# 1.top命令

top命令是Linux下常用的性能分析工具，能够实时显示系统中各个进程的资源占用状况，类似于Windows的任务管理器。下面详细介绍它的使用方法。

top - 01:06:48 up 1:22, 1 user, load average: 0.06, 0.60, 0.48  
Tasks: 29 total, 1 running, 28 sleeping, 0 stopped, 0 zombie  
Cpu(s): 0.3% us, 1.0% sy, 0.0% ni, 98.7% id, 0.0% wa, 0.0% hi, 0.0% si  
Mem: 191272k total, 173656k used, 17616k free, 22052k buffers  
Swap: 192772k total, 0k used, 192772k free, 123988k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND  
1379 root 16 0 7976 2456 1980 S 0.7 1.3 0:11.03 sshd  
14704 root 16 0 2128 980 796 R 0.7 0.5 0:02.72 top  
1 root 16 0 1992 632 544 S 0.0 0.3 0:00.90 init  
2 root 34 19 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/0  
3 root RT 0 0 0 0 S 0.0 0.0 0:00.00 watchdog/0

统计信息区  
前五行是系统整体的统计信息。第一行是任务队列信息，同 uptime 命令的执行结果。其内容如下：

01:06:48 当前时间  
up 1:22 系统运行时间，格式为时:分  
1 user 当前登录用户数  
load average: 0.06, 0.60, 0.48 系统负载，即任务队列的平均长度。  
三个数值分别为 1分钟、5分钟、15分钟前到现在的平均值。

第二、三行为进程和CPU的信息。当有多个CPU时，这些内容可能会超过两行。内容如下：

Tasks: 29 total 进程总数  
1 running 正在运行的进程数  
28 sleeping 睡眠的进程数  
0 stopped 停止的进程数  
0 zombie 僵尸进程数  
Cpu(s): 0.3% us 用户空间占用CPU百分比  
1.0% sy 内核空间占用CPU百分比  
0.0% ni 用户进程空间内改变过优先级的进程占用CPU百分比  
98.7% id 空闲CPU百分比  
0.0% wa 等待输入输出的CPU时间百分比  
0.0% hi  
0.0% si

最后两行为内存信息。内容如下：

Mem: 191272k total 物理内存总量  
173656k used 使用的物理内存总量  
17616k free 空闲内存总量  
22052k buffers 用作内核缓存的内存量  
Swap: 192772k total 交换区总量  
0k used 使用的交换区总量  
192772k free 空闲交换区总量  
123988k cached 缓冲的交换区总量。  
内存中的内容被换出到交换区，而后又被换入到内存，但使用过的交换区尚未被覆盖，  
该数值即为这些内容已存在于内存中的交换区的大小。  
相应的内存再次被换出时可不必再对交换区写入。

进程信息区  
统计信息区域的下方显示了各个进程的详细信息。首先来认识一下各列的含义。

序号 列名 含义  
a PID 进程id  
b PPID 父进程id  
c RUSER Real user name  
d UID 进程所有者的用户id  
e USER 进程所有者的用户名  
f GROUP 进程所有者的组名  
g TTY 启动进程的终端名。不是从终端启动的进程则显示为 ?  
h PR 优先级  
i NI nice值。负值表示高优先级，正值表示低优先级  
j P 最后使用的CPU，仅在多CPU环境下有意义  
k %CPU 上次更新到现在的CPU时间占用百分比  
l TIME 进程使用的CPU时间总计，单位秒  
m TIME+ 进程使用的CPU时间总计，单位1/100秒  
n %MEM 进程使用的物理内存百分比  
o VIRT 进程使用的虚拟内存总量，单位kb。VIRT=SWAP+RES  
p SWAP 进程使用的虚拟内存中，被换出的大小，单位kb。  
q RES 进程使用的、未被换出的物理内存大小，单位kb。RES=CODE+DATA  
r CODE 可执行代码占用的物理内存大小，单位kb  
s DATA 可执行代码以外的部分(数据段+栈)占用的物理内存大小，单位kb  
t SHR 共享内存大小，单位kb  
u nFLT 页面错误次数  
v nDRT 最后一次写入到现在，被修改过的页面数。  
w S 进程状态。  
D=不可中断的睡眠状态  
R=运行  
S=睡眠  
T=跟踪/停止  
Z=僵尸进程  
x COMMAND 命令名/命令行  
y WCHAN 若该进程在睡眠，则显示睡眠中的系统函数名  
z Flags 任务标志，参考 sched.h

默认情况下仅显示比较重要的 PID、USER、PR、NI、VIRT、RES、SHR、S、%CPU、%MEM、TIME+、COMMAND 列。可以通过下面的快捷键来更改显示内容。

更改显示内容  
通过 f 键可以选择显示的内容。按 f 键之后会显示列的列表，按 a-z 即可显示或隐藏对应的列，最后按回车键确定。

按 o 键可以改变列的显示顺序。按小写的 a-z 可以将相应的列向右移动，而大写的 A-Z 可以将相应的列向左移动。最后按回车键确定。

按大写的 F 或 O 键，然后按 a-z 可以将进程按照相应的列进行排序。而大写的 R 键可以将当前的排序倒转。

命令使用

1． 工具（命令）名称  
top  
2．工具（命令）作用  
显示系统当前的进程和其他状况； top是一个动态显示过程,即可以通过用户按键来不断刷新当前状态.如果在前台执行该命令,它将独占前台,直到用户终止该程序为止. 比较准确的说,top命令提供了实时的对系统处理器的状态监视.它将显示系统中CPU最“敏感”的任务列表.该命令可以按CPU使用.内存使用和执行时间 对任务进行排序；而且该命令的很多特性都可以通过交互式命令或者在个人定制文件中进行设定.  
3．环境设置  
在Linux下使用。  
4．使用方法  
4．1使用格式  
top [-] [d] [p] [q] [c] [C] [S] [s] [n]  
4．2参数说明  
d 指定每两次屏幕信息刷新之间的时间间隔。当然用户可以使用s交互命令来改变之。  
p 通过指定监控进程ID来仅仅监控某个进程的状态。  
q该选项将使top没有任何延迟的进行刷新。如果调用程序有超级用户权限，那么top将以尽可能高的优先级运行。  
S 指定累计模式  
s 使top命令在安全模式中运行。这将去除交互命令所带来的潜在危险。  
i 使top不显示任何闲置或者僵死进程。  
c 显示整个命令行而不只是显示命令名  
4.3其他  
　　下面介绍在top命令执行过程中可以使用的一些交互命令。从使用角度来看，熟练的掌握这些命令比掌握选项还重要一些。这些命令都是单字母的，如果在命令行选项中使用了s选项，则可能其中一些命令会被屏蔽掉。  
　　Ctrl+L 擦除并且重写屏幕。  
　　h或者? 显示帮助画面，给出一些简短的命令总结说明。  
　　k 终止一个进程。系统将提示用户输入需要终止的进程PID，以及需要发送给该进程什么样的信号。一般的终止进程可以使用15信号；如果不能正常结束那就使用信号9强制结束该进程。默认值是信号15。在安全模式中此命令被屏蔽。  
　　i 忽略闲置和僵死进程。这是一个开关式命令。  
　　q 退出程序。  
　　r 重新安排一个进程的优先级别。系统提示用户输入需要改变的进程PID以及需要设置的进程优先级值。输入一个正值将使优先级降低，反之则可以使该进程拥有更高的优先权。默认值是10。  
　　S 切换到累计模式。  
　　s 改变两次刷新之间的延迟时间。系统将提示用户输入新的时间，单位为s。如果有小数，就换算成m s。输入0值则系统将不断刷新，默认值是5 s。需要注意的是如果设置太小的时间，很可能会引起不断刷新，从而根本来不及看清显示的情况，而且系统负载也会大大增加。  
　　f或者F 从当前显示中添加或者删除项目。  
　　o或者O 改变显示项目的顺序。  
　　l 切换显示平均负载和启动时间信息。  
　　m 切换显示内存信息。  
　　t 切换显示进程和CPU状态信息。  
　　c 切换显示命令名称和完整命令行。  
　　M 根据驻留内存大小进行排序。  
　　P 根据CPU使用百分比大小进行排序。  
　　T 根据时间/累计时间进行排序。  
W 将当前设置写入~/.toprc文件中。这是写top配置文件的推荐方法。

# 2.vmstat

vmstat命令是最常见的Linux/Unix监控工具，可以展现给定时间间隔的服务器的状态值,包括服务器的CPU使用率，内存使用，虚拟内存 交换情况,IO读写情况。这个命令是我查看Linux/Unix最喜爱的命令，一个是Linux/Unix都支持，二是相比top，我可以看到整个机器的 CPU,内存,IO的使用情况，而不是单单看到各个进程的CPU使用率和内存使用率(使用场景不一样)。

一般vmstat工具的使用是通过两个数字参数来完成的，第一个参数是采样的时间间隔数，单位是秒，第二个参数是采样的次数，如:

root@ubuntu:~# vmstat 2 1  
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----  
 r b swpd free buff cache si so bi bo in cs us sy id wa  
 1 0 0 3498472 315836 3819540 0 0 0 1 2 0 0 0 100 0

2表示每个两秒采集一次服务器状态，1表示只采集一次。

实际上，在应用过程中，我们会在一段时间内一直监控，不想监控直接结束vmstat就行了,例如:

[复制代码](javascript:void(0);)

root@ubuntu:~# vmstat 2   
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----  
 r b swpd free buff cache si so bi bo in cs us sy id wa  
 1 0 0 3499840 315836 3819660 0 0 0 1 2 0 0 0 100 0  
 0 0 0 3499584 315836 3819660 0 0 0 0 88 158 0 0 100 0  
 0 0 0 3499708 315836 3819660 0 0 0 2 86 162 0 0 100 0  
 0 0 0 3499708 315836 3819660 0 0 0 10 81 151 0 0 100 0  
 1 0 0 3499732 315836 3819660 0 0 0 2 83 154 0 0 100 0

[复制代码](javascript:void(0);)

这表示vmstat每2秒采集数据，一直采集，直到我结束程序，这里采集了5次数据我就结束了程序。

好了，命令介绍完毕，现在开始实战讲解每个参数的意思。

**r** 表示运行队列(就是说多少个进程真的分配到CPU)，我测试的服务器目前CPU比较空闲，没什么程序在跑，当这个值超过了CPU数目，就会出现CPU瓶颈 了。这个也和top的负载有关系，一般负载超过了3就比较高，超过了5就高，超过了10就不正常了，服务器的状态很危险。top的负载类似每秒的运行队 列。如果运行队列过大，表示你的CPU很繁忙，一般会造成CPU使用率很高。

**b** 表示阻塞的进程,这个不多说，进程阻塞，大家懂的。

**swpd** 虚拟内存已使用的大小，如果大于0，表示你的机器物理内存不足了，如果不是程序内存泄露的原因，那么你该升级内存了或者把耗内存的任务迁移到其他机器。

**free**   空闲的物理内存的大小，我的机器内存总共8G，剩余3415M。

**buff**   Linux/Unix系统是用来存储，目录里面有什么内容，权限等的缓存，我本机大概占用300多M

**cache** cache直接用来记忆我们打开的文件,给文件做缓冲，我本机大概占用300多M(这里是Linux/Unix的聪明之处，把空闲的物理内存的一部分拿来做文件和目录的缓存，是为了提高 程序执行的性能，当程序使用内存时，buffer/cached会很快地被使用。)

**si**  每秒从磁盘读入虚拟内存的大小，如果这个值大于0，表示物理内存不够用或者内存泄露了，要查找耗内存进程解决掉。我的机器内存充裕，一切正常。

**so**  每秒虚拟内存写入磁盘的大小，如果这个值大于0，同上。

**bi**  块设备每秒接收的块数量，这里的块设备是指系统上所有的磁盘和其他块设备，默认块大小是1024byte，我本机上没什么IO操作，所以一直是0，但是我曾在处理拷贝大量数据(2-3T)的机器上看过可以达到140000/s，磁盘写入速度差不多140M每秒

**bo** 块设备每秒发送的块数量，例如我们读取文件，bo就要大于0。bi和bo一般都要接近0，不然就是IO过于频繁，需要调整。

**in** 每秒CPU的中断次数，包括时间中断

**cs** 每秒上下文切换次数，例如我们调用系统函数，就要进行上下文切换，线程的切换，也要进程上下文切换，这个值要越小越好，太大了，要考虑调低线程或者进程的 数目,例如在apache和nginx这种web服务器中，我们一般做性能测试时会进行几千并发甚至几万并发的测试，选择web服务器的进程可以由进程或 者线程的峰值一直下调，压测，直到cs到一个比较小的值，这个进程和线程数就是比较合适的值了。系统调用也是，每次调用系统函数，我们的代码就会进入内核 空间，导致上下文切换，这个是很耗资源，也要尽量避免频繁调用系统函数。上下文切换次数过多表示你的CPU大部分浪费在上下文切换，导致CPU干正经事的 时间少了，CPU没有充分利用，是不可取的。

**us** 用户CPU时间，我曾经在一个做加密解密很频繁的服务器上，可以看到us接近100,r运行队列达到80(机器在做压力测试，性能表现不佳)。

**sy** 系统CPU时间，如果太高，表示系统调用时间长，例如是IO操作频繁。

**id**  空闲 CPU时间，一般来说，id + us + sy = 100,一般我认为id是空闲CPU使用率，us是用户CPU使用率，sy是系统CPU使用率。

**wt** 等待IO CPU时间。

vmstat –s 可以查看page in/page out的数量

# 3.ps命令

Linux中的ps命令是Process Status的缩写。ps命令用来列出系统中当前运行的那些进程。ps命令列出的是当前那些进程的快照，就是执行ps命令的那个时刻的那些进程，如果想要动态的显示进程信息，就可以使用top命令。

要 对进程进行监测和控制，首先必须要了解当前进程的情况，也就是需要查看当前进程，而 ps 命令就是最基本同时也是非常强大的进程查看命令。使用该命令可 以确定有哪些进程正在运行和运行的状态、进程是否结束、进程有没有僵死、哪些进程占用了过多的资源等等。总之大部分信息都是可以通过执行该命令得到的。

ps 为我们提供了进程的一次性的查看，它所提供的查看结果并不动态连续的；如果想对进程时间监控，应该用 top 工具。

kill 命令用于杀死进程。

**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)

**ps工具标识进程的5种状态码:**

D 不可中断 uninterruptible sleep (usually IO)

R 运行 runnable (on run queue)

S 中断 sleeping

T 停止 traced or stopped

Z 僵死 a defunct (”zombie”) process

**1．命令格式：**

ps[参数]

**2．命令功能：**

用来显示当前进程的状态

**3．命令参数：**

a 显示所有进程

-a 显示同一终端下的所有程序

-A 显示所有进程

c 显示进程的真实名称

-N 反向选择

-e 等于“-A”

e 显示环境变量

f 显示程序间的关系

-H 显示树状结构

r 显示当前终端的进程

T 显示当前终端的所有程序

u 指定用户的所有进程

-au 显示较详细的资讯

-aux 显示所有包含其他使用者的行程

-C<命令> 列出指定命令的状况

--lines<行数> 每页显示的行数

--width<字符数> 每页显示的字符数

--help 显示帮助信息

--version 显示版本显示

**4．使用实例：**

**实例1：显示所有进程信息**

**命令：**

ps -A

**输出：**

[root@localhost test6]# ps -A

  PID TTY          TIME CMD

    1 ?        00:00:00 init

    2 ?        00:00:01 migration/0

    3 ?        00:00:00 ksoftirqd/0

    4 ?        00:00:01 migration/1

    5 ?        00:00:00 ksoftirqd/1

    6 ?        00:29:57 events/0

    7 ?        00:00:00 events/1

    8 ?        00:00:00 khelper

   49 ?        00:00:00 kthread

   54 ?        00:00:00 kblockd/0

   55 ?        00:00:00 kblockd/1

   56 ?        00:00:00 kacpid

  217 ?        00:00:00 cqueue/0

  ……省略部分结果

**说明：**

**实例2：显示指定用户信息**

**命令：**

ps -u root

**输出：**

[root@localhost test6]# ps -u root

  PID TTY          TIME CMD

    1 ?        00:00:00 init

    2 ?        00:00:01 migration/0

    3 ?        00:00:00 ksoftirqd/0

    4 ?        00:00:01 migration/1

    5 ?        00:00:00 ksoftirqd/1

    6 ?        00:29:57 events/0

    7 ?        00:00:00 events/1

    8 ?        00:00:00 khelper

   49 ?        00:00:00 kthread

   54 ?        00:00:00 kblockd/0

   55 ?        00:00:00 kblockd/1

   56 ?        00:00:00 kacpid

    ……省略部分结果

**说明：**

**实例3：显示所有进程信息，连同命令行**

**命令：**

ps -ef

**输出：**

[root@localhost test6]# ps -ef

UID        PID  PPID  C STIME TTY          TIME CMD

root         1     0  0 Nov02 ?        00:00:00 init [3]

root         2     1  0 Nov02 ?        00:00:01 [migration/0]

root         3     1  0 Nov02 ?        00:00:00 [ksoftirqd/0]

root         4     1  0 Nov02 ?        00:00:01 [migration/1]

root         5     1  0 Nov02 ?        00:00:00 [ksoftirqd/1]

root         6     1  0 Nov02 ?        00:29:57 [events/0]

root         7     1  0 Nov02 ?        00:00:00 [events/1]

root         8     1  0 Nov02 ?        00:00:00 [khelper]

root        49     1  0 Nov02 ?        00:00:00 [kthread]

root        54    49  0 Nov02 ?        00:00:00 [kblockd/0]

root        55    49  0 Nov02 ?        00:00:00 [kblockd/1]

root        56    49  0 Nov02 ?        00:00:00 [kacpid]

……省略部分结果

**说明：**

**实例4：** **ps 与grep 常用组合用法，查找特定进程**

**命令：**

ps -ef|grep ssh

**输出：**

[root@localhost test6]# ps -ef|grep ssh

root      2720     1  0 Nov02 ?        00:00:00 /usr/sbin/sshd

root     17394  2720  0 14:58 ?        00:00:00 sshd: root@pts/0

root     17465 17398  0 15:57 pts/0    00:00:00 grep ssh

**说明：**

**实例5：将目前属于您自己这次登入的 PID 与相关信息列示出来**

**命令：**

ps -l

**输出：**

[root@localhost test6]# ps -l

F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD

4 S     0 17398 17394  0  75   0 - 16543 wait   pts/0    00:00:00 bash

4 R     0 17469 17398  0  77   0 - 15877 -      pts/0    00:00:00 ps

**说明：**

各相关信息的意义：

F 代表这个程序的旗标 (flag)， 4 代表使用者为 super user

S 代表这个程序的状态 (STAT)，关于各 STAT 的意义将在内文介绍

UID 程序被该 UID 所拥有

PID 就是这个程序的 ID ！

PPID 则是其上级父程序的ID

C CPU 使用的资源百分比

PRI 这个是 Priority (优先执行序) 的缩写，详细后面介绍

NI 这个是 Nice 值，在下一小节我们会持续介绍

ADDR 这个是 kernel function，指出该程序在内存的那个部分。如果是个 running的程序，一般就是 "-"

SZ 使用掉的内存大小

WCHAN 目前这个程序是否正在运作当中，若为 - 表示正在运作

TTY 登入者的终端机位置

TIME 使用掉的 CPU 时间。

CMD 所下达的指令为何

在预设的情况下， ps 仅会列出与目前所在的 bash shell 有关的 PID 而已，所以， 当我使用 ps -l 的时候，只有三个 PID。

**实例6：列出目前所有的正在内存当中的程序**

**命令：**

ps aux

**输出：**

[root@localhost test6]# ps aux

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

root         1  0.0  0.0  10368   676 ?        Ss   Nov02   0:00 init [3]

root         2  0.0  0.0      0     0 ?        S<   Nov02   0:01 [migration/0]

root         3  0.0  0.0      0     0 ?        SN   Nov02   0:00 [ksoftirqd/0]

root         4  0.0  0.0      0     0 ?        S<   Nov02   0:01 [migration/1]

root         5  0.0  0.0      0     0 ?        SN   Nov02   0:00 [ksoftirqd/1]

root         6  0.0  0.0      0     0 ?        S<   Nov02  29:57 [events/0]

root         7  0.0  0.0      0     0 ?        S<   Nov02   0:00 [events/1]

root         8  0.0  0.0      0     0 ?        S<   Nov02   0:00 [khelper]

root        49  0.0  0.0      0     0 ?        S<   Nov02   0:00 [kthread]

root        54  0.0  0.0      0     0 ?        S<   Nov02   0:00 [kblockd/0]

root        55  0.0  0.0      0     0 ?        S<   Nov02   0:00 [kblockd/1]

root        56  0.0  0.0      0     0 ?        S<   Nov02   0:00 [kacpid]

……省略部分结果

**说明：**

USER：该 process 属于那个使用者账号的

PID ：该 process 的号码

%CPU：该 process 使用掉的 CPU 资源百分比

%MEM：该 process 所占用的物理内存百分比

VSZ ：该 process 使用掉的虚拟内存量 (Kbytes)

RSS ：该 process 占用的固定的内存量 (Kbytes)

TTY ：该 process 是在那个终端机上面运作，若与终端机无关，则显示 ?，另外， tty1-tty6 是本机上面的登入者程序，若为 pts/0 等等的，则表示为由网络连接进主机的程序。

STAT：该程序目前的状态，主要的状态有

R ：该程序目前正在运作，或者是可被运作

S ：该程序目前正在睡眠当中 (可说是 idle 状态)，但可被某些讯号 (signal) 唤醒。

T ：该程序目前正在侦测或者是停止了

Z ：该程序应该已经终止，但是其父程序却无法正常的终止他，造成 zombie (疆尸) 程序的状态

START：该 process 被触发启动的时间

TIME ：该 process 实际使用 CPU 运作的时间

COMMAND：该程序的实际指令

**实例7：列出类似程序树的程序显示**

**命令：**

ps -axjf

**输出：**

[root@localhost test6]# ps -axjf

Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.7/FAQ

 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND

    0     1     1     1 ?           -1 Ss       0   0:00 init [3]

    1     2     1     1 ?           -1 S<       0   0:01 [migration/0]

    1     3     1     1 ?           -1 SN       0   0:00 [ksoftirqd/0]

    1     4     1     1 ?           -1 S<       0   0:01 [migration/1]

    1     5     1     1 ?           -1 SN       0   0:00 [ksoftirqd/1]

    1     6     1     1 ?           -1 S<       0  29:58 [events/0]

    1     7     1     1 ?           -1 S<       0   0:00 [events/1]

    1     8     1     1 ?           -1 S<       0   0:00 [khelper]

    1    49     1     1 ?           -1 S<       0   0:00 [kthread]

   49    54     1     1 ?           -1 S<       0   0:00  \\_ [kblockd/0]

   49    55     1     1 ?           -1 S<       0   0:00  \\_ [kblockd/1]

   49    56     1     1 ?           -1 S<       0   0:00  \\_ [kacpid]

**说明：**

**实例8：找出与 cron 与 syslog 这两个服务有关的 PID 号码**

**命令：**

**输出：**

[root@localhost test6]# ps aux | egrep '(cron|syslog)'

root      2682  0.0  0.0  83384  2000 ?        Sl   Nov02   0:00 /sbin/rsyslogd -i /var/run/syslogd.pid -c 5

root      2735  0.0  0.0  74812  1140 ?        Ss   Nov02   0:00 crond

root     17475  0.0  0.0  61180   832 pts/0    S+   16:27   0:00 egrep (cron|syslog)

[root@localhost test6]#

**说明：**

**其他实例：**

1. 可以用 | 管道和 more 连接起来分页查看

命令：

ps -aux |more

2. 把所有进程显示出来，并输出到ps001.txt文件

命令：

ps -aux > ps001.txt

3. 输出指定的字段

命令：

ps -o pid,ppid,pgrp,session,tpgid,comm

输出：

[root@localhost test6]# ps -o pid,ppid,pgrp,session,tpgid,comm

  PID  PPID  PGRP  SESS TPGID COMMAND

17398 17394 17398 17398 17478 bash

17478 17398 17478 17398 17478 ps

[root@localhost test6]#

# 4.iostat

对于I/O-bond类型的进程，我们经常用iostat工具查看进程IO请求下发的数量、系统处理IO请求的耗时，进而分析进程与操作系统的交互过程中IO方面是否存在瓶颈。

下面通过iostat命令使用实例，说明使用iostat查看IO请求下发情况、系统IO处理能力的方法，以及命令执行结果中各字段的含义。

**1.不加选项执行iostat**

我们先来看直接执行iostat的输出结果：

[复制代码](javascript:void(0);)

linux # **iostat**

Linux 2.6.16.60-0.21-smp (linux) 06/12/12

avg-cpu: %user %nice %system %iowait %steal %idle

0.07 0.00 0.05 0.06 0.00 99.81

Device: tps Blk\_read/s Blk\_wrtn/s Blk\_read Blk\_wrtn

sda 0.58 9.95 37.47 6737006 25377400

sdb 0.00 0.00 0.00 824 0

[复制代码](javascript:void(0);)

单独执行iostat，显示的结果为从系统开机到当前执行时刻的统计信息。以上输出中，除最上面指示系统版本、主机名和日期的一行外，另有两部分：

**avg-cpu**: 总体cpu使用情况统计信息，对于多核cpu，这里为所有cpu的平均值

**Device**: 各磁盘设备的IO统计信息

对于cpu统计信息一行，我们主要看iowait的值，它指示cpu用于等待io请求完成的时间。Device中各列含义如下：

* **Device**: 以sdX形式显示的设备名称
* **tps**: 每秒进程下发的IO读、写请求数量
* **Blk\_read/s**: 每秒读扇区数量(一扇区为512bytes)
* **Blk\_wrtn/s**: 每秒写扇区数量
* **Blk\_read**: 取样时间间隔内读扇区总数量
* **Blk\_wrtn**: 取样时间间隔内写扇区总数量

我们可以使用-c选项单独显示avg-cpu部分的结果，使用-d选项单独显示Device部分的信息。

**2.指定采样时间间隔与采样次数**

与sar命令一样，我们可以以"**iostat interval [count]** ”形式指定iostat命令的采样间隔和采样次数：

[复制代码](javascript:void(0);)

linux # **iostat -d 1 2**

Linux 2.6.16.60-0.21-smp (linux) 06/13/12

Device: tps Blk\_read/s Blk\_wrtn/s Blk\_read Blk\_wrtn

sda 0.55 8.93 36.27 6737086 27367728

sdb 0.00 0.00 0.00 928 0

Device: tps Blk\_read/s Blk\_wrtn/s Blk\_read Blk\_wrtn

sda 2.00 0.00 72.00 0 72

sdb 0.00 0.00 0.00 0 0

[复制代码](javascript:void(0);)

以上命令输出Device的信息，采样时间为1秒，采样2次，若不指定采样次数，则iostat会一直输出采样信息，直到按”ctrl+c”退出命令。注意，第1次采样信息与单独执行iostat的效果一样，为从系统开机到当前执行时刻的统计信息。

**3.以kB为单位显示读写信息(-k选项)**

我们可以使用-k选项，指定iostat的部分输出结果以kB为单位，而不是以扇区数为单位：

[复制代码](javascript:void(0);)

linux # **iostat -d -k**

Linux 2.6.16.60-0.21-smp (linux) 06/13/12

Device: tps kB\_read/s kB\_wrtn/s kB\_read kB\_wrtn

sda 0.55 4.46 18.12 3368543 13686096

sdb 0.00 0.00 0.00 464 0

[复制代码](javascript:void(0);)

以上输出中，kB\_read/s、kB\_wrtn/s、kB\_read和kB\_wrtn的值均以kB为单位，相比以扇区数为单位，这里的值为原值的一半(1kB=512bytes\*2)

**4.更详细的io统计信息(-x选项)**

为显示更详细的io设备统计信息，我们可以使用-x选项，在分析io瓶颈时，一般都会开启-x选项：

[复制代码](javascript:void(0);)

linux # **iostat -x -k -d 1**

Linux 2.6.16.60-0.21-smp (linux) 06/13/12

……

Device: rrqm/s wrqm/s r/s w/s rkB/s wkB/s avgrq-sz avgqu-sz await svctm %util

sda 0.00 9915.00 1.00 90.00 4.00 34360.00 755.25 11.79 120.57 6.33 57.60

[复制代码](javascript:void(0);)

以上各列的含义如下：

* **rrqm/s**: 每秒对该设备的读请求被合并次数，文件系统会对读取同块(block)的请求进行合并
* **wrqm/s**: 每秒对该设备的写请求被合并次数
* **r/s**: 每秒完成的读次数
* **w/s**: 每秒完成的写次数
* **rkB/s**: 每秒读数据量(kB为单位)
* **wkB/s**: 每秒写数据量(kB为单位)
* **avgrq-sz**:平均每次IO操作的数据量(扇区数为单位)
* **avgqu-sz**: 平均等待处理的IO请求队列长度
* **await**: 平均每次IO请求等待时间(包括等待时间和处理时间，毫秒为单位)
* **svctm**: 平均每次IO请求的处理时间(毫秒为单位)
* **%util**: 采用周期内用于IO操作的时间比率，即IO队列非空的时间比率

对于以上示例输出，我们可以获取到以下信息：

1. 每秒向磁盘上写30M左右数据(wkB/s值)
2. 每秒有91次IO操作(r/s+w/s)，其中以写操作为主体
3. 平均每次IO请求等待处理的时间为120.57毫秒，处理耗时为6.33毫秒
4. 等待处理的IO请求队列中，平均有11.79个请求驻留

以上各值之间也存在联系，我们可以由一些值计算出其他数值，例如：

util = (r/s+w/s) \* (svctm/1000)

对于上面的例子有：util = (1+90)\*(6.33/1000) = 0.57603

# 5.netstat

简介

Netstat 命令用于显示各种网络相关信息，如网络连接，路由表，接口状态 (Interface Statistics)，masquerade 连接，多播成员 (Multicast Memberships) 等等。

输出信息含义

执行netstat后，其输出结果为

Active Internet connections (w/o servers)

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 2 210.34.6.89:telnet 210.34.6.96:2873 ESTABLISHED

tcp 296 0 210.34.6.89:1165 210.34.6.84:netbios-ssn ESTABLISHED

tcp 0 0 localhost.localdom:9001 localhost.localdom:1162 ESTABLISHED

tcp 0 0 localhost.localdom:1162 localhost.localdom:9001 ESTABLISHED

tcp 0 80 210.34.6.89:1161 210.34.6.10:netbios-ssn CLOSE

Active UNIX domain sockets (w/o servers)

Proto RefCnt Flags Type State I-Node Path

unix 1 [ ] STREAM CONNECTED 16178 @000000dd

unix 1 [ ] STREAM CONNECTED 16176 @000000dc

unix 9 [ ] DGRAM 5292 /dev/log

unix 1 [ ] STREAM CONNECTED 16182 @000000df

从整体上看，netstat的输出结果可以分为两个部分：

一个是Active Internet connections，称为有源TCP连接，其中"Recv-Q"和"Send-Q"指%0A的是接收队列和发送队列。这些数字一般都应该是0。如果不是则表示软件包正在队列中堆积。这种情况只能在非常少的情况见到。

另一个是Active UNIX domain sockets，称为有源Unix域套接口(和网络套接字一样，但是只能用于本机通信，性能可以提高一倍)。

Proto显示连接使用的协议,RefCnt表示连接到本套接口上的进程号,Types显示套接口的类型,State显示套接口当前的状态,Path表示连接到套接口的其它进程使用的路径名。

常见参数

-a (all)显示所有选项，默认不显示LISTEN相关

-t (tcp)仅显示tcp相关选项

-u (udp)仅显示udp相关选项

-n 拒绝显示别名，能显示数字的全部转化成数字。

-l 仅列出有在 Listen (监听) 的服務状态

-p 显示建立相关链接的程序名

-r 显示路由信息，路由表

-e 显示扩展信息，例如uid等

-s 按各个协议进行统计

-c 每隔一个固定时间，执行该netstat命令。

提示：LISTEN和LISTENING的状态只有用-a或者-l才能看到

实用命令实例

1. 列出所有端口 (包括监听和未监听的)

列出所有端口 netstat -a

# netstat -a | more

Active Internet connections (servers and established)

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 0 localhost:30037 \*:\* LISTEN

udp 0 0 \*:bootpc \*:\*

Active UNIX domain sockets (servers and established)

Proto RefCnt Flags Type State I-Node Path

unix 2 [ ACC ] STREAM LISTENING 6135 /tmp/.X11-unix/X0

unix 2 [ ACC ] STREAM LISTENING 5140 /var/run/acpid.socket

列出所有 tcp 端口 netstat -at

# netstat -at

Active Internet connections (servers and established)

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 0 localhost:30037 \*:\* LISTEN

tcp 0 0 localhost:ipp \*:\* LISTEN

tcp 0 0 \*:smtp \*:\* LISTEN

tcp6 0 0 localhost:ipp [::]:\* LISTEN

列出所有 udp 端口 netstat -au

# netstat -au

Active Internet connections (servers and established)

Proto Recv-Q Send-Q Local Address Foreign Address State

udp 0 0 \*:bootpc \*:\*

udp 0 0 \*:49119 \*:\*

udp 0 0 \*:mdns \*:\*

2. 列出所有处于监听状态的 Sockets

只显示监听端口 netstat -l

# netstat -l

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 0 localhost:ipp \*:\* LISTEN

tcp6 0 0 localhost:ipp [::]:\* LISTEN

udp 0 0 \*:49119 \*:\*

只列出所有监听 tcp 端口 netstat -lt

# netstat -lt

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 0 localhost:30037 \*:\* LISTEN

tcp 0 0 \*:smtp \*:\* LISTEN

tcp6 0 0 localhost:ipp [::]:\* LISTEN

只列出所有监听 udp 端口 netstat -lu

# netstat -lu

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State

udp 0 0 \*:49119 \*:\*

udp 0 0 \*:mdns \*:\*

只列出所有监听 UNIX 端口 netstat -lx

# netstat -lx

Active UNIX domain sockets (only servers)

Proto RefCnt Flags Type State I-Node Path

unix 2 [ ACC ] STREAM LISTENING 6294 private/maildrop

unix 2 [ ACC ] STREAM LISTENING 6203 public/cleanup

unix 2 [ ACC ] STREAM LISTENING 6302 private/ifmail

unix 2 [ ACC ] STREAM LISTENING 6306 private/bsmtp

3. 显示每个协议的统计信息

显示所有端口的统计信息 netstat -s

# netstat -s

Ip:

11150 total packets received

1 with invalid addresses

0 forwarded

0 incoming packets discarded

11149 incoming packets delivered

11635 requests sent out

Icmp:

0 ICMP messages received

0 input ICMP message failed.

Tcp:

582 active connections openings

2 failed connection attempts

25 connection resets received

Udp:

1183 packets received

4 packets to unknown port received.

.....

显示 TCP 或 UDP 端口的统计信息 netstat -st 或 -su

# netstat -st

# netstat -su

4. 在 netstat 输出中显示 PID 和进程名称 netstat -p

netstat -p 可以与其它开关一起使用，就可以添加 “PID/进程名称” 到 netstat 输出中，这样 debugging 的时候可以很方便的发现特定端口运行的程序。

# netstat -pt

Active Internet connections (w/o servers)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

tcp 1 0 ramesh-laptop.loc:47212 192.168.185.75:www CLOSE\_WAIT 2109/firefox

tcp 0 0 ramesh-laptop.loc:52750 lax:www ESTABLISHED 2109/firefox

5. 在 netstat 输出中不显示主机，端口和用户名 (host, port or user)

当你不想让主机，端口和用户名显示，使用 netstat -n。将会使用数字代替那些名称。

同样可以加速输出，因为不用进行比对查询。

# netstat -an

如果只是不想让这三个名称中的一个被显示，使用以下命令

# netsat -a --numeric-ports

# netsat -a --numeric-hosts

# netsat -a --numeric-users

6. 持续输出 netstat 信息

netstat 将每隔一秒输出网络信息。

# netstat -c

Active Internet connections (w/o servers)

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 0 ramesh-laptop.loc:36130 101-101-181-225.ama:www ESTABLISHED

tcp 1 1 ramesh-laptop.loc:52564 101.11.169.230:www CLOSING

tcp 0 0 ramesh-laptop.loc:43758 server-101-101-43-2:www ESTABLISHED

tcp 1 1 ramesh-laptop.loc:42367 101.101.34.101:www CLOSING

^C

7. 显示系统不支持的地址族 (Address Families)

netstat --verbose

在输出的末尾，会有如下的信息

netstat: no support for `AF IPX' on this system.

netstat: no support for `AF AX25' on this system.

netstat: no support for `AF X25' on this system.

netstat: no support for `AF NETROM' on this system.

8. 显示核心路由信息 netstat -r

# netstat -r

Kernel IP routing table

Destination Gateway Genmask Flags MSS Window irtt Iface

192.168.1.0 \* 255.255.255.0 U 0 0 0 eth2

link-local \* 255.255.0.0 U 0 0 0 eth2

default 192.168.1.1 0.0.0.0 UG 0 0 0 eth2

注意： 使用 netstat -rn 显示数字格式，不查询主机名称。

9. 找出程序运行的端口

并不是所有的进程都能找到，没有权限的会不显示，使用 root 权限查看所有的信息。

# netstat -ap | grep ssh

tcp 1 0 dev-db:ssh 101.174.100.22:39213 CLOSE\_WAIT -

tcp 1 0 dev-db:ssh 101.174.100.22:57643 CLOSE\_WAIT -

找出运行在指定端口的进程

# netstat -an | grep ':80'

10. 显示网络接口列表

# netstat -i

Kernel Interface table

Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flg

eth0 1500 0 0 0 0 0 0 0 0 0 BMU

eth2 1500 0 26196 0 0 0 26883 6 0 0 BMRU

lo 16436 0 4 0 0 0 4 0 0 0 LRU

显示详细信息，像是 ifconfig 使用 netstat -ie:

# netstat -ie

Kernel Interface table

eth0 Link encap:Ethernet HWaddr 00:10:40:11:11:11

UP BROADCAST MULTICAST MTU:1500 Metric:1

RX packets:0 errors:0 dropped:0 overruns:0 frame:0

TX packets:0 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

Memory:f6ae0000-f6b00000

11. IP和TCP分析

查看连接某服务端口最多的的IP地址

wss8848@ubuntu:~$ netstat -nat | grep "192.168.1.15:22" |awk '{print $5}'|awk -F: '{print $1}'|sort|uniq -c|sort -nr|head -20

18 221.136.168.36

3 154.74.45.242

2 78.173.31.236

2 62.183.207.98

2 192.168.1.14

2 182.48.111.215

2 124.193.219.34

2 119.145.41.2

2 114.255.41.30

1 75.102.11.99

TCP各种状态列表

wss8848@ubuntu:~$ netstat -nat |awk '{print $6}'

established)

Foreign

LISTEN

TIME\_WAIT

ESTABLISHED

TIME\_WAIT

SYN\_SENT

先把状态全都取出来,然后使用uniq -c统计，之后再进行排序。

wss8848@ubuntu:~$ netstat -nat |awk '{print $6}'|sort|uniq -c

143 ESTABLISHED

1 FIN\_WAIT1

1 Foreign

1 LAST\_ACK

36 LISTEN

6 SYN\_SENT

113 TIME\_WAIT

1 established)

最后的命令如下:

netstat -nat |awk '{print $6}'|sort|uniq -c|sort -rn