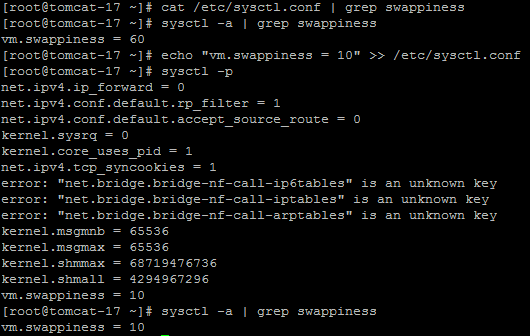
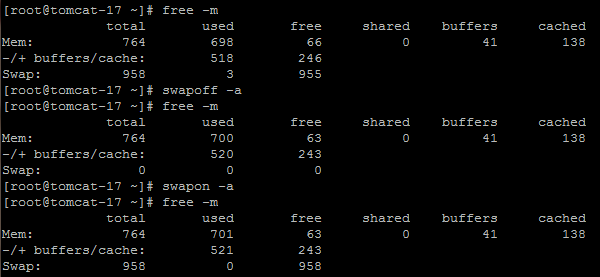
1. swap

当物理内存使用一定比例后会动用swap空间，在rhel和ubuntu系统中，swappiness 参数控制这个比例，默认的vm.swappiness值是60，可以改小一些降低swap的加载。对于swap的使用极少，可以将值设为0。这并不会禁止你对swap的使用，而是使你的系统对于swap的写入尽可能的少，同时尽可能多的使用你的实际内存。

cat /proc/sys/vm/swappiness



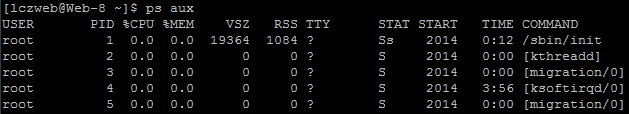
当swap空间已经使用很多时，可以将swap空间卸挂载，cache内容会被挤到物理内存中，然后再重新挂载swap

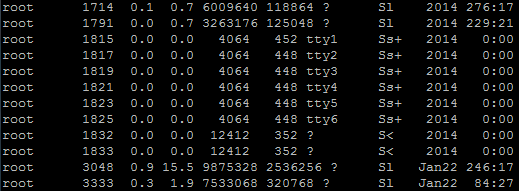


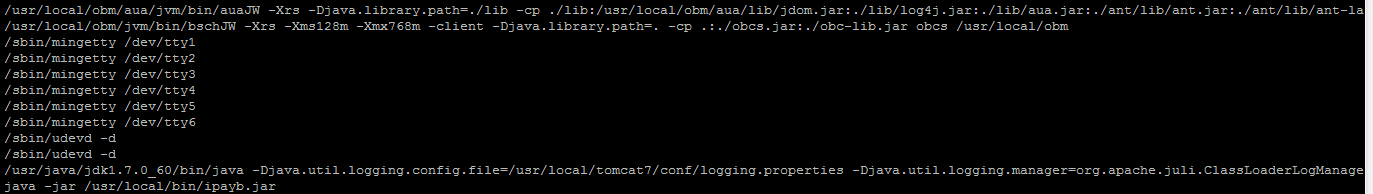
这里不探究多核CPU如何分配物理内存以及SWAP，这涉及到NUMA和SWAP的“恩怨”。

1. ps pmap free

ps aux命令分析每个使用的内存的脚本、命令和服务







au(x) 输出格式 :

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

USER: 行程拥有者

PID: pid

%CPU: 占用的 CPU 使用率

%MEM: 占用的记忆体使用率

VSZ: 占用的虚拟记忆体大小

RSS: 占用的记忆体大小

TTY: 终端的次要装置号码 (minor device number of tty)

STAT: 该行程的状态:

D: 不可中断的静止

R: 正在执行中

S: 静止状态

T: 暂停执行

Z: 不存在但暂时无法消除

W: 没有足够的记忆体分页可分配

<: 高优先序的行程

N: 低优先序的行程

L: 有记忆体分页分配并锁在记忆体内

+ 位于后台的进程组；

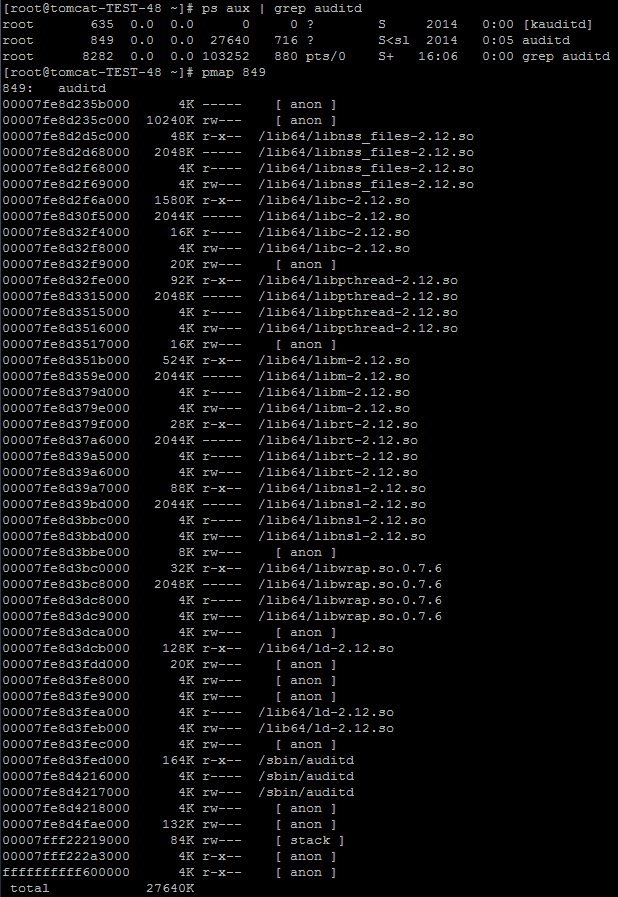
WCHAN 正在等待的进程资源；

START: 行程开始时间

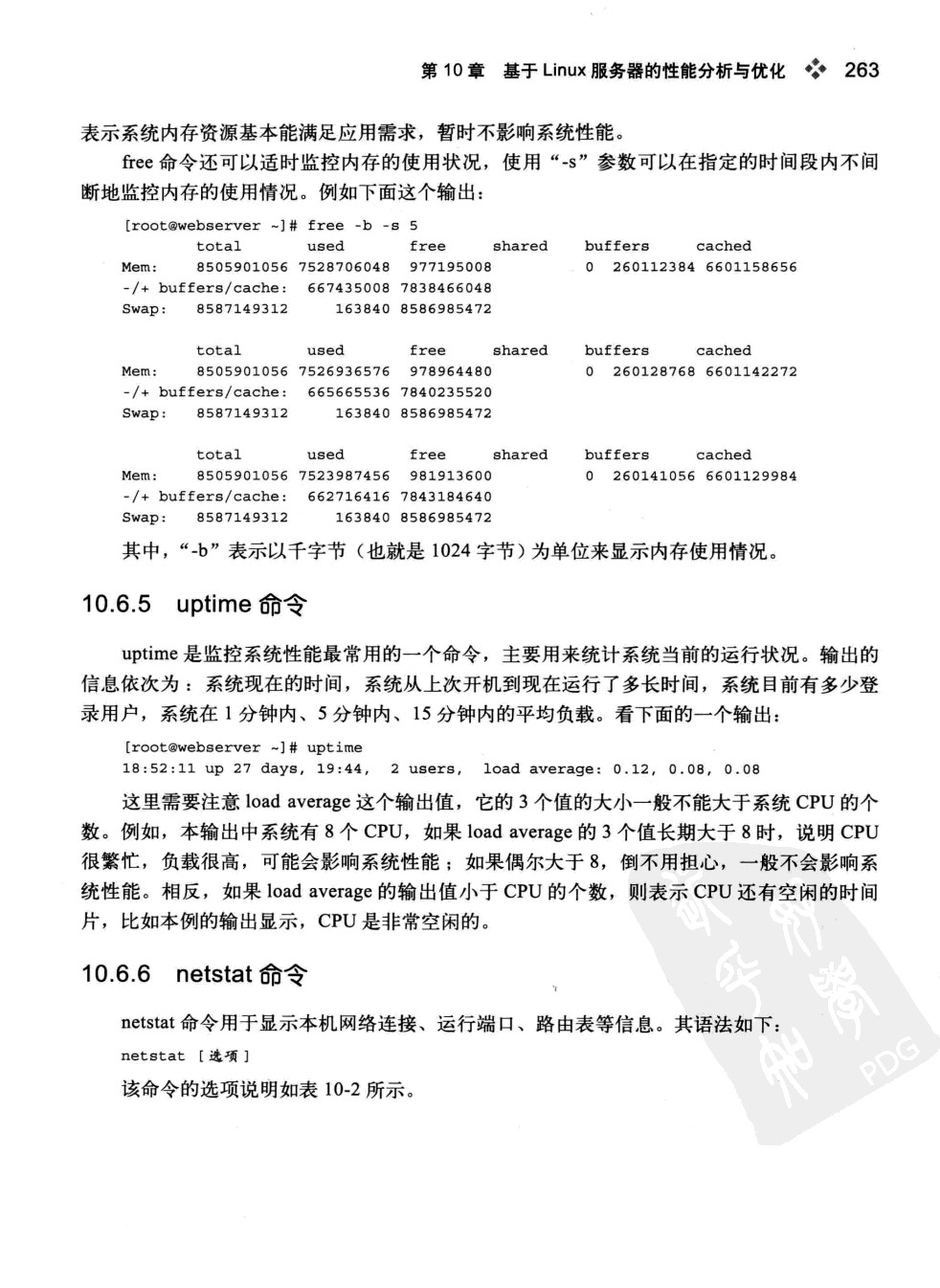
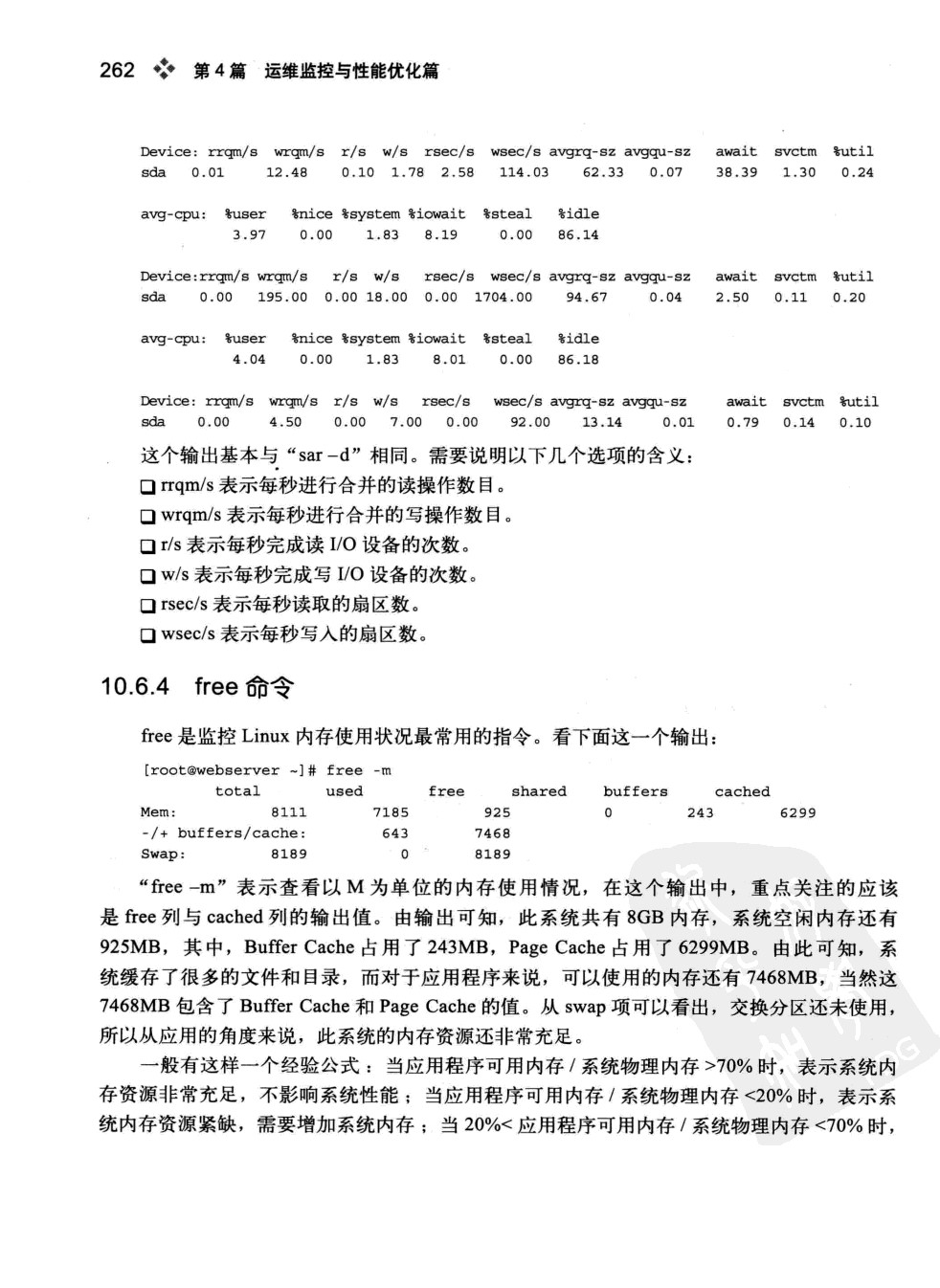
TIME: 执行的时间

COMMAND:所执行的指令

“pmap PID” 查看指定进程所有文件使用的内存



free



1. sshd

编辑/etc/ssh/sshd\_config文件

PermitRootLogin no



service sshd reload

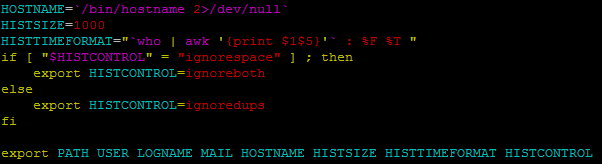


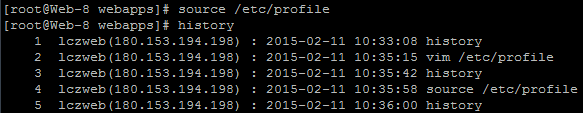
1. history

HISTSIZE系统变量设置操作记录history，在/etcprofile文件

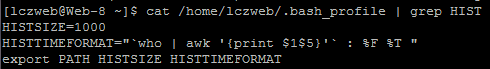


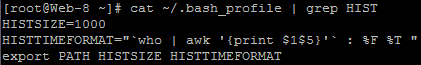
设置history记录1000行操作，并且以操作用户、IP地址、操作时间设置记录格式





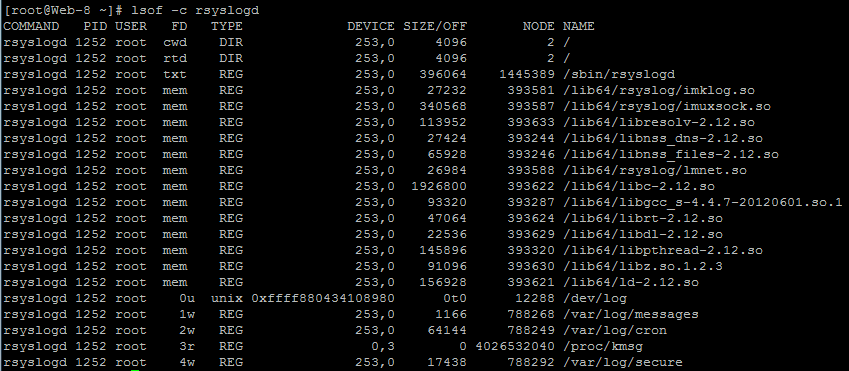
History记录追加在~/.bash\_history文件，可以将/etc/profile文件中的系统变量设置加到~/.bash\_profile，为指定用户设置系统变量



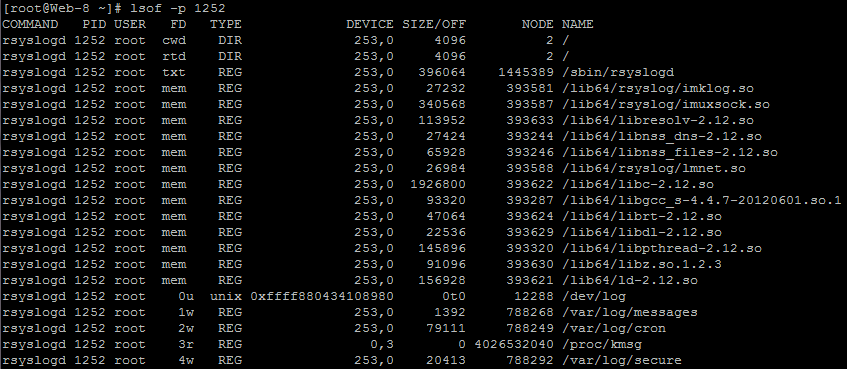


1. lsof 进程

lsof | grep 进程，或者lsof –c 进程

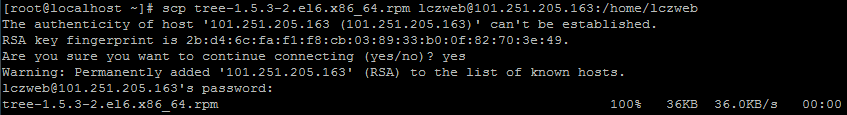


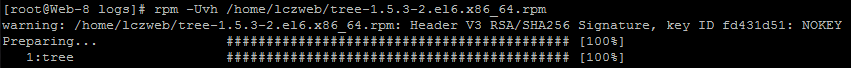
或者 lsof –p PID号



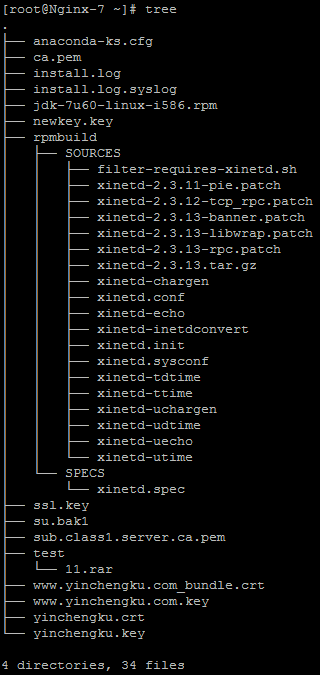
1. tree

首先将rhel 6.5光盘中的tree软件包上传到服务器，安装

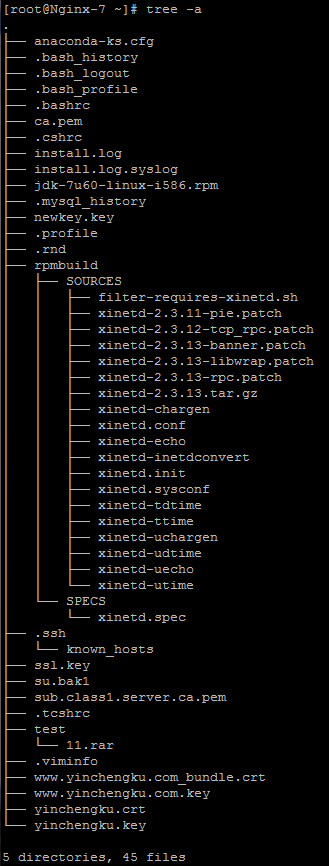




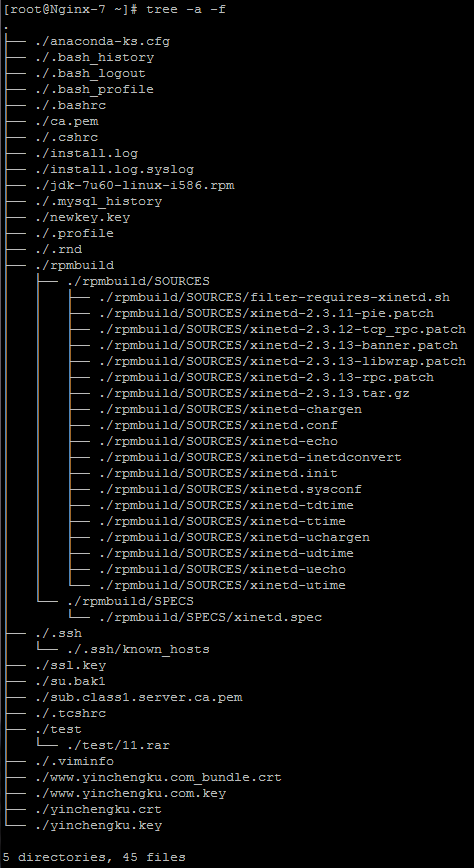
tree相当于pwd + ls



tree –a相当于pwd + ls -a

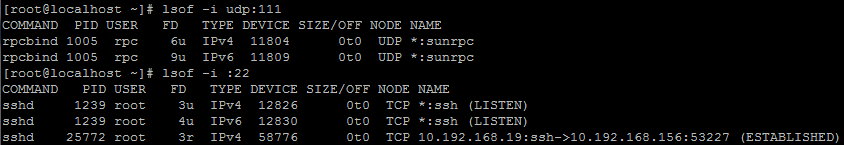


tree –a –f 查看所有隐藏文件、目录的完整路径



7．lsof 端口状态

lsof命令查看指定TCP、UDP端口的所有连接状态



8. while循环



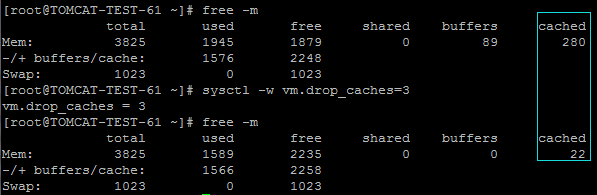
9.cached

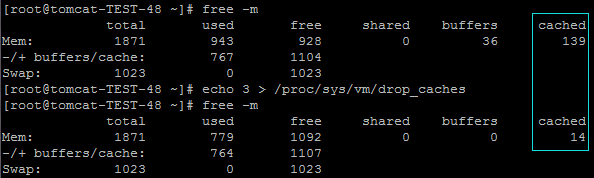
这两个命令效果是一样的





内存的cached部分很多时，使用该命令才有效果

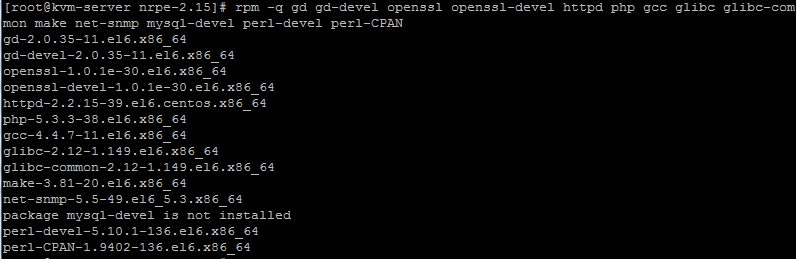




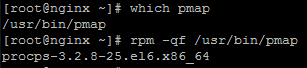
10.rpm

是否安装了这些软件包

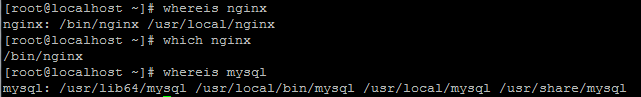
rpm -q gd gd-devel openssl openssl-devel httpd php gcc glibc glibc-common make net-snmp mysql-devel perl-devel perl-CPAN



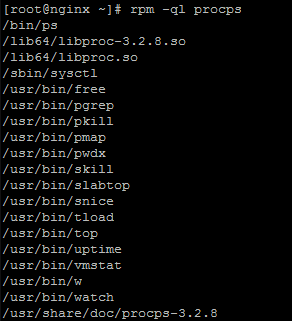
查找命令的安装包rpm -qf



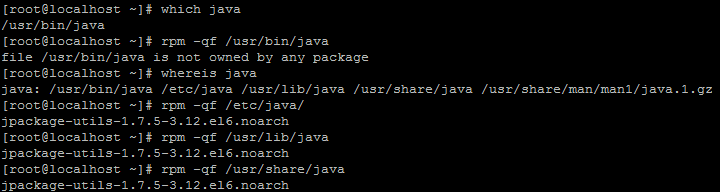
which或者whereis



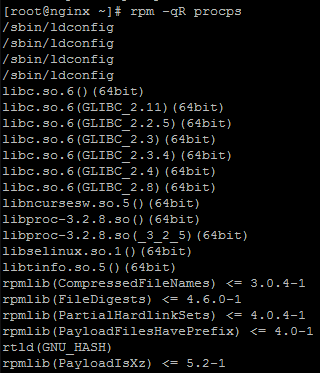
查找rpm包的安装路径rpm -ql



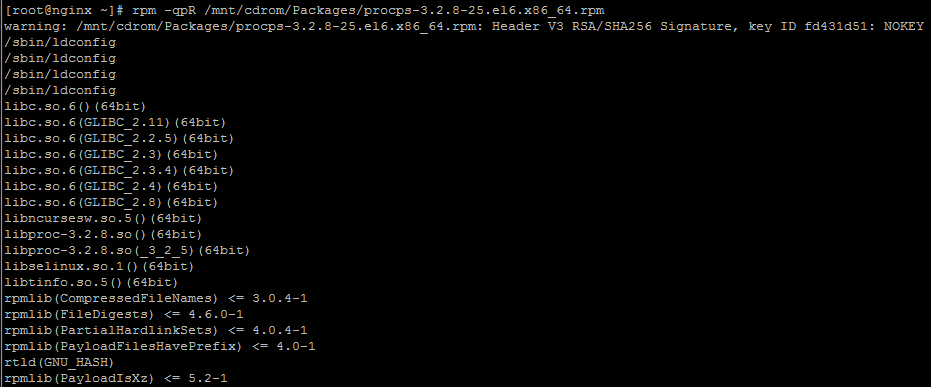
查询命令文件的路径whereis java



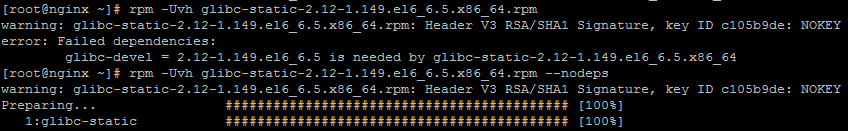
rpm包的依赖关系rpm -qR



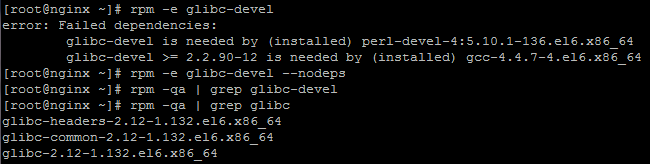
查看未安装的rpm包的依赖关系rpm -qpR



忽略依赖关系强制安装rpm包rpm -Uvh --nodeps

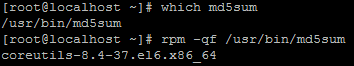


忽略依赖强制卸载rpm包rpm -e --nodeps

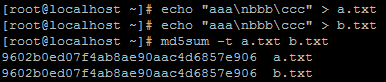


11.md5sum

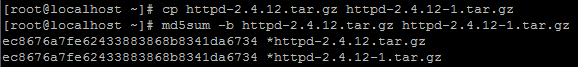
md5sum的命令来源



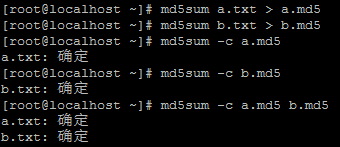
md5sum -t 以文本模式读入文件内容进行校验



md5sum -b 以二进制模式读入内容

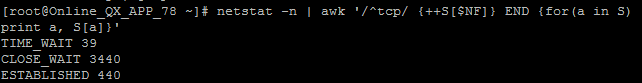


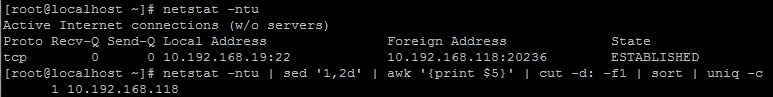
md5sum -c选项来对文件md5进行校验。校验时，根据已生成的md5来进行校验。生成当前文件的md5，并和之前已经生成的md5进行对比，如果一致，则返回OK，否则返回错误信息



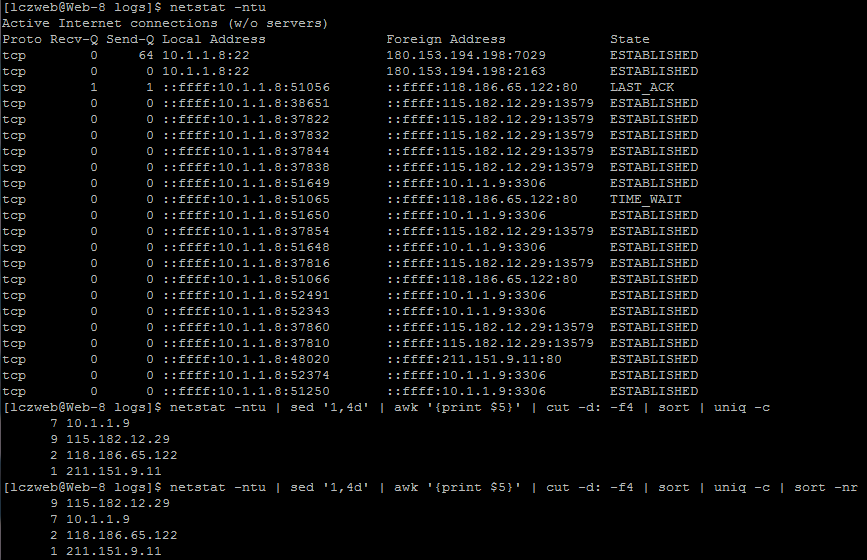
12. netstat

netstat -n | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}'



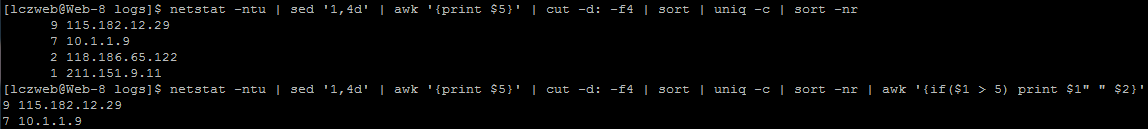
netstat -ntu | sed '1,2d' | awk '{print $5}' | cut -d: -f1 | sort | uniq -c

连接数从多到少排序netstat -ntu | sed '1,4d' | awk '{print $5}' | cut -d: -f4 | sort | uniq -c | sort -nr



只输入连接数过多的IP

netstat -ntu | sed '1,4d' | awk '{print $5}' | cut -d: -f4 | sort | uniq -c | sort -nr | awk '{if($1 > 5) print $1" " $2}'



13. ddos 防御

#!/bin/bash

netstat -ntu | sed '1,2d' | awk '{print $5}' | cut -d: -f1 | sort | uniq -c | awk '{if($1 > 5) print $2}' > denf.txt #没有NAT设备的写法

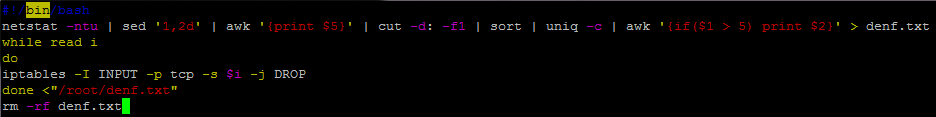
while read i

do

iptables -I INPUT -p tcp -s $i -j DROP

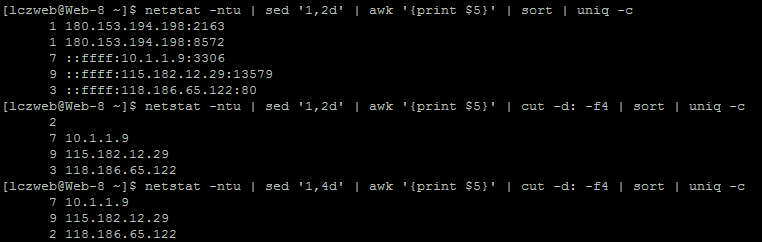
done <"/root/denf.txt"

rm -rf denf.txt

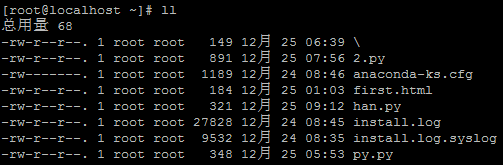


前方有NAT设备的写法

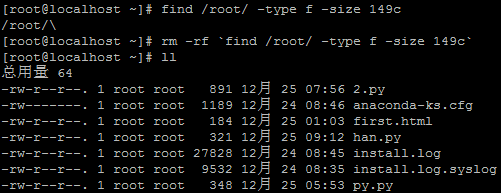
netstat -ntu | sed '1,4d' | awk '{print $5}' | cut -d: -f4 | sort | uniq -c



14. rm \



find命令查找文件夹的block大小



15．CPU

查看逻辑CPU个数： cat /proc/cpuinfo |grep "processor"|sort -u|wc -l

查看物理CPU个数： grep "physical id" /proc/cpuinfo|sort -u|wc -l

查看每个物理CPU内核个数： grep "cpu cores" /proc/cpuinfo|uniq

每个物理CPU上逻辑CPU个数： grep "siblings" /proc/cpuinfo|uniq

查看CPU型号：cat /proc/cpuinfo | grep 'model name' |uniq

cat /proc/cpuinfo | grep name | cut -f2 -d:



如果只想计算有多少个cpu内核，可以使用如下命令：

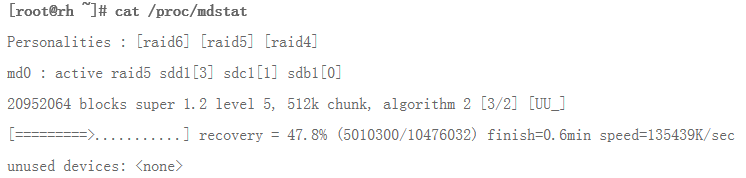
cat /proc/cpuinfo | grep 'model name' | wc -l



16.RAID

使用容量均为10.7 GB的sdb、sdc、sdd三块盘组成raid5 /dev/md0，三块磁盘均只有1个（主）分区：

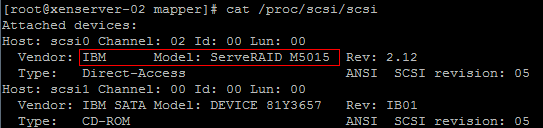
cat /proc/mdstat



查看服务器支持的硬raid

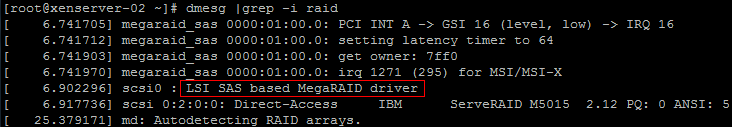
cat /proc/scsi/scsi

这是IBM IBM x3650 M3服务器，阵列卡是ServeRAID M5015

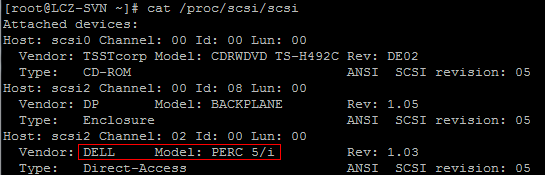


ServeRAID-M5015：支持RAID0、1、5、10、50，缓存512MB、带电池，加选件可支持RAID6、60，数据带宽6Gbps。单卡支持两块硬盘背板（8块2.5 SAS硬盘），可扩展硬盘背板。

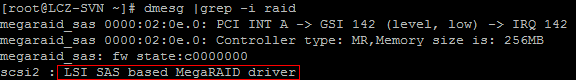
Raid卡的驱动是MegaRAID



这是Dell Power Edge 2900服务器，阵列卡是PERC 5/i



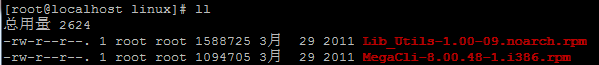
Raid卡的驱动是MegaRAID



去LSI官网下载MegaRAID Storage Manager (MSM)，<http://www.lsi.com/support/pages/download-results.aspx?component=Storage+Component&productfamily=RAID+Controllers&productcode=P00043&assettype=Management+Software+and+Tools&productname=MegaRAID+SAS+9240-4i>，下载14.11.01.00\_Linux-x64\_MSM.gz，地址：[http://www.lsi.com/downloads/Public/RAID%20Controllers/RAID%20Controllers%20Common%20Files/14.11.01.00\_Linux-x64\_MSM.gz](http://www.lsi.com/downloads/Public/RAID Controllers/RAID Controllers Common Files/14.11.01.00_Linux-x64_MSM.gz)

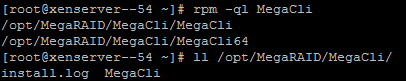


解压后就是两个rpm软件

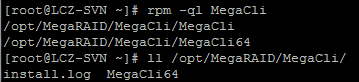


直接安装两个rpm软件，它会自动识别系统是32位还是64位，

32位系统安装后



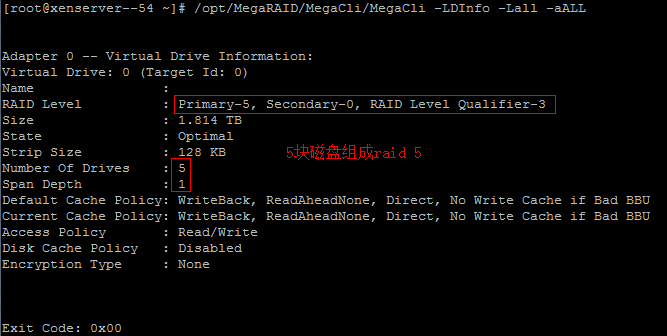
64位系统安装后



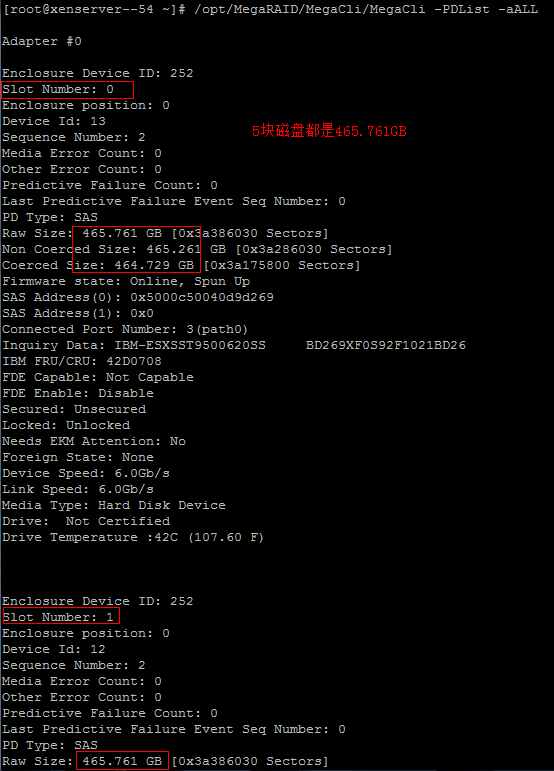
使用MegaCli或者MegaCli64命令查看raid阵列和磁盘信息：

这是IBM IBM x3650 M3服务器

/opt/MegaRAID/MegaCli/MegaCli -LDInfo -Lall -aALL

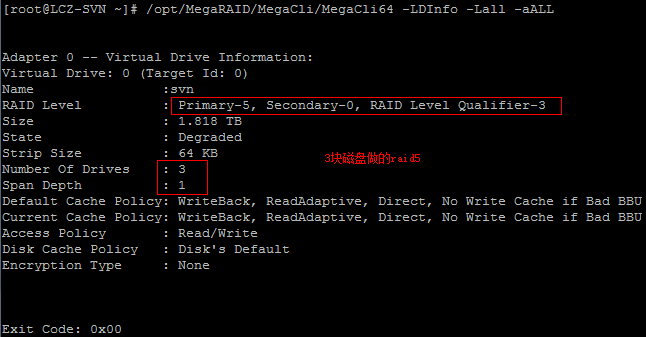


/opt/MegaRAID/MegaCli/MegaCli -PDList -aALL

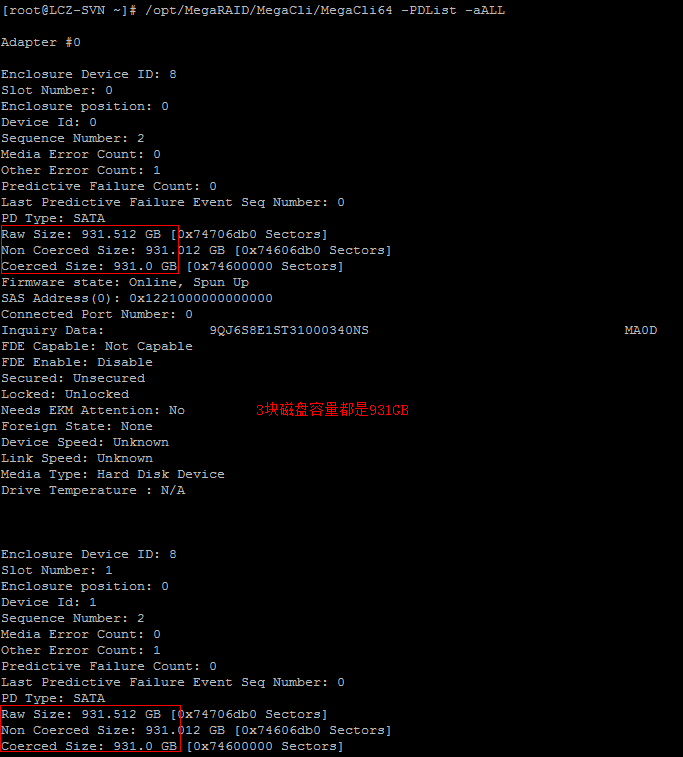


这是Dell Power Edge 2900服务器

/opt/MegaRAID/MegaCli/MegaCli64 -LDInfo -Lall -aALL

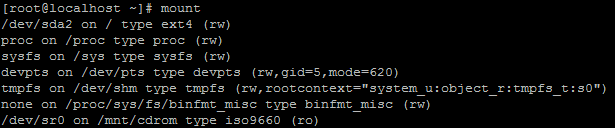


/opt/MegaRAID/MegaCli/MegaCli64 -PDList -aALL

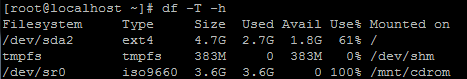


Raid5容量=（磁盘数量-1）\* 单独磁盘容量

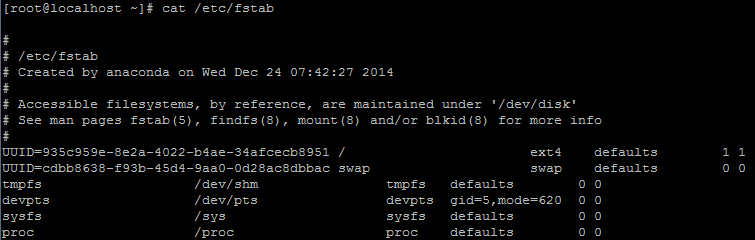
17.df

mount

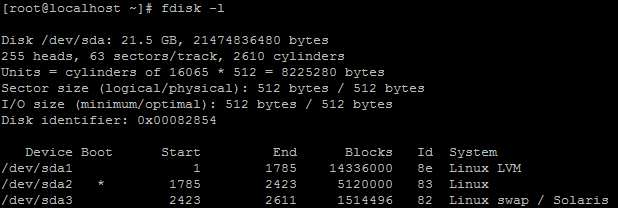
df -T -h



cat /etc/fstab



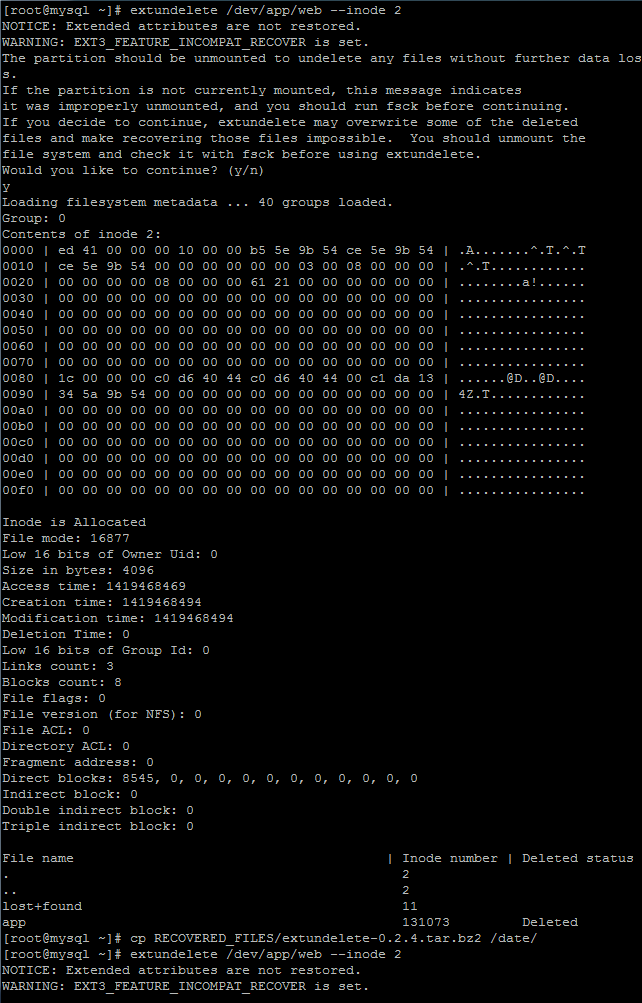
fdisk –l看磁盘

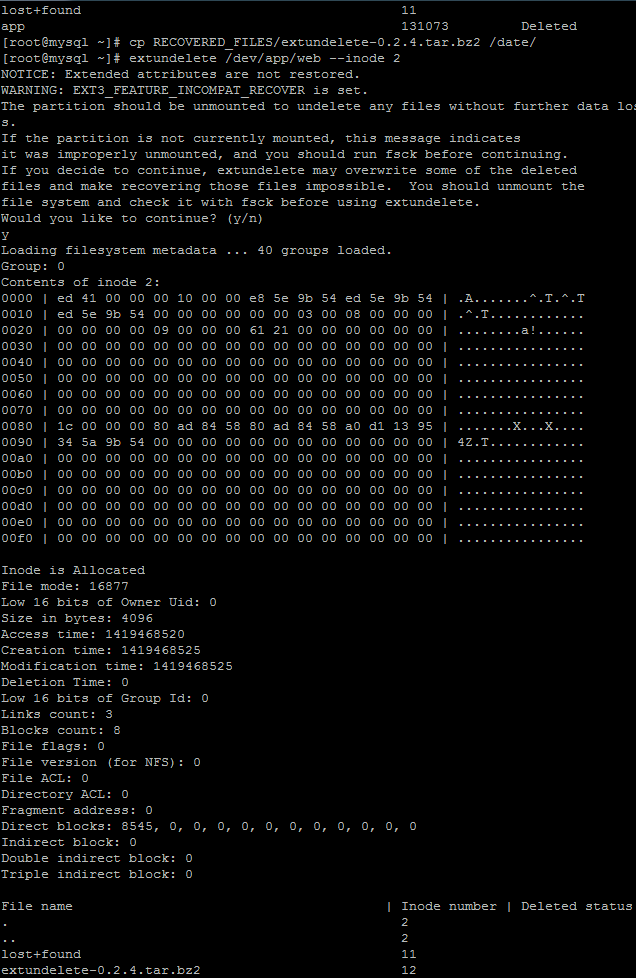


18. extundelete

被rm的文件能够使用extundelete恢复的前提条件：

Extundelete只能恢复删除后没有再写入文件的分区，第1：恢复的文件所在的分区必须能够卸载，如果要恢复的文件在根分区或者root家目录，就不能卸挂载从而恢复。所有能够恢复的前提是，被删除文件必须在LVM卷的分区；第2：文件被删除后，文件所在的分区没有继续写入文件。恢复的文件依靠inode值，文件被删除后它的inode值依然存在，但是再向这个分区写入时，被删除文件的inode值就被覆盖，extundelete就没法恢复了。





满足以上条件（恢复的不是根分区、root家目录，没有再向恢复分区写入文件），就可以在<http://extundelete.sourceforge.net/>官网下载extundelete-0.2.4，安装使用。

利用extundelete恢复文件时并不依赖特定文件格式，首先extundelete会通过文件系统的inode信息（根目录的inode一般为2）来获得当前文件系统下所有文件的信息，包括存在的和已经删除的文件，这些信息包括文件名和inode。然后利用inode信息结合日志去查询该inode所在的block位置，包括直接块，间接块等信息。最后利用dd命令将这些信息备份出来，从而恢复数据文件。

extundelete的官方网站是http://extundelete.sourceforge.net/ ，其目前的稳定版本是extundelete-0.2.4。，在安装extundelete之前需要yum install e2fsprogs e2fsprogs-devel两个依赖包。

下载extundelete-0.2.4.tar.bz2：wget http://nchc.dl.sourceforge.net/project/extundelete/extundelete/0.2.4/extundelete-0.2.4.tar.bz2

安装依赖包：yum install e2fsprogs e2fsprogs-devel

tar jxvf extundelete-0.2.4.tar.bz2

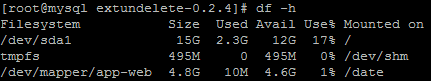
cd extundelete-0.2.4

./configure

make && make install

/dev/sda3分区为LVM逻辑卷，我划分了app/web逻辑卷挂载在/date目录下，



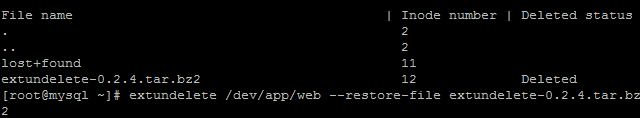


umount /dev/app/web

或者umount /date



extundelete /dev/app/web --inode 2



extundelete /dev/app/web --restore-file extundelete-0.2.4.tar.bz2恢复文件

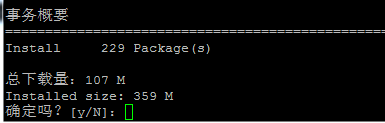
extundelete /dev/ app/web --restore-directory /date/app 恢复目录

在文件恢复成功后，extundelete命令默认会在执行命令的当前目录下创建一个RECOVERED\_FILES目录，此目录用于存放恢复出来的文件，所以执行extundelete命令的当前目录必须是可写的。根据上面的输出，通过md5sum命令校验，校验码与之前的完全一致，表明文件恢复成功。

19.rhel6.5 desktop

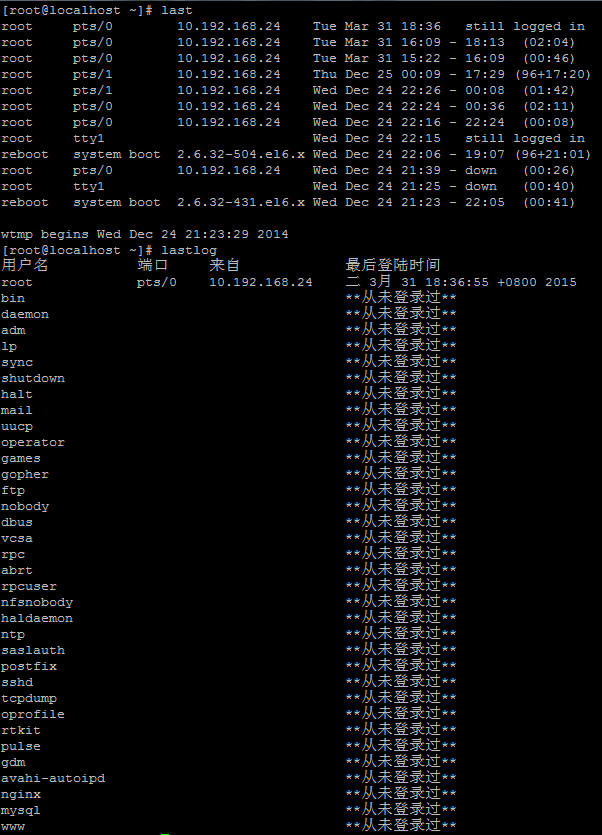
yum groupinstall "X Window System" " Desktop"





20.last

last命令反映只登录用户的登录情况，lastlog目录显示所有系统的登录情况：



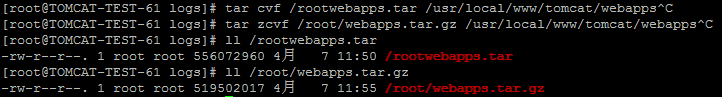
21.tar

将/usr/local/www/tomcat/webapps目录压缩成根目录下的rootwebapps.tar文件：

tar cvf /rootwebapps.tar /usr/local/www/tomcat/webapps

将/usr/local/www/tomcat/webapps目录压缩成root家目录下的rootwebapps.tar.gz文件：

tar zcvf /root/webapps.tar.gz /usr/local/www/tomcat/webapps



解压缩总结

1、\*.tar 用 tar -xvf 解压

2、\*.gz 用 gzip -d或者gunzip 解压

3、\*.tar.gz和\*.tgz 用 tar -xzf 解压

4、\*.bz2 用 bzip2 -d或者用bunzip2 解压

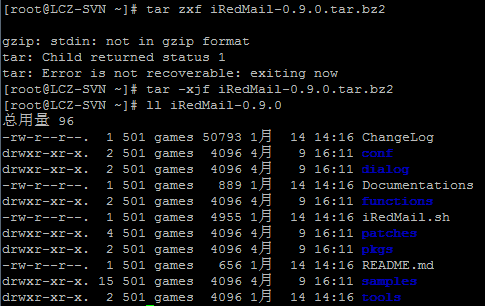
5、\*.tar.bz2用tar -xjf 解压

6、\*.Z 用 uncompress 解压

7、\*.tar.Z 用tar -xZf 解压

8、\*.rar 用 unrar e解压

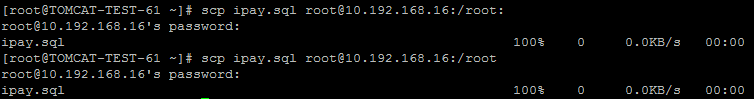
9、\*.zip 用 unzip 解压

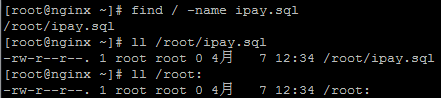


24.scp

使用scp命令，必须首先yum install openssh-clients，结构很简单：scp 本地文件 远程用户@远程主机IP：目的路径

必须注意，目的路径有“：”和没有“：”完全是两种结果：“目的路径：”表示复制到远程主机后的文件名字就是“路径：”；“目的路径”表示复制到远程主机的该目录下。实例如下：





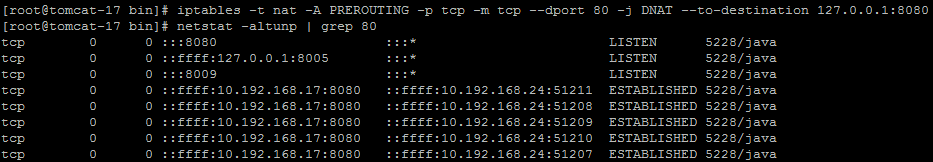
25.iptables

iptables -t nat -I PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 443



**80端口转8080**

**iptables -t nat -A PREROUTING -p tcp -m tcp --dport 80 -j DNAT --to-destination 127.0.0.1:8080**



**允许80、443端口：**

**iptables -A INPUT -p tcp --dport 80 -j ACCEPT**

**iptables -A INPUT -p tcp --dport 443 -j ACCEPT**



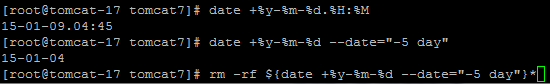
**删除端口转发**

**iptables -t nat -D PREROUTING -p tcp --dport 80 -j REDIRECT --to-port**



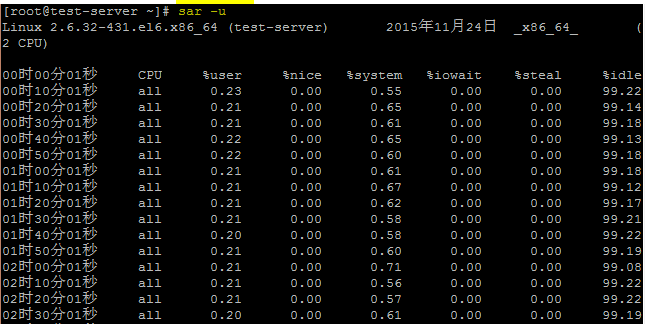
26.date

date +%y-%m-%d --date="-5 day"



27.sar

sar -u 从当天00：00开始、每隔10分钟的CPU使用情况



28.yum

epel源：rpm -Uvh <http://ftp.linux.ncsu.edu/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm>

rpmforge源：rpm -Uvh <http://apt.sw.be/redhat/el6/en/x86_64/rpmforge/RPMS/rpmforge-release-0.5.3-1.el6.rf.x86_64.rpm>

163yum源地址<http://mirrors.163.com/centos/6.7/os/x86_64/>，pgpkey<http://mirrors.163.com/centos/6.7/os/x86_64/>[RPM-GPG-KEY-CentOS-6](http://mirrors.163.com/centos/6/os/x86_64/RPM-GPG-KEY-CentOS-6)

rhel7的源：rpm -Uvh <http://ftp.linux.ncsu.edu/pub/epel/7/x86_64/e/epel-release-7-5.noarch.rpm>

nginx的源：rpm -Uvh <http://nginx.org/packages/centos/6/noarch/RPMS/nginx-release-centos-6-0.el6.ngx.noarch.rpm>

nginx源的内容

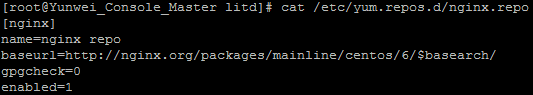
[nginx]

name=nginx repo

baseurl=http://nginx.org/packages/mainline/centos/6/$basearch/

gpgcheck=0

enabled=1



创建163yum源的repo文件

echo -e "[rhel-source]\nname=mirrors.163\nbaseurl=http://mirrors.163.com/centos/6.7/os/x86\_64/\ngpgcheck=1\ngpgkey=http://mirrors.163.com/centos/6.7/os/x86\_64/RPM-GPG-KEY-CentOS-6" > /etc/yum.repos.d/163.repo

[rhel-source]

name=mirrors.163

baseurl=http://mirrors.163.com/centos/6.7/os/x86\_64/

gpgcheck=1

gpgkey=http://mirrors.163.com/centos/6.7/os/x86\_64/RPM-GPG-KEY-CentOS-6

声明rpm关系依赖

rpm --import <http://mirrors.163.com/centos/6.7/os/x86_64/RPM-GPG-KEY-CentOS-6>

rpm --import <http://mirror.centos.org/centos-6/6/os/x86_64/RPM-GPG-KEY-CentOS-6>

创建yum仓库缓存 yum makecache

Centos源：<http://mirror.centos.org/centos-6/6/os/x86_64/，pgpkey>：http://mirror.centos.org/centos-6/6/os/x86\_64/[RPM-GPG-KEY-CentOS-6](http://vault.centos.org/6.5/os/x86_64/RPM-GPG-KEY-CentOS-6)

必要时卸载当前系统的yum包，下载163yum包并安装

rpm -e --nodeps yum yum-rhn-plugin yum-metadata-parser yum-utils yum-plugin-security

下载并安装这3个rpm包

wget <http://mirrors.163.com/centos/6/os/x86_64/Packages/yum-3.2.29-69.el6.centos.noarch.rpm>

wget <http://mirrors.163.com/centos/6/os/x86_64/Packages/yum-metadata-parser-1.1.2-16.el6.x86_64.rpm>

wget <http://mirrors.163.com/centos/6/os/x86_64/Packages/yum-plugin-fastestmirror-1.1.30-30.el6.noarch.rpm>

或者

下载安装yum、yum-metadata-parser、yum-fastestmirror包

[root@nginx ~]# wget <http://mirror.centos.org/centos-6/6/os/x86_64/Packages/yum-3.2.29-60.el6.centos.noarch.rpm>

[root@nginx ~]# wget <http://mirror.centos.org/centos-6/6/os/x86_64/Packages/yum-metadata-parser-1.1.2-16.el6.x86_64.rpm>

[root@nginx ~]# wget <http://mirror.centos.org/centos-6/6/os/x86_64/Packages/yum-plugin-fastestmirror-1.1.30-30.el6.noarch.rpm>

29. /var/log

/var/log/message 系统启动后的信息和错误日志，是Red Hat Linux中最常用的日志之一

/var/log/secure 与安全相关的日志信息

/var/log/maillog 与邮件相关的日志信息

/var/log/cron 与定时任务相关的日志信息

/var/log/spooler 与UUCP和news设备相关的日志信息

/var/log/boot.log 守护进程启动和停止相关的日志消息

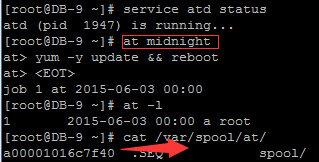
/var/log/wtmp 该日志文件永久记录每个用户登录、注销及系统的启动、停机的事件

30.at

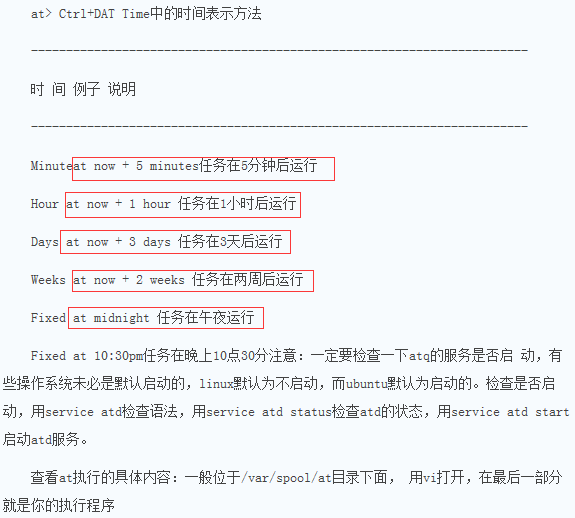
at 只执行一次，要想使用at 必须要有atd这个服务，命令输完后，以 ctrl+d 结束。

at –l 或者atq 列出at计划

at –d 计划编号 或者atrm 计划编号 删除任务计划



时间表示方法



31.route

添加默认路由条目：route add default gw 10.192.168.1

32.清理日志通用

统计大小为0的\*.log日志的数量

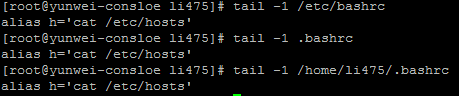
find / -name \*.log | xargs du -h | sort -nr | grep ^0 | wc -l

删除大小为0的\*.log日志

find / -name \*.log | xargs du -h | sort -nr | awk '{if($1==0) print $2}' | xargs rm -rf

34. alias

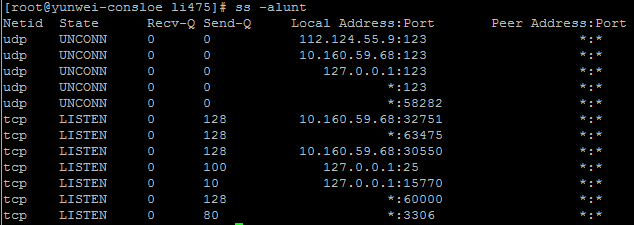
在当前用户的家目录下的.bashrc文件追加alias h='cat /etc/hosts'命令，在/etc/bashrc文件下也追加该命令，source /etc/bashrc生效



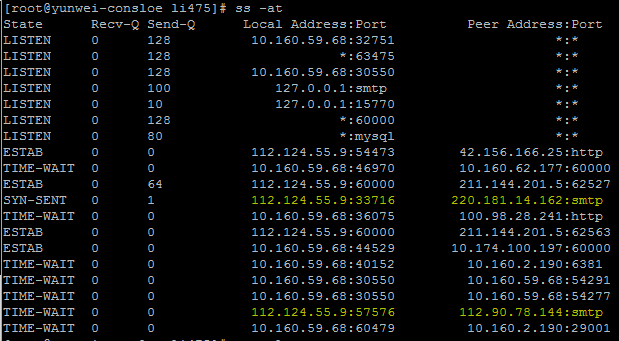
35. **ss**

为什么ss比netstat快：netstat是遍历/proc下面每个PID目录，ss直接读/proc/net下面的统计信息。

ss -alunt 粗略显示系统打开的TCP、UDP端口

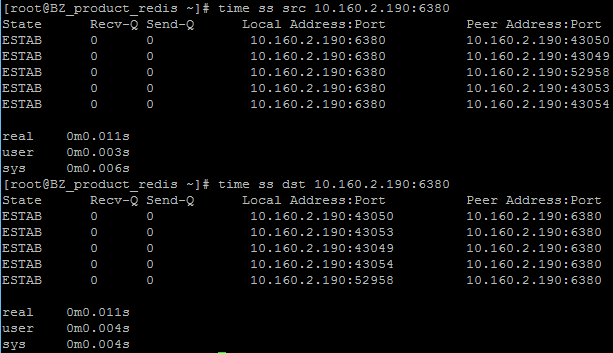


ss -at 详细显示系统所有TCP连接



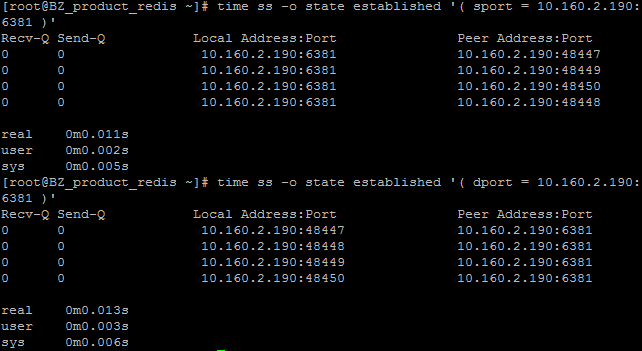
ss src 10.160.2.190:6380 显示源地址是10.160.2.190:6380的所有连接

ss dst 10.160.2.190:6380 显示目的地址是10.160.2.190:6380的所有连接



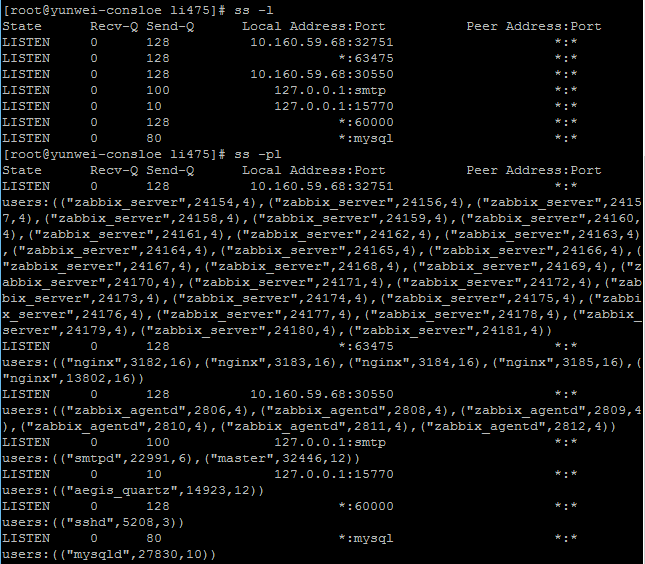
ss -o state established '( sport = 10.160.2.190:6381 )' 显示源地址是10.160.2.190:6380且TCP状态是established的所有连接

ss -o state established '( dport = 10.160.2.190:6381 )' 显示目的地址是10.160.2.190:6380且TCP状态是established的所有连接



ss -l 显示本地打开的所有端口

ss -pl 显示每个进程具体打开的socket



ss -t -a 显示所有tcp socket

ss -u -a 显示所有的UDP Socekt

ss -o state established '( dport = :smtp or sport = :smtp )' 显示所有已建立的SMTP连接

ss -s 列出当前socket详细信息

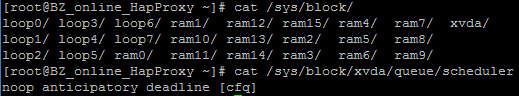
ss -atr 显示更多更详细的有关TCP和连接状态的信息

如果你无法使用ss命令，那么可能是缺少了iproute

36. I/O 调度算法

查看当前系统的I/O调度方法:

cat /sys/block/xvda/queue/scheduler



当前的I/O调度算法是cfg

临时更改I/O调度算法为deadline

echo deadline > /sys/block/sda/queue/scheduler

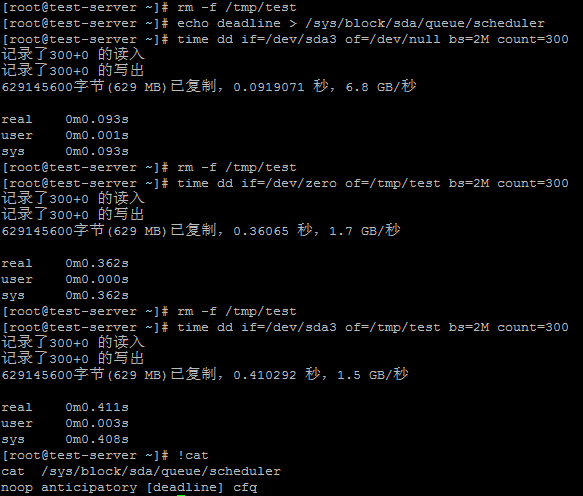


在I/O调度算法为deadline情况下，测试磁盘I/O性能

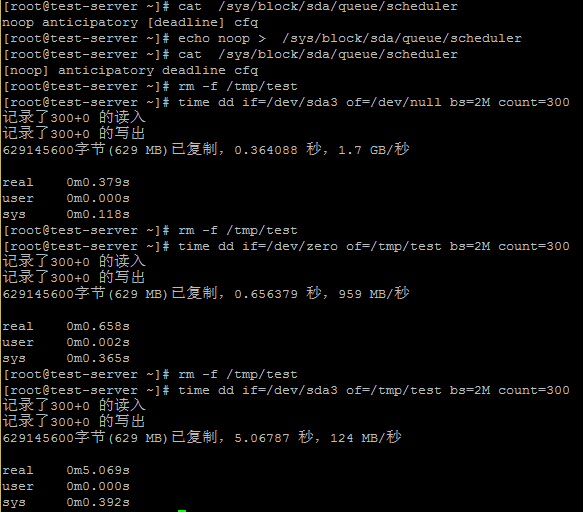
只读：time dd if=/dev/sda3 of=/dev/null bs=2M count=300

只写：time dd if=/dev/zero of=/tmp/test bs=2M count=300

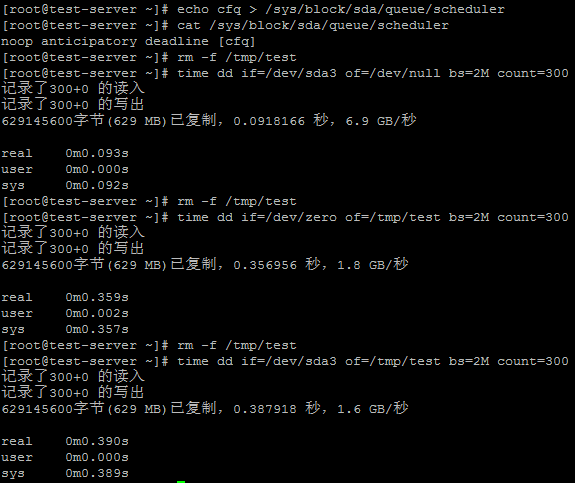
同时读写：time dd if=/dev/sda3 of=/tmp/test bs=2M count=300



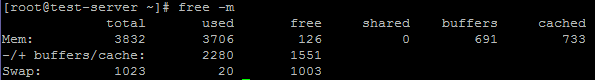
修改I/O调度算法为noop



修改I/O调度算法为cfq



以上数据由于系统I/O”预热“效果，导致初始测试结果和后测结果偏差较大。系统内存也快用光了：



可以看到，三种调度算法，noop和cfq性能最佳，deadline较差。而2.6.32-431.el6.x86\_64内核，默认的调度算法是cfq



想永久的更改I/O调度方法:

修改内核引导参数,加入elevator=调度程序名

vi /boot/grub/menu.lst

更改到如下内容:

kernel /boot/vmlinuz-2.6.18-8.el5 ro root=LABEL=/ elevator=deadline rhgb quiet

重启之后,查看调度方法:

cat /sys/block/sda/queue/scheduler

noop anticipatory [deadline] cfq

已经是deadline了

参考：[Linux I/O 调度算法](http://www.unixhot.com/article/4)

I/O调度的4种算法

1)CFQ(完全公平排队I/O调度程序)

特点:

在最新的内核版本和发行版中,都选择CFQ做为默认的I/O调度器,对于通用的服务器也是最好的选择.

CFQ试图均匀地分布对I/O带宽的访问,避免进程被饿死并实现较低的延迟,是deadline和as调度器的折中.

CFQ对于多媒体应用(video,audio)和桌面系统是最好的选择.

CFQ赋予I/O请求一个优先级,而I/O优先级请求独立于进程优先级,高优先级的进程的读写不能自动地继承高的I/O优先级.

工作原理:

CFQ为每个进程/线程,单独创建一个队列来管理该进程所产生的请求,也就是说每个进程一个队列,各队列之间的调度使用时间片来调度,

以此来保证每个进程都能被很好的分配到I/O带宽.I/O调度器每次执行一个进程的4次请求.

2)NOOP(电梯式调度程序)

特点:

在Linux2.4或更早的版本的调度程序,那时只有这一种I/O调度算法.

NOOP实现了一个简单的FIFO队列,它像电梯的工作主法一样对I/O请求进行组织,当有一个新的请求到来时,它将请求合并到最近的请求之后,以此来保证请求同一介质.

NOOP倾向饿死读而利于写.

NOOP对于闪存设备,RAM,嵌入式系统是最好的选择.

电梯算法饿死读请求的解释:

因为写请求比读请求更容易.

写请求通过文件系统cache,不需要等一次写完成,就可以开始下一次写操作,写请求通过合并,堆积到I/O队列中.

读请求需要等到它前面所有的读操作完成,才能进行下一次读操作.在读操作之间有几毫秒时间,而写请求在这之间就到来,饿死了后面的读请求.

3)Deadline(截止时间调度程序)

特点:

通过时间以及硬盘区域进行分类,这个分类和合并要求类似于noop的调度程序.

Deadline确保了在一个截止时间内服务请求,这个截止时间是可调整的,而默认读期限短于写期限.这样就防止了写操作因为不能被读取而饿死的现象.

Deadline对数据库环境(ORACLE RAC,MYSQL等)是最好的选择.

4)AS(预料I/O调度程序)

特点:

本质上与Deadline一样,但在最后一次读操作后,要等待6ms,才能继续进行对其它I/O请求进行调度.

可以从应用程序中预订一个新的读请求,改进读操作的执行,但以一些写操作为代价.

它会在每个6ms中插入新的I/O操作,而会将一些小写入流合并成一个大写入流,用写入延时换取最大的写入吞吐量.

AS适合于写入较多的环境,比如文件服务器

AS对数据库环境表现很差.

Anticipatory I/O scheduler 适用于大多数环境,但不太合适数据库应用

Deadline I/O scheduler 通常与Anticipatory相当,但更简洁小巧,更适合于数据库应用

CFQ I/O scheduler 为所有进程分配等量的带宽,适合于桌面多任务及多媒体应用，默认IO调度器

Default I/O scheduler

37.CPU iowait

kernel将cpu归为3个状态：idle，iowait，busy；idle是真正的idle，没活可干，没有进程是runnale的，cpu空闲；iowait时cpu没法干活（需要的数据还没就绪呢，所以叫wait），这时候cpu也空闲；busy就是在干活。idle和iowait时虽然cpu都是空闲的，但是将这两个状态区分开是很有意义的，因为iowait可以衡量由io问题导致的cpu空闲。

Zabbix监控CPU的选项中，就涉及idle、iowait、busy三项，当数据库的iowait值很大时，发现mysql导致的iowait飙升，持续时间为10seconds，无法手动解决，自然恢复常态。

可运行进程为0是iowait存在的前提，所以我们讨论iowait这个指标就是衡量cpu没有干活的时间，然后用这个指标来指导系统优化。无论是轻io还是重io的应用，都存在很多方法来降低io对整个系统的影响，比如最简单的把小的追加写在应用层汇集成一个大的追加写就会明显降低iowait，对于一个应用来说把iowait控制得越低程序效率会越高，因为这意味着cpu和io做了很好的隔离，比如像写log这种行为有的程序员可能会把系统的iowait搞成10%，有的程序员采用各种优化措施系统的iowait完全可以为0，真正的功夫就在这里。现在的linux kernel的设计和实现方式就决定了cpu、内存、io是互相干扰的，工程师就是在了解实现细节和约束之后把程序的效率搞到极致。（摘录：淘宝技术博客之[Iowait的成因、对系统影响及对策](http://www.searchtb.com/2013/02/iowait_why_and_optimization.html)）

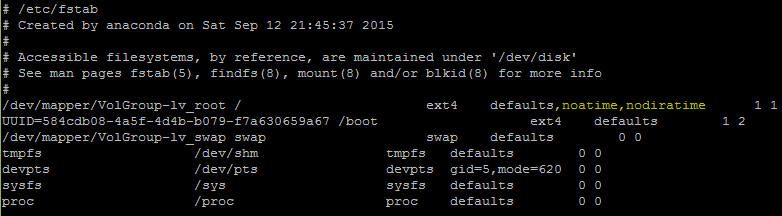
为什么人们会认为 %iowait 偏高是有I/O问题的迹象呢？他们的理由是：”%iowait 的第一个条件是CPU空闲，意即所有的进程都在休眠，第二个条件是仍有未完成的I/O请求，意味着进程休眠的原因是等待I/O，而 %iowait 升高则表明因等待I/O而休眠的进程数量更多了、或者进程因等待I/O而休眠的时间更长了。“ 听上去似乎很有道理，但是不对：首先 %iowait 确实表示CPU空闲、所有进程都在休眠，也确实有的进程在等待I/O，然而 %iowait 升高并不能证明等待I/O的进程数量增多了，也不能证明等待I/O的总时间增加了。。。。。。这就是为什么说 %iowait 所含的信息量非常少的原因，它是一个非常模糊的指标，如果看到 %iowait 升高，还需检查I/O量有没有明显增加，avserv/avwait/avque等指标有没有明显增大，应用有没有感觉变慢，如果都没有，就没什么好担心的。（参考：[理解 %iowait (%wio)](http://linuxperf.com/?p=33)）

38. relatime和noatime

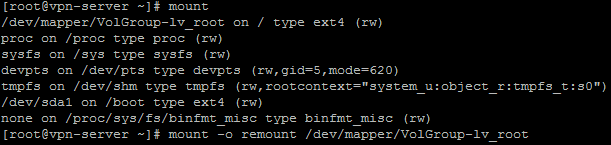
默认的方式下linux会把文件访问的时间atime做记录，这在绝大部分的场合都是没有必要的，如果遇到机器IO负载高或是CPU WAIT高的情况，可以尝试使用noatime和nodiratime。

测试：

修改/etc/fstab文件，在defaults后添加noatime,nodiratime



然后执行mount -o remount /重新挂载根分区



如果不想改fstab，直接用mount命令：

mount -o noatime -o nodiratime -o remount /

noatime就是不记录文件访问时间。以此来节约不必要的磁盘开支，尤其用于读取异常频繁的分区。

relatime在linux 2.6.20内核以后，是默认的磁盘分区挂载参数。在centos6.2中已是默认参数。他的解释有点复杂。简单的说，就是只有在文件的atime比文件的ctime或者mtime更老的情况下，当有最新的访问到来的时候，atime才会被记录并更新。如果atime已经比ctime或者mtime更新了，atime将不会再做任何更新。

未指定"noatime,nodiratime"的情况下：

read文件的时候会导致atime更新，不会导致mtime和ctime更新

write文件只会导致mtime和ctime更新，不会导致atime更新。

指定"noatime,nodiratime"的情况下：

read文件的时候不会导致atime、mtime、ctime改变

write文件只会导致mtime和ctime更新，不会导致atime更新。

参考：[使用noatime属性优化文件系统读取性能](http://hackershell.cn/?p=651)

使用 noatime, nodiratime 或 relatime 可以提升 ext2， ext3 及 ext4 格式磁盘的性能。 Linux 在默认情况下使用atime选项，每次在磁盘上读取（或写入）数据时都会产生一个记录。这是为服务器设计的，在桌面使用中意义不大。默认的 atime 选项最大的问题在于即使从页面缓存读取文件(从内存而不是磁盘读取)，也会产生磁盘写操作！

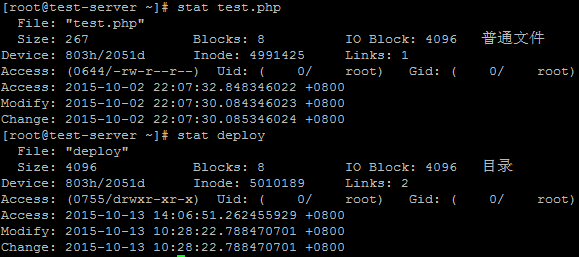
使用 noatime 选项阻止了读文件时的写操作。大部分应用程序都能很好工作。只有少数程序如Mutt需要这些信息。Mutt 的用户应该使用 relatime 选项。使用 relatime 选项后，只有文件被修改时才会产生文件访问时间写操作。nodiratime 选项仅对目录禁用了文件访问时间。relatime 是比较好的折衷，Mutt等程序还能工作，但是仍然能够通过减少访问时间更新提升系统性能。 (参考：[atime](https://wiki.archlinux.org/index.php/Fstab_(简体中文)" \l "atime_.E5.8F.82.E6.95.B0) 参数)

39. stat

stat查看文件、目录的inode值、最后修改访问时间

stat test.php

stat deploy



40.strace跟踪进程

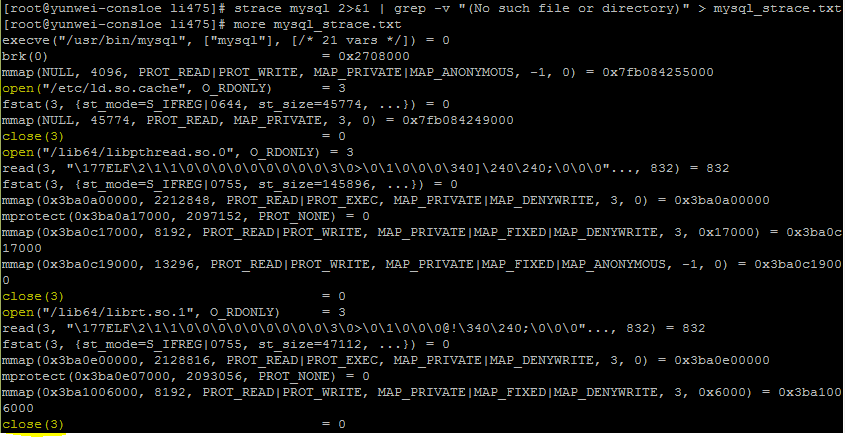
在调试的时候，strace能帮助你追踪到一个程序所执行的系统调用。当你想知道程序和操作系统如何交互的时候，这是极其方便的，比如你想知道执行了哪些系统调用，并且以何种顺序执行。

strace参数

-c 统计每一系统调用的所执行的时间,次数和出错的次数等.  
-d 输出strace关于标准错误的调试信息.  
-f 跟踪由fork调用所产生的子进程.  
-ff 如果提供-o filename,则所有进程的跟踪结果输出到相应的filename.pid中,pid是各进程的进程号.  
-F 尝试跟踪vfork调用.在-f时,vfork不被跟踪.  
-h 输出简要的帮助信息.  
-i 输出系统调用的入口指针.  
-q 禁止输出关于脱离的消息.  
-r 打印出相对时间关于,,每一个系统调用.  
-t 在输出中的每一行前加上时间信息.  
-tt 在输出中的每一行前加上时间信息,微秒级.  
-ttt 微秒级输出,以秒了表示时间.  
-T 显示每一调用所耗的时间.  
-v 输出所有的系统调用.一些调用关于环境变量,状态,输入输出等调用由于使用频繁,默认不输出.  
-V 输出strace的版本信息.  
-x 以十六进制形式输出非标准字符串  
-xx 所有字符串以十六进制形式输出.  
-a column设置返回值的输出位置.默认 为40.  
-e expr指定一个表达式,用来控制如何跟踪.格式如下:[qualifier=][!]value1[,value2]…  
qualifier只能是 trace,abbrev,verbose,raw,signal,read,write其中之一.value是用来限定的符号或数字.默认的 qualifier是 trace.感叹号是否定符号.例如:-eopen等价于 -e trace=open,表示只跟踪open调用.而-etrace!=open表示跟踪除了open以外的其他调用.有两个特殊的符号 all 和 none.  
注意有些shell使用!来执行历史记录里的命令,所以要使用\\.  
-e trace=set只跟踪指定的系统 调用.例如:-e trace=open,close,rean,write表示只跟踪这四个系统调用.默认的为set=all.  
-e trace=file只跟踪有关文件操作的系统调用.  
-e trace=process只跟踪有关进程控制的系统调用.  
-e trace=network跟踪与网络有关的所有系统调用.  
-e strace=signal跟踪所有与系统信号有关的 系统调用  
-e trace=ipc跟踪所有与进程通讯有关的系统调用  
-e abbrev=set设定 strace输出的系统调用的结果集.-v 等与 abbrev=none.默认为abbrev=all.  
-e raw=set将指 定的系统调用的参数以十六进制显示.  
-e signal=set指定跟踪的系统信号.默认为all.如 signal=!SIGIO(或者signal=!io),表示不跟踪SIGIO信号.  
-e read=set输出从指定文件中读出 的数据.例如:  
-e read=3,5  
-e write=set输出写入到指定文件中的数据.  
-o filename将strace的输出写入文件filename  
-p pid跟踪指定的进程pid.  
-s strsize指定输出的字符串的最大长度.默认为32.文件名一直全部输出.  
-u username以username 的UID和GID执行被跟踪的命令

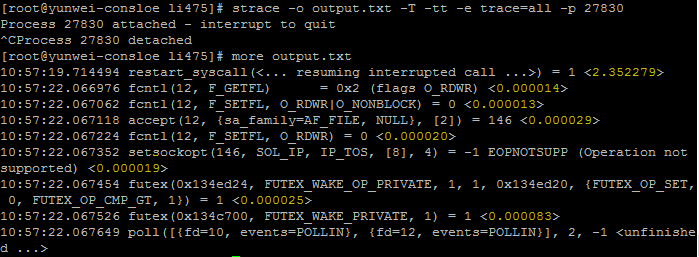
查看某个命令会调用哪些文件，如mysql

strace mysql 2>&1 | grep -v "(No such file or directory)"



利用某个进程的PID，跟踪该进程的所有系统调用（-e trace=all），并统计系统调用的花费时间，以及开始时间（并以可视化的时分秒格式显示）

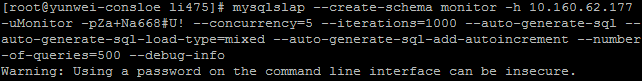
strace -o output.txt -T -tt -e trace=all -p 27830



41.IO TPS

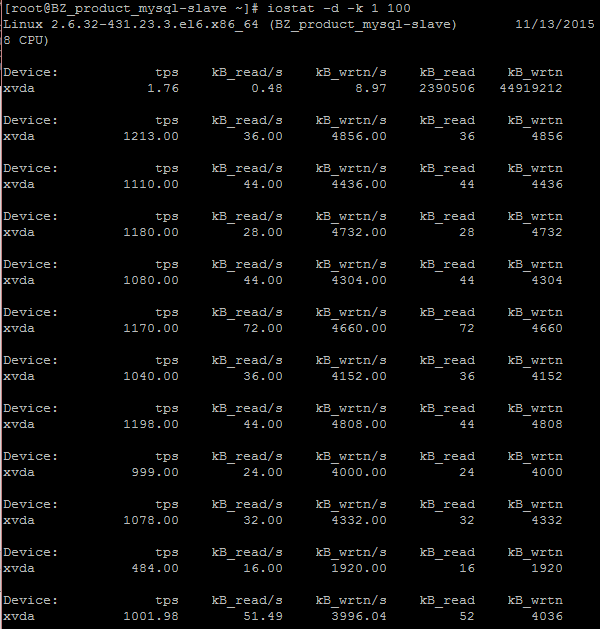
Iostat可以查看IO TPS

利用mysqlslap对mysql进行压测，同时用iostat显示TPS

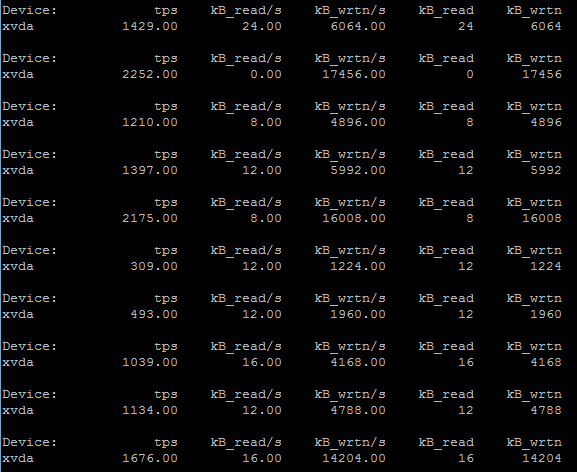


iostat -d -k 1 100参数 -d 表示，显示设备（磁盘）使用状态；-k某些使用block为单位的列强制使用Kilobytes为单位；1 100表示，数据显示每隔1秒刷新一次，共显示100次。

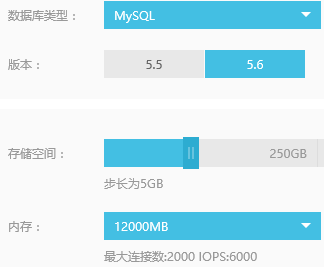
这是阿里ECS 8核16G SSD底层存储的IO TPS，平均为1100；



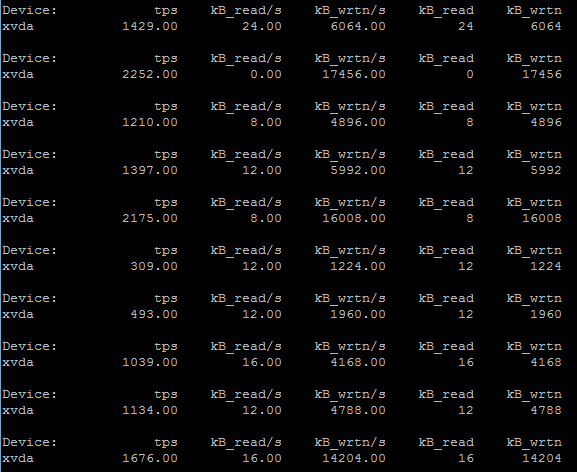
最大已达2250

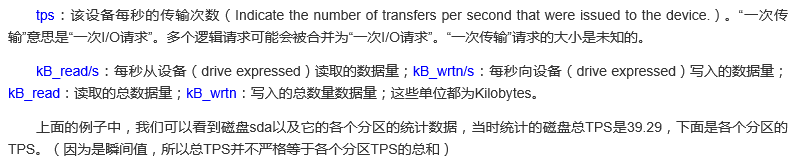


而12G内存的RDS IOTPS标注是6000

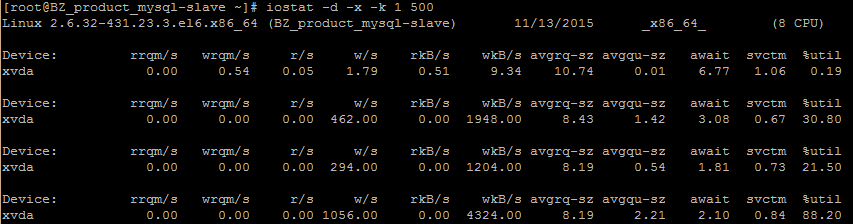


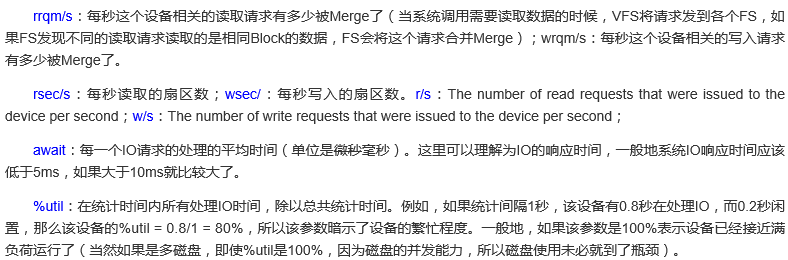
结果分析：

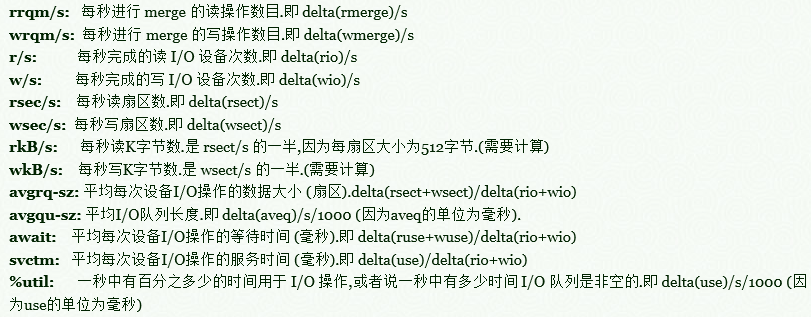




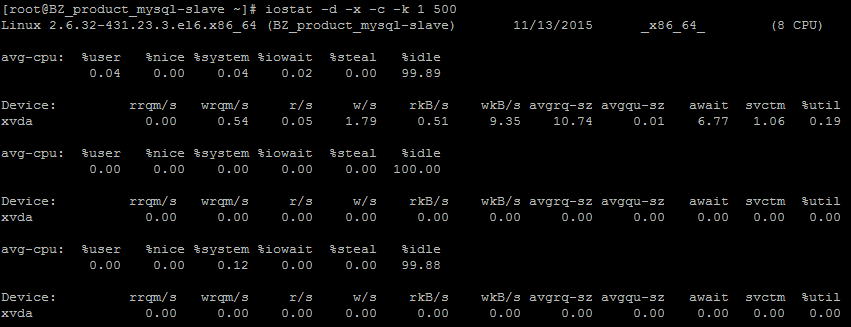
iostat -d -x -k 1 100 查看设备使用率（%util）、响应时间（await）





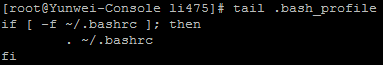


iostat -d -x -c -k 1 500 还获取CPU部分信息

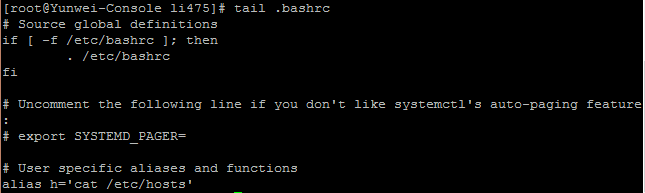


42.alias

自定义alias要写在~/.bashrc，原理是用户登录时会调用~/.bashrc文件



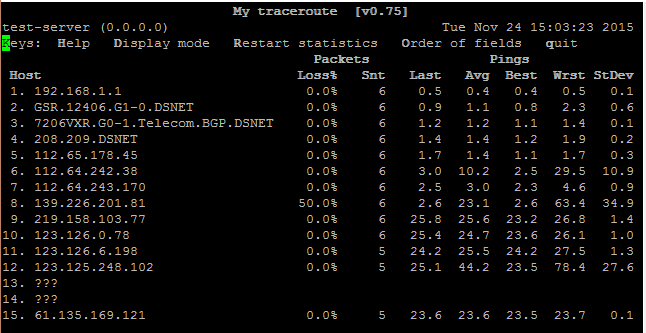
echo "alias h='cat /etc/hosts'" >> .bashrc



43.mtr

mtr – mtr在单一的网络诊断工具上，结合了traceroute和ping程序的功能。它是我见过最神奇的命令行工具

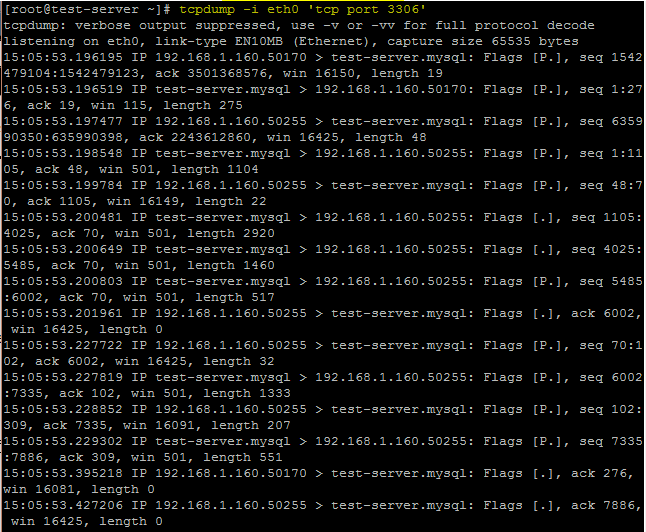
mtr www.baidu.com 结果如下



44. tcpdump

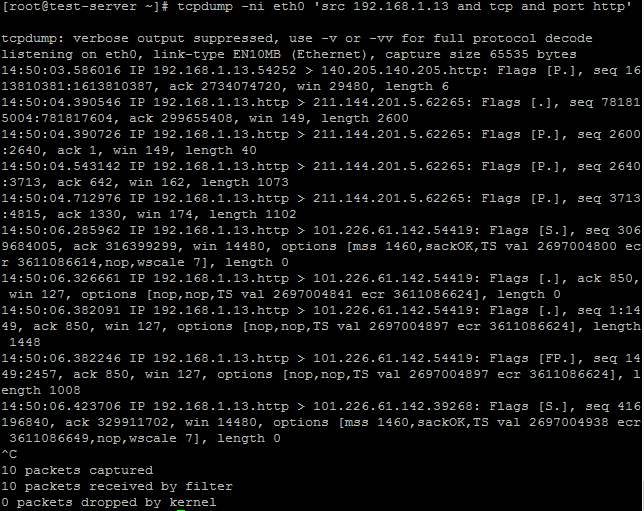
抓取eth0的TCP 3306端口的数据包

tcpdump -i eth0 'tcp port 3306'



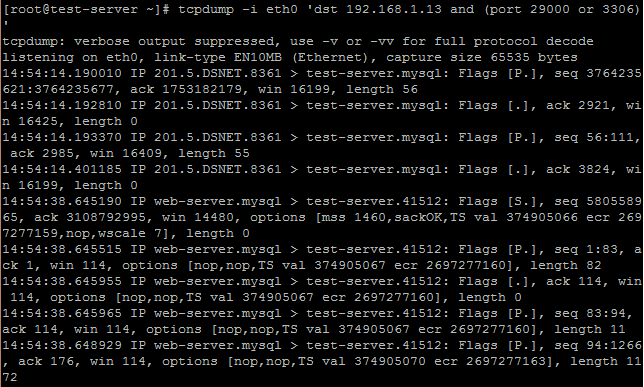
显示从192.168.1.13出发的所有HTTP会话信息：

tcpdump -ni eth0 'src 192.168.1.13 and tcp and port http'



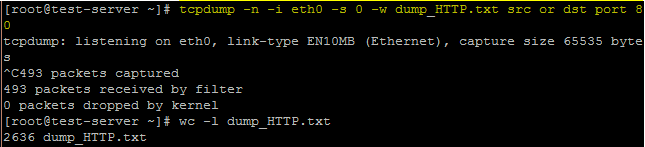
显示目标IP是192.168.1.13，端口号是29000、3306

tcpdump -i eth0 'dst 192.168.1.13 and (port 29000 or 3306)'



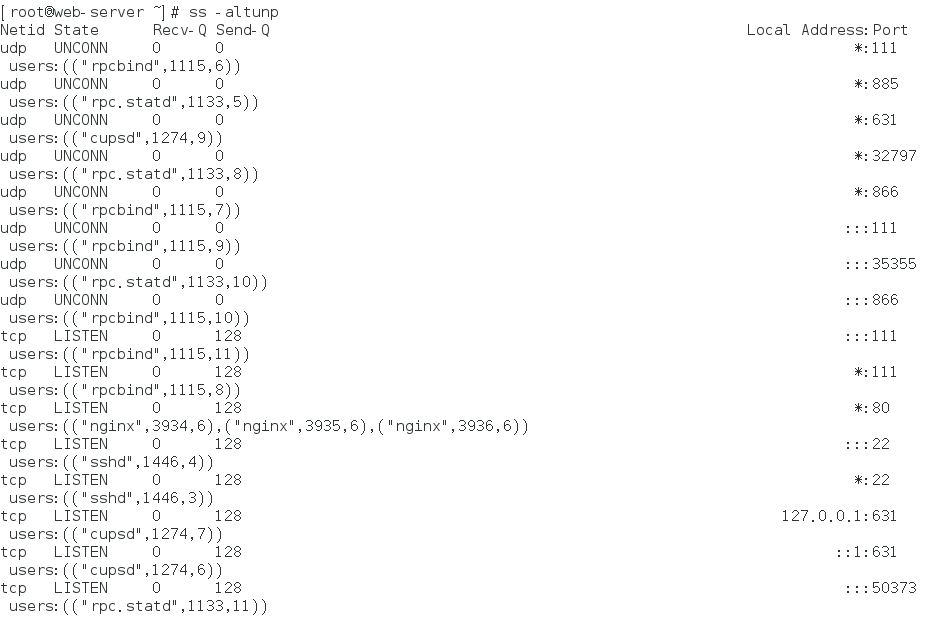
获取TCP 80端口的源、目标数据，将抓包数据存放到dump\_HTTP.txt文件（二进制文件，需要用Wireshark查看）

tcpdump -n -i eth0 -s 0 -w dump\_HTTP.txt src or dst port 80

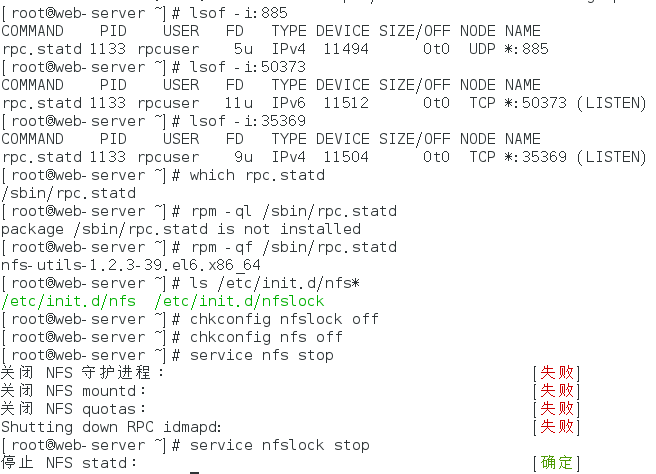
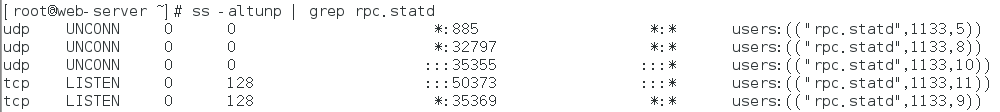


45.灭掉不必要的服务

通过ss命令查看到有一些莫名其妙的端口

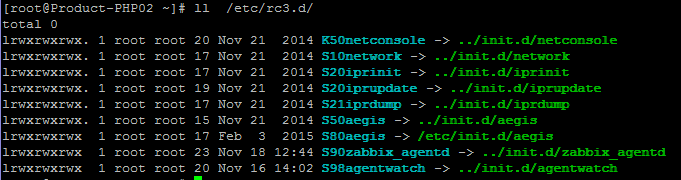


其中，udp 32797、35355和tcp 885、32797都是rpc.statd产生的，进程号是1133，看看它正在执行哪个命令：使用lsof找到开启这些端口的命令，用which找到此命令的来源，用rpm -qf找到此文件的rpm安装包，查找该软件包对应的服务程序



45.查看服务的启动顺序

ll /etc/rc3.d/



46.第三方系统工具

nmap – 扫描主机的端口开放情况.

ntop web based tool – Ntop是一款监控网络流量工具，它显示的网络状况更加直观、详细。Ntop甚至可以列出每个节点计算机的网络带宽利用率。它是一个灵活的、功能齐全的，用来监控和解决局域网问题的工具；可以自动从网络中识别有用的信息；将截获的数据包转换成易于识别的格式；对网络环境中通信失败的情况进行分析；探测网络通信的时间和过程等。

Conky - Conky是x-window下，一款免费的，轻量级系统监控的工具。它能够监控许多系统环境的状态，其中包括的CPU，内存，交换空间，磁盘存储，温度，进程，网络接口，电池电量，系统消息，电子邮件收件箱等。

GKrellM – 它来用于监测CPU状态，内存，硬盘，网络接口，本地和远程邮箱，以及其他的东西等。

vnstat – vnstat是一个基于控制台的网络流量监控软件，它会保持每月，每天，每小时，监视并记录所选定网络接口的网络通信状况。

htop – htop是一个增强版本的top，同时也是一个交互式进程查看器，它可以以树状结构来显示进程列表。

iptraf是一款彩色的交互式的IP局域网监控工具。这是一个基于ncurses的IP LAN监视工具，它会统计网络中产生的各种数据，包括TCP信息，UDP连接数，ICMP和OSPF信息，以太网负载信息，节点统计，IP校验和错误，以及其他信息。并且提供：

Network traffic statistics by TCP connection

IP traffic statistics by network interface

Network traffic statistics by protocol

Network traffic statistics by TCP/UDP port and by packet size

Network traffic statistics by Layer2 address