[uptime,Top,Free,PS,Pstree,VMStat,SysStat](http://blog.csdn.net/xiangqiao123/article/details/37658325)

一 uptime

uptime命令用于查看服务器运行了多长时间以及有多少个用户登录，快速获知服务器的负荷情况。  
  
uptime的输出包含一项内容是load average，显示了最近1，5，15分钟的负荷情况。它的值代表等待CPU处理的进程数，如果CPU没有时间处理这些进程，load average值会升高；反之则会降低。  
load average的最佳值是1，说明每个进程都可以马上处理并且没有CPU cycles被丢失。对于单CPU的机器，1或者2是可以接受的值；而在一个多CPU的系统中这个值应除以物理CPU的个数,假设CPU个数为4,而load average为8或者10,那结果也是在2多点而已。  
也可以使用uptime命令来判断网络性能。例如，某个网络应用性能很低，通过运行uptime查看服务器的负荷是否很高，如果不是，那么问题应该是网络方面造成的。

也可以查看/proc/loadavg 和/proc/uptime两个文件来获取相关的信息。

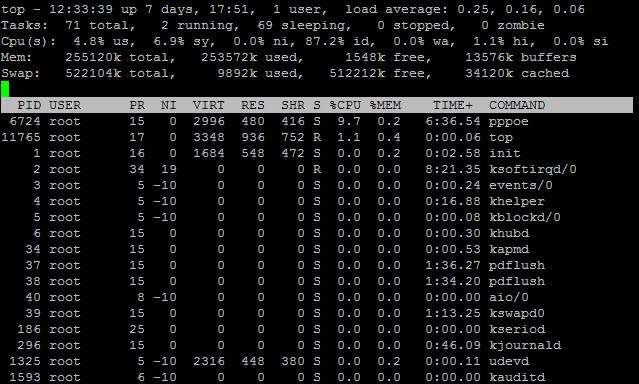
以下是uptime的运行实例：



二 Top

Top命令显示了实际CPU使用情况，默认情况下，它显示了服务器上占用CPU的任务信息并且每5秒钟刷新一次。你可以通过多种方式分类它们，包括PID、时间和内存使用情况。  
  
下面是输出值的介绍：  
  
    PID：进程标识  
    USER；进程所有者的用户名  
    PR：进程的优先级  
    NI：nice级别  
    RES：进程使用的物理内存数量  
    SHR：该进程和其他进程共享内存的数量  
    CODE:代码的大小  
    DATA:数据+堆栈大小  
    S：进程的状态：S＝休眠状态，R＝运行状态，T＝停止状态，D＝中断休眠状态，Z＝僵尸状态  
    %CPU：共享的CPU使用  
    %MEM；共享的物理内存  
    TIME：进程占用CPU的时间  
    COMMAND：启动任务的命令行（包括参数）  
  
进程的优先级和nice级别  
    进程优先级是一个决定进程被CPU执行优先顺序的参数，内核会根据需要调整这个值。Nice值是一个对优先权的限制。进程优先级的值不能低于nice值。（nice值越低优先级越高）  
    进程优先级是无法去手动改变的，只有通过改变nice值去间接的调整进程优先级。如果一个进程运行的太慢了，你可以通过指定一个较低的nice值去为它分配更多的CPU资源。当然，这意味着其他的一些进程将被分配更少的CPU资源，运行更慢一些。Linux支持nice值的范围是19（低优先级）到-20（高优先级），默认的值是0。如果需要改变一个进程的nice值为负数（高优先级），必须使用su命令登陆到root用户。下面是一些调整nice值的命令示例，  
    以nice值-5开始程序xyz  
    #nice –n -5 xyz  
  
    改变已经运行的程序的nice值  
    #renice level pid  
  
    将pid为2500的进程的nice值改为10  
    #renice 10 2500  
  
僵尸进程  
    当一个进程被结束，在它结束之前通常需要用一些时间去完成所有的任务（比如关闭打开的文件），在一个很短的时间里，这个进程的状态为僵尸状态。在进程完成所有关闭任务之后，会向父进程提交它关闭的信息。有些情况下，一个僵尸进程不能关闭它自己，这时这个进程状态就为z（zombie）。不能使用kill命令杀死僵尸进程，因为它已经标志为“dead”。如果你无法摆脱一个僵尸进程，你可以杀死它的父进程，这个僵尸进程也就消失了。然而，如果父进程是init进程，你不能杀死init进程，因为init是一个重要的系统进程，这种情况下你只能通过一次重新启动服务器来摆脱僵尸进程。也必须分析应用为什么会导致僵死？

Top运行示例：

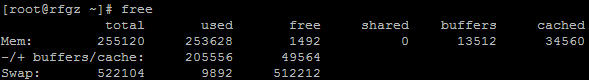


三 Free

free命令显示系统的所有内存的使用情况，包括空闲内存、被使用的内存和交换内存空间。Free命令显示也包括一些内核使用的缓存和缓冲区的信息。     当使用free命令的时候，需要记住Linux的内存结构和虚拟内存的管理方法，比如空闲内存数量的限制，还有swap空间的使用并不标志一个内存瓶颈的出现。

也可以通过 cat /proc/meminfo来查看内存的使用情况。

Free运行示例：

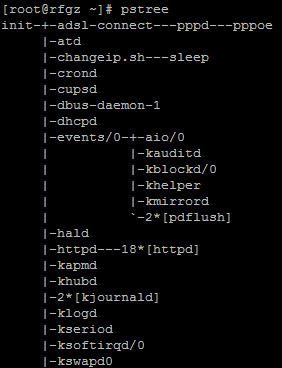


[root@Linux /tmp]# free  
             total       used       free     shared    buffers     cached  
Mem:       4149156    4130412      18744          0      13220    2720160  
-/+ buffers/cache:    1397032    2752124  
Swap:      6289408        144    6289264第1行  
total 内存总数: 4149156  
used 已经使用的内存数: 4130412  
free 空闲的内存数: 18744  
shared 当前已经废弃不用，总是0  
buffers Buffer Cache内存数: 13220  
cached Page Cache内存数: 2720160  
  
关系：total = used + free  
  
第2行：  
-/+ buffers/cache的意思相当于：  
-buffers/cache 的内存数：1397032 (等于第1行的 used - buffers - cached)  
+buffers/cache 的内存数: 2752124 (等于第1行的 free + buffers + cached)  
  
可见-buffers/cache反映的是被程序实实在在吃掉的内存，而+buffers/cache反映的是可以挪用的内存总数。  
  
第三行单独针对交换分区, 就不用再说了.  
  
为了提高磁盘存取效率, Linux做了一些精心的设计, 除了对dentry进行缓存(用于VFS,加速文件路径名到inode的转换), 还采取了两种主要Cache方式：Buffer Cache和Page Cache。前者针对磁盘块的读写，后者针对文件inode的读写。这些Cache有效缩短了 I/O系统调用(比如read,write,getdents)的时间。

四 PS 和 Pstree

ps和pstree命令是系统分析最常用的基本命令。  
  
ps命令提供了一个正在运行的进程的列表，列出进程的数量取决于命令所附加的参数。通常地我们可以使用如下命令来查找某进程是否在运行： ps -aux | grep XXX 。  
  
pstree命令可以以树状结构来显示所有的进程信息并且可以整合子进程的信息。Pstree命令对分析进程的来源十分有用。

示例：



五VMStat

Vmstat命令提供了对进程、内存、页面I/O块和CPU等信息的监控，vmstat可以显示检测结果的平均值或者取样值，取样模式可以提供一个取样时间段内不同频率的监测结果。  
  
注：在取样模式中需要考虑在数据收集中可能出现的误差，将取样频率设为比较低的值可以尽可能的减小误差的影响。  
    下面介绍一下各列的含义  
  
    ·process（procs）  
    r：等待运行时间的进程数量  
    b：处在不可中断睡眠状态的进程  
    w：被交换出去但是仍然可以运行的进程，这个值是计算出来的  
    ·memory  
    swpd：虚拟内存的数量  
    free：空闲内存的数量  
    buff：用做缓冲区的内存数量  
    ·swap  
    si：从硬盘交换来的数量  
    so：交换到硬盘去的数量  
    ·IO  
    bi：向一个块设备输出的块数量  
    bo：从一个块设备接受的块数量  
    ·system  
    in：每秒发生的中断数量， 包括时钟  
    cs：每秒发生的context switches的数量  
    ·cpu(整个cpu运行时间的百分比)  
    us：非内核代码运行的时间（用户时间，包括nice时间）  
    sy：内核代码运行的时间（系统时间）  
    id：空闲时间，在Linux 2.5.41之前的内核版本中，这个值包括I/O等待时间  
    wa：等待I/O操作的时间，在Linux 2.5.41之前的内核版本中这个值为0

示例：



六 SysStat

http://sebastien.godard.pagesperso-orange.fr/index.html

iostat： 报告CPU的统计和设备，分区，网络文件系统的IO的统计；   
mpstat： 报告进程相关的统计；   
pidstat：报告Linux的IO，CPU，memory等的统计；   
sar： 收集，报告，保存系统活动信息(CPU, memory, disks, interrupts, network interfaces, TTY, kernel tables,etc.)；