

[Fwd: timer counter chip access mystery]

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/performance/2003-07/0030.html>

From: Jin Guojun [DSD] (j_guojun_at_lbl.gov)

Date: 07/16/03

Date: Wed, 16 Jul 2003 14:37:08 -0700

To: freebsd-performance@freebsd.org

```
i386/isa/clock.c, line 1207
i8254_get_timecount(...)
{
    outb(TIMER_MODE, TIMER_SEL0 | TIMER_LATCH);

    low = inb(TIMER_CNTR0);
    high = inb(TIMER_CNTR0);
    ...
    inb(IO_ICU1);
return counter.
}
```

This routine takes about 4000 ns. It makes `gettimeofday()` cost over 4000 ns.

measure_i8254_get_timecount: Two_TIMER_CNTR0 2362 ns

measure_i8254_get_timecount: IO_ICU1 936 ns

and `outb()` looks like to take another 700 ns.

The Linux uses the same process to get the time counter (see below). It also comments that this is from Steve McCanne's `microtime-i386` for BSD. However, `gettimeofday()` under Linux costs 900 ns. See attached GIF file for a table of comparison.

Any idea why linux is 4 times faster than FreeBSD in reading the same time counter chip?

-Jin

----- Linux source code for the same routine -----

```
arch/x86_64/kernel/time.c (line 68)
void do_gettimeofday(struct timeval *tv)
{
    unsigned long flags, t;
    unsigned int sec, usec;
```

frebsd-performance: [Fwd: timer counter chip access mystery]

```
read_lock_irqsave(&xtime_lock, flags);
spin_lock(&time_offset_lock);

sec = xtime.tv_sec;
usec = xtime.tv_usec;

t = (jiffies - wall_jiffies) * (1000000L / HZ) + do_gettimeoffset();
if (t > timeoffset) timeoffset = t;
usec += timeoffset;

spin_unlock(&time_offset_lock);
read_unlock_irqrestore(&xtime_lock, flags);

tv->tv_sec = sec + usec / 1000000;
tv->tv_usec = usec % 1000000;
}
```

arch/i386/kernel/time.c (line 127)

```
/* This function must be called with interrupts disabled
 * It was inspired by Steve McCanne's microtime-i386 for BSD. -- jrs
 *
 * However, the pc-audio speaker driver changes the divisor so that
 * it gets interrupted rather more often - it loads 64 into the
 * counter rather than 11932! This has an adverse impact on
 * do_gettimeoffset() -- it stops working! What is also not
 * good is that the interval that our timer function gets called
 * is no longer 10.0002 ms, but 9.9767 ms. To get around this
 * would require using a different timing source. Maybe someone
 * could use the RTC - I know that this can interrupt at frequencies
 * ranging from 8192Hz to 2Hz. If I had the energy, I'd somehow fix
 * it so that at startup, the timer code in sched.c would select
 * using either the RTC or the 8253 timer. The decision would be
 * based on whether there was any other device around that needed
 * to trample on the 8253. I'd set up the RTC to interrupt at 1024 Hz,
 * and then do some jiggery to have a version of do_timer that
 * advanced the clock by 1/1024 s. Every time that reached over 1/100
 * of a second, then do all the old code. If the time was kept correct
 * then do_gettimeoffset could just return 0 - there is no low order
 * divider that can be accessed.
 *
 * Ideally, you would be able to use the RTC for the speaker driver,
 * but it appears that the speaker driver really needs interrupt more
 * often than every 120 us or so.
 *
 * Anyway, this needs more thought.... pjs (1993-08-28)
 *
 * If you are really that interested, you should be reading
 * comp.protocols.time.ntp!
 */
```

[Fwd: timer counter chip access mystery]

freebsd-performance: [Fwd: timer counter chip access mystery]

```
static unsigned long do_slow_gettimeoffset(void)
{
    int count;

    static int count_p = LATCH; /* for the first call after boot */
    static unsigned long jiffies_p = 0;

    /*
     * cache volatile jiffies temporarily; we have IRQs turned off.
     */
    unsigned long jiffies_t;

    /* gets recalled with irq locally disabled */
    spin_lock(&i8253_lock);
    /* timer count may underflow right here */
    outb_p(0x00, 0x43); /* latch the count ASAP */

    count = inb_p(0x40); /* read the latched count */

    jiffies_t = jiffies;

    count |= inb_p(0x40) << 8;

    /* VIA686a test code... reset the latch if count > max + 1 */
    if (count > LATCH) {
        outb_p(0x34, 0x43);
        outb_p(LATCH & 0xff, 0x40);
        outb(LATCH >> 8, 0x40);
        count = LATCH - 1;
    }

    spin_unlock(&i8253_lock);
    ...
    return count;
}
```

freebsd-performance@freebsd.org mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-performance>

To unsubscribe, send any mail to "freebsd-performance-unsubscribe@freebsd.org"