

基于多核 PC 的 Linux 系统实时性改造

刘 胜, 王丽芳, 蒋泽军

(西北工业大学 计算机学院, 陕西 西安 710129)

摘 要: 分析了 Linux 系统所存在的非实时因素, 提出一种基于多核 PC 的 Linux 实时性改造方案. 方案从以下几个方面改进 Linux 的实时性: 实现满足实时调度要求的高精度定时机制; 重新分配系统处理器资源; 指定分发系统外部中断以减少中断延迟对实时任务的影响. 以内核可加载模块的方式实现此方案, 保证了实时模块和原有系统之间的相互独立, 同时也便于移植和维护. 经过测试表明所提出的方案能够很好地提高 Linux 系统的实时性.

关键词: Linux 实时性; 高精度定时; 指定分发中断; 内核可加载模块

中图分类号: TP391

文献标识码: A

文章编号: 1000-7180(2013)08-0120-04

Reconstruct Real-time of Linux Based on Multicore Computer

LIU Sheng, WANG Li-fang, JIANG Ze-jun

(School of Computer College, Northwestern Polytechnical University, Xi'an 710129, China)

Abstract: This article analyzes the non-real-time presence of Linux system, and proposes a Linux real-time rehabilitation program based on the multi-core PC. The program improves the real-time nature of the Linux through the following aspects. Firstly, the program achieves high-precision timing mechanism to meet the requirements of real-time scheduling; Secondly, it reallocates system processor resources; And finally it distributes system external interrupts to reduce the interrupt latency of real-time tasks. The program ensures the flexibility and ease of maintenance of the real-time module through kernel loadable module. The test shows that the proposed program can improve the real-time performance of the Linux system.

Key words: Linux real-time rehabilitation; high-precision timing; distributes system interrupts; kernel loadable module

1 引言

对运行在 PC 上的 Linux 系统的实时性进行分析可以得知, Linux 系统本身支持实时任务^[1-2], 然而其实时性却受到很大的制约, 很难满足对实时性要求较高的场合, 究其原因, 在 Linux 系统中存在很多因素影响系统实时性. 这些非实时性因素主要体现在: 系统时钟精度比较低导致调度周期很长; 非实时任务对 CPU 资源的抢占导致不能满足实时任务调度的确定性要求; 不确定的中断延迟导致实时任务得不到及时调度. 这三个因素的存在, 使得 Linux 实时任务得不到合理的调度, 不能按时完成, 从而保证不了实时性.

2 Linux 实时化改造方案分析

通过学习现有的比较著名的 Linux 系统实时化改造方案, 可以得知对 Linux 系统的实时化改造, 从实现途径上来讲, 目前主要有以下两种方式: ①直接对标准的 Linux 内核进行修改, 增强其实施性能; ②对标准的 Linux 内核进行外部实时性扩展, 以内核可加载模块的方式实现一个实时微内核, 将所有实时任务都交由实时内核来处理.

对标准的 Linux 内核进行外部扩展主要是使用 Linux 内核可加载模块的方式实现的. 使用可加载模块的方式实现相对于直接修改 Linux 内核的方式实现, 由于其没有对原有系统做出改动, 所以保证了

原有系统内核的独立性和稳定性。同时,使用可加载模块的方式实现,可以方便地加载和卸载实时微内核,保证了实时模块本身的灵活性和可移植性,维护起来也更加方便,实现起来也更加简单。其次,使用可加载模块的方式实现不影响Linux内核本身的版本升级和功能完善,可以让系统用户体验到Linux系统在不断完善的过程中出现的新的功能特性。因此,本文采用Linux内核可加载模块的方式实现Linux实时性扩展。

3 高精度定时方案

随着Linux内核版本的提高,Linux内核结构和功能日臻完善,也出现了一些新的功能特性,例如:Linux内核的可抢占性增大,Linux系统的时间机制中使用了Tickless机制,Linux内核调度算法的改进。这些新的特性为Linux系统实时化改造提供了诸多便利。尤其是在较新版本Linux系统中实现的Tickless机制将为解决实时系统高精度定时的的问题提供更加简单有效的途径。

所谓的Tickless机制^[3]是相对于传统的tick机制而言,针对传统tick机制的缺陷实现的一种新的时间机制。其旨在省略掉系统内核不必要的时钟中断,使时钟中断更具有实际意义并且在精准的时间点上产生中断来驱动系统中各时钟模块。从Linux 2.6.16版本的内核开始,Linux开始革命性地改进其时间系统。Tickless机制被采用和实现,其主要思想是基于实时操作系统kurt-Linux实现的UTIME定时机制发展而来的。

通过对Tickless机制的分析,可以得知在条件满足的情况下,Linux系统在启动之初就会尝试从原有的周期性定时机制切换到高精度模式,并且使用Hrtimer^[4]来模拟原有的tick,这个条件首先是硬件性能的支持,如今的大多数计算机的硬件配置都满足高精度定时要求。

Hrtimer相对于原有的动态定时器,具有非常高的精度,并且Hrtimer的事件处理函数可以选择在硬件中断的环境下执行,这一方面保证了时间精度,另一方面也保证了有更高的执行效率。可以设想使用Hrtimer来实现一种满足于实时调度的时钟机制。通过在普通配置的PC机上测试,发现Hrtimer的定时粒度最小都可以达到数微秒。通常来讲,实时系统的定时粒度^[5]需要在100 μs 以内,因此,使用Hrtimer可以达到实时系统的定时要求。并且使用基于Hrtimer来实现实时定时机制,可以避免传统

实时系统中周期性定时所带来很多额外系统开销,可以根据实时调度的需求,在特定的时刻发出时钟中断。

使用Hrtimer作为实时调度的时钟中断还是存在一定的定时延迟,其原因在于Hrtimer自身的处理过程所造成的时间开销,通过测试可以发现,这个定时延迟在多次定时中都比较稳定。因此,可以通过对定时延迟的统计得到一个平均的定时延迟,在每一次设定定时器期限的时候都减去这个延迟就能得到一个非常理想的定时精度。由于Linux内核在高精度模式下时钟中断信号也由Hrtimer提供,所以在CPU资源进行分区管理(下文将提到)以后可以通过调用内核提供的tick_cancel_sched_timer函数,将实时CPU核的自身的时钟定时器sched_timer停掉。这样一方面降低系统不必要的开销,另一方面也便于基于这种定时机制来实现自己的实时调度器^[6]。

4 处理器资源分区管理

在Linux系统中每个CPU都维持有各自的runqueue、寄存器、IDT等,都可以单独进行任务的调度、挂起、唤醒等操作。所以,从PC上独立出一个实时CPU只用来运行实时任务,剩余的非实时CPU资源用来执行Linux的非实时任务,从而可以确保非实时任务不影响实时任务。系统的CPU资源重新分配后,即使非实时CPU的负载非常大,用来运行实时任务的CPU都是空负载,这也就是前面所说的独立性,从而能够保证当有实时任务到来时,立即给它分配实时CPU资源,使实时任务能够立即得到执行,并可以独占实时CPU运行,直至结束。

在多核计算机中,进程有一个很重要的特性就是可以指定其在哪一个CPU子集上运行,把进程能够指定其在某一个CPU子集上执行的特性称为进程的亲缘性(affinity)。Linux系统中所有的进程都有一个相关的结构task_struct来标识,在这个结构中有一个成员cpus_allowed表征的位掩码用来指定该进程本身可以运行的CPU子集。这个位掩码由n位组成,与系统中n个逻辑处理器一一对应,如果为给定的进程设置了给定的位,那么这个进程就可以在相关的CPU上运行。在缺省情况下所有的位掩码都是1,表示进程可以在任何CPU上运行,并且能够根据需要在处理器之间迁移。

Linux内核调度器本身就具有称为软亲缘性的

特性,这意味着进程通常不会在处理器之间频繁的迁移,有助于减少因为进程频繁迁移产生的负载.具有软亲缘性的进程,系统可根据需要均衡不同处理器之间的负载而将任务从繁忙的处理器迁移到空闲的处理器.这种机制称为 Linux 系统的负载均衡机制.而设置了硬亲缘性的进程则只能在指定的 CPU 子集内进行迁移.在 Linux 内核中提供了设置任务硬亲缘性的函数 `sched_setaffinity`.使用亲缘性设置函数可以根据进程的 `pid` 来设置进程可以运行的 CPU 的掩码,从而让进程运行在指定的 CPU 上.

对于线程来说,在 Linux 系统中线程是以轻量级进程来实现的,因此对于用户态创建的线程在系统中都有其对应的 `task_struct` 结构,通过找到线程对应的 `task_struct` 结构就可以得到其进程号,从而设置亲缘性. Linux 内核中提供了宏 `for_each_process` 宏用来遍历系统中所有的进程,在本实现方案中,可以使用此宏来将系统中所有非实时任务绑定到非实时 CPU 核上,而将实时任务绑定到实时 CPU 核上.在对系统 CPU 资源进行重新分配后,保证了实时任务可以独占实时 CPU 核的运行,相对于未分配 CPU 资源的 PC 系统来说提高了实时性.

在 Linux 系统中每一个 CPU 都维护有各自的运行队列 `runqueue`,对于运行在实时 CPU 核上的实时任务,可以通过使用内核函数 `dequeue_task` 和 `enqueue_task` 来将实时任务移入和移除当前 CPU 的运行队列 `runqueue`,从而实现对实时任务的调度控制.需要注意的是,在 Linux 内核中指定 CPU 的 `runqueue` 并不可以直接使用,需要使用 `__per_cpu_offset[cpu]` 来得到指定 CPU 的 `per-cpu` 变量的偏移量加上 `runqueues` 的地址来得到,而 `runqueues` 的地址也需要使用 `kallsyms_lookup_name("runqueues")` 来查找内核符号表相应的表项来得到.(注:在前文提到的内核函数以及下文提到的内核函数大多数也并不是可以直接使用的内核 API,需要通过查找内核符号表 `kallsyms` 来得到函数地址再使用)

5 中断机制的实时化改造

在现今的多核计算机的中断系统大都采用高级可编程控制器 APIC^[7].每个 CPU 都有自己的 LocalAPIC,并且还有一个 I/O APIC 是安装在主板上的,由 I/O APIC 和 LocalAPIC 组成的多级 APIC 系统构成了计算机的中断处理系统,I/O APIC 在中断处理过程中主要用于接受外部设备发出的中断

信号,并且把中断信号分发给各个 CPU 的 LocalAPIC.

类似与上文中系统进程亲缘性的绑定, Linux 系统也支持中断亲缘性的绑定,在内核态下使用 `irq_set_affinity` 函数来设置某一个中断号的中断亲缘性.然而这种亲缘性的绑定只能针对于某一个中断号的中断,由于中断号可能由于外部设备的不固定增加、减少或者变动,不能很方便地实现对系统外部中断的完全绑定.

一种更直接的方法是对计算机的 I/OAPIC 进行编程.通过对 I/OAPIC 的相应寄存器进行编程从而完成将外部中断分发到指定的 CPU 上.通过参考 Intel 用户手册^[8]可以知道,I/O APIC 具有对 24 个 64 位的寄存器,对应保存 24 根中断请求信号线的中断重定向表.这 24 个中断重定向表的每一项都可以连接外部中断并且实现中断“路由”功能,其中的 56~63 位表示把此表项对应的中断发送到哪个 CPU 进行处理,这八位表示的是最多 8 个 CPU 的掩码,通过对这 8 位其中的某一位置 1 即标识中断可以分发到该 CPU 上,置 0 标识中断不会分发到该 CPU 上.以双核计算机为例,将第 56 位置为 1,而将 57~63 位都置为 0 即可以实现将系统中所有外部中断都转发到 CPU0 上去(这里我们假定 CPU0 为非实时 CPU 核),而 CPU1(实时 CPU 核)上运行的实时任务将不会受到外部中断的影响.

对 I/O APIC 重定向表的编程也就是对 I/O APIC 的 24 个寄存器读写操作,在 Linux 系统中这 24 个寄存器的基地址为 0XFE00000,此外在 I/OAPIC 中还有一个选择寄存器用于选择这 24 个寄存器进行读写操作,结合这两步操作就可以完成对 I/OAPIC 中断重定向表进行编程.在具体的实现中,由于这个地址是物理地址,并不能直接对其进行读写操作,需要将其映射到内存中进行操作.通过查阅 Linux 内核源代码,可以发现内核中实现了 `io_apic_write` 和 `io_apic_read` 两个函数,其本质上也是将 I/OAPIC 寄存器映射到内存再进行读写操作,然而这两个函数并没有导出(也没有出现在内核符号表中),因此无法直接使用.可以仿照内核的代码实现,来实现自己的读写函数,这样就可以实现对 Linux 系统中的外部中断指定分发.

6 Linux 改造后的实时性测试

为了验证上述实时性改进方案的有效性,对方案中高精度定时机制定时误差进行测试,可以看出

提出的高精度定时机制在进行定时误差优化以后能够达到很高的定时精度(图中定时误差不超过 400 ns)能够很好地满足实时调度的定时要求。

其次,验证处理器资源重新分配以后实时任务的执行情况可以看出在对 CPU 资源重新分配以后,实时任务的执行时间相对于处理器资源未分配时有了较大的差距,表明了对 CPU 资源重新分配能有效提高 Linux 系统实时性。

最后,验证方案中对系统外部中断进行指定分发处理以后实时任务的执行情况可以看出对外部中断指定分发以后,实时任务的执行时间相对于外部中断未指定分发时有了较大的差距,表明了对外部中断进行指定分发能有效提高 Linux 系统实时性。

7 结束语

本文从制约 Linux 系统实时性因素出发,从提高时钟精度、对处理器资源进行重新分配以及对中断进行指定分发三个方面来提高 Linux 系统的实时性。通过实验分别验证了此方案中所做出的改进对 Linux 的实时性有显著性的提高。以此方案为基础在实现一个更加简单的实时调度器运行在实时 CPU 核,采用合适的实时调度算法,并且再以内存映射机制为基础实现一套实时化内存管理机制就能实现一个完善的基于多核 PC 的 Linux 实时系统,这将是进一步研究的目标。

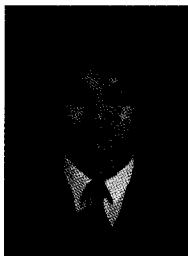
参考文献:

- [1] Daniel P Bovet, Marco Cesati. 深入理解 Linux 内核

[M]. 3 版. 北京:中国电力出版社,2007.

- [2] Robert Love Linux 内核设计与实现[M]. 2 版. 北京:机械工业出版社,2006.
- [3] Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman. Linux 设备驱动程序[M]. 3 版. 北京:中国电力出版社,2006.
- [4] 秋铁,周玉,邓莹莹. Linux 内核 API 完全参考手册[M]. 北京:机械工业出版社,2011.
- [5] 李小群,赵慧斌,叶以民,等. 一种基于时钟粒度细化的 Linux 实时化方案[D]. 北京:中国科学院软件研究所,2003.
- [6] 赖娟. Linux 内核分析及实时性改造[D]. 成都:电子科技大学,2007.
- [7] Vaidyanathan Srinivasan, Gautham R Shenoy, Srivatsa Vaddagiri, et al. energy-aware task and interrupt management in Linux[C]//proceedings of the Linux Symposium. Ottawa, ON, Canada: Linux Symposium, 2008:187-198.
- [8] Intel. Intel Corporation IA-32 Intel Architecture Software Developer's Manual Volume 3: System Programming Guide[M]. USA: Intel Corporation, 2006.

作者简介:



刘胜男, (1987—), 硕士研究生. 研究方向操作系统内核实时化改造、嵌入式计算及其应用。

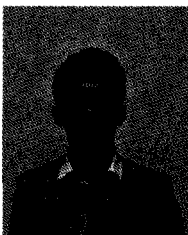
(上接第 119 页)

参考文献:

- [1] 潘汉. 基于 Windows 2000 的实时扩展研究[D]. 西安:西安电子科技大学, 2009.
- [2] 廖小勇. 基于 PC/Windows 环境的实时系统研究与实现技术[D]. 上海:同济大学, 2002.
- [3] Intel. Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3A: System Programming Guide, Part 1[M]. US: Intel Corporation, 2006.
- [4] Intel. 82093AA I/O Advanced Programmable Interrupt Controller (IOAPIC) [M]. US: Intel Corporation, 1996.
- [5] 张银奎. 软件调试[M]. 北京:电子工业出版社, 2008.
- [6] 谭文, 杨潇, 邵坚磊. 寒江独钓: Windows 内核安全编程[M]. 北京:电子工业出版社, 2009.

- [7] 张帆, 史彩成. Windows 驱动开发技术详解[M]. 北京:电子工业出版社, 2008.
- [8] 毛德操. Windows 内核情景分析—采用开源代码 ReactOS[M]. 北京:电子工业出版社, 2010.

作者简介:



蒋善锋男, (1989—), 硕士研究生. 研究方向为操作系统内核实时化改造、嵌入式计算及其应用。