Linux时钟精度：毫秒？微妙？纳秒？ - William.Wu

chezxiaoqiang 2021-12-14

Linux时钟精度：毫秒？微妙？纳秒？

最近被内核时钟精度弄的很是郁闷。具体情况如下：

扫盲：1秒=1000毫秒=1000000微妙=1000000000纳秒

首先：linux有一个很重要的概念——节拍，它的单位是（次/秒）。2.6内核这个值是1000，系统中用一个HZ的宏表征这个值。同时有全局的jiffies变量，表征从开机以来经过的节拍次数（这里面还有故事，后面说，先记住这个）。当然还有wall_jiffies的墙上jiffies来表示从 07-01-1970 到现在的节拍数。每个节拍里面执行一次时钟中断。就是说，它的精度是毫秒。

接着：内核中还有一个变量xtime表征系统的实际时间（墙上时间），定义如下。其中xtime.tv_sec以秒为单位，存放从Unix祖宗定的纪元时间（19700701）到现在的秒数。xtime.tv_nsec以纳秒为单位，记录从上一秒开始经过的纳秒数。就是说，它的精度是纳秒。

```
struct timespec xtime;

struct timespec{
    time_t tv_sec;   //秒
    long tv_nsec;    //纳秒
};
```

最后：linux提供一个gettimeofday的系统调用，它会返回一个timeval的结构体，定义如下。解释同上，tv_sec代表墙上时间的秒，tv_usec表示从上一秒到现在经过的微秒数。就是说，它的精度是微妙。

```
struct timeval{
    long tv_sec;     //秒
    long tv_usec;    //微妙
};
```

精彩的来了：

1. 内核中的xtime会在每个时钟中断的时候被更新一次，也就是每个节拍更新一次。你妹！！每毫秒更新一次怎么能冒出来纳秒的精度？？而且，内核还有可能丢失节拍。怎

猜你喜欢

Linux时钟精度：毫秒 2021-12-04
时间换算,秒、毫秒、 2021-12-13
TimeStamp 毫秒和纳 2021-12-30
秒,毫秒,微秒,纳秒,皮 2021-12-14
秒的换算：皮秒、纳 2021-12-14

相关资源

jQuery实现的精确到 2023-02-01
冰清玉洁QQ空间秒赞 2022-12-05
嵐 - 蒙纳清华 Regular 2023-01-18
蒙纳幼雅丽 Regular 2023-01-03
蒙纳中综艺 Regular 2023-02-10

相似解决方案

SqlAlchemy mysql 毫 2015-06-25
文件时间戳精度 - ext3 2023-03-17
Linux' hrtimer - 微秒精 2023-02-03
具有毫秒精度的 Linux 1970-01-01
如何在Linux中从C打 2013-04-22
R中的Delta时间戳解 2019-08-01
以微秒或纳秒精度解 2015-07-20
从微秒转换为小时、 2013-01-12
为什么 MySQL 不支持 2017-02-13

么能是纳秒？？

2. 各种书上说，gettimeofday系统调用是读取的xtime的值。日，为啥读出来之后精度丢了？变成微妙了？

寻寻觅觅终于理清了故事：

针对问题1：在linux启动的时候，一个节拍的时间长度还会以纳秒为单位初始化到tick_nsec中，初始化值为999848ns，坑爹啊！不到一毫秒！节拍大约为1000.15Hz。靠！实际的节拍竟然不是准确的1000！所以在每个时钟中断通过wall_jiffies去更新xtime的时候得到的就是一个以纳秒为最小单位的的值。所以！xtime的粒度应该是不到1毫秒，也就是精度是不到1毫秒。

针对问题2：gettimeday系统调用的读xtime代码部分如下：

```
do{
    unsigned long lost;
    seq = read_seqbegin(&xtime_lock);

    usec = timer->get_offset();    //在计时器中取从上一次时钟中断到现在的微秒数
    lost = jiffies - wall_jiffies;
    if(lost)
        usec += lost*(1000000/HZ); //HZ是节拍宏，值1000
    sec = xtime.tv_sec;
    usec += (xtime.tv_nsec/1000);  //由纳秒转为微妙

}while(read_seqretry(&xtime_lock, seq))
```

while部分使用了seg锁，只看中间的就好了。加了注释之后就很清晰了。由于节拍可能会丢失，所以lost是丢失的节拍数（不会很多）。至于计时器就比较麻烦了，timer可能有下面四种情况。

a. 如果cur_timer指向timer_hpet对象，该方法使用HPET定时器——Inter与Microsoft开发的高精度定时器频率至少10MHz，也就是说此时可提供真正的微妙级精度。

b. 如果cur_timer指向timer_pmtmr对象，该方法使用ACPI PMT计时器（电源管理定时器）平率大约3.58MHz，也就是说也可以提供真正的微妙级精度。

c. 如果cur_timer指向timer_tsc对象，该方法使用时间戳计数器，内置在所有8086处理器，每个CPU时钟，计数器增加一次，频率就是CPU频率，所以timer精度最高。完全可以胜任微妙级的精度。

d. 如果cur_timer指向timer_pit对象，该方法使用PIT计数器，也即是最开始提到的节拍计数，频率大概是1000Hz，此时显然不能提供精度达到微妙的时间。所以只有这种情况是假毫秒精度！

综上：如果使用gettimeofday系统调用，只要不要使用节拍计数器就可以保证达到微妙

精度的时间（刨除进程上下文时间误差）。至于网上说的可以拿到纳秒精度的时间，看起来都是错的。除非通过修改内核，使用时间戳计数器实现。Over！

最后最后说一个事情：jiffies的定义的是4字节，你可能猜想它初始值是0。实际上，事实并非如此！linux中jiffies被初始化为0xfffb6c20,它是一个32位有符号数，正好等于-300 000。因此，计数器会在系统启动5分钟内溢出。这是为了使对jiffies溢出处理有缺陷的内核代码在开发阶段被发现，避免此类问题出现在稳定版本中。

参考《深入理解linux内核》

分类：

技术点：

相关文章：

秒,毫秒,微秒,纳秒,皮秒,飞秒 - jtlgb 2021-12-14

oracle精确到毫秒纳秒 2021-06-15

java的计时：毫秒、纳秒 2022-12-23

时钟周期及秒(s) 毫秒(ms) 微秒(μs) 纳秒(ns) 皮秒(ps)之间转换 2021-12-28

java获取当前系统毫秒，纳秒 2022-12-23

秒,毫秒,微秒,纳秒,皮秒,飞秒 - 中道学友 2021-12-14

linux下的定时或计时操作（gettimeofday等的用法，秒，微妙，纳秒(转… 2022-12-23