Linux下查看各种性能指标

- 前言：

各种操作系统无一例外都拥有性能指标，就像我们用windows时不时就会查看一下电脑的CPU占用率啊，内存占用率啊之类的，我们用Linux肯定也有想查看性能的时候。但Linux的查看会比windows复杂一点，另一方面却有利于我们理解一些硬件方面的东西。该小节十分需要读者动手上机，接下来我们一边操作一边学习吧。

- 正文：

1、查看硬盘状况：

df大致查看，fdisk狠狠地查看。

```
[root@localhost ~]# df
Filesystem              1K-blocks    Used Available Use% Mounted on
devtmpfs                   495388       0    495388   0% /dev
tmpfs                      507376       0    507376   0% /dev/shm
tmpfs                      507376    6888    500488   2% /run
tmpfs                      507376       0    507376   0% /sys/fs/cgroup
/dev/mapper/centos-root   6486016 1529696   4956320  24% /
/dev/sda1                 1038336  143492    894844  14% /boot
tmpfs                      101476       0    101476   0% /run/user/0
[root@localhost ~]# fdisk -l

Disk /dev/sda: 8589 MB, 8589934592 bytes, 16777216 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000cfea9

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200    16777215     7339008   8e  Linux LVM

Disk /dev/mapper/centos-root: 6652 MB, 6652166144 bytes, 12992512 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/centos-swap: 859 MB, 859832320 bytes, 1679360 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

[root@localhost ~]# _
```

如图，在Linux中，一切都是文件，我们的硬盘也无一例外的用文件来表示，那么哪个文件是我们的硬盘呢？答案是：/dev/sda。我们第一反应肯定是，怎么它的名字那么奇怪，在windows中都是C盘D盘啊，而它叫sda。这是因为，一般的个人计算机硬盘分为IDE老式硬盘和SATA新式硬盘，现在社会一般都是SATA硬盘，所以知道为什么叫sd了吧，另外的a表示这是第一块（最开始创建系统的时候就只是分配了8G的一个盘而已），如果还有很多块硬盘的话，就会被标上sdb、sdc等等姓名。接下来来看fdisk后的DeviceBoot列，有/dev/sda1、/dev/sda2两个分区，当然如果有更多分区肯定按sda3等这样排下去的。另外，读者可能会发现一个问题，最开始用df查看硬盘时，只能看到sda1区的情况，我们的sda2呢？其实/dev/mapper/centos-root就是sda2，只不过换了个名字而已。

```
[root@localhost ~]# df -h
Filesystem               Size  Used Avail Use% Mounted on
devtmpfs                 484M     0  484M   0% /dev
tmpfs                    496M     0  496M   0% /dev/shm
tmpfs                    496M  6.8M  489M   2% /run
tmpfs                    496M     0  496M   0% /sys/fs/cgroup
/dev/mapper/centos-root  6.2G  1.5G  4.8G  24% /
/dev/sda1               1014M  141M  874M  14% /boot
tmpfs                    100M     0  100M   0% /run/user/0
[root@localhost ~]#
```

以后，我们加-h参数，让它人性化显示好看点，现在我们可以看到各个区的大小(size)，使用情况(used)，和可用空间(avail)以及入口(mounted on)。现在，我们可以在对应的分区路径里，dd if=/dev/zero of=swapfile bs=1M count=200新建200MB的文件，然后再查看硬盘的使用量，是不是改变了呢?

2、查看内存状况：

free快速查看内存指标。

```
[root@localhost ~]# free
              total        used        free      shared  buff/cache   available
Mem:        1014756      163784      738452        6880      112520      719808
Swap:        839676           0      839676
[root@localhost ~]# free -h
              total        used        free      shared  buff/cache   available
Mem:           990M        159M        721M        6.7M        109M        703M
Swap:          819M          0B        819M
[root@localhost ~]# _
```

如图，我们先简单介绍一下各列都代表了什么：①total：内存总量；②used：正在使用的内存量；③free：完全空闲的内存量；④shared：共享内存（几乎没用）；⑤buff/cache：缓存缓冲内存；⑥available：真实可用的内存。那么问题来了，free和available看哪个呢？其实是看available，因为available=free+buff/cache，free是完全没用到的内存，而buff/cache是指这样一个东西：

（1）cache：内存的速度远远比不上CPU，为了不让CPU浪费时间等待内存响应，CPU第一次通过内存拿数据时，会看看cache有没有相应的缓存（好比我们上网时早已记录好的账号密码一样），但第一次肯定是空的。于是CPU去从硬盘找出要的数据，放入内存，执行完这一次操作后，把本次访问的数据放到cache里，以备下一次使用。等到下一次执行相同或需要一些相同数据时，CPU直接从cache里读取，这样就能大大提高读速度了。

（2）buff：buff缓冲区应该是我们最常听的名词了，它是一个数据存放区。每当有大量的数据要流入硬盘时，先把零碎的数据放在buff中，然后等buff满了再一次性写入硬盘。这个设置可以大大提高写数据的速度和效率。

接下来，我们实践验证缓存的存在：

先free -m查看buff/cache的数量，然后运行time find，查看执行find命令所用的时间；然后再次free -m查看buff/cache的数量，会发现数量增多了，最后再一次time find可以看到执行时间减少了，这是因为第一次执行find指令后把一些相关数据放在cache里了，下一次调用时直接先看cache有无符合要求的数据，发现有，那么直接调用，所以find所用的时间减少了。

理论上是这样没错的，但我的Linux系统快到只能看到cache增加了，但是执行时间太短看不出区别。总而言之明白这个缓存区的作用即可。

3、查看CPU性能指标：

再此之前，我们需要理解进程这个概念。常用windows的朋友肯定不陌生，说白了进程就是一个个开着的软件啊服务啊之类的。没错，确实是这样的，但是Linux要看到进程还是有点麻烦的，不像windows那么方便。我们直接上机实践实践，先使用ping命令来开启一个发包的进程：

```
[Teemo@localhost ~]$ ping www.baidu.com
PING www.a.shifen.com (183.232.231.174) 56(84) bytes of data.
```

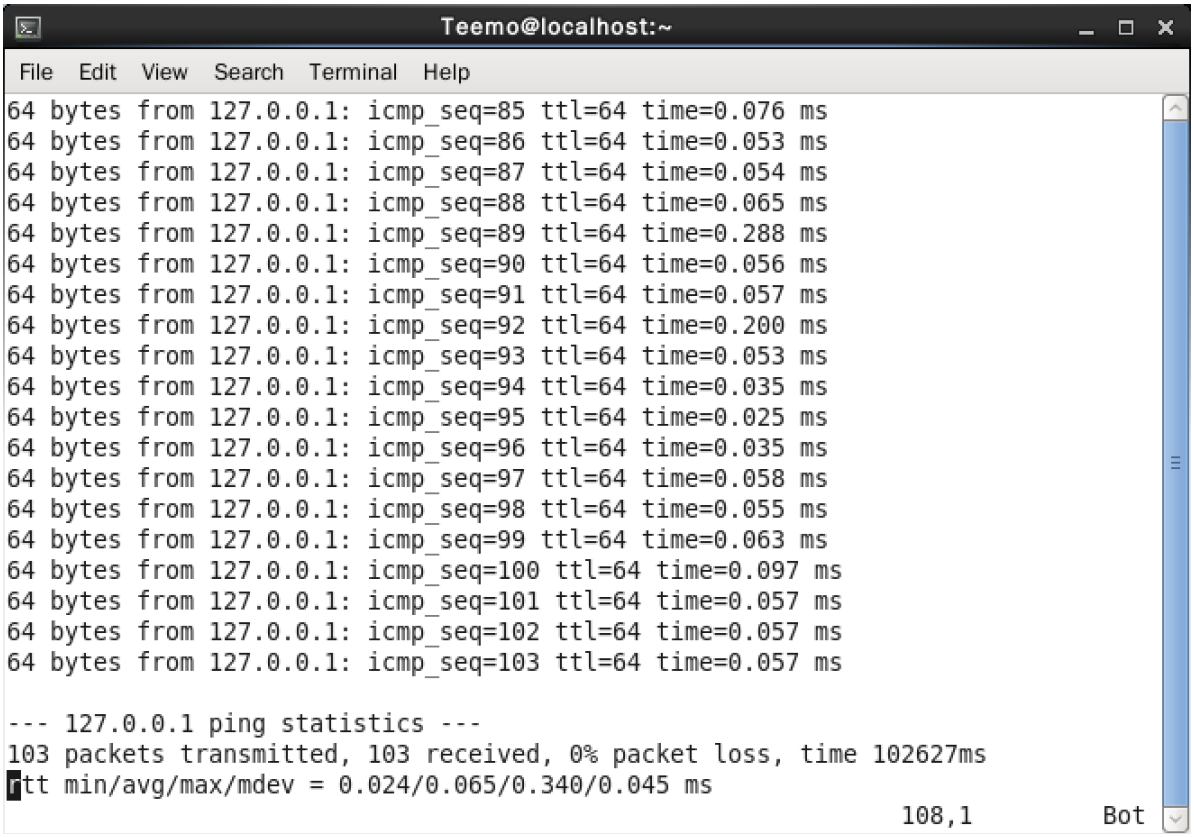
由于我的新Linux系统没有接网，所以我用了我原来的Linux，可以看到在ping了百度后，没接受到数据，系统“卡住了”。其实，ping是一个前台进程，它霸占了命令行终端，所以我不能操作别的命令了，最后只能Ctrl+C停止它。那么，如果我想它一直运行，但不要影响我们的其他前台操作，怎么办呢？接下来我们学学怎么挂后台进程操作：

```
[Teemo@localhost ~]$ ping 127.0.0.1 >> ping.log &
[1] 2388
[Teemo@localhost ~]$ jobs
[1]+  Running                 ping 127.0.0.1 >> ping.log &
[Teemo@localhost ~]$ tail -f ping.log
64 bytes from 127.0.0.1: icmp_seq=45 ttl=64 time=0.057 ms
64 bytes from 127.0.0.1: icmp_seq=46 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=47 ttl=64 time=0.340 ms
64 bytes from 127.0.0.1: icmp_seq=48 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=49 ttl=64 time=0.057 ms
64 bytes from 127.0.0.1: icmp_seq=50 ttl=64 time=0.056 ms
64 bytes from 127.0.0.1: icmp_seq=51 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=52 ttl=64 time=0.056 ms
64 bytes from 127.0.0.1: icmp_seq=53 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=54 ttl=64 time=0.050 ms
64 bytes from 127.0.0.1: icmp_seq=55 ttl=64 time=0.199 ms
64 bytes from 127.0.0.1: icmp_seq=56 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=57 ttl=64 time=0.060 ms
64 bytes from 127.0.0.1: icmp_seq=58 ttl=64 time=0.108 ms
```

我们ping百度收不到数据，那我们ping本机地址，然后在末尾加个 >> ping.log 表示信息流到ping.log文件里，这个是不是有点像C++的流向符呢，哈哈。我们再在最后加个&符号，表示挂后台，然后会发现这个进程到了后台，标号为【1】。在这里，我们可以jobs查看当前的后台进程，注意，jobs只能看到那些被我们用“&”挂到后台的进程，这不意味着我们的Linux只有这一个后台进程。随后，用tail -f ping.log持续查看该文件，发现有连续不断的信息出来，这证明了它确实挂在后台。

```
[Teemo@localhost ~]$ fg % 1
ping 127.0.0.1 >> ping.log
^C[Teemo@localhost ~]$
```

如果要后台进程回到前台，fg % （进程标号）即可，如图，我让ping命令回到前台，但由于我命令其一切信息输入文件，所以看不到打印的信息。命令行一直“卡”在那里，所以我就手动暂停了该进程，完美落幕了属于是。



打开文件一看，乱七八糟的一堆数据，我们就不管它了。

那么我们怎么才能看到更多的进程呢？现在不妨试试用ps命令从用户自己的进程开始看起吧：

```
[Teemo@localhost ~]$ ps
  PID TTY          TIME CMD
 2346 pts/0    00:00:00 bash
 2407 pts/0    00:00:00 ps
[Teemo@localhost ~]$ su - root
Password:
[root@localhost ~]# ps
  PID TTY          TIME CMD
 2408 pts/0    00:00:00 su
 2414 pts/0    00:00:00 bash
 2434 pts/0    00:00:00 ps
[root@localhost ~]#
```

可以看到，开始只能看到bash和ps两个进程，bash是命令行脚本，ps是我们的命令没啥好说的，换上了root用户后多了一个su命令的进程，这里有话说。值得注意的是，如果我们是多个用户同时使用一台Linux服务器，虽然进程名字可能相同，但是PID（进程唯一ID）不同就是不同的进程了。例如开始的两个是2346和2407，后面两个同名字进程是2414和2434，就说明了是不同的进程，所以说，ps只是观察用户自己的进程而已，看不到别人的。至于那个TTY什么东西，我查了一下，是指各种类型的终端设备：①tty1-tty6 说明是在本地机器的命令行下登录的， tty7说明是在本地机器的图形界面下登录的；②pts说明是用远程工具连接的，比如xshell，后面的数字代表登录的时间顺序，越小证明登录的越早。

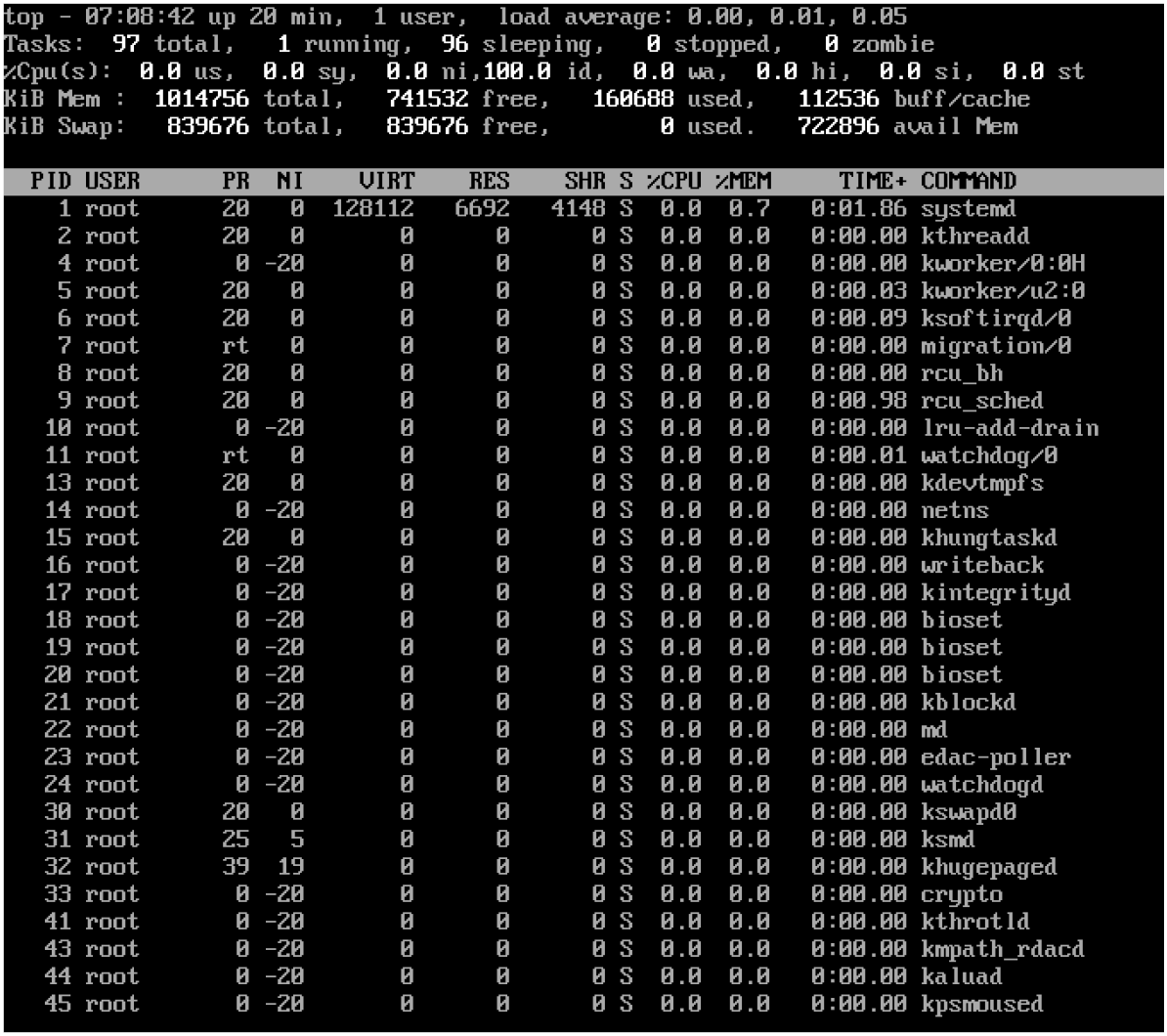
```
[root@localhost ~]# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 16:59 ?        00:00:01 /sbin/init
root         2     0  0 16:59 ?        00:00:00 [kthreadd]
root         3     2  0 16:59 ?        00:00:00 [migration/0]
root         4     2  0 16:59 ?        00:00:00 [ksoftirqd/0]
root         5     2  0 16:59 ?        00:00:00 [stopper/0]
root         6     2  0 16:59 ?        00:00:00 [watchdog/0]
root         7     2  0 16:59 ?        00:00:01 [events/0]
root         8     2  0 16:59 ?        00:00:00 [events/0]
root         9     2  0 16:59 ?        00:00:00 [events_long/0]
root        10     2  0 16:59 ?        00:00:00 [events_power_ef]
root        11     2  0 16:59 ?        00:00:00 [cgroup]
root        12     2  0 16:59 ?        00:00:00 [khelper]
root        13     2  0 16:59 ?        00:00:00 [netns]
root        14     2  0 16:59 ?        00:00:00 [async/mgr]
root        15     2  0 16:59 ?        00:00:00 [pm]
root        16     2  0 16:59 ?        00:00:00 [sync_supers]
root        17     2  0 16:59 ?        00:00:00 [bdi-default]
root        18     2  0 16:59 ?        00:00:00 [kintegrityd/0]
root        19     2  0 16:59 ?        00:00:00 [kblockd/0]
root        20     2  0 16:59 ?        00:00:00 [kacpid]
root        21     2  0 16:59 ?        00:00:00 [kacpi_notify]
root        22     2  0 16:59 ?        00:00:00 [kacpi_hotplug]
```

如图，ps -ef查看全局进程，这才是我们最喜欢的操作😊，但同时这个还真不好理解。我们细细道来：

我们先理解一下各列指标意思。①UID：启动该进程的用户；②PID：该进程的ID号，唯一的；③PPID：父进程的ID，先有父进程再有字进程，例如图中的PID为3的进程的父进程是2，说明是它是2创建出来了；④C：我也不知道是什么；⑤STIME：StartTime该进程何时启动的；⑥TTY：上面说过了；⑦TIME：该进程执行的时间，占用的CPU时间；⑧CMD：执行的命令。

看到这，可能有的小伙伴疑惑了，我16:59开的Linux，这些最多执行了1s，最少甚至是0，怎么回事？当然不是我一开机就ps -ef啦，就算手速再快也不该这样的，而且我截图的时候是17:30了。其实，这和CPU的底层知识有关。平时我们在windows查看的CPU使用率，你有想过怎么计算的吗，CPU是一直对开着的进程运作的吗，CPU是完成一个任务再开下一个任务吗？并不是，CPU采用的是分时系统，举个例子，一分钟里，CPU把时间分成了60份，每份一秒，然后给进程分配份数，给A份20份，给B分5分，这样一分钟里，CPU给A20s的处理时间，然后又给B5s处理时间，处理完没有别的分配了就立刻摆烂。事实上，CPU分的时间片自然比这细细细细细细细的多，这也是为什么我们查看CPU使用率它会不断刷新，它可能是分割的1秒的样子，然后每秒统计一下各进程的时间片分配占比。

所以回到图片，可能自从开机以来，CPU根本没给这个进程多少个时间片，所以总计运行时间1秒甚至比1秒还少。



top命令查看CPU使用率。每几秒会刷新一次，如果连续按Enter键，会迅速刷新。上半部分显示系统的整体状况，下半部分显示一个个进程的情况。

下面的我们就不细看了，上面都说的差不多了，我们直奔%Cpu(s)：0.0 us……这一行。CPU(s)是指整个计算机CPU的平均使用率，如果你是双核CPU的，按下1，会显示两个核各自的情况，注意下面的进程的占用率算的是占一个核的使用率。例如，一个进程%CPU是99.2%，但是%Cpu(s)一栏的使用率(us)是33.3%，而空闲率(id)是64.8%，可不是用了99.2%吗，才33.3%？这时候你该留意你的CPU是不是多核，按下1，你会发现你一个核使用率100us、0id，另一个0us、100id。当然这些数据是乱举的，明白就好。剩下的参数什么意思，建议自己动手查查，我摆烂了。

- 后记：

学会查看这些指标是很重要的一点，其中有大把指标参数，不懂的自己查查嗷~