这个改变记录，同时呢IO线程启动dump，发送二进制日志，保存到从上边的中继日志上。然后sql线程读取二进制日志，是主从达到一致。**

**数据库每天都要承担的网站的数据更新，所以大量的读写任务都在一个机器上压力非常大。**

如何对Mysql数据库进行备份和恢复？

1.mysqldump是最常用的备份工具，支持myisam或innodb引擎，备份出来的数据是一个sql文件，含有创建表的sql语句，和包含表中每一条数据的插入sql语句。不管是备份还是恢复，效率都比较低，会锁表，如果数据量不大，使用mysqldump是很方便的。

2.innobackupex提供了myisam表备份的能力。当数据量较大时，适合使用innobackupex来备份，效率更快。

3.binlog，实时备份，在高可用架构中，通常使用主从或者主主，当一台宕机，另外一台可以马上替上去，数据可能会有少量丢失，我们可以通过查看binlog日志完全恢复。另外，做增量备份时，也可以只备份binlog文件，恢复时通过binlog文件可以恢复到指定某一时刻的数据状态

socket和tcp访问mysql的区别？

socket访问是通过服务器上的一个socket文件来和mysql服务通信，速度快，但是只能在同一台服务器连接。tcp是通过网络来通信，可以在另外的机器连接。

**mysql如何减少主从复制延迟？**

**如果延迟比较大，就先确认以下几个因素：**

**1. 从库硬件比主库差，导致复制延迟**

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

**3. 慢SQL语句过多**

**4. 网络延迟**

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

**6. slave负载，一般的做法是，使用多台slave来分摊读请求**

# 使用读写分离可以有效的缓解数据库压力

**6：TCP/IP原理说一下？TCP有哪几个状态，分别是什么意思？**

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

**简述TCP三次握手的过程？**

**在TCP/IP协议中，TCP协议提供可靠的连接服务，采用三次握手建立一个连接。**

**第一次建立连接时，客户端发送syn包到服务器，并进入请求连接（SYN\_SEND）状态，等待服务器确认。服务器收到后，确认客户的SYN，同时自己也发送SYN+ACK包，此时服务器进入等待（SYN\_RECV）状态。最后客户端向服务器发送ACK确认包，发送完毕后客户端和服务器进入建立连接（ESTABLISHED）状态，完成三次握手，客户端与服务器开始传送数据**

**为什么连接的时候是三次握手，关闭的时候却是四次握手？**

**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。故需要四步握手。**

**我们都知道，dns既采用了tcp协议，又采用了udp协议，什么时候采用tcp协议？什么时候采用udp协议？为什么要这么设计？**

**DNS主备服务器之间传输数据时使用tcp，而客户端与DNS服务器之间数据传输时用的是udp。tcp和udp最大的区别在于tcp需要三次握手来验证数据包的完整性和准确性，udp不需要验证，速度比tcp要快。DNS主备数据传输需要保证数据准确性，有必要使用tcp通信，而客户端请求服务端，更需要快，所以用udp**

**7：如何优化Linux系统？**

**1登录系统：不使用root登录，通过sudo授权管理，使用普通用户登录。**

**2禁止SSH远程：更改默认的远程连接SSH服务及禁止root远程连接。**

**3时间同步：定时自动更新服务器时间（chronyc makestep）**

**4配置yum更新源，从国内更新下载安装rpm包。**

**5调整文件描述符数量，进程及文件的打开都会消耗文件描述符。**

**6精简开机启动服务（crond、sshd、network、rsyslog）**

**7Linux内核参数优化/etc/sysctl.conf，执行sysct -p生效。**

**8清空/etc/issue，去除系统及内核版本登陆前的屏幕显示**

**8：从运维角度讲一下怎么预防cc攻击和ddos攻击**

**CC主要是用来攻击页面的，通过模拟多个用户进行大并发访问，导致网页打开的速度比较慢。防御CC攻击可以通过多种方法，比如，可以分析攻击的请求头信息，针对这些请求做一些限制。也可以分析请求的ip，利用iptables来限制ip。将网站做成静态页面，也可以有效降低服务器资源使用。另外，还可以限制连接数量，修改最大超时时间等。**

**ddos攻击就是利用合理的服务请求来占用过多的服务资源，从而使合法用户无法得到服务的响应。当攻击量很大，这时候只能临时在上层网络把目标IP封掉，这样牺牲单个ip而保全大局。也可以接入第三方的防ddos攻击的cdn。**

**9：如果你们公司的网站访问很慢，你会如何排查？**

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

**网站打不开，排查故障的思路？(网络-服务端口-流量-负载-进程-配置-抓包)**

**A. 先ping一下域名，看看网络是否通**

**B. Telnet 一下服务器的80端口，看是否通**

**C. 用sar命令查看服务器流量是否正常**

**D. 用w查看服务器是否有负载，ps查看服务是否开启**

**E. 查看配置是否正确**

**F. 用tcpdump抓包查看数据包是否正常，看是否有攻击**

**简单介绍一下您所知道的减少和加速web服务器的请求的方法？**

**A. 针对静态的访问，做过期时间，把静态文件缓存在客户端上**

**B. 在前端搭建squid反向代理，把静态的文件缓存在squid上**

**C. 接入CDN**

**D. 合并js、css以及小图片**

**你在工作的过程中,遇到过你印象最深的是什么故障问题,你又是如何解决?**

**1. 网站访问特别慢,经过查询,发现是开发的一个 bug,查询数据库创建了 80G 的临时表,修复**

**bug 即可**

**2. 网站被植入恶意代码,通过数据对比,找到注入点,通过备份的数据修复**

**3. 时间数据不同步(导致所有证书异常,网站无法访问)**

简单介绍一下您所知道的减少和加速web服务器的请求的方法？

A. 针对静态的访问，做过期时间，把静态文件缓存在客户端上

B. 在前端搭建squid反向代理，把静态的文件缓存在squid上

C. 接入CDN

D. 合并js、css以及小图片

网站打不开，排查故障的思路？(网络-服务端口-流量-负载-进程-配置-抓包)

A. 先ping一下域名，看看网络是否通

B. Telnet 一下服务器的80端口，看是否开启

C. 没端口再查看日志，看看配置文件是否正确

D. 用sar命令查看服务器流量是否正常

E. 用w查看服务器是否有负载

F. 用tcpdump抓包查看数据包是否正常，看是否有攻击

**突然发现一台Linux系统只读，应该怎么处理，请写出详细步骤？**

**文件系统只读，可能是误操作，比如挂载的时候加了ro的挂载选项。也可能是文件系统损坏，这时候可以使用fsck -y 分区 来尝试修复一下，但在修复之前最好是把重要数据做一个备份以防万一。如果修复失败，那说明是磁盘损坏，这就需要更换磁盘了。**

**一台Linux服务器负载高，连接慢，怎么查看？  
先用w看负载多少，用top看哪个进程占用cpu高，同时用top按M看哪个进程占用内存多，用iotop看哪个进程读写频发，用sar命令或者nload命令查看网卡流量，是否跑满带宽**

**如何判断当前服务器的80端口是否被监听，如果被监听，怎么知道被哪个进程监听?**

**用netstat –lnpt |grep 80 命令来查看80端口是否被监听**

**查看进程：使用命令lsof –i :80看第一列**

**在Linux下，如何分析一个程序达到性能瓶颈的原因，请分别从CPU、内存、IO、网络的角度判断是谁导致的瓶颈？**

**A. 用w命令看系统负载高不高，如果高可能是CPU不够用，进程太多运行太慢，也可能是磁盘I/O太高了。**

**B. 用vmstat 1命令来查看各个指标，着重分析r,b,swpd,si,so,bi,bo,us,wa. 若r列或者us列数值偏高则说明cpu有瓶颈，若b列或者wa列数值偏高同时bi或bo的数字很大，则说明磁盘有压力，若swpd一直变化，并且si和so一直不为0，则说明内存不够了。**

**C. 用top命令来分析哪个进程耗费cpu最多，默认按使用cpu多少排序，按M也可以按内存使用多少排序。**

**D. 用sar -n DEV 1 10 或者 nload 命令可以查看网卡的流量（若没有nload命令，请使用yum install epel-release; yum install -y nload 安装）**

**E. 用iotop命令可以查看磁盘读写速度 （yum install -y iotop 安装）**

**网站时常出现too many connection的错误，请给出解决的方案？**

**Too many connections是因为连接mysql的客户端数量超过了它的最大限定值。mysql默认最大连接数为100个，而程序在打开连接后，并未能及时的关闭mysql的连接，造成连接数超出默认值。临时解决方案：在my.cnf里配置max\_connections=1024，更改后需要重启mysqld服务，若不能重启服务，也可以进入mysql中，执行set global max\_connections = 1024;。当然，最好的解决方案就是和开发人员一起找出不能正常关闭连接的问题，比如要通过查看慢查询日志分析执行慢的sql**

**Tomcat和Resin有什么区别，工作中你怎么选择？  
区别：Tomcat用户数多，可参考文档多，Resin用户数少，可考虑文档少  
最主要区别则是Tomcat是标准的java容器，不过性能方面比resin的要差一些  
但稳定性和java程序的兼容性，应该是比resin的要好。工作中选择：现在大公司都是用resin，追求性能；而中小型公司都是用Tomcat，追求稳定和程序的兼容**

**tomcat进程出现假死的情况？**

**先排查假死时机器的负载，是否因为内存不足，cpu使用过高，或者磁盘问题等导致。临时解决办法是写个脚本监控tomcat进程状况和模拟用户访问web站点，当出现不正常时，重启tomcat。另外还需要写一个监控系统各个硬件状态的脚本，综合分析tomcat假死的原因，当然根据tomcat、系统相关的日志来获取一些信息也是非常有必要的。**

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

**我们自己测了都没问题，只是这个客户访问有问题，那肯定是要先联系到这个客户，能远程最好，问一下客户的网络是不是正常的，访问其它的网站有没有问题，如果访问其它网站有问题，那叫客户解决本身网络问题。如果访问其它网站都没问题，用ping和nslookup解析一下我们的网站是不是正常的，让客户用IP来访问我们的网站是否可行，如果IP访问没问题，那就是客户的DNS服务器有问题或者DNS服务器解析不到我们的网站。还有一种可能就是跨运营商访问的问题，一般是靠CDN解决**

**怎么查看两台服务器之间的网络是不是正常的，服务器是禁ping的？\*\***

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

**Linux系统中病毒怎么解决？**

1. **最简单有效的方法就是重装系统  
   、 2）要查的话就是找到病毒文件然后删除**
2. **中毒之后一般机器cpu、内存使用率会比较高机器向外发包等异常情况，排查方法简单介绍下**
3. **top 命令找到cpu使用率最高的进程**
4. **一般病毒文件命名都比较乱，可以用 ps aux 找到病毒文件位置**
5. **rm -f 命令删除病毒文件**
6. **检查计划任务、开机启动项和病毒文件目录有无其他可以文件等**
7. **由于即使删除病毒文件不排除有潜伏病毒，所以最好是把机器备份数据之后重装一下**
8. **清空/etc/issue，去除系统及内核版本登录前的屏幕显示**

**10：网站出现500,502,503,400,403,404都是什么意思，怎么排查和解决？**

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

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

**503：服务器超载**

**504: 网关超时**

**400：语法错误**

**401：用户名或密码错误**

**403：服务器拒绝请求，禁止访问。检查服务器配置，是不是对客户端做了限制。**

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

**414：请求数据包头过长**

**499：服务端处理时间过长，客户端主动关闭了连接**

**301永久重定向 302临时重定向**

**11：请简述devops是什么?**

**答:DevOps是一种实现Dev(开发)与Ops(运维)工作流有效联合的思想。**

**持续集成/持续部署（以下称为CI/CD）是现代软件开发实践中构建、测试和部署应用到生产环境的支柱。CI/CD在弥合开发与各受影响团队之间的差距方面发挥着举足轻重的作用。CI有助于降低风险，通过自动处理项目不同开发人员的代码变更实现生产均等化。另一方面，CD使开发人员能够无缝地将集成代码交付给生产，从而提供快速有效的自动化流程，以便在不费工夫的情况下向客户发布新功能和更新。**

**简述集中式版本控制系统与分布式版本控制系统的区别?**

**答:集中式控制系统,必须联网才能工作,如果中央服务器挂了那么就完蛋了 分布式版本控制系统可以不用联网工作,因为参与开发的每人都有一个版本库, 并且分布式的没有中央服务器,可靠性高,也可以有中央服务器但是只是用来合 大家修改的代码.**

**请简述git本地仓库有哪三大区, 中间的那个区主要有什么作用?**

**答:工作区和暂存区和本地仓库,中间的区主要用来暂存你修改或更新的代码, 但是还不是最终的结果,又不想提交到版本库里,因为修改一点就提交到版本库 那样会造成版本库很乱,不方便管理,并且提交到版本库里之后删除不了.所以可以吧 修改好但又不确定是不是最终的修改的都可以先放到暂存区**

**CI/CD流程：**

**开发人员编写代码，将代码上传到GitLab服务器，Jenkins从GitLab服务器上面拉取最新代码数据到本地，根据Jenkins项目配置，将代码拉取到本地后，可以进行编译、测试、打包等工作，最后将数据包写入到共享服务器，应用服务器最后从共享服务器拉取数据实现上线功能。**

**请说明下面git命令的作用**

**git init 答:初始化git仓库**

**git add \* 答:把文件或修改的代码添加到暂存区**

**git rm xxx 答:删除暂存区的代码或文件**

**git status 答:查看工作区里有无修改或新添加的代码文件还未添加到暂存区**

**git commit -m "xxx" 答:把暂存区的代码或文件提交到版本库**

**git checkout -- xxx 答:撤销修改**

**git reset HEAD 答:回退版本**

**git log 答:查看提交的历史版本信息**

**git reflog 答:查看所有的操作历史。**

**git reset --hard xxx 答:还原版本**

**git branch 答:查看有多少个分支默认只要一个master分支**

**git branch xxx 答:创建自定义的分支**

**git checkout xxx 答:切换自定义的分支**

**git merge xxx 答:合并分支里的新内容**

**请简述github与gitlab分别是什么，各自的应用场景**

**答:github是开源项目代码托管平台 gitlab是一个开源的版本管理系统**

**github提供了公有仓库和私有仓库但是私有仓库是要付费的.可通过web界面访问 可以实现远程代码管理,跨区域管理**

**gitlab是一个自托管的git项目仓库可通过web界面访问,他和github应用场景相似, 可以实现远程代码管理,跨区域管理**

**持续集成的工作原理是将小的代码块推送到Git仓库中托管的应用程序代码库中，并且每次推送时，都要运行一系列脚本来构建、测试和验证代码更改，然后再将其合并到主分支中。**

**持续交付和部署相当于更进一步的CI，可以在每次推送到仓库默认分支的同时将应用程序部署到生产环境。**

**CI/CD流水线的13大好处**

**1. 卓越的代码质量2. 减少变更和审核时间**

**3. 加速发布周期4. 故障检测和隔离**

**5. 增强测试可靠性6. 减少任务堆积**

**7. 改善平均解决时间8. 降低成本**

**9. 增强透明度和责任制10. 简化沟通**

**11. 频繁的更新和维护12. 提升客户满意度**

**13. 增强的性能指标**

**说一下你们公司怎么发版的（代码怎么发布的）？**

**发布：jenkins配置好代码路径（GIT），然后拉代码，打tag。需要编译就编译，编译之后推送到发布服务器（jenkins里面可以调脚本），然后从分发服务器往下分发到业务服务器上，有时候也会先灰度发布一两台生产环境**

**回滚：按照版本号到发布服务器找到对应的版本推送**

说一下你们公司怎么发版的（代码怎么发布的）？

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

开发人员先在自己的电脑上开发功能代码，然后把代码推送到gitlab，测试通过webhook自动把代码拉取到测试机，开发人员先做一次测试，功能没问题，会把代码发布到预上线的机器上，此时测试人员会测试，才会正式发布到生产环境。有时候也会先灰度发布一两台生产环境

**12：简述raid0 raid1 raid5 三种工作模式的工作原理及特点\*\***

**RAID，可以把硬盘整合成一个大磁盘，还可以在大磁盘上再分区，放数据，还有一个大功能，多块盘放在一起可以有冗余（备份）**

**RAID整合方式有很多，常用的：0 1 5 10**

**RAID 0，可以是一块盘和N个盘组合**

**其优点读写快，是RAID中最好的**

**缺点：没有冗余，一块坏了数据就全没有了**

**RAID 1，只能2块盘，盘的大小可以不一样，以小的为准**

**10G+10G只有10G，另一个做备份。它有100%的冗余**

**缺点：浪费资源，成本高**

**RAID 5 ，3块盘，容量计算10\*（n-1）,损失一块盘，**

**特点，读写性能一般，读还好一点，写不**

描述raid 0、1、5的特点和优点?

Raid0条带卷，可以高效读写，硬盘空间利用率100%，raid1是复制卷可以实现数据的高可靠读写，硬盘空间利率50%，raid5兼得以上两种优点，硬盘空间利用率N-1，仅可损坏一块硬盘。

**13：现在给你三百台服务器，你怎么对他们进行管理？**

**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ansiable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。便于查阅每台服务器上的各种信息记录。**

**14：常用的Nginx模块，用来做什么？**

**rewrite地址重写模块**

**access访问控制模块**

**ssl安全加密模块**

**ngx\_http\_gzip\_module：网络传输压缩模块**

**ngx\_http\_proxy\_module 代理模块**

**ngx\_http\_upstream\_module定义后端服务器列表（用于定义可通过fastcgi传递、proxy传递、uwsgi传递、memcached传递和scgi传递指令来引用的服务器组）**

**Nginx 如何实现后端服务的健康检查？**

**方式一，利用 nginx 自带模块 ngx\_http\_proxy\_module 和 ngx\_http\_upstream\_module 对后端节点做健康检查。**

**方式二，利用 nginx\_upstream\_check\_module 模块对后端节点做健康检查。**

**为什么Nginx性能这么高？**

**因为它用的是异步非阻塞事件处理机制：运用了epoll模型，提供了一个队列，排队解决请求，所以性能高**

**Nginx怎么处理请求的？**

**nginx接收一个请求后，首先由listen（监听端口）和server\_name指令匹配server模块，再匹配server模块里的location（匹配用户的请求）**

**Nginx 如何实现页面的动静分离?**

**Nginx 可以使用 location匹配用户的请求,根据正则表达式（~\*）判断用户访问的是静态页面还是动态页面**

**nginx安全？**

**1.修改版本信息，并隐藏具体的版本号：server\_tokens off;**

**2.限制并发量：ngx\_http\_limit\_req\_module模块，可以有效降低DDOS攻击的风险**

**3.拒绝非法的请求：一般仅需要get和post**

**4.防止buffer溢出：**

**client\_body\_buffer\_size 1k;**

**client\_header\_buffer\_size 1k;**

**client\_max\_body\_size 1k;**

**large\_client\_header\_buffers 2 1k;**

**LVS：使用Linux内核集群实现一个高性能、高可用的负载均衡服务器，工作在4层（也就是TCP/IP中的传输层），一般LVS都是位于整个集群系统的最前端，由一台或者多台负载调度器（DS）组成，分发给应用服务器（RS）。基于IP负载均衡技术的IPVS模块来实现的，它具有很好的可伸缩性、可靠性和可管理性**

**LVS的优点是：**

**1、抗负载能力强、工作在网络4层之上仅作分发用，没有流量的产生这个特点也决定了它在负载均衡软件里的性能最强的，对内存和cpu资源消耗比较低，保证了均衡器IO的性能不会收到大流量的影响。**

**2、配置性比较低，这是一个缺点也是一个优点，因为没有可太多配置的东西所以并不需要太多接触，大大减少了人为出错的几率**

**3、工作稳定，因为其本身抗负载能力很强，自身有完整的双机热备方案**

**如LVS+Keepalived**

**4、工作在4层，应用范围较广，几乎可对所有应用做负载均衡**

**LVS的缺点是：**

**1、软件本身不支持正则表达式处理，不能做动静分离，现在许多网站在这方面都有较强的需求，这个是Nginx/HAProxy+Keepalived的优势所在**

**2、如果是网站应用比较庞大的话，LVS实施起来比较复杂**

**HAProxy：HAProxy提供高可用性、负载均衡以及基于TCP和HTTP应用的代理，支持虚拟主机，它是免费、快速并且可靠的一种解决方案。HAProxy特别适用于那些负载特大的web站点，这些站点通常又需要会话保持或七层处理。HAProxy运行在当前的硬件上，完全可以支持数以万计的并发连接。并且它的运行模式使得它可以很简单安全的整合进您当前的架构中，同时可以保护你的web服务器不被暴露到网络上。**

**特点：**

**1、HAProxy也是支持虚拟主机的，HAProxy的优点能够补充Nginx的一些缺点，比如支持Session的保持，Cookie的引导，同时支持通过获取指定的url来检测后端服务器的状态**

**2、HAProxy跟LVS类似，本身就只是一款负载均衡软件，单纯从效率上来讲HAProxy会比Nginx有更出色的负载均衡速度，在并发处理上也是优于Nginx的**

**3、HAProxy支持TCP协议的负载均衡转发，可以对MySQL读进行负载均衡**

**LVS、Nginx、HAproxy有什么区别？工作中你怎么选择？  
LVS： 是基于四层的转发，只能做端口的转发  
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**

**网站做了负载均衡，发现有些用户访问的时候经常需要重新登录是什么原因？**

**这是因为session没有会话保持，用户登录后，他的请求被分发到了另外一台服务器上。本来已经在A服务器上保存了session，可下次访问到了B服务器，而B服务器又不存在刚才保存的session信息，所以就会显示未登录。要解决这个问题很容易，做一个会话保持。比如可以设定1小时内用户的请求固定在一台服务器上。如果是nginx也可以设置ip\_hash。当然最好的解决方案是，使用redis存取session，实现session共享。**

**apache与nginx的区别？**

**1）nginx 轻量级，同样web 服务，比apache 占用更少的内存及资源 抗高并发能力大约是apache的10倍以上，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**

**2）nginx的静态处理性能力比apache强,nginx处理动态不行，一般动态请求要apache去做，nginx只适合静态和反向，**

**3）核心区别，apache是同步多进程，一个连接对应一个进程，nginx是异步的，很多个连接可以对应一个进程**

**使用nginx做负载均衡；后端一主一备；测试发现主正常的情况下，会有10%左右的流量访问备，请分析原因？**

**可能是nginx配置文件的max\_fails（失败超过指定次数会暂停或请求转往其它服务器）太小**

**15：**Docker

1. 3大服务模式

IAAS基础设施服务：CPU、内存、存储、网络和其它基本的计算资源

PAAS平台服务：中间件服务，编程语言、运行环境、技术支持

SAAS软件服务：根据需求购买应用软件服务

1. 容器概述
2. 核心技术：

Cgroups 资源管理

SELinux 安全

NameSpace 命名空间：UTS、NETWORK、MOUNT、USER、PID、IPC

1. 容器优点：

-相比于传统的虚拟化技术，容器更加简洁高效

-传统虚拟机需要安装操作系统

-容器使用共享公共库和程序

缺点：

-容器的隔离性没有虚拟化强

-共用Linux内核，安全性有先天缺陷

三、容器间通信模式

host模式，与宿主机共享网络

container模式，共享其他容器的网络命名空间

none模式，无网络模式

bridge模式，默认桥接网络

自定义网络

1. 镜像命令

Docker images //查看本机镜像

Docker search 镜像名称 //查找镜像

Docker pull 镜像名称:标签 //下载镜像

Docker push 镜像名称:标签 //上传镜像

Docker save 镜像名称:标签 -o 备份镜像名称.tar //备份镜像

Docker load -i 备份镜像名称:标签 //导入镜像

Docker rmi 镜像名称:标签 //删除镜像

Docker history 镜像名称:标签 //查看镜像制作历史

Docker inspect 镜像名称:标签 //查看镜像详细信息

Docker tag 镜像名称:标签 新镜像名称:新标签 //创建新的镜像

Docker commit 容器id 镜像名称:标签 //打包镜像

1. 容器命令

Docker run -itd 镜像名称:标签 启动命令 //创建启动并进入容器

Docker ps //查看容器

Docker ps -aq //查看所有容器和id

Docker rm -f //删除容器

Docker start | stop | restart 容器id //启动、停止、重启容器

Docker cp 本机文件路径 容器id:容器内路径 //拷贝本机文件到容器内

Docker cp 容器id:容器内路径 本机文件路径 //拷贝容器文件到本机内

Docker inspect 容器id //查看容器详细信息

Docker attach 容器id //进入容器，退回关闭容器

Docker exec -it 容器id 启动命令 //进入容器，退出不关闭容器

Docker top 查看容器输出

1. Dockerfile语法

FROM 基础镜像

RUN 制作镜像时执行的命令，可多个

ADD 复制文件到镜像，自动解压

COPY 复制文件到镜像，不解压

EXPOSE 开放端口

ENV 设置环境变量

WORKDIR 工作目录

CMD 容器启动时执行的命令，仅一条

创建镜像：Docker build -t 镜像名称:标签 Dockerfile所在目录

1. 发布容器

Docker run -itd -p宿主机端口:容器端口 镜像名称:标签

1. 容器共享卷

Docker run -itd -v 宿主机对象:容器内对象 镜像名称:标签

1. Docker私有仓库

1.仓库配置：yum -y install docker-distribution

systemctl enable --now docker-distribution

2.客户端配置：vim /etc/docker/daemon.json

{

"exec-opts": ["native.cgroupdriver=systemd"],

"registry-mirrors": ["https://hub-mirror.c.163.com"],

"insecure-registries":["192.168.1.100:5000", "registry:5000"]

}

1. 上传镜像

docker tag myos:latest 192.168.1.100:5000/myos:latest

docker push 192.168.1.100:5000/myos:latest

附：

docker images |awk '$2!="TAG"{print $1,$2}'|while read \_f \_v;\*do

docker tag ${\_f}:${\_v} 192.168.1.100:5000/${\_f##\*/}:${\_v};

docker push 192.168.1.100:5000/${\_f##\*/}:${\_v};

docker rmi ${\_f}:${\_v};

done

1. 查看

curl <http://仓库IP:5000/v2/_catalog>

curl <http://仓库IP:5000/v2/镜像名称/tags/list>

16：k8s

1. 核心
2. master 管理节点

apiserver（6443整个系统对外接口，供客户端和其它组件调用）

scheduler（10251负责对集群内部资源进行调度）

etcd（2380分布式键值数据库）

controllermanager（10252负责管理控制器）

2.node 计算节点

由kubelet（创建修改删除pod）、kube-proxy（k8s核心组件，实现service通信与负载均衡）、docker组成

kube-proxy代理模式：

k8s v1.0 用户空间代理模式

k8s v1.1 iptables代理模式

k8s v1.8 ipvs代理模式

3.image 镜像仓库

1. 语法：kubectl [command] [type] [name] [flags]

command子命令、type资源类型、name资源名称、flags附加参数

1. 系统命名空间

default 默认命名空间

kube-node-lease 高可用提供心跳监视的命名空间

kube-public 公共数据

kube-system 系统服务对象使用的命名空间

1. 排错命令
2. kubectl get 查看资源状态信息
3. kubectl describe 查看资源详细信息
4. kubectl logs 查看日志（容器终端信息）
5. 创建容器

kubectl run 资源名称 -it --image=私有仓库镜像:标签

1. 资源控制器
2. Deployment（为RS提供滚动更新）

ReplicaSet资源控制器、创建管理POD、可以扩容和缩容

1. DaemonSet（每个node节点启动运行pod）

系统服务类型：kube-proxy、flannel

1. job（单任务控制器）
2. CronJob（多次任务控制器，保留3个pod）
3. StatefulSet（有状态相关的pod）
4. Horizontal Pod Autoscaling（自动水平扩展pod，提高资源利用率）

\*污点策略

NoSchedule 不会被调度

PreferNoSchedule 尽量不调度

NoExecute 驱逐节点

设置污点标签：kubectl taint node node-0001 key=value:NoSchedule

删除污点标签：kubectl taint node node-0001 key-

查看污点标签：kubectl describe nodes | grep Taints

\*资源文件

create 创建资源对象

apply 声明更新资源对象

delete 删除资源对象

1. 查询资源对象文件格式（json、yaml）：kubectl get 资源对象 资源名称 -o 格式
2. 扩容：kubectl scale deployment.apps apache --replicas=3
3. 修改配置即时生效：kubectl edit deployment.apps apache
4. 集群更新与回滚：Recreate 重建式更新、rollingUpdate滚动更新

查看历史版本，回滚：

kubectl rollout history deployment myapache

kubectl rollout undo deployment myapache --to-revision=1





1. POD

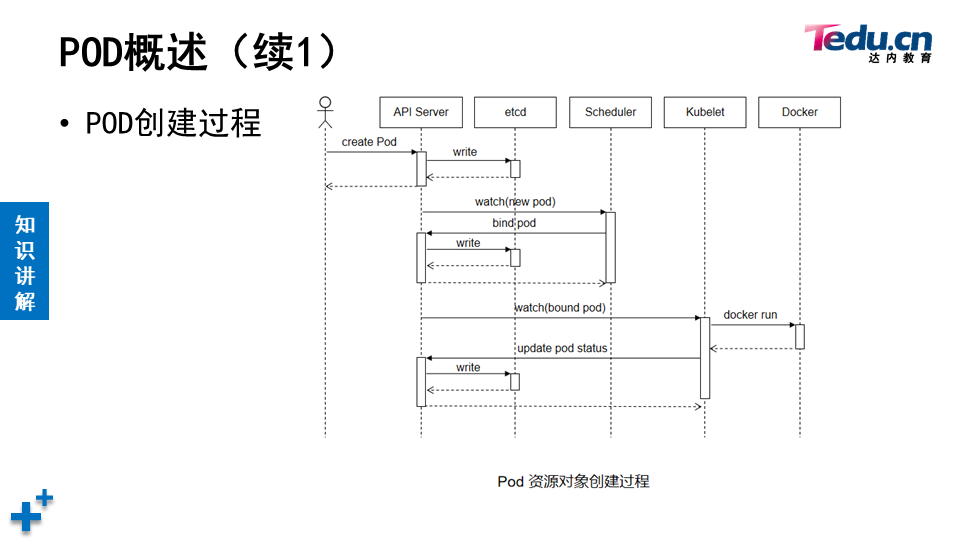
概念：pod是kubernetes调度的基本单元，一个pod包涵一个或多个容器，这些容器使用相同的网络命名空间和端口号，pod是一个服务的聚合单元，支持横向扩展和复制

同一个Pod共享进程（PID）

同一个Pod共享网络IP及权限（NETWORK）

同一个Pod共享IPC通信信号（IPC）

同一个Pod共享主机名（UTS）

\*POD创建过程

具体的创建步骤包括：

1. 客户端通过kubectl命令行（或API Server的Restful API）提交创建Pod请求。

2、API Server处理用户请求，存储Pod数据到etcd

3、调度器通过API Server查看未绑定的Pod。尝试为Pod分配主机

1. 过滤主机 (调度预选)：调度器用一组规则过滤掉不符合要求的主机。比如Pod指定了所需要的资源量，那么可用资源比Pod需要的资源量少的主机会被过滤掉
2. 主机打分(调度优选)：对上一步筛选出的符合要求的主机进行打分。在主机打分阶段，调度器会考虑一些整体优化策略，使用最低负载的主机
3. 选择主机：选择打分最高的主机，进行绑定操作，并将结果存储到etcd中

7、kubelet根据调度结果执行Pod创建操作： 绑定成功后，scheduler会调用APIServer的API在etcd中创建一个boundpod对象，描述在一个工作节点上绑定运行的所有pod信息。运行在每个工作节点上的kubelet也会定期与etcd同步boundpod信息，一旦发现应该在该工作节点上运行的boundpod对象没有更新，则调用Docker API创建并启动pod内的容器

\*Pod生命周期：创建主容器（main container）为必需操作，可选操作包括运行初始化容器（init container）、容器启动后钩子（post start hook）、容器存活性探测（liveness probe）、就绪性探测（readiness probe）、容器终止前钩子（pre stop hook）

\*Pod phase（相位）

Pending 容器创建过程中，但它尚未被调度完成

Running 所有容器创建完成

Succeeded 所有容器都已经成功终止了并不会被重启

Failed Pod 所有容器中至少有一个容器退出是非0状态

Unknown 无法正常获取到Pod对象的状态信息

1. node标签管理
2. 查看标签：kubectl get node --show-labels
3. 设置标签：kubectl label nodes <node-name> k=v
4. 删除标签： kubectl label nodes <node-name> -

九、服务发现



服务端口：

port：service暴露在cluster ip上的端口，供集群内部服务访问使用

targetPort：pod上容器服务监听的端口

17：SQL语句分类

1. 数据查询语言（DQL：select）

2.数据定义语言（DDL：create、alter、drop）

3.数据操纵语言（DML：insert、update、delete）

4.数据控制语言（DCL：grant、revoke）

1. 约束
2. 用于限制表中的数据，保证表中数据的准确性和可靠性
3. 分类：（表级pfu）primary key主键、foreigh key外键（不支持列级）、

unique唯一、not null非空、default默认值

3.表结构：字段名、类型、空、键值、默认值、额外设置

三、事务

1.数据库事务指的是一组数据操作，可以用来维护数据库的完整性，保证成批的SQL语句要么全部执行，要么全都不执行；只有使用了innodb数据库引擎的库或表才支持事务。

1. 事务必须满足的4个条件（ACID）

A原子性：事务要么都执行要么都不执行，在执行过程中发生错误会被回滚到开始的状态

C一致性：在事务的开始和结束，数据库的完整性没有被破坏

I隔离性：允许多个并发事务同时对数据进行读写和修改，防止交叉执行导致数据的不一致

D持久性：事务处理结束后，对数据的修改是永久的

1. 事务控制语句：begin开始、rollback回滚、commit提交
2. 事务隔离：可重复读、不可重复读、脏读、幻读
3. 事务隔离级别：读未提交、读提交、可重复读（mysql默认）、串行化

隔离级别 | 出现脏读 | 出现不可重读 | 出现幻读 |

读未提交 | 可能 |可能 | 可能 |

读提交 | 不可能 |可能 | 可能 |

可重复读 | 不可能 | 不可能 | 可能 |

串行化 | 不可能 | 不可能 | 不可能 |

1. 视图：视图是由数据库中的一个表或多个表导出的虚拟表，方便用户对数据的操作。可以隐藏一些数据，可以使复杂的查询易于理解和使用
2. 存储过程：是可编程的函数，由一组SQL语句和控制结构组成，提高了代码的重用性，减少了编译次数和数据库的连接次数，提高了效率
3. 索引（index）
4. 索引是帮助mysql高效获取数据的数据结构，可以用来快速查询数据库表中的特定记录，主要有B+Tree和Hash索引
5. 优点：可以大大提高Mysql的检索速度；减小服务器需要扫描的数据量；可以帮助服务器避免排序和临时表
6. 缺点：降低了更新表的速度，在对表增删改时，要保存数据和索引文件；会占用磁盘空间；
7. 分类：普通索引、唯一索引、全文索引、单列索引、多列索引
8. 用户授权（笔记第一周的6天）
9. 备份
10. 备份策略：完全备份、差异备份、增量备份
11. 备份命令

mysqldump -u用户名 -p密码 > 路径/文件名.sql

例：备份多个库/所有/某个库 表

mysqldump -uroot -p密码 -B/-A（--all-databases)/某库名 表名） > 路径/文件名.sql

1. Percona
2. 解决了物理备份跨平台性差、备份时间长、冗余备份浪费存储空间的缺点；

解决了mysqldump备份过程中会阻塞操作、效率较低和还原速度慢、锁表的缺点

2.Percona特点：在线热备不锁表

3.组件：xtrabackup、innobackupex

4.完全备份格式

innobackupex -u用户名 -p密码 备份目录名

选项：

--no-timestamp 不用日期命名生成子目录名

--apply-log 准备恢复数据

--copy-back 恢复数据

--databases=库1 库2/库.表

1. 增量备份格式（需要一个完全备份）

innobackupex -u用户名 -p密码 --incremental 完全备份 --incremental-basedir=备份目录名

选项：

--apply-log 准备恢复数据

--redo-only 日志合并

--incremental-dir=完全备份目录名

一、Mysql主从同步

1.原理：当开启主从复制时，主库会生成一个logdump线程，用来给从库的I/O线程传binlog。从库会生成两个线程，一个I/O线程去请求主库的binlog并写到自己的relaylog（中继日志）文件中，另外一个SQL线程会读取relaylog文件中的日志，来实现主从操作一致，最终达到主从数据同步。

1. 主服务器需要做哪些配置?

设置server\_id；启用binlog日志；授权用户；查看binlog日志信息

1. 从服务器需要做哪些配置?

设置server\_id；确保与主库数据一致；指定主库信息；启动slave程序；查看状态信息

1. 主从同步结构：一主一从、一主多从、主从从、互为主从
2. 复制模式

同步复制：主库执行完一次事务后，等待所有的从库执行完该事务，才将结果返回给客户端

异步复制：主库执行完一次事务后立即将结果返回给客户端，不关心从库是否已经同步数据

半同步复制：介于同步复制和异步复制之间，主库执行完一次事务后，至少等待一台从库同步数据完成才将结果返回给客户端

1. 导致从服务器 IO线程报错的原因有哪些?

答: IO线程报错的原因有两个,第一是指定主库信息时参数信息有误;第二是安全限制(firewall,selinux)

5.导致从服务器 SQL 线程报错的原因有哪些?

答:SQL 线程:执行本机中继日志文件里的 SQL 命令,把数据写进本机数据库里;

出错原因:一般是执行中继日志文件里 sql 命令用到的库或表在本机不存在;

修复 SQL线程:查看本机和主数据库服务器中的库和表使本机有主数据库服务器中的库或表

1. MySQL 主从同步延迟是怎么产生的？如何解决?

答：主从复制单线程，如果主库写并发太大，来不及传到从库，就会导致延迟；还有慢SQL语句过多，网络延迟，从库硬件较差都会导致数据同步延迟。解决：数据库前端加缓存层；使用更高版本的数据库支持多线程复制；使用多台slave来分摊读请求；提高硬件性能等

1. Mysql读写分离
2. 原因：数据库的读取频率高于写入频率，单个数据库实例在写入的时候会影响性能
3. 实现：主服务器负责增删改请求；从服务器负责读请求
4. 工具：mysql-proxy、mycat、maxscale
5. Mysql数据分片
6. 原因：关系型数据库本身比较容易成为系统瓶颈，数据切分的目的在于减少数据库的负担，缩短查询时间
7. 实现：数据切分就是将数据分散存储到多个数据库中，通过扩充主机的数量缓解单一数据库的性能问题，从而达到提升数据库操作性能的目地
8. 方式：垂直切分；水平切分
9. Mycat（端口：8066）
10. mycat是基于java的分布式数据库系统中间件，为高并发环境的分布式存储提供解决方案，适合数据大量写入的存储需求
11. 支持Mysql、Oracle、Sqlserver、Mongodb
12. 分配规则：枚举法、固定分片、求模法......
13. 工作过程：解析SQL命令涉及到的表，然后看对表的配置，如果有分片规则，则获取SQL命令里分片字段的值，并匹配分片函数，获得分片列表，然后将SQL命令发往对应的服务器去执行，最后收集和处理分片数据，将结果返回给客户端
14. MHA集群
15. 介绍：MHA是一套优秀的实现Mysql高可用的解决方案，数据库的自动故障切换能做到在0-30s内完成，确保在故障切换过程中最大限度保证数据的一致性，以达到真正意义上的高可用
16. 组成：MHA Manager管理节点、MHA Node数据节点
17. 工作过程：由管理节点manager定时探测集群和总的master节点，当主节点故障时，管理节点自动将拥有最新数据的slave提升为新的主

4.MHA 集群中,配置 MySQL 主从同步时,你认为使用那种复制模式合理?

答：半同步复制；既提高了客户端的体验度,保证了至少有一个从库保存了主上的 binlog 日志文件在本机的中继日志上

1. PXC
2. 介绍：是基于galera的mysql高可用集群解决方案
3. PXC 集群中有哪些端口,各种的作用是什么?

答：3306+4567 (数据库服务+集群通信)

1. PXC 集群与 MHA 相比较有哪些优点?

答：数据强一致性,无同步延迟；

多线程复制,部署使用简单；

没有主从切换操作,无需使用虚拟 VIP；

支持 INNODB 存储引擎；

支持节点自动加入,无需手动拷贝数据

1. Redis（6379）
2. 关系型数据库：Mysql、MariaDB、Oracle、DB2、SQL Server

按照预先设置的组织结构，将数据存储在物理介质上，数据之间可以做关联操作

1. 非关系型数据库（nosql）：Redis、Memcached、MongoDB

不需要预先定义数据存储结构，每条记录可以有不同的数据类型和字段个数

1. redis是c语言开发的一个开源高性能键值对的内存数据库，是一种nosql数据库，性能优秀，数据在内存中，读写速度非常快；支持数据持久化，可以将内存中数据保存在磁盘中；可以做主从复制，哨兵，高可用
2. redis数据类型：字符串、散列（hash）、列表、集合、有序集合
3. redis数据过期处理：定期删除、惰性删除、内存淘汰机制
4. 缓存雪崩：是指在某一个时间段，缓存集中过期失效，大量的请求都跳过缓存直接到数据库中 解决：不设置超时时间，避免过期；在失效时间上加随机的过期时间，避免集中过期；有相应的限流降级策略
5. 缓存击穿：是指大量的请求集中对一个热点key进行访问，当这个key失效的瞬间，持续的大并发直接落到数据库上，就在这个key的点上击穿了缓存 解决：加互斥锁，避免大量请求同时打库；设置key不过期，更新时直接覆盖
6. 缓存穿透：指缓存和数据库中都没有数据，而用户不断发起请求，这会给服务器带来很大的压力 解决：把空对象存到缓存中，并设置一个较短的过期时间；接口层增加校验；利用布隆过滤器

<https://www.cnblogs.com/Fairy-02-11/p/6182478.html>

<http://blog.csdn.net/mishifangxiangdefeng/article/details/48977269>

1.哨兵服务默认端口?

答：26379

1. 集群通信默认端口?

答：16379

3.哨兵服务是如何工作的?

答：主库宕机后,会选举一台新的主服务器,客户端访问新的主服务器写入数据后,再重启原先宕机的主后会发现他自动成为当前新主的从服务器,并同步更新宕机后所写入的数据

4.你认为配置几台哨兵服务器比较合理,为什么?

答：通常哨兵配置数量为奇数；偶数投票表决时易出现平票,就有可能会出现双主,出现了双

主集群就崩了,所以不管怎么样只能同时有一个主存在

5.Redis 集群最低要求几主几从?为什么?

答：官方说明必须要三个 Master 节点和三个Slave节点；节点的有效性是靠会,50%的节点认为失效,就算失效。

6.简述redis集群原理和 hash 槽的作用?

答：在redis集群中，所有的redis节点彼此互联，节点内部使用二进制协议优化传输速度和带宽，当一个节点挂掉后，集群中超过半数的节点检测失效时才认为该节点失效；Redis 集群中内置了 16384 个哈希槽,当需要在 Redis 集群中放置一个 key-value 时,redis先对 key 使用 crc16 算法算出一个结果,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,redis会根据节点数量大致均等的将哈希槽映射到不同的节点。

1. redis集群的三种模式：

主从模式、sentine模式、cluster模式

8.什么情况下集群无法存/取数据?(默认 16 个数据库)

答：redis集群使用了主从复制模型,每个主节点都会有一个或多个从节点，在没有复制模型的情况下,如果某个主节点故障了,那么整个集群就会以为缺少某个范围的槽而不可用。

9.redis和memcached的区别

1. 存储方式上：memcached把数据全部存在内存，断电即失效

redis有部分数据存在硬盘上，保证数据的持久性

1. 数据类型上：memcached只支持简单的key-value，而redis支持5种数据类型
2. value大小：redis可以达到1GB，而memcached只有1MB
3. 存储引擎
4. 概念：数据库存储引擎是数据库底层软件组织，决定了一个数据库的性能效率
5. 分类：MyISAM、InnoDB、Memory......  
   3.查看：show engines；
6. 特点

MyISAM（5.5版本前默认）：访问速度快，不支持事务和外键，适用于对事务的完整性没有要求的场景，在web、数据仓库中应用广泛

InnoDB（5.5后默认）：支持事务和外键，相比于myisam，innodb写的处理效率差一点，并且会占用更多的磁盘空间来存储数据和索引，在计费系统、财务系统中应用广泛

Memory:是用保存在内存中的数据来创建表，默认使用hash索引，访问速度特别快，缺点就是数据库服务关闭，数据就会丢失，对表的大小也有限制

1. 数据持久化
2. RDB：创建一个子进程来执行持久化，先将数据写入临时文件，持久化过程结束后，再用这个临时文件替换上次持久化好的文件；适合大规模数据恢复，对数据完整性要求不是非常高的场合；意外宕机时，丢失最后一次持久化的所有数据
3. AOF：追加方式记录写操作的文件，默认没有开启，可以灵活的设置持久化方式，意外宕机时仅可能丢失1s的数据；缺点：持久化文件的体积通常会大于RDB方式，执行fsync策略时的速度可能会比RDB慢
4. AOF写操作方式：

appendfsync always：时时记录，并完成磁盘同步

appendfsync everyses：每秒记录一次，并完成磁盘同步

appendfsync no：写入aof，，不执行磁盘同步

18：HTTP常见状态码？

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

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

503：服务器超载

504：网关超时，根据后端服务器实际处理情况，调整后端请求超时时间。

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

401：用户名或密码错误

403：服务器拒绝请求，禁止访问。检查服务器配置，是不是对客户端做了限制。

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

414：请求数据包头过长

499：服务端处理时间过长，客户端主动关闭了连接，根据实际Nginx后端服务器的处理时间修改客户端超时时间。

301永久重定向 302临时重定向

nginx应用场景：

1.http服务器，可以做网页静态服务器

2.虚拟主机，可以实现在一台服务器虚拟出多个网站

3.反向代理，负载均衡

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

last : 表示完成当前的rewrite规则

break : 停止执行当前虚拟主机的后续rewrite指令集

permanent : 返回301永久重定向，地址栏会显示跳转后的地址

redirect : 返回302临时重定向，地址栏会显示跳转后的地址

nginx反向代理模式（负载均衡策略）

1）轮询：每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器 down 掉，能自动剔除。

2）权重：指定轮询几率，weight和访问比率成正比，用于后端服务器性能不均的情况。

3）ip\_hash：每个请求按访问ip的hash结果分配，这样每个访客固定访问一个后端服务器，可以解决session的问题。

4）fair（第三方）：按后端服务器响应时间来分配,时间短的优先分配。

5）url\_hash（第三方）：按访问url的hash结果来分配请求，使同样的url定向到同一个后端服务器，后端服务器为缓存时比较有效。

常用的Nginx模块，用来做什么？

rewrite地址重写模块（缩短URL、隐藏实际路径提高安全性、易于用户记忆和输入、易于被搜索引擎收录）

access访问控制模块

ssl安全加密模块

ngx\_http\_gzip\_module 网络传输压缩模块

ngx\_http\_proxy\_module 代理模块

ngx\_http\_upstream\_module定义后端服务器列表（用于定义可通过fastcgi传递、proxy传递、uwsgi传递、memcached传递和scgi传递指令来引用的服务器组）

Nginx 如何实现后端服务的健康检查？

方式一，利用 nginx 自带模块 ngx\_http\_proxy\_module 和 ngx\_http\_upstream\_module 对后端节点做健康检查。

方式二，利用 nginx\_upstream\_check\_module 模块对后端节点做健康检查。

在proxy模块中你配置过哪些参数？

proxy\_buffer\_size 4k：设置缓冲区大小，设置的过小，可能会产生502错误

proxy\_connect\_timeout：设置服务器链接的超时时间，设置的太小，会报504错误

proxy\_read\_timeout 60配置在http,server,location，用于设置服务器读取应答内容的超时时间

为什么 Nginx 不使用多线程？

Nginx: 采用单线程来异步非阻塞处理请求(epoll)，不会为每个请求分配 cpu 和内存资源，节省了大量资源消耗，同时也减少了CPU 的上下文切换。所以才使得 Nginx 支持更高的并发。

为什么Nginx性能这么高？

因为它用的是异步非阻塞事件处理机制：运用了epoll模型，提供了一个队列，排队解决请求，所以性能高

Nginx怎么处理请求的？

nginx接收一个请求后，首先由listen和server\_name指令匹配server模块，再匹配server模块里的location

Nginx 如何实现页面的动静分离?

Nginx 可以使用 location匹配用户的请求,根据正则表达式（~\*）判断用户访问的是静态页面还是动态页面

nginx安全？

1.修改版本信息，并隐藏具体的版本号：server\_tokens off;

2.限制并发量：ngx\_http\_limit\_req\_module模块，可以有效降低DDOS攻击的风险

3.拒绝非法的请求：一般仅需要get和post

4.防止buffer溢出：

client\_body\_buffer\_size 1k;

client\_header\_buffer\_size 1k;

client\_max\_body\_size 1k;

large\_client\_header\_buffers 2 1k;

nginx调优？

优化并发数量，设置worker\_processes进程与CPU核心数一致

启用页面压缩功能，开启高效传输模式，设置连接超时时间

增加浏览器缓存机制

设计自定义报错页面

Nginx最大打开文件数，worker\_rlimit\_nofile 65535

Nginx采用epoll事件模型，处理效率高

fastcgi调优、内核参数优化等

lvs/nginx/haproxy优缺点

Nginx的优点是：

1、工作在网络的第7层，可以针对http应用做一些分流的策略，它的正则比HAProxy更为强大，

2、Nginx对网络稳定性的依赖非常小，安装和配置比较简单，测试起来比较方便，它基本能把错误用日志打印出来

3、可以承担高并发且稳定，负载度比LVS相对小些

4、Nginx可以通过端口检测到服务器内部的故障，比如根据服务器处理网页返回的状态码、超时等，不过其中缺点就是不支持url来检测。

5、Nginx不仅仅是一款优秀的负载均衡器和反向代理软件，它同时也是功能强大的Web应用服务，LNMP也是近几年非常流行的web架构，在高流量的环境中稳定性也很好。

6、Nginx也可作为静态网页和图片服务器，还有Nginx社区非常活跃，第三方模块也很多

Nginx的缺点是：

1、Nginx仅能支持http、https和Email协议，这样就在适用范围上面小些，这个是它的缺点

2、对后端服务器的健康检查，只支持通过端口来检测，不支持通过url来检测，不支持Session的直接保持，但能通过ip\_hash来解决

LVS：使用Linux内核集群实现一个高性能、高可用的负载均衡服务器，工作在第4层（也就是TCP/IP中的传输层），一般LVS都是位于整个集群系统的最前端，由一台或者多台负载调度器（DS）组成，分发给应用服务器（RS）。基于IP负载均衡技术的IPVS模块来实现的，它具有很好的可伸缩性、可靠性和可管理性

LVS的优点是：

1、抗负载能力强、工作在4层仅作分发用，没有流量的产生这个特点也决定了它在负载均衡软件里的性能最强的，对内存和cpu资源消耗比较低，保证了均衡器IO的性能不会收到大流量的影响。

2、配置性比较低，这是一个缺点也是一个优点，因为没有可太多配置的东西所以并不需要太多接触，大大减少了人为出错的几率

3、工作稳定，因为其本身抗负载能力很强，自身有完整的双机热备方案：LVS+Keepalived

4、工作在4层，应用范围较广，几乎可对所有应用做负载均衡

LVS的缺点是：

1、软件本身不支持正则表达式处理，不能做动静分离，现在许多网站在这方面都有较强的需求，这个是Nginx/HAProxy+Keepalived的优势所在

2、如果是网站应用比较庞大的话，LVS实施起来比较复杂

HAProxy：HAProxy提供高可用性、负载均衡以及基于TCP和HTTP应用的代理，支持虚拟主机，它是免费、快速并且可靠的一种解决方案。HAProxy特别适用于那些负载特大的web站点，这些站点通常又需要会话保持或七层处理。HAProxy运行在当前的硬件上，完全可以支持数以万计的并发连接。并且它的运行模式使得它可以很简单安全的整合进您当前的架构中，同时可以保护你的web服务器不被暴露到网络上。

特点：

1、HAProxy也是支持虚拟主机的，HAProxy的优点能够补充Nginx的一些缺点，比如支持Session的保持，Cookie的引导，同时支持通过获取指定的url来检测后端服务器的状态

2、HAProxy跟LVS类似，本身就只是一款负载均衡软件，单纯从效率上来讲HAProxy会比Nginx有更出色的负载均衡速度，在并发处理上也是优于Nginx的

3、HAProxy支持TCP协议的负载均衡转发，可以对MySQL读进行负载均衡

HAProxy的负载均衡算法有4种：

①roundrobin，轮询，这个不多说，这个是负载均衡基本都具备的；

② static-rr，权重，建议关注；

③leastconn，最少连接者先处理，建议关注；

④ source，根据请求源IP

Keepalived的工作原理？

Keepalived是一个保证集群高可用的服务软件，用来防止单点故障，使用VRRP协议实现。在master和backup之间通过master主动降低自己的权值或者backup检测到master出现故障时，backup将会接管master的工作，继续服务。

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、维护及全局配置文件的加载和解析。check负责健康检查，包括常见的各种检查方式，vrrp模块是来实现VRRP协议的

LVS三种模式的工作过程？

LVS 有三种负载均衡的模式，分别是VS/NAT（nat 模式） VS/DR(路由模式) VS/TUN（隧道模式）

LVS、Nginx、HAproxy有什么区别？工作中你怎么选择？  
LVS： 是基于四层的转发，只能做端口的转发  
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  
  
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是专业的代理服务器配置简单，所以中小型企业推荐使用HAproxy

什么是中间件？什么是jdk？   
中间件是一种独立的系统软件或服务程序，是连接两个独立应用程序或独立系统的软件

分布式应用借助这种软件在不同的技术之间共享资源；jdk是Java的开发工具包

讲述一下Tomcat8005、8009、8080三个端口的含义？  
8080一般应用使用

8005关闭时使用

8009为AJP端口，即容器使用，如Apache能通过AJP协议访问Tomcat的8009端口

tomcat优化？

修改tomcat最大线程连接数；tomcat内存优化，在启动时告诉JVM要多大内存

？

to mcat进程出现假死的情况

先排查假死时机器的负载，是否因为内存不足，cpu使用过高，或者磁盘问题等导致。临时解决办法是写个脚本监控tomcat进程状况和模拟用户访问web站点，当出现不正常时，重启tomcat。另外还需要写一个监控系统各个硬件状态的脚本，综合分析tomcat假死的原因，当然根据tomcat、系统相关的日志来获取一些信息也是非常有必要的。

Tomcat工作模式

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

进入Tomcat的请求可以根据Tomcat的工作模式分为如下两类：

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

Tomcat和Resin有什么区别，工作中你怎么选择？  
区别：Tomcat用户数多，可参考文档多，Resin用户数少，可考虑文档少  
最主要区别则是Tomcat是标准的java容器，不过性能方面比resin的要差一些  
但稳定性和java程序的兼容性，应该是比resin的要好。工作中选择：现在大公司都是用resin，追求性能；而中小型公司都是用Tomcat，追求稳定和程序的兼容

使用nginx做负载均衡；后端一主一备；测试发现主正常的情况下，会有10%左右的流量访问备，请分析原因？

可能是nginx配置文件的max\_fails（失败超过指定次数会暂停或请求转往其它服务器）太小

apache与nginx的区别？

1）nginx 轻量级，同样web 服务，比apache 占用更少的内存及资源 抗高并发能力大约是apache的10倍以上，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

2）nginx的静态处理性能力比apache强,nginx处理动态不行，一般动态请求要apache去做，nginx只适合静态和反向，

3）核心区别，apache是同步多进程，一个连接对应一个进程，nginx是异步的，很多个连接可以对应一个进程

网站做了负载均衡，发现有些用户访问的时候经常需要重新登录是什么原因？

这是因为session没有会话保持，用户登录后，他的请求被分发到了另外一台服务器上。本来已经在A服务器上保存了session，可下次访问到了B服务器，而B服务器又不存在刚才保存的session信息，所以就会显示未登录。要解决这个问题很容易，做一个会话保持。比如可以设定1小时内用户的请求固定在一台服务器上。如果是nginx也可以设置ip\_hash。当然最好的解决方案是，使用redis存取session，实现session共享。