

Linux 内核中一种高精度定时器的设计与实现

周 鹏, 周明天

(电子科技大学 计算机学院, 四川 成都 610054)

摘 要: CGL(Carrier Grade Linux)是由开源组织(OSDL, Open Source Development Lab)发起的、专门针对电信级服务的 Linux。CGL 在标准 Linux 的基础上,增加了一组为适应电信运营环境而设计的特性。某些电信应用对实时性有较高要求,普通 Linux 在实时性方面和电信平台的要求之间还存在一定的差距。为增强系统的软实时能力, CGL 要求提供一种精度在 0.1 毫秒以上高精度定时器(high-resolution timer)。首先介绍 Linux 内核 2.6.x 中时钟与定时器的情况,然后详细阐述这种符合 POSIX1003.1b API 标准的高精度定时器的设计与实现,最后总结该定时器的性能并得出结论。

关键词: Linux 内核;时钟;定时器;Carrier Grade Linux

中图分类号: TP311

文献标识码: A

文章编号: 1005-3751(2006)04-0073-03

Design and Implementation of a High-Resolution Timer in Linux Kernel

ZHOU Peng, ZHOU Ming-tian

(Computer Sci. and Techn. Coll., Univ. of Electronic Sci. and Techn., Chengdu 610054, China)

Abstract: CGL(Carrier Grade Linux), which focuses on telecom service, is initialized by open source organization (OSDL, Open Source Development Lab). Based on generic release of Linux, CGL adds a series of features to meet the telecom grade needs. High-resolution timer is an enhancement for generic Linux soft real-time performance whose interfaces conform to the POSIX1003.1b API. This paper firstly introduces the clock and timer in standard kernel 2.6.x. Secondly describes design and implementation of high-resolution timer. At last summarizes the performance of high-resolution timer and draws a conclusion.

Key words: Linux kernel; clock; timer; carrier grade Linux

1 内核 2.6.x 中的时钟与定时器

时钟和定时器对 Linux 内核来说十分重要。首先,内核要管理系统的运行时间(uptime)和当前墙上时间(wall time),即当前实际时间。其次,内核中大量的活动由时间驱动(time driven)。其中一些活动是周期性的,比如平衡调度器(scheduler)中的运行队列(runqueue)或刷新屏幕这样的活动,它们以固定的频率定时发生;同时,内核要非周期性地调度某些函数在未来某个时间发生,比如推迟执行的磁盘 I/O 操作等。

1.1 实时时钟

内核必须借助硬件的帮助才能管理时间。实时时钟(real time clock)是用来持久存放系统时间的设备,它与 CMOS 集成在一起,并通过主板电池供电,所以即便在关闭计算机系统之后,实时时钟仍能继续工作。

系统启动时,内核读取实时时钟,将所读的时间存放在变量 xtime 中作为墙上时间(wall time), xtime 保存着

从 1970 年 1 月 1 日 0:00 到当前时刻所经历的秒数。虽然在 Intel x86 机器上,内核会周期性地当前时间存入实时时钟中,但应该明确,实时时钟的主要作用就是在启动时初始化墙上时间 xtime。

1.2 系统定时器与动态定时器

周期性发生的事件都是由系统定时器(system timer)驱动。在 x86 体系结构上,系统定时器通常是一种可编程硬件芯片(如 8254 CMOS 芯片),又称为可编程间隔定时器(PIT, Programmable Interval Timer),其产生的中断就是时钟中断(timer interrupt)。时钟中断对应的中断处理程序负责更新系统时间和执行周期性运行的任务。系统定时器的频率称为节拍率(tick rate),在内核中表示为 Hz。Hz 的大小与系统结构相关,内核在文件 <asm/param.h>中定义了 Hz 的值:

```
# define Hz 1000 /* Internal kernel timer frequency */
```

以 x86 为例,在 2.4 之前的内核中其大小是 100;从内核 2.6 开始, Hz = 1000,也就是说每秒钟时钟中断发生 1000 次。这一变化使得系统定时器的精度(resolution)由 10ms 提高到 1ms,这大大提高了系统对于时间驱动事件调度的准确性。但过于频繁的时钟中断不可避免地增加了系统开销(overhead),总的来说,在现代计算机系统上,

收稿日期:2005-08-01

作者简介: 周 鹏(1981-),男,四川仁寿人,硕士研究生,从事计算机软件与理论的研究;周明天,博士生导师,教授,研究方向为网络计算、网络安全技术。

Hz = 1000 不会导致难以接受的系统开销^[1]。

与系统定时器相对的是动态定时器(dynamic timer),它是调度事件在未来某个时刻发生的机制。内核可以动态地创建或销毁动态定时器。

系统定时器及其中断处理程序是内核管理机制的中枢,下面是一些利用系统定时器周期执行的工作:

- * 更新系统运行时间(uptime);
- * 更新当前墙上时间(wall time);
- * 在对称多处理器系统(SMP)上,均衡调度各处理器上的运行队列;
- * 检查当前进程的是否用完了时间片(time slice),如用尽,则进行重新调度;
- * 运行超时的动态定时器;
- * 更新资源消耗和处理器时间的统计值。

内核动态定时器依赖于系统时钟中断,因为只有当系统时钟中断发生后内核才会去检查当前是否有超时的动态定时器。

为精确获取当前墙上时间 xtime,需要利用硬件时钟源(TSC, ACPI-PM 等)计算[jiffies, jiffies + 1)之间所流逝的时间。这些硬件时钟源提供了从最近一次时钟中断到当前时刻的 ΔT ,内核通过结构 timer_opts 中的 get_offset()获取该值:

```
struct timer_opts{
    char * name;
    int (* init)(char * override);
    void (* mark_offset)(void);
    unsigned long (* get_offset)(void);
    unsigned long long (* monotonic_clock)(void);
    void (* delay)(unsigned long);
};
```

每次时钟中断处理程序执行时调用 mark_offset()记录 ΔT ,以后内核就可以调用 get_offset()获得 ΔT 。

1.3 内核中的全局变量 jiffies

全局变量 jiffies 用来记录自系统启动以来产生的节拍(tick)总数,系统启动时将 jiffies 初始化为 0。此后,每次系统时钟中断处理程序都会将 jiffies 加一;jiffies 在一秒内增加的值就是 Hz。系统运行时间就可以表示为 jiffies/Hz 秒。jiffies 在内核文件<linux/jiffies.h>中定义:

```
extern unsigned long volatile jiffies;
```

1.4 其他的时钟资源

x86 体系结构中的时钟资源还包括 CPU 本地 APIC(local Advanced Programmable Interrupt Controller)中的定时器和时间戳计时器 TSC(timestamp counter)。高精度定时器将使用 CPU 本地 APIC 作为高精度定时中断源。

2 高精度定时器的设计与实现

x86 体系结构中,内核 2.6.x 的 Hz = 1000,即系统时钟中断执行粒度为 1ms,这意味着系统中周期事情最快为

1ms 执行一次,而不可能有更好的精度。动态定时器随时都可能超时,但由于只有在系统时钟中断到来时内核才会检查,执行超时的动态定时器,所以动态定时器的平均误差大约为半个系统时钟中断周期(即 0.5ms)。

对于实时要求较高的电信应用来说,普通 Linux 在实时性方面^[2]与电信平台的要求之间还存在一定的差距。CGL 为增强 Linux 的软实时能力,在以下方面对内核进行了改进:提供高精度的动态定时器;提供可抢占式内核(preemption kernel)等^[3]。下面主要介绍该高精度定时器的设计思想及实现,该实现遵循 POSIX 1003.1b 中时钟和定时器相关 API 标准^[4],方便应用程序开发人员使用。

高精度定时器的基本设计思想为:用(jiffies + sub_jiffie)表示动态定时器的超时时间,PIT 仍然按频率 Hz = 1000 产生系统时钟中断。如果在一个时钟中断 tick 与下一个时钟中断(tick + 1)之间,即[jiffies, jiffies + 1)之间,有高精度动态定时器等待处理(其超时时间表示为(jiffies + sub_jiffie), sub_jiffies < 1),那么用最近的动态定时器的超时值 sub_jiffie 对硬件定时器(PIT 或 local APIC)进行设定,使其在时刻(jiffies + sub_jiffie)产生中断,通知内核对该高精度定时器进行处理。而不必总是等到系统时钟中断到来后才检查执行所有超时的定时器,从而达到提高动态定时器精度的目的。

2.1 高精度定时器的中断源

高精度定时器内核中,仍然使用可编程间隔定时器 PIT 产生每秒 Hz 次的系统时钟中断。对于采用哪种硬件定时器产生高精度动态定时器中断则取决于 CPU 上是否有本地 APIC。若 CPU 上没有本地 APIC,那么仍可使用 PIT 产生高精度定时中断,虽然这种让 PIT 既产生系统时钟中断又产生高精度定时器中断的做法效率不高。

下面两个函数是调度系统时钟中断和高精度定时器中断的核心函数:

```
schedule_jiffies_int() //在下一个 jiffies 产生时钟中断(中断源为 PIT)
```

```
schedule_hr_timer_int() //在指定的时刻产生高精度定时中断(中断源为 local APIC 或 PIT)
```

2.2 获取高精度定时器的发生时间

高精度定时器发生在连续两个 jiffies 之间(即时刻(jiffies + sub_jiffie)),要确定其产生时间,就必须确定 sub_jiffie 的大小。通过函数 get_arch_cycles(ref_jiffies)可获取 sub_jiffie 的值,sub_jiffie 以 CPU 时钟周期为最小计时单位。函数具体实现思想是,通过访问计数器 TSC,计算从上一个 jiffies 到当前时刻 ref_jiffies 之间的 TSC 差值,最终确定 sub_jiffie 的大小。

2.3 动态定时器链表的处理

所有动态定时器都以链表的形式存放在一起。为了寻找超时的定时器而遍历整个链表的做法是低效的;同样,将链表以超时时间进行排序也不是一种很高效的做法,这样做使得在链表中插入和删除定时器的开销较大。

2.6.x 内核将定时器按它们的超时值划分为 5 组^[5]。当定时器超时值接近时,定时器将随组一起下移。采用这种分组定时器的方法能尽可能减少内核搜索超时定时器所带来的开销。但由于高精度定时器以(jiffies + sub-jiffie)表示动态定时器的超时值,所以需要修改当前 2.6.x 内核中动态定时器链表的结构,这在一定程度上不可避免地降低了原 2.6.x 定时器管理的高效性。为尽可能地减小这种影响,高精度定时器链表采用 hash 表结构,将所有具体相同 jiffies 值的定时器链在同一个 hash 表项上,这些定时器再根据 sub-jiffie 的大小进行排序,如图 1 所示。

(R) Xeon(TM) 1.60GHz 处理器,内存大小为 2GB;内核中高精度定时器中断源为本地 APIC;并在内核中将定时器精度配置为 10 μ s。

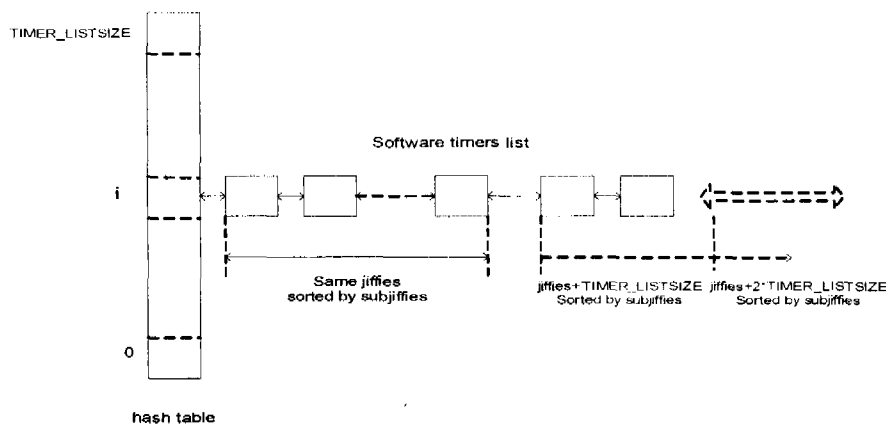


图 1 定时器 hash 表结构

每次按下列步骤处理定时器链表:

- 1) 获取当前时间(jiffies + sub-jiffie);
- 2) 获得链表上下一个定时器;
- 3) 若该定时器已超时,调用其中断处理函数并返回 2);
- 4) 若该定时器是高精度定时器,则调用 schedule_hr_timer_int();
- 5) 若在当前 jiffies 到 jiffies + 1 之间没有高精度定时器,则调用 schedule_jiffies_int()。

该 hash 表结构的设计在一定程度上减少了改变 2.6.x 中定时器链表结构所带来的系统开销。

3 高精度定时器的性能测试

该高精度定时器的实现为用户提供了遵循 POSIX1003.1b API 标准的函数接口^[4]:

clock_gettime(), clock_gettime(), clock_settime(), clock_nanosleep();

timer_create(), timer_delete(), timer_gettime(), timer_settime(), timer_getoverrun();

下面将给出基于内核 2.6.10 的高精度定时器的相关测试结果。测试硬件环境为双 Intel

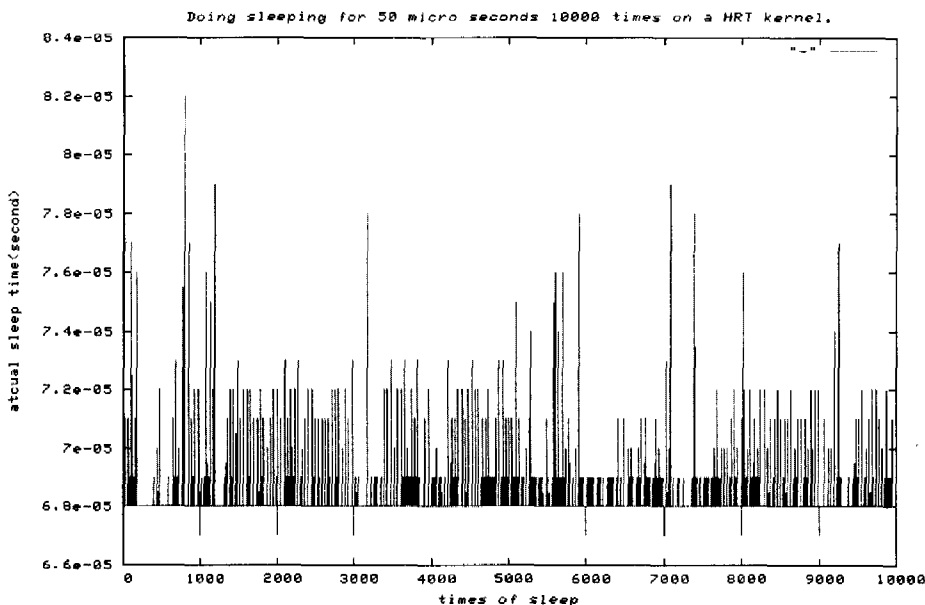


图 2 在高精度定时器内核上睡眠 50 微秒 10000 次

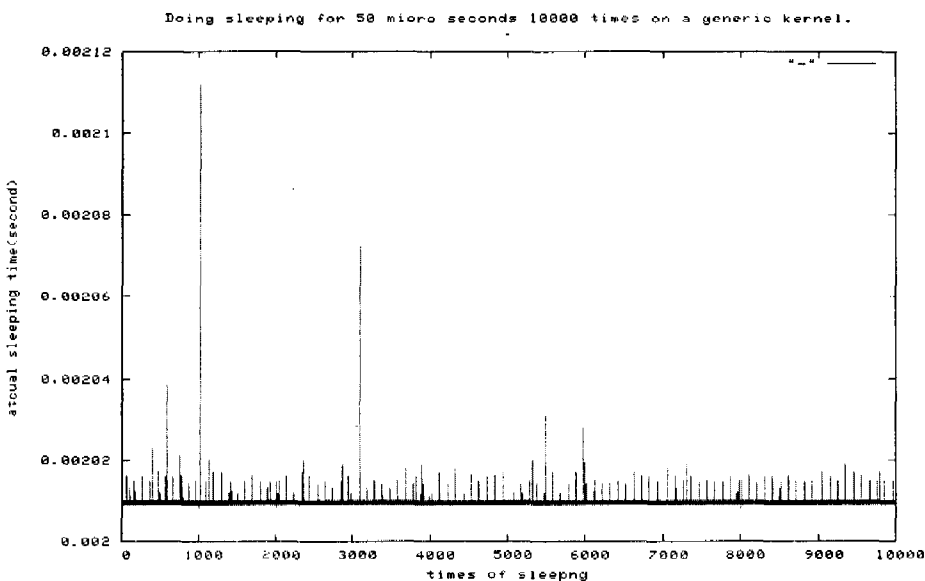


图 3 在普通内核上睡眠 50 微秒 10000 次

(下转第 78 页)

为了更清楚地理解决策树的知识表示,可以把它转化成规则的形式,如下:

If(道岔信息 = True) then (非故障)
If(道岔信息 = False) and (转向架水平加速度方差 $\leq a1$) then (非故障)
If(道岔信息 = False) and (转向架水平加速度方差 $> a1$) and (车体水平加速度方差 $\leq b1$) then (非故障)
If(道岔信息 = False) and (转向架水平加速度方差 $> a1$) and (车体水平加速度方差 $> b1$) and (车体垂直加速度方差 $\leq c1$) then (非故障)
If(道岔信息 = False) and (转向架水平加速度方差 $> a1$) and (车体水平加速度方差 $> b1$) and (车体垂直加速度方差 $> c1$) then (轨道故障)

图 2 中 $a1, b1, c1$ 分别表示转向架水平加速度方差的分类门限、车体水平加速度方差的分类门限以及车体垂直加速度方差的分类门限;省略号表示在试验过程中,不断完善加入新的属性以及类别时的决策树结构。

应用决策树进行轨道故障判决流程如图 3 所示。

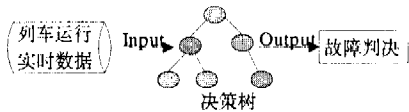


图 3 决策树判决流程

1.3 实例分析

现以列车在沈大线上运行采集的数据为例,对算法进行验证,试验结果如表 3 所示,通过统计判决正确概率基本达到预期效果。本文方法中属性主要还只是考虑了 3 个加速度的方差值,类别也只是分了轨道故障和无故障两类。在以后的试验以及工作中根据实际遇到的情况,可以加入更多的相关属性以及可以更加详细地对故障分类,使得故障类别更加清楚,判决更趋于合理化,同时判决的正

确概率也得到提高。

表 3 试验结果

里程 (m)	速度 (km/h)	车体垂直加速度方差	车体水平加速度方差	转向架水平加速度方差	道岔信息	...	判决结果
5793	85	0.079934	0.077401	0.520618	false	...	轨道故障
7152	91	0.099115	0.080663	0.392309	false	...	轨道故障
8636	83	0.113714	0.106451	0.362360	true	...	无故障
9266	84	0.076295	0.059881	0.282721	false	...	无故障
10100	86	0.064951	0.076098	0.440309	false	...	无故障
31458	88	0.092874	0.084715	0.379678	false	...	轨道故障
...

2 结束语

决策树在市场划分、金融风险、产品开发以及故障诊断中已经得到了比较广泛的应用。C4.5 算法是决策树的一个经典算法,文中把 C4.5 算法应用到列车轨道故障的判决中,通过对样本数据的学习训练生成决策树,根据生成的决策树来对未知的输入数据进行决策,实现故障检测的自动化和智能化,具有广阔的应用前景。

参考文献:

[1] 唐桂峰,李 宁,陈世福.高速公路路面破损智能识别系统的设计与实现[J].计算机科学,2004,31:12-15.
[2] Quinlan J R. C4.5 Programs for Machine Learning[M]. San Mateo:Morgan Kaufmann Publishers, Inc,1993.
[3] 陈文伟,黄金才.数据挖掘技术[M].北京:北京工业大学出版社,2002.23-25.
[4] 唐海兵,秦怀青.利用决策树改进基于特征的人侵检测系统[J].微机发展,2005,15(4):102-105.
[5] 朱 明.数据挖掘[M].合肥:中国科学技术大学出版社,2002.67-72.

(上接第 75 页)

表 1 列出了在高精度定时器内核和普通 2.6.10 内核上睡眠不同时间(从 100ns 到 4s)的平均延迟,可以看出平均延迟时间有效地从毫秒级减小到 10 微秒级。

表 1 平均睡眠延迟对比

高精度定时器内核 (精度 = 10 微秒)	普通内核 (精度 = 1 毫秒 = 1000 微秒)
24934 纳秒	1996892 纳秒

图 2、图 3 分别绘制了在普通内核和高精度定时器内核上睡眠 50 μ s10000 次所测得的实际睡眠时间。与表 1 结果相似,50 μ s 的实际睡眠时间从(2.008~2.112)ms 降低到(67~83) μ s。

4 结 论

高精度定时器用(jiffies + sub_jiffie)表示定时器超时值,用本地 APIC 或 PIT 作为高精度定时器中断源,有效地提高了内核动态定时器的精度,加强了 Linux 在电信级应用的软实时能力,符合 CGL3.0^[3]的要求,有利于将电

信应用平台向 Linux 移植。

参考文献:

[1] Love R. Linux Kernel Development[M]. USA: Sams Publishing, 2004.
[2] Bover D P, Cesati M. Understanding the Linux Kernel (2nd edition)[M]. [s.l.]: O'Reilly Publishing, 2002.
[3] Abeni L, Goel A, Krasic C, et al. A Measurement - Based Analysis of the Real - Time Performance of Linux[A]. Proceedings of the Eighth IEEE Real - Time and Embedded Technology and Applications Symposium (RTAS'02)[C]. Washington, DC, USA: IEEE Computer Society, 2002. 133 - 142.
[4] IEEE Std 1003.1 - 2001, System Interfaces, Issue 6 - for descriptions of interfaces of POSIX clocks and timers[EB/OL]. <http://standards.ieee.org/catalog/olis/posix.html>, 2001.
[5] Open Source Development Labs. Carrier grade Linux requirements definition (Version 3.0)[EB/OL]. <http://www.osdl.org/docs/cgl-perf-req-def-30.pdf>, 2005.