原文地址：http://blog.csdn.net/zjl\_1026\_2001/article/details/2294044

CPU整体性能监控工具  
Linux中常用的监控CPU整体性能的工具有：  
 mpstat： mpstat 不但能查看所有CPU的平均信息，还能查看指定CPU的信息。♣  
 vmstat：只能查看所有CPU的平均信息；查看cpu队列信息；♣  
 iostat: 只能查看所有CPU的平均信息。♣  
 sar： 与mpstat 一样，不但能查看CPU的平均信息，还能查看指定CPU的信息。♣  
 top：显示的信息同ps接近，但是top可以了解到CPU消耗，可以根据用户指定的时间来更新显示。♣  
 oprofile：OProfile 是一个低开销的系统全局的性能监视工具。它使用处理器上的性能监视硬件来检索关于内核以及系统上的可执行文件的信息，例如内存是何时被引用的；L2 缓存请求数量；收到的硬件中断数量等♣  
  
3.2.1. /proc/stat/  
包含了所有CPU活动的信息，该文件中的所有值都是从系统启动开始累计到当前时刻。  
  
[work@builder ~]$ cat /proc/stat  
cpu 432661 13295 86656 422145968 171474 233 5346  
cpu0 123075 2462 23494 105543694 16586 0 4615  
cpu1 111917 4124 23858 105503820 69697 123 371  
cpu2 103164 3554 21530 105521167 64032 106 334  
cpu3 94504 3153 17772 105577285 21158 4 24  
intr 1065711094 1057275779 92 0 6 6 0 4 0 3527 0 0 0 70 0 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7376958 0 0 0 0 0 0 0 1054602 0 0 0 0 0 0 0 30 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
ctxt 19067887  
btime 1139187531  
processes 270014  
procs\_running 1  
procs\_blocked 0  
  
  
输出解释  
CPU 以及CPU0、CPU1、CPU2、CPU3每行的每个参数意思（以第一行为例）为：  
参数 解释  
user (432661) 从系统启动开始累计到当前时刻，用户态的CPU时间（单位：jiffies） ，不包含 nice值为负进程。1jiffies=0.01秒  
nice (13295) 从系统启动开始累计到当前时刻，nice值为负的进程所占用的CPU时间（单位：jiffies）   
system (86656) 从系统启动开始累计到当前时刻，核心时间（单位：jiffies）   
idle (422145968) 从系统启动开始累计到当前时刻，除硬盘IO等待时间以外其它等待时间（单位：jiffies）   
iowait (171474) 从系统启动开始累计到当前时刻，硬盘IO等待时间（单位：jiffies） ，  
irq (233) 从系统启动开始累计到当前时刻，硬中断时间（单位：jiffies）   
softirq (5346) 从系统启动开始累计到当前时刻，软中断时间（单位：jiffies）   
  
CPU时间=user+system+nice+idle+iowait+irq+softirq  
  
“intr”这行给出中断的信息，第一个为自系统启动以来，发生的所有的中断的次数；然后每个数对应一个特定的中断自系统启动以来所发生的次数。  
“ctxt”给出了自系统启动以来CPU发生的上下文交换的次数。  
“btime”给出了从系统启动到现在为止的时间，单位为秒。  
“processes (total\_forks) 自系统启动以来所创建的任务的个数目。  
“procs\_running”：当前运行队列的任务的数目。  
“procs\_blocked”：当前被阻塞的任务的数目。  
  
3.2.2. /proc/loadavg  
该文件中的所有值都是从系统启动开始累计到当前时刻。该文件只给出了所有CPU的集合信息，不能该出每个CPU的信息。  
  
[root@localhost ~]# cat /proc/loadavg   
4.61 4.36 4.15 9/84 5662  
  
每个值的含义为：  
参数 解释  
lavg\_1 (4.61) 1-分钟平均负载  
lavg\_5 (4.36) 5-分钟平均负载  
lavg\_15(4.15) 15-分钟平均负载  
nr\_running (9) 在采样时刻，运行队列的任务的数目，与/proc/stat的procs\_running表示相同意思  
nr\_threads (84) 在采样时刻，系统中活跃的任务的个数（不包括运行已经结束的任务）  
last\_pid(5662) 最大的pid值，包括轻量级进程，即线程。  
  
假设当前有两个CPU，则每个CPU的当前任务数为4.61/2=2.31  
  
3.2.3. uptime  
  
uptime是Linux系统常用的命令，用来报告系统已经运行多长时间，依此显示的信息：现在时间，系统已经运行了的时间，目前有多少登陆用户， 1分钟系统平均负载，5分钟系统平均负载，15分钟系统平均负载。该命令从/proc/loadavg 中获得load average的信息。  
范例1：系统只用一个CPU  
  
[root@localhost ~]# uptime  
12:20:49 up 3 days,9:20, 5 users, load average 1.10 1.32 1.15   
  
对于一个CPU的系统来说，范例1中的平均负载高了些。通常来说：如果系统有n个CPU而且平均负载小于n,则说明某些CPU还有空闲的时间片。通过该命令，你能知道CPU是否繁忙，但是无法知道为什么忙。  
  
3.2.4. mpstat  
  
mpstat是Multiprocessor Statistics的缩写，是实时系统监控工具。其报告与CPU的一些统计信息，这些信息存放在/proc/stat文件中。在多CPUs系统里，其不但能查看所有CPU的平均状况信息，而且能够查看特定CPU的信息。下面只介绍 mpstat与CPU相关的参数，mpstat的语法如下：   
mpstat [-P {|ALL}] [internal [count]]   
  
参数的含义如下：  
  
参数 解释  
-P {|ALL} 表示监控哪个CPU， cpu在[0,cpu个数-1]中取值  
internal 相邻的两次采样的间隔时间  
count 采样的次数，count只能和delay一起使用  
  
当没有参数时，mpstat则显示系统启动以后所有信息的平均值。有interval时，第一行的信息自系统启动以来的平均信息。从第二行开始，输出为前一个interval时间段的平均信息。与CPU有关的输出的含义如下：  
  
参数 解释 从/proc/stat获得数据  
CPU 处理器ID   
total\*100δusr/δuser 在internal时间段里，用户态的CPU时间（%） ，不包含 nice值为负 进程   
total\*100δnice/δnice 在internal时间段里，nice值为负进程的CPU时间（%）   
total\*100δsystem/δsystem 在internal时间段里，核心时间（%）   
total\*100δiowait/δiowait 在internal时间段里，硬盘IO等待时间（%）   
total\*100δirq/δirq 在internal时间段里，软中断时间（%）   
total\*100δsoftirq/δsoft 在internal时间段里，软中断时间（%）   
total\*100δidle/δidle 在internal时间段里，CPU除去等待磁盘IO操作外的因为任何原因而空闲的时间闲置时间 （%）   
total\*100δintr/δintr/s 在internal时间段里，每秒CPU接收的中断的次数   
  
CPU总的工作时间=total\_cur=user+system+nice+idle+iowait+irq+softirq  
total\_pre=pre\_user+ pre\_system+ pre\_nice+ pre\_idle+ pre\_iowait+ pre\_irq+ pre\_softirq  
user=user\_cur – user\_preδ  
total=total\_cur-total\_preδ  
其中\_cur 表示当前值，\_pre表示interval时间前的值。上表中的所有值可取到两位小数点。  
  
范例1：average mode (粗略信息)  
当mpstat不带参数时，输出为从系统启动以来的平均值。  
  
[work@builder linux-2.6.14]$ mpstat   
Linux 2.6.9-5.31AXsmp (builder.redflag-linux.com) 12/16/2005  
  
09:38:46 AM CPU %user %nice %system %iowait %irq %soft %idle intr/s  
09:38:48 AM all 23.28 0.00 1.75 0.50 0.00 0.00 74.47 1018.59  
  
  
范例2: 每2秒产生了4个处理器的统计数据报告  
下面的命令可以每2秒产生了4个处理器的统计数据报告，一共产生三个interval 的信息，然后再给出这三个interval的平均信息。默认时，输出是按照CPU 号排序。第一个行给出了从系统引导以来的所有活跃数据。接下来每行对应一个处理器的活跃状态。。  
  
  
[work@builder linux-2.6.14]$ mpstat -P ALL 2 3  
Linux 2.6.9-5.31AXsmp (builder.redflag-linux.com) 12/16/2005  
  
09:38:46 AM CPU %user %nice %system %iowait %irq %soft %idle intr/s  
09:38:48 AM all 23.28 0.00 1.75 0.50 0.00 0.00 74.47 1018.59  
09:38:48 AM 0 2.01 0.00 1.01 0.50 0.00 0.00 96.48 5.03  
09:38:48 AM 1 51.26 0.00 3.02 1.01 0.00 0.00 44.72 7.04  
09:38:48 AM 2 17.09 0.00 2.01 0.50 0.00 0.00 81.41 0.00  
09:38:48 AM 3 22.61 0.00 1.01 0.00 0.00 0.00 76.38 1006.03  
  
09:38:48 AM CPU %user %nice %system %iowait %irq %soft %idle intr/s  
09:38:50 AM all 24.22 0.00 1.25 1.25 0.00 0.00 73.28 1049.75  
09:38:50 AM 0 1.00 0.00 1.00 0.00 0.00 0.00 98.01 28.86  
09:38:50 AM 1 93.03 0.00 3.48 0.00 0.00 0.00 2.99 25.37  
09:38:50 AM 2 1.99 0.00 0.50 4.98 0.00 0.00 92.54 0.00  
09:38:50 AM 3 1.00 0.00 0.50 0.00 0.00 0.00 98.51 995.52  
  
09:38:50 AM CPU %user %nice %system %iowait %irq %soft %idle intr/s  
09:38:52 AM all 23.69 0.00 2.37 0.37 0.00 0.12 73.44 1028.36  
09:38:52 AM 0 2.99 0.00 1.49 0.00 0.00 0.50 95.02 111.94  
09:38:52 AM 1 52.74 0.00 2.99 1.00 0.00 0.00 42.29 4.48  
09:38:52 AM 2 14.93 0.00 1.99 0.00 0.00 0.00 82.09 4.48  
09:38:52 AM 3 23.38 0.00 1.99 0.50 0.00 0.00 73.13 907.46  
  
Average: CPU %user %nice %system %iowait %irq %soft %idle intr/s  
Average: all 23.73 0.00 1.79 0.71 0.00 0.04 73.73 1032.28  
Average: 0 2.00 0.00 1.16 0.17 0.00 0.17 96.51 48.75  
Average: 1 65.72 0.00 3.16 0.67 0.00 0.00 29.95 12.31  
Average: 2 11.31 0.00 1.50 1.83 0.00 0.00 85.36 1.50  
Average: 3 15.64 0.00 1.16 0.17 0.00 0.00 82.70 969.55  
[work@builder linux-2.6.14]$   
  
  
范例3：比较带参数和不带参数的mpstat的结果。  
在后台开一个2G的文件  
> cat 1.img —/\* 1.img=2GBytes  
  
然后在另一个终端运行mpstat命令  
  
[root@localhost ~]# mpstat  
Linux 2.6.13 (localhost.localdomain) 2005年12月15日  
  
13时48分52秒 CPU %user %nice %system %iowait %irq %soft %idle intr/s  
13时48分52秒 all 1.57 0.00 0.24 0.15 0.12 0.01 97.92 308.12  
[root@localhost ~]# mpstat  
Linux 2.6.13 (localhost.localdomain) 2005年12月15日  
  
13时48分53秒 CPU %user %nice %system %iowait %irq %soft %idle intr/s  
13时48分53秒 all 1.57 0.00 0.24 0.15 0.12 0.01 97.92 308.12  
[root@localhost ~]# mpstat  
Linux 2.6.13 (localhost.localdomain) 2005年12月15日  
  
13时48分53秒 CPU %user %nice %system %iowait %irq %soft %idle intr/s  
13时48分53秒 all 1.57 0.00 0.24 0.15 0.12 0.01 97.92 308.12  
[root@localhost ~]# mpstat  
Linux 2.6.13 (localhost.localdomain) 2005年12月15日  
  
13时48分54秒 CPU %user %nice %system %iowait %irq %soft %idle intr/s  
13时48分54秒 all 1.57 0.00 0.24 0.15 0.12 0.01 97.92 308.12  
[root@localhost ~]#  
  
然后再运行命令：mpstat 3 100 /\*每隔3秒给出一次信息，共给出100次\*/  
[root@localhost ~]# mpstat 3 100  
Linux 2.6.13 (localhost.localdomain) 2005年12月15日  
  
13时48分25秒 CPU %user %nice %system %iowait %irq %soft %idle intr/s  
13时48分28秒 all 96.00 0.00 4.00 0.00 0.00 0.00 0.00 401.67  
13时48分31秒 all 96.01 0.00 3.65 0.00 0.33 0.00 0.00 393.36  
13时48分34秒 all 95.36 0.00 4.30 0.00 0.33 0.00 0.00 532.45  
13时48分37秒 all 94.00 0.00 5.67 0.00 0.33 0.00 0.00 400.33  
13时48分40秒 all 95.00 0.00 5.00 0.00 0.00 0.00 0.00 330.33  
13时48分43秒 all 95.33 0.00 4.33 0.00 0.33 0.00 0.00 337.33  
13时48分46秒 all 94.68 0.00 4.98 0.00 0.33 0.00 0.00 329.57  
13时48分49秒 all 95.67 0.00 4.00 0.00 0.33 0.00 0.00 417.33  
13时48分52秒 all 95.67 0.00 3.67 0.00 0.67 0.00 0.00 448.00  
13时48分55秒 all 94.02 0.00 5.32 0.00 0.33 0.33 0.00 410.96  
13时48分58秒 all 95.35 0.00 4.32 0.00 0.33 0.00 0.00 466.45  
  
[root@localhost ~]#  
  
  
上两表显示出当要正确反映系统的情况，需要正确使用命令的参数。vmstat 和iostat 也需要注意这一问题。  
  
3.2.5. vmstat  
  
vmstat是Virtual Meomory Statistics（虚拟内存统计）的缩写, 是实时系统监控工具。该命令通过使用knlist子程序和/dev/kmen伪设备驱动器访问这些数据，输出信息直接打印在屏幕。vmstat反馈的与CPU相关的信息包括：  
  
（1）多少任务在运行  
（2）CPU使用的情况  
（3）CPU收到多少中断  
（4）发生多少上下文切换  
  
下面只介绍 Vmstat与CPU相关的参数  
vmstat的语法如下：   
　　vmstat [delay [count]]   
  
参数的含义如下：  
参数 解释  
delay 相邻的两次采样的间隔时间  
count 采样的次数，count只能和delay一起使用  
  
当没有参数时，vmstat则显示系统启动以后所有信息的平均值。有delay时，第一行的信息自系统启动以来的平均信息。从第二行开始，输出为前一个delay时间段的平均信息。当系统有多个CPU时，输出为所有CPU的平均值。  
  
与CPU有关的输出的含义 (采用进一法)  
参数 解释 从/proc/stat获得数据  
任务的信息   
r 在internal时间段里，运行队列里等待CPU的任务（任务）的个数，即不包含vmstat进程 procs\_running-1  
b 在internal时间段里，被资源阻塞的任务数（I/0，页面调度，等等.） ，通常情况下是接近0的 procs\_blocked  
CPU信息 所有值取整（四舍五入）   
total\*100δnice)/δuser+δus 在internal时间段里，用户态的CPU时间(%)，包含 nice值为负进程 (  
total\*100δsoftirq)/δirq+δsystem+δsy 在internal时间段里，核心态的CPU时间(%) (  
total\*100δidle/δid 在internal时间段里，cpu空闲的时间，不包括等待i/o的时间(%)   
total\*100δiowait/δwa 在internal时间段里，等待i/o的时间(%)   
系统信息   
intr/intervalδin 在internal时间段里，每秒发生中断的次数   
ctxt/intervalδcs 在internal时间段里，每秒上下文切换的次数，即每秒内核任务交换的次数   
  
total\_cur=user+system+nice+idle+iowait+irq+softirq  
total\_pre=pre\_user+ pre\_system+ pre\_nice+ pre\_idle+ pre\_iowait+ pre\_irq+ pre\_softirq  
total=total\_cur-total\_preδ  
  
  
范例1：average mode (粗略信息)  
当vmstat不带参数时，对应的输出值是从系统启动以来的平均值，而r和b则对应的是完成这一命令时，系统的值。从下面例子，可以看出系统基本出去闲置状态（idle）。自启动以来，CPU在用户态消耗时间为5%，在核心态消耗为本1%，剩下的为闲置时间。需要指出的是：这里的用户态时间包括nice值为负的进程的时间。  
  
[root@localhost ~]# vmstat  
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 4580 428 98516 0 0 49 6 15 19 2 1 96 1  
[root@localhost ~]#  
  
  
范例2：average mode (详细信息)  
命令格式：  
vmstat –s  
这里只讨论与CPU相关信息。“CPU ticks”表示自系统启动CPU运行时间，这里以tick为时间单位。用tick来西安市us,sy id 和wa的时间；forks指自从系统启动以来，所创建的新任务的个数。这些信息从/proc/stat 的第一行和”processes”行获得。  
  
[root@localhost ~]# vmstat -s  
255280 total memory  
244216 used memory  
206624 active memory  
21208 inactive memory  
11064 free memory  
628 buffer memory  
91396 swap cache  
255992 total swap  
24 used swap  
255968 free swap  
973400 non-nice user cpu ticks  
477 nice user cpu ticks  
206168 system cpu ticks  
43567714 idle cpu ticks  
373234 IO-wait cpu ticks  
62732 IRQ cpu ticks  
1972 softirq cpu ticks  
22366502 pages paged in  
88756936 pages paged out  
0 pages swapped in  
0 pages swapped out  
135634319 interrupts  
137288441 CPU context switches  
1134440368 boot time  
208990 forks  
[root@localhost ~]#  
  
结果解释  
  
参数 描述 /proc/stat  
non-nice user cpu ticks 自系统启动以来，CPU在用户态下运行非nice进程的时间，单位为jiffies user  
nice user cpu ticks 自系统启动以来，CPU在用户态下运行nice进程的时间，单位为jiffies nice  
system cpu ticks 自系统启动以来，CPU处于系统状态的时间，单位为jiffies sys  
idle cpu ticks 自系统启动以来，CPU处于闲置状态的时间，单位为jiffies idle  
IO-wait cpu ticks 自系统启动以来，CPU处理IO中断的时间，单位为jiffies iowait  
IRQ cpu ticks 自系统启动以来，CPU处理硬中断的时间，单位为jiffies irq  
softing cpu ticks 自系统启动以来，CPU处理软中断的时间，单位为jiffies Softirq  
interrupts 自系统启动以来，发生的所有的中断的次数目 Intr  
CPU context switches 自系统启动以来，发生的上下文交换的次数 Ctxt  
boot time 自系统启动以来到现在运行的时间，单位为秒。 btime  
forks 自系统启动以来所创建的任务的个数目。 Process  
  
  
范例3：定期采样(delay [count])  
定期采样数据是指每隔delay时间，采样一次。当count 为0时，vmstat 将不停地定期报告信息；否则当报告count次后，vmstat 命令停止运行。  
第一行的信息如同范例1，是自系统启动以来的平均信息。从第二行开始，每行的意思是：r和b采样那一时刻系统运行队列和等待队列的情况；而usystem参数（in,cs）以及CPU参数（us,sy,id,wa）对应的输出值是系统在前一个delay的情况。  
从下面例子可以看出上下文交换的次数小于中断的发生次数。当系统大部分时间是空闲并且中断大部分是时间中断时，这种现象极可能发生。当时间中断发生时, 因为调度器没有什么任务可调度，所以很少发生上下文切换。  
  
[root@localhost ~]# vmstat 2 4  
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----  
r b swpd free buff cache si so bi bo in cs us sy id wa  
1 0 24 11032 652 91396 0 0 49 6 15 19 2 1 96 1  
0 0 24 11032 652 91396 0 0 0 0 377 464 1 0 99 0  
0 0 24 11024 652 91396 0 0 0 0 387 476 1 0 100 0  
0 0 24 11024 652 91396 0 0 0 0 323 377 0 0 100 0  
[root@localhost ~]#  
  
  
3.2.6. iostat  
  
iostat [-t] [-c] [interval [count]]   
  
参数的含义如下：  
参数 解释  
T 表示输出采用的时间  
C 表示只显示CPU的信息  
Internal 相邻的两次采样的间隔时间  
count 采样的次数，count只能和delay一起使用  
  
当没有参数时，iostat则显示系统启动以后所有信息的平均值。  
  
与CPU有关的输出的含义  
  
参数 解释 从/proc/stat获得  
CPU 处理器ID   
total\*100δusr/δuser 在internal时间段里，用户态的CPU时间（%） ，不包含 nice值为负进程   
total\*100δnice/δnice 在internal时间段里，nice值为负进程的CPU时间（%）   
total\*100δsoftirq)/δirq+δsystem+δsys 在internal时间段里，核心时间（%） (  
total\*100δiowait/δiowait 在internal时间段里，硬盘IO等待时间（%）   
total\*100δidle/δidle 在internal时间段里，CPU除去等待磁盘IO操作外的因为任何原因而空闲的时间闲置时间 （%）   
  
total\_cur=user+system+nice+idle+iowait+irq+softirq  
total\_pre=pre\_user+ pre\_system+ pre\_nice+ pre\_idle+ pre\_iowait+ pre\_irq+ pre\_softirq  
total=total\_cur-total\_preδ  
  
有interval时，第一行的信息自系统启动以来的平均信息。从第二行开始，输出为前一个interval时间段的平均信息。  
  
范例1：  
  
  
[root@localhost /]# dd if=/dev/zero of=/1.img bs=1024 count=1000000&  
[1] 15854  
[root@localhost /]# iostat -c 1  
Linux 2.6.13 (localhost.localdomain) 2005年12月19日  
  
avg-cpu: %user %nice %sys %iowait %idle  
1.94 0.00 0.56 0.75 96.75  
  
avg-cpu: %user %nice %sys %iowait %idle  
3.65 0.00 13.87 82.48 0.00  
  
avg-cpu: %user %nice %sys %iowait %idle  
  
  
3.2.7. sar  
sar是System Activity Reporter（系统活跃情况报告）的缩写。顾名思义，sar工具将对系统当前的状态进行采样，然后通过计算数据和比例来表达系统的当前运行状态。它的特点是可以连续对系统采样，获得大量的采样数据；采样数据和分析的结果都可以存入文件，所需的负载很小。这些是检查历史数据和一些近来的系统事件。sar用于检查的性能数据类似于vmstat, mpstat和 iostat的显示。 sar的数据是一段时间保存的内容，因此可以察看过去的信息。 lastcomm可以现在系统最近被执行的命令。这些可以用在系统审计中。sa 可以在\*BSD和Linux中找到，它给用户在系统审计中更多的选项来收集信息。  
在反馈CPU整体信息方面，sar 反馈的与CPU相关的信息包括：  
  
（1）多少任务在运行  
（2）CPU使用的情况  
（3）CPU收到多少中断  
（4）发生多少上下文切换  
  
sar的语法如下：   
  
sar [-options] [interval [count]]  
  
其中，internal是两次采样的间隔时间；count是指采样的次数；与CPU相关的options有：  
参数的含义如下：  
参数 解释  
-c 表示输出采用的时间  
-e hh:mm:ss 表示只显示CPU的信息  
-i {irq |SUM|ALL|XALL} 相邻的两次采样的间隔时间  
-P {cpu|ALL}   
-q 显示在采样的时刻，可运行队列的任务的个数，以及系统平均负载  
-u CPU 使用的情况，报告了cpu的用户态，系统态，等待I/O和空闲时间上的百分比。   
-w：每秒上下文交换率  
-o：filename 将结果放在文件里  
-f：filename 表示从file文件中取出数据，如果没有指定-f file，则从标准数据文件  
  
范例1：  
首先在后台运行一个创建1GB文件的任务，然后输入：“sar -c -q -w 3 2”，表示每3秒采样一次，采样两次，可以看到系统有82个任务，但是在10时12分08秒时有一个任务在运行，在下一次采样10时12分11秒时，没有任务在运行。  
  
[root@localhost /]# dd if=/dev/zero of=/1.img bs=1024 count=1000000&  
[1] 16213  
[root@localhost /]# sar -c -q -w -I SUM 3 2  
Linux 2.6.13 (localhost.localdomain) 2005年12月19日  
  
10时12分05秒 proc/s  
10时12分08秒 0.00  
  
10时12分05秒 cswch/s  
10时12分08秒 692.33  
  
10时12分05秒 INTR intr/s  
10时12分08秒 sum 895.33  
  
10时12分05秒 runq-sz plist-sz ldavg-1 ldavg-5 ldavg-15  
10时12分08秒 1 82 0.20 0.08 0.09  
  
10时12分08秒 proc/s  
10时12分11秒 0.00  
  
10时12分08秒 cswch/s  
10时12分11秒 648.50  
  
  
10时12分08秒 INTR intr/s  
10时12分11秒 sum 461.79  
  
10时12分08秒 runq-sz plist-sz ldavg-1 ldavg-5 ldavg-15  
10时12分11秒 0 82 0.59 0.16 0.12  
  
Average: proc/s  
Average: 0.00  
  
Average: cswch/s  
Average: 670.38  
  
Average: INTR intr/s  
Average: sum 678.20  
  
Average: runq-sz plist-sz ldavg-1 ldavg-5 ldavg-15  
Average: 0 82 0.40 0.12 0.10   
  
  
与CPU有关的输出的含义  
参数 解释 从/proc/stat获得数据  
total\*100δprocesses/δproc/s 在internal时间段里，每秒上下文切换率   
total\*100δctxt/δcswch 在internal时间段里，每秒上下文切换率   
total\*100δidle/δintr/s 在internal时间段里，每秒CPU接收的中断的次数   
从/proc/loadavg获得数据  
runq-sz 采样时，运行队列中任务的个数，不包括vmstat 进程。 procs\_running-1  
plist-sz 采样时，系统中活跃的任务的个数 nr\_threads  
ldavg-1 采样的前一秒钟系统的负载(%) lavg\_1  
ldavg-5 采样的5秒钟系统的负载(%) lavg\_5  
ldavg-15 采样的前15秒钟系统的负载(%) lavg\_15  
  
下面输入命令“sar 3 2”,表示每隔3秒显示CPU的状态，共显示2次。此时创建1GB文件的任务还在后台进行。  
  
[root@localhost /]# sar 3 2  
Linux 2.6.13 (localhost.localdomain) 2005年12月19日  
10时12分37秒 CPU %user %nice %system %iowait %idle  
10时12分40秒 all 2.91 0.00 14.18 82.91 0.00  
10时12分43秒 all 2.66 0.00 14.95 82.39 0.00  
Average: all 2.78 0.00 14.58 82.64 0.00  
[root@localhost /]#  
  
  
与CPU有关的输出的含义  
参数 解释 从/proc/stat获得数据  
CPU 处理器ID   
total\*100δusr/δuser 在internal时间段里，用户态的CPU时间（%） ，不包含 nice值为负进程   
total\*100δnice/δnice 在internal时间段里，nice值为负进程的CPU时间（%）   
total\*100δsoftirq)/δirq+δsystem+δsys 在internal时间段里，核心时间（%） (  
total\*100δiowait/δiowait 在internal时间段里，硬盘IO等待时间（%）   
total\*100δirq/δidle 在internal时间段里，CPU除去等待磁盘IO操作外的因为任何原因而空闲的时间闲置时间 （%）   
  
输入命令“sar”,当不带任何参数时，表示每一秒采样一次，每次显示从系统到该采样时系统的平均信息。从Fig.8看出在10时12分43秒，系统idle应该为0,而Fig.9  
  
[root@localhost /]# sar  
Linux 2.6.13 (localhost.localdomain) 2005年12月19日  
  
00时00分01秒 CPU %user %nice %system %iowait %idle  
00时10分01秒 all 0.11 0.00 0.16 0.00 99.73  
00时20分01秒 all 0.13 0.00 0.15 0.00 99.72  
.  
.  
.  
08时10分01秒 all 0.11 0.00 0.14 0.00 99.75  
08时20分01秒 all 0.13 0.00 0.15 0.00 99.72  
08时30分01秒 all 0.13 0.00 0.16 0.00 99.71  
08时40分01秒 all 3.96 0.00 0.57 0.34 95.12  
08时50分01秒 all 7.42 0.00 0.77 0.74 91.07  
09时00分01秒 all 4.93 0.00 1.97 8.14 84.95  
09时10分01秒 all 4.81 0.00 0.69 0.29 94.21  
09时20分01秒 all 3.09 0.00 0.93 1.72 94.25  
09时30分01秒 all 7.59 0.00 1.59 1.27 89.55  
09时40分01秒 all 2.27 0.00 0.74 0.32 96.67  
09时50分01秒 all 4.81 0.00 0.81 0.07 94.31  
10时00分02秒 all 1.84 0.00 2.17 7.78 88.21  
10时10分01秒 all 0.63 0.00 0.24 0.20 98.93  
10时20分02秒 all 1.70 0.00 1.85 7.95 88.50  
10时30分01秒 all 0.50 0.00 0.21 0.21 99.08  
Average: all 0.81 0.00 0.33 0.49 98.37  
[root@localhost /]#  
  
total\_cur=user+system+nice+idle+iowait+irq+softirq  
total\_pre=pre\_user+ pre\_system+ pre\_nice+ pre\_idle+ pre\_iowait+ pre\_irq+ pre\_softirq  
total=total\_cur-total\_preδ  
上表中的所有值可取到两位小数点。  
  
范例2：  
  
  
[work@builder linux-2.6.14]$ sar –P ALL 2 3  
Linux 2.6.9-5.31AXsmp (builder.redflag-linux.com) 12/28/2005  
  
08:28:45 AM CPU %user %nice %system %iowait %idle  
08:28:47 AM all 0.00 0.00 0.00 0.00 100.00  
08:28:47 AM 0 0.00 0.00 0.00 0.00 100.00  
08:28:47 AM 1 0.00 0.00 0.00 0.00 100.00  
08:28:47 AM 2 0.00 0.00 0.00 0.00 100.00  
08:28:47 AM 3 0.00 0.00 0.00 0.00 100.00  
  
08:28:47 AM CPU %user %nice %system %iowait %idle  
08:28:49 AM all 0.00 0.00 0.00 0.00 100.00  
08:28:49 AM 0 0.00 0.00 0.00 0.00 100.00  
08:28:49 AM 1 0.00 0.00 0.00 0.00 100.00  
08:28:49 AM 2 0.00 0.00 0.00 0.00 100.00  
08:28:49 AM 3 0.00 0.00 0.00 0.00 100.00  
  
08:28:49 AM CPU %user %nice %system %iowait %idle  
08:28:51 AM all 0.00 0.00 0.12 0.25 99.63  
08:28:51 AM 0 0.00 0.00 0.00 0.00 100.00  
08:28:51 AM 1 0.00 0.00 0.00 0.50 99.50  
08:28:51 AM 2 0.00 0.00 0.00 0.00 100.00  
08:28:51 AM 3 0.00 0.00 0.00 0.00 100.00  
  
Average: CPU %user %nice %system %iowait %idle  
Average: all 0.00 0.00 0.04 0.08 99.88  
Average: 0 0.00 0.00 0.00 0.00 100.00  
Average: 1 0.00 0.00 0.00 0.17 99.83  
Average: 2 0.00 0.00 0.00 0.00 100.00  
Average: 3 0.00 0.00 0.00 0.00 100.00  
Fig.1.   
  
  
3.2.8. top  
top命令提供了实时的对系统处理器的状态监视，即可以通过用户按键来不断刷新当前状态。如果前台执行该命令，它将独占前台，直到用户终止该程序为止，但是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。  
下面是该命令的语法格式：  
top [-] [d delay] [options]  
  
一些参数的含义如下：  
参数 解释  
d delay 相邻的两次采样的间隔时间  
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  
-l 不显示空闲进程，默认时，top显示空闲进程  
-S 将系统进程信息也显示于屏幕，默认时，不显示。  
  
  
与CPU有关的输出的含义  
参数 解释  
CPU信息 从/proc/stat获得信息  
us 在internal时间段里，用户态的CPU时间(%)，不包含nice值为负的任务占用的CPU的时间   
total\*100δusr/δ  
total\*100δnice/δni 在internal时间段里，nice值为负的任务的用户态的Niced进程的CPU时间(%)   
total\*100δsystem/δsy 在internal时间段里，核心态的CPU时间(%)   
total\*100δidle/δid 在internal时间段里，cpu空闲的时间，不包括等待i/o的时间(%)   
total\*100δiowait/δwa 在internal时间段里，等待i/o的时间(%)   
total\*100δirq/δhi 在internal时间段里，硬中断时间(%)   
total\*100δsoftirq/δsi 在internal时间段里，软中断时间(%)   
任务   
total 采样时，系统全部进程的个数，包括正在运行的top进程，不包括线程。 =running+sleeping+stopped+zombie  
running 采样时，运行队列的进程的个数，但不包含top这个进程 通过判断/proc/pid/status中stat的值。  
sleeping 采样时，可中断和不可中断状态的进程的个数   
stopped 采样时，停止状态的进程的个数   
zombie 采样时，僵尸状态的进程的个数   
  
  
运行中的参数的含义  
参数 解释  
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  
S 切换到累计模式  
L 切换显示平均负载和启动时间信息  
T 切换显示进程和CPU状态信息  
M 切换显示内存信息  
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  
  
　　  
范例1：查看 键入top命令查看系统状况  
  
>top  
top - 16:52:17 up 3 min, 1 user, load average: 0.10, 0.27, 0.14  
Tasks: 55 total, 1 running, 54 sleeping, 0 stopped, 0 zombie  
top - 16:52:31 up 4 min, 1 user, load average: 0.08, 0.26, 0.13  
Tasks: 55 total, 1 running, 54 sleeping, 0 stopped, 0 zombie  
Cpu(s): 0.3% us, 0.3% sy, 0.0% ni, 99.0% id, 0.0% wa, 0.3% hi, 0.0% si  
Mem: 251924k total, 238168k used, 13756k free, 4420k buffers  
Swap: 0k total, 0k used, 0k free, 165200k cached  
  
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND  
4171 root 15 0 48404 12m 2220 S 0.7 5.0 0:01.36 X  
4404 root 15 0 66540 16m 13m S 0.3 6.9 0:02.31 kdeinit  
1 root 16 0 1772 544 464 S 0.0 0.2 0:00.61 init  
2 root 34 19 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/0  
3 root 5 -10 0 0 0 S 0.0 0.0 0:00.00 events/0  
4 root 5 -10 0 0 0 S 0.0 0.0 0:00.02 khelper  
5 root 15 -10 0 0 0 S 0.0 0.0 0:00.00 kacpid  
19 root 5 -10 0 0 0 S 0.0 0.0 0:00.00 kblockd/0  
29 root 20 0 0 0 0 S 0.0 0.0 0:00.00 pdflush  
30 root 15 0 0 0 0 S 0.0 0.0 0:00.01 pdflush  
32 root 14 -10 0 0 0 S 0.0 0.0 0:00.00 aio/0  
20 root 15 0 0 0 0 S 0.0 0.0 0:00.00 khubd  
31 root 15 0 0 0 0 S 0.0 0.0 0:00.01 kswapd0  
618 root 25 0 0 0 0 S 0.0 0.0 0:00.00 kseriod  
  
  
第一行的项目依次为当前时间、系统启动时间、当前系统登录用户数目、平均负载。  
第二行为进程情况，依次为进程总数、休眠进程数、运行进程数、僵死进程数、终止进程数。  
第三行为CPU状态，依次为用户占用、系统占用、优先任务占用、闲置任务占用。  
第四行为内存状态，依次为平均可用内存、已用内存、空闲内存、共享内存、缓存使用内存。  
第五行为交换状态，依次为平均可用交换容量、已用容量、闲置容量、交换高速缓存容量。  
然后下面就是和ps相仿的各进程情况列表了。  
总的来说，top命令的功能强于ps，但需要长久占用前台，所以用户应该根据自己的情况来使用这个命令。   
  
3.2.9. oprofile  
  
Oprofile这个性能监视工具是利用处理器上所包含的专用的性能监视硬件（若没有性能监视硬件则使用一个基于计时器的代用品）来收集与性能相关的数据样品（samples）。它获得关于内核以及系统上的可执行文件的信息，例如内存是何时被引用的；L2缓存请求的数量；收到的硬件中断数量等。  
下表提供了对 oprofile 软件包中包括的工具的总览。   
  
命令 描述  
opcontrol 配置要收集的数据  
op\_help 显示系统处理器的可用事件以及每个事件的简单描述  
opreport 获得有用的profile数据  
Opannotate 如果应用程序使用调试符号编译了，创建带注解  
oprofiled 作为守护进程来运行，定期把样品数据写入磁盘  
Oparchive 收集可执行文件信息和调试信息,将采样文件归档  
op\_import 把样品数据库文件从异类二进制格式转换成系统的本地原始格式。只有在分析不同体系的样品数据库时才使用该选项。  
  
  
（1）opcontrol  
在运行 OProfile 之前，必须用opcontrol配置监视环境。在 opcontrol 命令被执行时，设置选项就会被保存到 /root/.oprofile/daemonrc 文件中。   
  
命令格式  
opcontrol [--vmlinux] [--start] [--stop] [--dump] [--shutdown] [--save=filename]  
参数解释  
参数 描述  
vmlinux 用来配置是否监视内核。要监视内核，以根用户身份执行以下命令：   
  
>opcontrol --vmlinux=/usr/src/linux-2.6.13/vmlinux  
要配置 OProfile 不监视内核，以根用户身份执行以下命令：   
>opcontrol --no-vmlinux  
这个命令还会载入 oprofile 内核模块（如果还没有被载入），并创建 /dev/oprofile/ 目录（如果不存在）。  
start 开始监视系统  
要使用 OProfile 来开始监视系统，以根用户身份执行以下命令：   
>opcontrol --start  
Reading module info.  
Using log file /var/lib/oprofile/oprofiled.log  
Daemon started.  
Profiler running.  
  
/root/.oprofile/daemonrc 中的设置被使用。OProfile 守护进程 oprofiled 被启动；它定期把样品数据写入 /var/lib/oprofile/samples/ 目录。该守护进程的日志位于 /var/lib/oprofile/oprofiled.log。如果 OProfile 使用不同的配置选项被重新启动，以前会话中的样品文件就会被自动备份到 /var/lib/oprofile/samples/session-N 目录中，这里的 N 是前一次备份会话数量再加1。  
  
stop 停止监视  
Shutdown 要停止建档器，以根用户身份执行以下命令：   
>opcontrol –shutdown  
Save save 保存数据  
要保存当前的抽样文件集合，执行以下命令，把 替换成当前会话中的独特描述性名称。 opcontrol --save= 目录 /var/lib/oprofile/samples/name/ 被创建，当前的抽样文件被复制到其中。  
Dump dump OProfile 守护进程 oprofiled 定期收集样品，并把它们写入 /var/lib/oprofile/samples/ 目录。在读取数据之前，请以根用户身份执行以下命令来确定所有数据都被写入这个目录中了