最后一次周考

！！！以下答案仅供参看，部分题目答案详见下面链接：
http://m.7624.net/iyule/5005629/20180914A167D200.html

http://blog.51cto.com/sandshell/2072733

1. 简单（50 分）

简述 raid0 raid1 raid5 三种工作模式及特点

RAID0：条带模式，至少 2 块磁盘，通过并发读写提高效率
RAID1：镜像模式，至少 2 块磁盘，通过镜像备份提高磁盘设备的可靠性
RAID5：高性价比模式，至少 3 块磁盘，其中 1 块磁盘容量用来存放恢复校验数据
RAID10：条带+镜像模式，相当于 RAID1 +RAID0，至少 4 块磁盘，读写效率及可靠性都更高

squid、Varnish 和 Nginx 有什么区别？工作中你怎么选择

Squid、Varinsh 和 Nginx 都是代理服务器

什么是代理服务器：
能当替用户去访问公网，并且能把访问到的数据缓存到服务器本地，等用户下次再访问相同的源的时候，代理服务器直接从本地回应给用户，当本地没有的时候，我代替你去访问公网，我收你的请求，我先在我自己的本地缓存找，如果我本地缓存有，我直接从我本地的缓存里回复，如果我在我本地没有找到你要访问的缓存的数据，那么代理服务器就会代替你去访问公网

区别：
1）Nginx 本来是反向代理/web 服务器，用了插件可以做做这个副业是本身不支持特性挺多，只能缓存静态文件
2）从这些功能上。varnish 和 squid 是专业的 cache 服务，而 nginx 这些是第三方模块完成
3）varnish 本身的技术上优势要高于 squid，它采用了可视化页面缓存技术在内存的利用上，Varnish 比 Squid 具有优势，性能要比 Squid 高。还有强大的通过 Varnish 管理端口，可以使用正则表达式快速、批量地清除部分缓存它是内存缓存，速度一流，但是内存缓存也限制了其容量，缓存页面和图片一般是挺好的
4）squid 的优势在于完整的庞大的 cache 技术资料，和很多的应用生产环境

工作中选择：
要做 cache 服务的话，我们肯定是要选择专业的 cache 服务，优先选择 squid 或者 varnish。

什么是 VLAN？作用是什么？

Vlan ，即虚拟局域网
作用：广播控制、 提高安全性、 带宽利用、 减少延迟
默认交换机接口在 vlan1 中

网线 568B 线序是什么？用于数据通信的是哪几根？

白橙 橙 白绿 蓝 白蓝 绿 白棕 棕
白橙、橙来发送数据，白绿、绿来接收数据

说说 OSI 七层模型

物理层 数据链路层 网络层 传输层 会话层 表示层 应用层

2. 中等（100 分）

mysql 主从同步原理

（1）Slave 上面的 IO 线程连接上 Master，并请求从指定日志文件的指定位置之后的日志内容
（2）Master 接收到来自 Slave 的 IO 线程的请求后，通过负责复制的 IO 线程根据请求信息读取指 定日志指定位置之后的日志信息，返回给 Slave 端的 IO 线程
（3）Slave 的 IO 线程接收到信息后，将接收到的日志内容依次写入到 Slave 端的 Relay Log 文件
（4）Slave 的 SQL 线程检测到 Relay Log 中新增加了内容后，会马上解析该 Log 文件中的内容成为在 Master 端真实执行时候的那些可执行的 Query 语句，并在自身执行这些 query 语句

对 docker 的理解？为什么要有 docker？docker 重要的命令

Docker 是完整的一套容器管理系统，它提供了一组命令，让用户方便直接的使用容器技术，而不需过多的关心底层内核技术,相比于传统的虚拟技术，容器更加计划简洁高效，容器使用共享的公共库和程序

- docker images //查看镜像列表
- docker history //查看镜像制作历史
- docker inspect //查看镜像底层信息
- docker pull //下载镜像
- docker push //上传镜像

- docker rmi //删除本地镜像
- docker save //镜像另存为 tar 包
- docker load //使用 tar 包导入镜像
- docker search //搜索镜像
- docker tag //修改镜像名称和标签
- docker run //运行容器
- docker ps //查看容器列表
- docker stop //关闭容器
- docker start //启动容器
- docker restart //重启容器
- docker attach|exec //进入容器
- docker inspect //查看容器底层信息
- docker top //查看容器进程列表
- docker rm //删除容器
- docker ps 查看正在运行的容器
- docker ps -a 查看所有容器列表
- docker ps -aq 仅显示容器 id

svn 与 git 区别 awk

（1）git 是分布式的；SVN 是集中式的
（2）git 有暂存区的概念，提交修改是先提交至暂存区，然后再从暂存区提交至 master 分支，且提交至 master 分支后，暂存区就没有内容了；SVN 是直接提交至中央服务器
（3）权限分配方面：SVN 能给用户分配读写的权限；git 需要借助工具才能分配
（4）使用 git 切分支比 SVN 操作简单方便，很多倍
（5） Git 把内容按元数据方式存储，而 SVN 是按文件。在更新、提交的速度上 git 有优势

KVM，docker 区别

（1）Docker 容器的启动可以在秒级实现，这相比 KVM 方式要快得多
（2）容器除了运行其中应用外，基本不消耗额外的系统资源，使得应用的性能很高，同时系统的开销尽量小。
（3）虚拟化技术依赖物理 CPU 和内存，是硬件级别的；而 docker 构建在操作系统上，甚至可以在虚拟机上运行

常用的 nginx 模块，用来做什么？

rewrite 模块，实现重写功能
access 模块：来源控制
ssl 模块：安全加密
http_gzip_module：网络传输压缩模块
http_proxy_module 模块实现代理 awk

http_upstream_module 模块实现定义后端服务器列表
cache_purge 实现缓存清除功能

keepalived 用到的协议
VRRP 虚拟路由冗余协议

给了日志文件，统计 30 天内，访问 IP 最多的前 5

通过 t>="1/Sep/2018:16:35:57" && t<"30/Sep/2018:16:35:57"来设置 30 天时间段

awk -F'[]|[|+]' '{t=$2; if(t>="1/Sep/2018:16:35:57" && t<"30/Sep/2018:16:35:57") print}' access.log | awk '{print $1}' | sort -nr | uniq -c | head -5

什么是中间件？

中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源。中间件位于客户机/服务器的操作系统之上，管理计算机资源和网络通讯是连接两个独立应用程序或独立系统的软件。相连接的系统，即使它们具有不同的接口但通过中间件相互之间仍能交换信息。执行中间件的一个关键途径是信息传递通过中间件，应用程序可以工作于多平台或 OS 环境。

什么叫 CDN？

CDN 的全称是 Content Delivery Network，即内容分发网络 DN 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

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

如何将本地 80 端口的请求转发到 8080，当前主机 ip 192.168.1.1 （iptables）

iptables -A PREROUTING -d 192.168.1.1 -p tcp -dport 80 -j DNAT --to-destination 192.168.1.1:8080

困难（100）

ansible 如何保证批量命令的正确性？

Ansible -playbook --syntax-check python.yaml

ansible 对于不同的硬件主机，如何保证所有命令的兼容性分组？

/etc/ansible/ansible.cfg 默认配置文件路径
-inventory 是定义托管主机地址配置文件
-首先编辑 /etc/ansible/hosts 文件,写入一些进程主机的地址

测试
– ansible [组名称] --list-hosts
对于需要 ansible 进行管理的主机写入到配置文件/etc/ansible/hosts 中，在该配置文件中可以是可以对主机进行分组的
对于相同的服务比如 web 服务器可以设置为一个 web 组，在 web 组内添加相应的 web 主机的 ip 地址或者可以通过解析后使用的

lvs 的三种模式，区别

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）

一、NAT 模式（VS-NAT）

原理：就是把客户端发来的数据包的 IP 头的目的地址，在负载均衡器上换成其中一台 RS 的 IP 地址并发至此 RS 来处理,RS 处理完后把数据交给负载均衡器,负载均衡器再把数据包原 IP 地址改为自己的 IP。将目的地址改为客户端 IP 地址即可期间,无论是进来的流量,还是出去的流量,都必须经过负载均衡器

优点：集群中的物理服务器可以使用任何支持 TCP/IP 操作系统，只有负载均衡器需要一个合法的 IP 地址

缺点：扩展性有限。当服务器节点（普通 PC 服务器）增长过多时,负载均衡器将成为整个系统的瓶颈因为所有的请求包和应答包的流向都经过负载均衡器。当服务器节点过多时大量的数据包都交汇在负载均衡器那，速度就会变慢！

二、IP 隧道模式（VS-TUN）

原理：首先要知道，互联网上的大多 Internet 服务的请求包很短小，而应答包通常很大那么隧道模式就是，把客户端发来的数据包，封装一个新的 IP 头标记(仅目的 IP)发给 RSRS 收到后,先把数据包的头解开,还原数据包,处理后,直接返回给客户端,不需要再经过负载均衡器。注意,由于 RS 需要对负载均衡器发过来的数据包进行还原,所以说必须支持 IPTUNNEL 协议，所以,在 RS 的内核中,必须编译支持 IPTUNNEL 这个选项

优点：负载均衡器只负责将请求包分发给后端节点服务器，而 RS 将应答包直接发给用户所以，减少了负载均衡器的大量数据流动，负载均衡器不再是系统的瓶颈，就能处理很巨大的请求量这种方式，一台负载均衡器能够为很多 RS 进行分发。而且跑在公网上就能进行不同地域的分发。

缺点：隧道模式的 RS 节点需要合法 IP，这种方式需要所有的服务器支持”IP Tunneling”(IP

Encapsulation)协议，服务器可能只局限在部分 Linux 系统上

三、直接路由模式（VS-DR）

原理：负载均衡器和 RS 都使用同一个 IP 对外服务但只有 DR 对 ARP 请求进行响应所有 RS 对本身这个 IP 的 ARP 请求保持静默也就是说,网关会把对这个服务 IP 的请求全部定向给 DR 而 DR 收到数据包后根据调度算法,找出对应的 RS,把目的 MAC 地址改为 RS 的 MAC（因为 IP 一致）并将请求分发给这台 RS 这时 RS 收到这个数据包,处理完成之后，由于 IP 一致，可以直接将数据返给客户，则等于直接从客户端收到这个数据包无异,处理后直接返回给客户端，由于负载均衡器要对二层包头进行改换,所以负载均衡器和 RS 之间必须在一个广播域，也可以简单的理解为在同一台交换机上

优点：和 TUN（隧道模式）一样，负载均衡器也只是分发请求，应答包通过单独的路由方法返回给客户端与 VS-TUN 相比，VS-DR 这种实现方式不需要隧道结构，因此可以使用大多数操作系统做为物理服务器。

缺点：（不能说缺点，只能说是不足）要求负载均衡器的网卡必须与物理网卡在一个物理段上。

tomcat 如何搭建，优化

搭建：

1）使用 RPM 安装 JDK 环境

```
[root@web1 ~]# yum  -y install  java-1.8.0-openjdk
[root@web1 ~]# yum -y install java-1.8.0-openjdk-headless
```

2）安装 apache-tomcat-8.0.30.tar.gz 软件包

```
[root@web1 ~]# tar -xf  apache-tomcat-8.0.30.tar.gz
[root@web1 ~]# mv apache-tomcat-8.0.30  /usr/local/tomcat
```

3）启动服务

```
[root@web1 ~]# /usr/local/tomcat/bin/startup.sh
[root@web1 ~]# firewall-cmd --set-default-zone=trusted
[root@web1 ~]# setenforce 0
```

4）服务器验证端口信息

```
[root@web1 ~]# netstat -nutlp |grep java          //查看 java 监听的端口
tcp        0      0 :::8080         :::*                LISTEN      2778/java
tcp     0    0 ::ffff:127.0.0.1:8005     :::*        LISTEN      2778/java
```

5）客户端浏览测试页面

```
[root@client ~]# firefox http://ip:8080
```

优化：

Tomcat 作为 Web 服务器，它的处理性能直接关系到用户体验，下面是几种常见的优化措施：

一、掉对 web.xml 的监视，把 jsp 提前编辑成 Servlet。有富余物理内存的情况，加大 tomcat 使用的 jvm 的内存

二、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。我们之前就遇到过内存分配不足，导致虚拟机一直处于 full GC，从而导致处理能力严重下降。

(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。最好的办法还是利用下面提到的缓存。

三、利用缓存和压缩

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，为了能加快网络传输速度，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。曾经我就见过一个图片从 300 多 kb 压缩到几十 kb，自己几乎看不出来区别。

四、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。可以参考之前写的《利用 nginx+tomcat+memcached 组建 web 服务器负载均衡》。

五、 优化 tomcat 参数

这里以 tomcat7 的参数配置为例，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。

```
<Connector port="8080"
           protocol="org.apache.coyote.http11.Http11NioProtocol"
           connectionTimeout="20000"
           redirectPort="8443"
           maxThreads="500"
```

```
            minSpareThreads="20"
            acceptCount="100"
            disableUploadTimeout="true"
            enableLookups="false"
            URIEncoding="UTF-8" />
```

https://www.cnblogs.com/xuwc/p/8523681.html

nginx 优化

一、一般来说 nginx 配置文件中对优化比较有作用的为以下几项：

1. worker_processes 8;

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

2. worker_cpu_affinity 00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000;

为每个进程分配 cpu，上例中将 8 个进程分配到 8 个 cpu，当然可以写多个，或者将一个进程分配到多个 cpu。

3. worker_rlimit_nofile 65535;

这个指令是指当一个 nginx 进程打开的最多文件描述符数目，理论值应该是最多打开文件数（ulimit -n）与 nginx 进程数相除，但是 nginx 分配请求并不是那么均匀，所以最好与 ulimit -n 的值保持一致。

现在在 linux 2.6 内核下开启文件打开数为 65535，worker_rlimit_nofile 就相应应该填写 65535。

这是因为 nginx 调度时分配请求到进程并不是那么的均衡，所以假如填写 10240，总并发量达到 3-4 万时就有进程可能超过 10240 了，这时会返回 502 错误。

查看 linux 系统文件描述符的方法：

```
[root@web001 ~]# sysctl -a | grep fs.file
fs.file-max = 789972
fs.file-nr = 510 0 789972
```

4. use epoll;

使用 epoll 的 I/O 模型

(

补充说明:

与 apache 相类，nginx 针对不同的操作系统，有不同的事件模型

A）标准事件模型

Select、poll 属于标准事件模型，如果当前系统不存在更有效的方法，nginx 会选择 select 或 poll

B）高效事件模型

Kqueue：使用于 FreeBSD 4.1+, OpenBSD 2.9+, NetBSD 2.0 和 MacOS X. 使用双

处理器的 MacOS X 系统使用 kqueue 可能会造成内核崩溃。
Epoll: 使用于 Linux 内核 2.6 版本及以后的系统。
/dev/poll：使用于 Solaris 7 11/99+, HP/UX 11.22+ (eventport), IRIX 6.5.15+ 和 Tru64 UNIX 5.1A+。
Eventport：使用于 Solaris 10. 为了防止出现内核崩溃的问题， 有必要安装安全补丁。
)

5. worker_connections 65535;

每个进程允许的最多连接数， 理论上每台 nginx 服务器的最大连接数为 worker_processes*worker_connections。

6. keepalive_timeout 60;

keepalive 超时时间。

7. client_header_buffer_size 4k;

客户端请求头部的缓冲区大小，这个可以根据你的系统分页大小来设置，一般一个请求头的大小不会超过 1k，不过由于一般系统分页都要大于 1k，所以这里设置为分页大小。
分页大小可以用命令 getconf PAGESIZE 取得。

```
[root@web001 ~]# getconf PAGESIZE
4096
```

但也有 client_header_buffer_size 超过 4k 的情况，但是 client_header_buffer_size 该值必须设置为“系统分页大小”的整倍数。

8. open_file_cache max=65535 inactive=60s;

这个将为打开文件指定缓存，默认是没有启用的，max 指定缓存数量，建议和打开文件数一致，inactive 是指经过多长时间文件没被请求后删除缓存。

9. open_file_cache_valid 80s;

这个是指多长时间检查一次缓存的有效信息。

10. open_file_cache_min_uses 1;

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

php 优化

https://blog.csdn.net/mengzuchao/article/details/78818566
https://blog.csdn.net/xq123joes/article/details/77980802
https://baijiahao.baidu.com/s?id=1600263247562743733&wfr=spider&for=pc

数据库优化

（1）升级硬件
（2）加大网络带宽
（3）sql 语句优化
（4）索引优化

（5）加缓存
（6）读写分离
（7）分区
（8）分布式数据库（垂直切分）
（9）水平切分
https://blog.csdn.net/weixin_38112233/article/details/79054661

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

mysql 的 innodb 如何定位锁问题:

在使用 show engine innodb status 检查引擎状态时，发现了死锁问题
在 5.5 中，information_schema 库中增加了三个关于锁的表（MEMORY 引擎）
innodb_trx ## 当前运行的所有事务
innodb_locks ## 当前出现的锁
innodb_lock_waits ## 锁等待的对应关系

mysql 如何减少主从复制延迟:

如果延迟比较大，就先确认以下几个因素：
1.从库硬件比主库差，导致复制延迟
2.主从复制单线程，如果主库写并发太大，来不及传送到从库就会导致延迟。更高版本的 mysql 可以支持多线程复制
3.慢 SQL 语句过多
4.网络延迟
5.master 负载
主库读写压力大，导致复制延迟，架构的前端要加 buffer 及缓存层
6.slave 负载
　　一般的做法是，使用多台 slave 来分摊读请求，再从这些 slave 中取一台专用的服务器
只作为备份用，不进行其他任何操作.另外， 2 个可以减少延迟的参数:
-slave-net-timeout=seconds 单位为秒 默认设置为 3600 秒
#参数含义：当 slave 从主数据库读取 log 数据失败后，等待多久重新建立连接并获取数据
-master-connect-retry=seconds 单位为秒 默认设置为 60 秒
#参数含义：当重新建立主从连接时，如果连接建立失败，间隔多久后重试
通常配置以上 2 个参数可以减少网络问题导致的主从数据同步延迟

MySQL 数据库主从同步延迟解决方案
最简单的减少 slave 同步延时的方案就是在架构上做优化，尽量让主库的 DDL 快速

执行还有就是主库是写，对数据安全性较高，比如 sync_binlog=1，innodb_flush_log_at_trx_commit= 1 之类的设置，而 slave 则不需要这么高的数据安全，完全可以讲 sync_binlog 设置为 0 或者关闭 binlog innodb_flushlog 也可以设置为 0 来提高 sql 的执行效率。另外就是使用比主库更好的硬件设备作为 slave

LVS、Nginx、HAproxy 优缺点？工作中你怎么选择

（1）Nginx：工作在网络的 7 层之上，可以针对 http 应用做一些分流的策略它的正则规则比 HAProxy 更为强大和灵活；Nginx 对网络稳定性的依赖非常小；Nginx 安装和配置比较简单，测试起来比较方便；Nginx 仅能支持 http、https 和 Email 协议，这样就在适用范围上面小些。

（2）抗负载能力强、是工作在网络 4 层之上仅作分发之用，没有流量的产生；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；无流量，LVS 只分发请求，而流量并不从它本身出去，这点保证了均衡器 IO 的性能不会收到大流量的影响；软件本身不支持正则表达式处理，不能做动静分离

（3）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；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。

keepalive 的工作原理和如何做到健康检查？

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

虚拟路由冗余协议，可以认为是实现路由器高可用的协议，即将 N 台提供相同功能的路由器组成一个路由器组这个组里面有一个 master 和多个 backup，master 上面有一个对外提供服务的 vip（该路由器所在局域网内其他机器的默认路由为该 vip），master 会发组播，当 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、维护及全局配置文件的加载和解析。check 负责健康检查，包括常见的各种检查方式，vrrp 模块是来实现 VRRP 协议的

Keepalived 健康检查方式配置

```
HTTP_GET|SSL_GET
HTTP_GET | SSL_GET
{
url {
path /# HTTP/SSL 检查的 url 可以是多个
digest <STRING> # HTTP/SSL 检查后的摘要信息用工具 genhash 生成
```

```
status_code 200# HTTP/SSL 检查返回的状态码
}
connect_port 80 # 连接端口
bindto<IPADD>
connect_timeout 3 # 连接超时时间
nb_get_retry 3 # 重连次数
delay_before_retry 2 #连接间隔时间
}
```

3. 生产环境（100）

如果你在生产服务器上执行了 rm -rf/*怎么办？

(1)迅速按下 ctrl + C ， 保持心态稳定
(2)重新安装系统
(3)从其他服务器备份信息上恢复数据

IO 性能不足，你如何优调？

详细调优见 https://blog.csdn.net/doitsjz/article/details/50837569

清空一个几百万行数据库的表 A 数据，通常用的 sql 语句？

```
truncate table  A
```

生产环境下数据库备份的有哪些方式？

逻辑备份和物理备份

zabbix 监控到数据库 cpu 跑满，怎么处理？

在服务器上执行 mysql -u root -p 之后，通过 show processlist 命令找到耗时最长的
先杀掉该进程
通过慢查询日志找到具体的 sql 语句
使用 explain 优化 sql 语句
调整 my.cnf 的 query_cache_size 和 tmp_table_size 的值

16 核的 cpu，负载为 13.5，是负载高还是负载低？如果高了，多少合适？

因为 16*0.7=11.2 < 13.5，所以负载高了
负载低于 11.2 合适

现在给你三百台服务器，你怎么对他们进行管理？

1）设定跳板机，使用统一账号登录。
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、应用的 cmdb 信息管理。

Linux 系统中病毒怎么办？

（1）top 命令找到 cpu 使用率最高的进程，一般是病毒文件
（2）可以用 ps aux 找到病毒文件位置
（3）rm -f 命令删除病毒文件
（4）删除病毒文件不排除有潜伏病毒，把机器备份数据之后重装一下系统。

已知 apache 服务的访问日志按天记录在本地目录/var/app/logs 下，由于磁盘空间紧张现在要求只能保留最近 7 天的访问日志！ 请问如何解决？

方法一：find /var/app/logs/ -type f -name "*.log"-mtime +7 | xargs rm -f
方法二 rm -f $(find /var/app/logs/ -type f -name"*.log" -mtime +7)
方法三：find /var/app/logs/ -type f -name "*.log"-mtime +7 -exec rm -f {} \;

如何优化 Linux 系统

不用 root，添加普通用户，通过 sudo 授权管理
更改默认的远程连接 SSH 服务端口及禁止 root 用户远程连接
定时自动更新服务器时间
配置国内 yum 源
关闭 selinux 及 iptables（iptables 工作场景如果有外网 IP 一定要打开，高并发除外）
调整文件描述符的数量
精简开机启动服务（crond rsyslog network sshd）
内核参数优化（/etc/sysctl.conf）
更改字符集，支持中文，但建议还是用英文字符集，防止乱码
锁定关键系统文件
清空/etc/issue，去除系统及内核版本登录前的屏幕显示