CPU 负载

Linux通过``/proc/stat``和``/proc/uptime``导出各种信息,用户空间工具如top(1)使用这些信息计算系统花费在某个特定状态的平均时间。例如:

```
$ iostat Linux 2.6.18.3-exp (linmac) 02/20/2007
avg-cpu: %user %nice %system %iowait %steal %idle
10.01 0.00 2.92 5.44 0.00 81.63
```

这里系统认为在默认采样周期內有10.01%的时间工作在用户空间, 2.92%的时间用在系统空间, 总体上有81.63%的时间是空闲的。

大多数情况下``/proc/stat``的信息几乎真实反映了系统信息,然而,由于内 核采集这些数据的方式/时间的特点,有时 这些信息根本不可靠。

那么这些信息是如何被搜集的呢?每当时间中断触发时,内核查看此刻运行的进程类型,并增加与此类型/状态进程对应的计数器的值。这种方法的问题是在两次时间中断之间系统(进程)能够在多种状态之间切换多次,而计数器只增加最后一种状态下的计数。

举例 ---

假设系统有一个进程以如下方式周期性地占用cpux

在上面的情况下,根据``/proc/stat``的信息(由于当系统处于空闲状态时,时间中断经常会发生)系统的负载将会是0大家能够想象内核的这种行为会发生在许多情况下,这将导致``/proc/stat`` 中存在相当古怪的信息:

```
/* gcc -o hog smallhog.c */
#include <time.h>
#include <limits.h>
#include <signal.h>
#include <sys/time.h>
#define HIST 10
static volatile sig atomic t stop;
static void sighandler (int signr)
(void) signr;
stop = 1;
static unsigned long hog (unsigned long niters)
stop = 0;
while (!stop && --niters);
return niters;
int main (void)
struct itimerval it = { .it_interval = { .tv_sec = 0, .tv_usec = 1 },
                        .it value = { .tv sec = 0, .tv usec = 1 } };
sigset t set;
unsigned long v[HIST];
double tmp = 0.0;
unsigned long n;
signal (SIGALRM, &sighandler);
setitimer (ITIMER REAL, &it, NULL);
hog (ULONG MAX);
for (i = 0; i < HIST; ++i) v[i] = ULONG MAX - hog (ULONG MAX);
for (i = 0; i < HIST; ++i) tmp += v[i];
tmp /= HIST;
n = tmp - (tmp / 3.0);
sigemptyset (&set);
sigaddset (&set, SIGALRM);
```

```
for (;;) {
       hog (n);
sigwait (&set, &i);
return 0;
```

参考 ---

- https://lore.kernel.org/r/loom.20070212T063225-663@post.gmane.org
 Documentation/filesystems/proc.rst (1.8)

谢谢 ---

Con Kolivas, Pavel Machek