[**每天一个linux命令（46）：vmstat命令**](http://www.cnblogs.com/peida/archive/2012/12/25/2833108.html)

vmstat是Virtual Meomory Statistics（虚拟内存统计）的缩写，可对操作系统的虚拟内存、进程、CPU活动进行监控。他是对系统的整体情况进行统计，不足之处是无法对某个进程进行深入分析。vmstat 工具提供了一种低开销的系统性能观察方式。因为 vmstat 本身就是低开销工具，在非常高负荷的服务器上，你需要查看并监控系统的健康情况,在控制窗口还是能够使用vmstat 输出结果。在学习vmstat命令前，我们先了解一下Linux系统中关于物理内存和虚拟内存相关信息。

**物理内存和虚拟内存区别：**

我们知道，直接从物理内存读写数据要比从硬盘读写数据要快的多，因此，我们希望所有数据的读取和写入都在内存完成，而内存是有限的，这样就引出了物理内存与虚拟内存的概念。

物理内存就是系统硬件提供的内存大小，是真正的内存，相对于物理内存，在linux下还有一个虚拟内存的概念，虚拟内存就是为了满足物理内存的不足而提出的策略，它是利用磁盘空间虚拟出的一块逻辑内存，用作虚拟内存的磁盘空间被称为交换空间（Swap Space）。

作为物理内存的扩展，linux会在物理内存不足时，使用交换分区的虚拟内存，更详细的说，就是内核会将暂时不用的内存块信息写到交换空间，这样以来，物理内存得到了释放，这块内存就可以用于其它目的，当需要用到原始的内容时，这些信息会被重新从交换空间读入物理内存。

linux的内存管理采取的是分页存取机制，为了保证物理内存能得到充分的利用，内核会在适当的时候将物理内存中不经常使用的数据块自动交换到虚拟内存中，而将经常使用的信息保留到物理内存。

要深入了解linux内存运行机制，需要知道下面提到的几个方面：

首先，Linux系统会不时的进行页面交换操作，以保持尽可能多的空闲物理内存，即使并没有什么事情需要内存，Linux也会交换出暂时不用的内存页面。这可以避免等待交换所需的时间。

其次，linux进行页面交换是有条件的，不是所有页面在不用时都交换到虚拟内存，linux内核根据”最近最经常使用“算法，仅仅将一些不经常使用的页面文件交换到虚拟内存，有时我们会看到这么一个现象：linux物理内存还有很多，但是交换空间也使用了很多。其实，这并不奇怪，例如，一个占用很大内存的进程运行时，需要耗费很多内存资源，此时就会有一些不常用页面文件被交换到虚拟内存中，但后来这个占用很多内存资源的进程结束并释放了很多内存时，刚才被交换出去的页面文件并不会自动的交换进物理内存，除非有这个必要，那么此刻系统物理内存就会空闲很多，同时交换空间也在被使用，就出现了刚才所说的现象了。关于这点，不用担心什么，只要知道是怎么一回事就可以了。

最后，交换空间的页面在使用时会首先被交换到物理内存，如果此时没有足够的物理内存来容纳这些页面，它们又会被马上交换出去，如此以来，虚拟内存中可能没有足够空间来存储这些交换页面，最终会导致linux出现假死机、服务异常等问题，linux虽然可以在一段时间内自行恢复，但是恢复后的系统已经基本不可用了。

因此，合理规划和设计linux内存的使用，是非常重要的。

**虚拟内存原理：**

在系统中运行的每个进程都需要使用到内存，但不是每个进程都需要每时每刻使用系统分配的内存空间。当系统运行所需内存超过实际的物理内存，内核会释放某些进程所占用但未使用的部分或所有物理内存，将这部分资料存储在磁盘上直到进程下一次调用，并将释放出的内存提供给有需要的进程使用。

在Linux内存管理中，主要是通过“调页Paging”和“交换Swapping”来完成上述的内存调度。调页算法是将内存中最近不常使用的页面换到磁盘上，把活动页面保留在内存中供进程使用。交换技术是将整个进程，而不是部分页面，全部交换到磁盘上。

分页(Page)写入磁盘的过程被称作Page-Out，分页(Page)从磁盘重新回到内存的过程被称作Page-In。当内核需要一个分页时，但发现此分页不在物理内存中(因为已经被Page-Out了)，此时就发生了分页错误（Page Fault）。

当系统内核发现可运行内存变少时，就会通过Page-Out来释放一部分物理内存。经管Page-Out不是经常发生，但是如果Page-out频繁不断的发生，直到当内核管理分页的时间超过运行程式的时间时，系统效能会急剧下降。这时的系统已经运行非常慢或进入暂停状态，这种状态亦被称作thrashing(颠簸)。

**1．命令格式：**

vmstat [-a] [-n] [-S unit] [delay [ count]]

vmstat [-s] [-n] [-S unit]

vmstat [-m] [-n] [delay [ count]]

vmstat [-d] [-n] [delay [ count]]

vmstat [-p disk partition] [-n] [delay [ count]]

vmstat [-f]

vmstat [-V]

**2．命令功能：**

用来显示虚拟内存的信息

**3．命令参数：**

-a：显示活跃和非活跃内存

-f：显示从系统启动至今的fork数量 。

-m：显示slabinfo

-n：只在开始时显示一次各字段名称。

-s：显示内存相关统计信息及多种系统活动数量。

delay：刷新时间间隔。如果不指定，只显示一条结果。

count：刷新次数。如果不指定刷新次数，但指定了刷新时间间隔，这时刷新次数为无穷。

-d：显示磁盘相关统计信息。

-p：显示指定磁盘分区统计信息

-S：使用指定单位显示。参数有 k 、K 、m 、M ，分别代表1000、1024、1000000、1048576字节（byte）。默认单位为K（1024 bytes）

-V：显示vmstat版本信息。

**4．使用实例：**

**实例1：显示虚拟内存使用情况**

**命令：**

vmstat

**输出：**

[root@localhost ~]# vmstat 5 6  
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------  
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st  
 0  0      0 3029876 199616 690980    0    0     0     2    3    2  0  0 100  0  0  
 0  0      0 3029752 199616 690980    0    0     0    41 1009   39  0  0 100  0  0  
 0  0      0 3029752 199616 690980    0    0     0     3 1004   36  0  0 100  0  0  
 0  0      0 3029752 199616 690980    0    0     0     4 1004   36  0  0 100  0  0  
 0  0      0 3029752 199616 690980    0    0     0     6 1003   33  0  0 100  0  0

 0  0      0 3029752 199616 690980    0    0     0     5 1003   33  0  0 100  0  0

**说明：**

字段说明：

Procs（进程）：

r: 运行队列中进程数量

b: 等待IO的进程数量

Memory（内存）：

swpd: 使用虚拟内存大小

free: 可用内存大小

buff: 用作缓冲的内存大小

cache: 用作缓存的内存大小

Swap：

si: 每秒从交换区写到内存的大小

so: 每秒写入交换区的内存大小

IO：（现在的Linux版本块的大小为1024bytes）

bi: 每秒读取的块数

bo: 每秒写入的块数

系统：

in: 每秒中断数，包括时钟中断。

cs: 每秒上下文切换数。

CPU（以百分比表示）：

us: 用户进程执行时间(user time)

sy: 系统进程执行时间(system time)

id: 空闲时间(包括IO等待时间),中央处理器的空闲时间 。以百分比表示。

wa: 等待IO时间

备注： 如果 r经常大于 4 ，且id经常少于40，表示cpu的负荷很重。如果pi，po 长期不等于0，表示内存不足。如果disk 经常不等于0， 且在 b中的队列 大于3， 表示 io性能不好。Linux在具有高稳定性、可靠性的同时，具有很好的可伸缩性和扩展性，能够针对不同的应用和硬件环境调整，优化出满足当前应用需要的最佳性能。因此企业在维护Linux系统、进行系统调优时，了解系统性能分析工具是至关重要的。

命令：

vmstat 5 5

表示在5秒时间内进行5次采样。将得到一个数据汇总他能够反映真正的系统情况。

**实例2：显示活跃和非活跃内存**

**命令：**

vmstat -a 2 5

**输出：**

[root@localhost ~]# vmstat -a 2 5  
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------  
 r  b   swpd   free  inact active   si   so    bi    bo   in   cs us sy id wa st  
 0  0      0 3029752 387728 513008    0    0     0     2    3    2  0  0 100  0  0  
 0  0      0 3029752 387728 513076    0    0     0     0 1005   34  0  0 100  0  0  
 0  0      0 3029752 387728 513076    0    0     0    22 1004   36  0  0 100  0  0  
 0  0      0 3029752 387728 513076    0    0     0     0 1004   33  0  0 100  0  0  
 0  0      0 3029752 387728 513076    0    0     0     0 1003   32  0  0 100  0  0

[root@localhost ~]#

**说明：**

使用-a选项显示活跃和非活跃内存时，所显示的内容除增加inact和active外，其他显示内容与例子1相同。

字段说明：

Memory（内存）：

inact: 非活跃内存大小（当使用-a选项时显示）

active: 活跃的内存大小（当使用-a选项时显示）

**实例3：查看系统已经fork了多少次**

**命令：**

vmstat -f

**输出：**

[root@SCF1129 ~]# vmstat -f

     12744849 forks

[root@SCF1129 ~]#

**说明：**

这个数据是从/proc/stat中的processes字段里取得的

**实例4：查看内存使用的详细信息**

**命令：**

vmstat -s

**输出：**

[root@localhost ~]# vmstat -s  
      4043760  total memory  
      1013884  used memory  
       513012  active memory  
       387728  inactive memory  
      3029876  free memory  
       199616  buffer memory  
       690980  swap cache  
      6096656  total swap  
            0  used swap  
      6096656  free swap  
        83587 non-nice user cpu ticks  
          132 nice user cpu ticks  
       278599 system cpu ticks  
    913344692 idle cpu ticks  
       814550 IO-wait cpu ticks  
        10547 IRQ cpu ticks  
        21261 softirq cpu ticks  
            0 stolen cpu ticks  
       310215 pages paged in  
     14254652 pages paged out  
            0 pages swapped in  
            0 pages swapped out  
    288374745 interrupts  
    146680577 CPU context switches  
   1351868832 boot time  
       367291 forks

**说明：**

这些信息的分别来自于/proc/meminfo,/proc/stat和/proc/vmstat。

**实例5：查看磁盘的读/写**

**命令：**

vmstat -d

**输出：**

[root@localhost ~]# vmstat -d  
disk- ------------reads------------ ------------writes----------- -----IO------  
       total merged sectors      ms  total merged sectors      ms    cur    sec  
ram0       0      0       0       0      0      0       0       0      0      0  
ram1       0      0       0       0      0      0       0       0      0      0  
ram2       0      0       0       0      0      0       0       0      0      0  
ram3       0      0       0       0      0      0       0       0      0      0  
ram4       0      0       0       0      0      0       0       0      0      0  
ram5       0      0       0       0      0      0       0       0      0      0  
ram6       0      0       0       0      0      0       0       0      0      0  
ram7       0      0       0       0      0      0       0       0      0      0  
ram8       0      0       0       0      0      0       0       0      0      0  
ram9       0      0       0       0      0      0       0       0      0      0  
ram10      0      0       0       0      0      0       0       0      0      0  
ram11      0      0       0       0      0      0       0       0      0      0  
ram12      0      0       0       0      0      0       0       0      0      0  
ram13      0      0       0       0      0      0       0       0      0      0  
ram14      0      0       0       0      0      0       0       0      0      0  
ram15      0      0       0       0      0      0       0       0      0      0  
sda    33381   6455  615407   63224 2068111 1495416 28508288 15990289      0  10491  
hdc        0      0       0       0      0      0       0       0      0      0  
fd0        0      0       0       0      0      0       0       0      0      0  
md0        0      0       0       0      0      0       0       0      0      0  
[root@localhost ~]#

**说明：**

这些信息主要来自于/proc/diskstats.

merged:表示一次来自于合并的写/读请求,一般系统会把多个连接/邻近的读/写请求合并到一起来操作.

**实例6：查看/dev/sda1磁盘的读/写**

**命令：**

**输出：**

[root@SCF1129 ~]# df

文件系统                 1K-块      已用      可用 已用% 挂载点

/dev/sda3            1119336548  27642068 1034835500   3% /tmpfs                 32978376         0  32978376   0% /dev/shm

/dev/sda1              1032088     59604    920056   7% /boot

[root@SCF1129 ~]# vmstat -p /dev/sda1

sda1          reads   read sectors  writes    requested writes

               18607    4249978          6         48[root@SCF1129 ~]# vmstat -p /dev/sda3

sda3          reads   read sectors  writes    requested writes

              429350   35176268   28998789  980301488[root@SCF1129 ~]#

**说明：**

这些信息主要来自于/proc/diskstats。

reads:来自于这个分区的读的次数。

read sectors:来自于这个分区的读扇区的次数。

writes:来自于这个分区的写的次数。

requested writes:来自于这个分区的写请求次数。

**实例7：查看系统的slab信息**

**命令：**

vmstat -m

**输出：**

[root@localhost ~]# vmstat -m  
Cache                       Num  Total   Size  Pages  
ip\_conntrack\_expect           0      0    136     28  
ip\_conntrack                  3     13    304     13  
ip\_fib\_alias                 11     59     64     59  
ip\_fib\_hash                  11     59     64     59  
AF\_VMCI                       0      0    960      4  
bio\_map\_info                100    105   1064      7  
dm\_mpath                      0      0   1064      7  
jbd\_4k                        0      0   4096      1  
dm\_uevent                     0      0   2608      3  
dm\_tio                        0      0     24    144  
dm\_io                         0      0     48     77  
scsi\_cmd\_cache               10     10    384     10  
sgpool-128                   32     32   4096      1  
sgpool-64                    32     32   2048      2  
sgpool-32                    32     32   1024      4  
sgpool-16                    32     32    512      8  
sgpool-8                     45     45    256     15  
scsi\_io\_context               0      0    112     34  
ext3\_inode\_cache          51080  51105    760      5  
ext3\_xattr                   36     88     88     44  
journal\_handle               18    144     24    144  
journal\_head                 56     80     96     40  
revoke\_table                  4    202     16    202  
revoke\_record                 0      0     32    112  
uhci\_urb\_priv                 0      0     56     67  
UNIX                         13     33    704     11  
flow\_cache                    0      0    128     30  
msi\_cache                    33     59     64     59  
cfq\_ioc\_pool                 14     90    128     30  
cfq\_pool                     12     90    216     18  
crq\_pool                     16     96     80     48  
deadline\_drq                  0      0     80     48  
as\_arq                        0      0     96     40  
mqueue\_inode\_cache            1      4    896      4  
isofs\_inode\_cache             0      0    608      6  
hugetlbfs\_inode\_cache         1      7    576      7  
Cache                       Num  Total   Size  Pages  
ext2\_inode\_cache              0      0    720      5  
ext2\_xattr                    0      0     88     44  
dnotify\_cache                 0      0     40     92  
dquot                         0      0    256     15  
eventpoll\_pwq                 3     53     72     53  
eventpoll\_epi                 3     20    192     20  
inotify\_event\_cache           0      0     40     92  
inotify\_watch\_cache           1     53     72     53  
kioctx                        0      0    320     12  
kiocb                         0      0    256     15  
fasync\_cache                  0      0     24    144  
shmem\_inode\_cache           254    290    768      5  
posix\_timers\_cache            0      0    128     30  
uid\_cache                     0      0    128     30  
ip\_mrt\_cache                  0      0    128     30  
tcp\_bind\_bucket               3    112     32    112  
inet\_peer\_cache               0      0    128     30  
secpath\_cache                 0      0     64     59  
xfrm\_dst\_cache                0      0    384     10  
ip\_dst\_cache                  5     10    384     10  
arp\_cache                     1     15    256     15  
RAW                           3      5    768      5  
UDP                           5     10    768      5  
tw\_sock\_TCP                   0      0    192     20  
request\_sock\_TCP              0      0    128     30  
TCP                           4      5   1600      5  
blkdev\_ioc                   14    118     64     59  
blkdev\_queue                 20     30   1576      5  
blkdev\_requests              13     42    272     14  
biovec-256                    7      7   4096      1  
biovec-128                    7      8   2048      2  
biovec-64                     7      8   1024      4  
biovec-16                     7     15    256     15  
biovec-4                      7     59     64     59  
biovec-1                     23    202     16    202  
bio                         270    270    128     30  
utrace\_engine\_cache           0      0     64     59  
Cache                       Num  Total   Size  Pages  
utrace\_cache                  0      0     64     59  
sock\_inode\_cache             33     48    640      6  
skbuff\_fclone\_cache           7      7    512      7  
skbuff\_head\_cache           319    390    256     15  
file\_lock\_cache               1     22    176     22  
Acpi-Operand               4136   4248     64     59  
Acpi-ParseExt                 0      0     64     59  
Acpi-Parse                    0      0     40     92  
Acpi-State                    0      0     80     48  
Acpi-Namespace             2871   2912     32    112  
delayacct\_cache              81    295     64     59  
taskstats\_cache               4     53     72     53  
proc\_inode\_cache           1427   1440    592      6  
sigqueue                      0      0    160     24  
radix\_tree\_node           13166  13188    536      7  
bdev\_cache                   23     24    832      4  
sysfs\_dir\_cache            5370   5412     88     44  
mnt\_cache                    26     30    256     15  
inode\_cache                2009   2009    560      7  
dentry\_cache              60952  61020    216     18  
filp                        479   1305    256     15  
names\_cache                   3      3   4096      1  
avc\_node                     14     53     72     53  
selinux\_inode\_security      994   1200     80     48  
key\_jar                       2     20    192     20  
idr\_layer\_cache              74     77    528      7  
buffer\_head              164045 164800     96     40  
mm\_struct                    51     56    896      4  
vm\_area\_struct             1142   1958    176     22  
fs\_cache                     35    177     64     59  
files\_cache                  36     55    768      5  
signal\_cache                 72    162    832      9  
sighand\_cache                68     84   2112      3  
task\_struct                  76     80   1888      2  
anon\_vma                    458    864     24    144  
pid                          83    295     64     59  
shared\_policy\_node            0      0     48     77  
Cache                       Num  Total   Size  Pages  
numa\_policy                  37    144     24    144  
size-131072(DMA)              0      0 131072      1  
size-131072                   0      0 131072      1  
size-65536(DMA)               0      0  65536      1  
size-65536                    1      1  65536      1  
size-32768(DMA)               0      0  32768      1  
size-32768                    2      2  32768      1  
size-16384(DMA)               0      0  16384      1  
size-16384                    5      5  16384      1  
size-8192(DMA)                0      0   8192      1  
size-8192                     7      7   8192      1  
size-4096(DMA)                0      0   4096      1  
size-4096                   110    111   4096      1  
size-2048(DMA)                0      0   2048      2  
size-2048                   602    602   2048      2  
size-1024(DMA)                0      0   1024      4  
size-1024                   344    352   1024      4  
size-512(DMA)                 0      0    512      8  
size-512                    433    480    512      8  
size-256(DMA)                 0      0    256     15  
size-256                   1139   1155    256     15  
size-128(DMA)                 0      0    128     30  
size-64(DMA)                  0      0     64     59  
size-64                    5639   5782     64     59  
size-32(DMA)                  0      0     32    112  
size-128                    801    930    128     30  
size-32                    3005   3024     32    112  
kmem\_cache                  137    137   2688      1

这组信息来自于/proc/slabinfo。

slab:由于内核会有许多小对象，这些对象构造销毁十分频繁，比如i-node，dentry，这些对象如果每次构建的时候就向内存要一个页(4kb)，而其实只有几个字节，这样就会非常浪费，为了解决这个问题，就引入了一种新的机制来处理在同一个页框中如何分配小存储区，而slab可以对小对象进行分配,这样就不用为每一个对象分配页框，从而节省了空间，内核对一些小对象创建析构很频繁，slab对这些小对象进行缓冲,可以重复利用,减少内存分配次数。