Linux下CPU使用率

linux下提供了用于监控CPU使用率的途径，这里首先介绍几个通常的文件

**/proc/uptime**

该文件中记录两个数值，第一个数值代表系统总的启动时间，第二个数值则代表系统空闲的时间，单位都是秒。

$ cat   /proc/uptime  
5711847.49   3339234.87

**/proc/[pid]/stat**

|  |
| --- |
| $ cat /proc/1/stat 1 (init) S 0 1 1 0 -1 4202752 2349 1719861150 8 23067 20 3681 34030168 1016826 20 0 1 0 1 2969600 328 4294967295 1 1 0 0 0 0 0 4096 536962595 4294967295 0 0 0 5 0 0 28 0 0 |

再看一下每个字段代表的具体的含义

|  |
| --- |
| pid= 进程(包括轻量级进程，即线程)号 comm= 应用程序或命令的名字 task\_state= 任务的状态，R:runnign, S:sleeping (TASK\_INTERRUPTIBLE), D:disk sleep (TASK\_UNINTERRUPTIBLE), T: stopped, T:tracing stop,Z:zombie, X:dead ppid= 父进程ID pgid= 线程组号 sid=该任务所在的会话组ID tty\_nr= (pts/3) 该任务的tty终端的设备号，INT（34817/256）=主设备号，（34817-主设备号）=次设备号 tty\_pgrp= 终端的进程组号，当前运行在该任务所在终端的前台任务(包括shell 应用程序)的PID。 task->flags= 进程标志位，查看该任务的特性 min\_flt= 该任务不需要从硬盘拷数据而发生的缺页（次缺页）的次数 cmin\_flt= 累计的该任务的所有的waited-for进程曾经发生的次缺页的次数目 maj\_flt= 该任务需要从硬盘拷数据而发生的缺页（主缺页）的次数 cmaj\_flt= 累计的该任务的所有的waited-for进程曾经发生的主缺页的次数目 utime= 该任务在用户态运行的时间，单位为jiffies stime= 该任务在核心态运行的时间，单位为jiffies cutime= 累计的该任务的所有的waited-for进程曾经在用户态运行的时间，单位为jiffies cstime= 累计的该任务的所有的waited-for进程曾经在核心态运行的时间，单位为jiffies priority= 任务的动态优先级 nice= 任务的静态优先级 num\_threads= 该任务所在的线程组里线程的个数 it\_real\_value= 由于计时间隔导致的下一个 SIGALRM 发送进程的时延，以 jiffy 为单位. start\_time= 该任务启动的时间，单位为jiffies vsize=（page） 该任务的虚拟地址空间大小 rss= (page) 该任务当前驻留物理地址空间的大小 Number of pages the process has in real memory,minu 3 for administrative purpose. 这些页可能用于代码，数据和栈。 rlim=（bytes） 该任务能驻留物理地址空间的最大值 start\_code= 该任务在虚拟地址空间的代码段的起始地址 end\_code= 该任务在虚拟地址空间的代码段的结束地址 start\_stack= 该任务在虚拟地址空间的栈的结束地址 kstkesp= esp(32 位堆栈指针) 的当前值, 与在进程的内核堆栈页得到的一致. kstkeip= 指向将要执行的指令的指针, EIP(32 位指令指针)的当前值. pendingsig= 待处理信号的位图，记录发送给进程的普通信号 block\_sig= 阻塞信号的位图 sigign= 忽略的信号的位图 sigcatch= 被俘获的信号的位图 wchan= 如果该进程是睡眠状态，该值给出调度的调用点 nswap 被swapped的页数，当前没用 cnswap 所有子进程被swapped的页数的和，当前没用 exit\_signal= 该进程结束时，向父进程所发送的信号 task\_cpu(task)= 运行在哪个CPU上 task\_rt\_priority= 实时进程的相对优先级别 task\_policy= 进程的调度策略，0=非实时进程，1=FIFO实时进程；2=RR实时进程 |

通常情况下，进程占用CPU的运行时间的统计方法为

procpu= utime(该任务在用户态运行的时间) + stime(该任务在核心态运行的时间) + cutime(累计的该任务的所有的waited-for进程曾经在用户态运行的时间) + cstime(累计的该任务的所有的waited-for进程曾经在核心态运行的时间)  
cutime=所有已死线程在用户态运行的时间  cstime=所有已死在核心态运行的时间

**/proc/cpuinfo**

该文件中存放了有关 cpu的相关信息(型号，缓存大小等)。

**/proc/stat**

 该文件包含了所有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。不同内核版本中该文件的格式可能不大一致，以下通过实例来说明数据该文件中各字段的含义。

|  |
| --- |
| fjzag@fjzag-desktop:~$ cat /proc/stat  cpu  38082 627 27594 893908 12256 581 895 0 0  cpu0 22880 472 16855 430287 10617 576 661 0 0  cpu1 15202 154 10739 463620 1639 4 234 0 0  intr 120053 222 2686 0 1 1 0 5 0 3 0 0 0 47302 0 0 34194 29775 0 5019 845 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  ctxt 1434984  btime 1252028243  processes 8113  procs\_running 1  procs\_blocked 0 |

第一行的数值表示的是CPU总的使用情况，所以我们只要用第一行的数字计算就可以了。下表解析第一行各数值的含义：

**参数          解析（单位：jiffies）**

**user** (38082)从系统启动开始累计到当前时刻，处于用户态的运行时间，不包含 nice值为负进程。

**nice** (627)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

**system** (27594)从系统启动开始累计到当前时刻，处于核心态的运行时间

**idle** (893908)从系统启动开始累计到当前时刻，除IO等待时间以外的其它等待时间

**iowait (**12256**)**从系统启动开始累计到当前时刻，IO等待时间

**irq (**581**)**从系统启动开始累计到当前时刻，硬中断时间

**softirq** (895)从系统启动开始累计到当前时刻，软中断时间

**stealstolen**(0) which is the time spent in other operating systems when running in a virtualized environment

**guest**(0) which is the time spent running a virtual  CPU  for  guest operating systems under the control of the Linux kernel

通常情况下 总的cpu时间totalCpuTime = user + nice + system + idle + iowait + irq + softirq + stealstolen  +  guest

**/proc/<pid>/task/<tid>/stat**

该文件包含了某一进程所有的活动的信息，该文件中的所有值都是从系统启动开始累计到当前时刻。该文件的内容格式以及各字段的含义同/proc/<pid>/stat文件。

         注意，该文件中的tid字段表示的不再是进程号，而是linux中的轻量级进程(lwp)，即我们通常所说的线程。

**jiffies**

Linux中对于时间的计算，用了两种计量单位：秒和jiffies。jiffies是一个全局变量，用来记录自系统启动以来产生的节拍的总数。启动时，内核将该变量初始化为0，此后，每次时钟中断处理程序都会增加该变量的值。一秒内时钟中断的次数等于Hz，所以jiffies一秒内增加的值也就是Hz，通常1jiffies对应10毫秒，也就是CPU每秒中触发100次中断。

**/proc/meminfo** 获取系统内存的详细信息。

|  |
| --- |
| $ cat /proc/meminfo MemTotal:        8131024 kB MemFree:         6895932 kB Buffers:          502416 kB Cached:           307764 kB SwapCached:         4132 kB |

**proc文件系统**

/proc文件系统是一个伪文件系统，它只存在内存当中，而不占用外存空间。它以文件系统的方式为内核与进程提供通信的接口。用户和应用程序可以通过/proc得到系统的信息，并可以改变内核的某些参数。由于系统的信息，如进程，是动态改变的，所以用户或应用程序读取/proc目录中的文件时，proc文件系统是动态从系统内核读出所需信息并提交的。

/proc目录中有一些以数字命名的目录，它们是进程目录。系统中当前运行的每一个进程在/proc下都对应一个以进程号为目录名的目录/proc/pid，它们是读取进程信息的接口。此外，中/proc/pid目录中有一个task目录，/proc/pid/task目录中也有一些以该进程所拥有的线程的线程号命名的目录/proc/pid/task/tid，它们是读取线程信息的接口。

**单核情况下Cpu使用率的计算**

**基本思想**

通过读取/proc/stat 、/proc/<pid>/stat、/proc/<pid>/task/<tid>/stat以及/proc/cpuinfo这几个文件获取总的Cpu时间、进程的Cpu时间、线程的Cpu时间以及Cpu的个数的信息，然后通过一定的算法进行计算(采样两个足够短的时间间隔的Cpu快照与进程快照来计算进程的Cpu使用率)。

**总的Cpu使用率计算**

**计算方法：**

1、  采样两个足够短的时间间隔的Cpu快照，分别记作t1,t2，其中t1、t2的结构均为：

(user、nice、system、idle、iowait、irq、softirq、stealstolen、guest)的9元组;

2、  计算总的Cpu时间片totalCpuTime

a)         把第一次的所有cpu使用情况求和，得到s1;

b)         把第二次的所有cpu使用情况求和，得到s2;

c)         s2 - s1得到这个时间间隔内的所有时间片，即totalCpuTime = j2 - j1 ;

3、计算空闲时间idle

idle对应第四列的数据，用第二次的第四列 - 第一次的第四列即可

idle=第二次的第四列 - 第一次的第四列

6、计算cpu使用率

pcpu =100\* (total-idle)/total

**某一进程Cpu使用率的计算**

**计算方法：**

1．               采样两个足够短的时间间隔的cpu快照与进程快照，

a)         每一个cpu快照均为(user、nice、system、idle、iowait、irq、softirq、stealstolen、guest)的9元组;

b)         每一个进程快照均为 (utime、stime、cutime、cstime)的4元组；

2．               计算出两个时刻的总的cpu时间与进程的cpu时间，分别记作：totalCpuTime1、totalCpuTime2、processCpuTime1、processCpuTime2

3．               计算该进程的cpu使用率pcpu = 100\*( processCpuTime2 – processCpuTime1) / (totalCpuTime2 – totalCpuTime1) (按100%计算，如果是多核情况下还需乘以cpu的个数);

**某一线程Cpu使用率的计算**

**计算方法：**

1．               采样两个足够短的时间隔的cpu快照与线程快照，

a)         每一个cpu快照均为(user、nice、system、idle、iowait、irq、softirq、stealstealon、guest)的9元组;

b)         每一个线程快照均为 (utime、stime)的2元组；

2．               分别根据**结论2、结论4**计算出两个时刻的总的cpu时间与线程的cpu时间，分别记作：totalCpuTime1、totalCpuTime2、threadCpuTime1、threadCpuTime2

3．               计算该线程的cpu使用率pcpu = 100\*( threadCpuTime2 – threadCpuTime1) / (totalCpuTime2 – totalCpuTime1) (按100%计算，如果是多核情况下还需乘以cpu的个数);

**系统中有关进程cpu使用率的常用命令**

|  |  |
| --- | --- |
| ps | 查看系统中相关进程的Cpu使用率的信息 |
| top | 查看系统的运行状况 |
|  |  |

**ps 命令**

通过ps命令可以查看系统中相关进程的Cpu使用率的信息。以下在linux man文档中对ps命令输出中有关cpu使用率的解释：

**top命令**

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

参考：

http://blog.csdn.net/zg\_hover/article/details/4356210