文章编号:1000 - 582X(2005)03 - 0061 - 04

Linux 系统实时性探讨*

蒋 溢,李琳皓,陈 龙,熊安萍
(重庆邮电学院 计算机科学与技术学院,重庆 400065)

摘 要:标准 Linux 是典型的分时系统,具有较差的实时性。随着 Linux 操作系统在实时应用领域的不断扩展,增强 Linux 系统的实时性变得尤为必要。以分析 Linux 内核调度策略及算法为出发点,从整个 Linux 系统的角度讨论了几种不同的增强 Linux 系统实时性的方法,并总结了当前 Linux 系统实时性研究的方向。

关键词:Linux;实时性;调度策略;调度算法
中图分类号:TP316.8 **文献标识码**:A

标准 Linux 是一个多任务多用户的分时操作系统,尽管系统通过为实时任务赋予较高的优先级使得系统具有一定的实时性,但对于大多数时限要求较高的实时任务来说,远远达不到其对于时间约束的要求。而 Linux 是一个开源的、廉价的、性能卓著的系统,为了有效地利用这个资源,越来越多的厂商及技术开发人员热衷于改善 Linux 的实时性,使其除在桌面应用外,还广泛应用于嵌入式系统,实时控制等领域。

实时任务一般可以分为软实时和硬实时两种类型。硬实时任务要求系统要确保任务执行最坏情况下的服务时间,即对于实时事件的响应时间的截止期限必须得到满足,否则会破坏系统或造成致命的错误。而软实时任务虽然也有一个与之关联的最后期限,并希望满足该时限的要求,但并不是强制的,即使过了最后期限,任务的执行仍然是有意义的,不会给系统带来致命的错误[1]。近年来,围绕增强 Linux 系统软实时性及硬实时性方面都提出了许多相应的解决方案及算法,同时也出现了各种各样各具特色的具有实时调度功能的 Linux。

1 Linux 内核的调度策略及算法

1.1 Linux 内核调度策略

为了同时支持实时和非实时两种进程,Linux 的调度策略简单讲就是优先级加上时间片。当系统中有实时进程到来时,系统赋予它最高的优先级。在 Linux 中[2],是由进程描述符中 policy 域确定调度策略的,它的取值有 3 种,分别表示不同的调度策略:SCHED_FIFO(先进先出)、SCHED_RR(轮转)、SCHED_OTHER(其它)。其中,SCHED_FIFO 及 SCHED_RR 是实时进程的调度策略,而 SCHED_OTHER 是指用于非实时进程的基于优先级的轮转策略。

1.2 Linux 内核调度算法

系统在进行调度算法实现时,会根据每个任务的 4 个与调度相关的参数:rt_priority(实时进程的静态优先级)、policy(进程调度策略)、nice(用户设定的优先级)、counter(进程本轮调度剩余时间片)加以进行。

对于普通进程,采用 SCHED_OTHER 调度策略,调度器会选择 nice + counter 值最大的进程调度执行,其中,nice 一旦由用户进程设定就不再变化,取值范围一般为(-20,19),它代表了该进程的优先级,也代表该进程在每一个调度周期中能够得到的时间片的多少;而 counter 则是一个可以动态变化的值,它反映了一个进程在当前的调度周期中还剩下的时间片。在每一个调度周期的开始,nice 的值被赋给 counter,在每个时钟周期 counter 值都减 1,当 counter 值为零时,表示该进程用完自己在本调度周期中的时间片,不再参与本调度周期的进程调度;当所有进程的时间片都用完时,一个调度周期结束,然后周而复始。由此可以看

* 收稿日期:2004 - 10 - 16
基金项目:教育部科学技术研究重点项目(F2003 - 02);重庆市科技攻关计划项目(7220 - B - 21);重庆市教委科学技术研究项目(020504)
作者简介:蒋溢(1969 -),男,湖北孝感市人,重庆邮电学院工程师,主要从事计算机应用技术的研究。

出 Linux 系统中的调度周期是动态变化的,SCHED_OTHER 调度策略本质上是一种比例共享的调度策略 SD[3]。

对于实时进程,系统使用基于实时优先级 rt_priority 的调度策略,直接将实时进程的优先权值提高到高于所有的非实时进程。对于实时进程的调度,如果采用 SCHED_FIFO 策略,实时进程就没有时间片的概念,其调度只取决于实时进程到来的先后;如果采用 SCHED_RR 策略,与 SCHED_FIFO 调度是一样的,但它存在时间片,一个正在执行的实时进程所分配的时间片到了,那么该进程就放弃执行,时间片的长度可以通过 sched_rr_get_interval 调用得到。

从 Linux 的调度策略及算法可以得知,Linux 系统的调度实时性是非常简单的,并且从 Linux 进程调度的时机来看[4],Linux 只能实现基于进程的抢占策略,在内核是不可抢占的,即除非进程主动放弃处理机,否则处理机工作在核心态下时是不会产生进程调度的。这样一来,Linux 不可抢占内核调度很难满足硬实时任务的要求,另外,Linux 的时钟粒度过大、屏蔽中断等问题都影响了 Linux 内核对于实时任务的响应延迟,因此,Linux 内核很难达到对于实时任务的 QoS(服务质量保证),于是出现了很多关于改善 Linux 实时性的解决方案及策略。

2 常用增强 Linux 实时性的方法

增强 Linux 系统实时性,主要从以下几个方面进行:增加 Linux 内核可抢占性、细化 Linux 系统时钟粒度、改善屏蔽中断处理、改善并增加实时调度算法。下面,就围绕这几方面所采取的一些增强 Linux 系统实时性的方法进行讨论。

2.1 增强 Linux 内核抢占性

当进程运行在核心态下时,有 2 种实现 Linux 内核抢占的思想:一是完全剥夺抢占方式,另外一个是插入抢占点的方式[5]。完全剥夺方式核心思想是利用了 Linux 支持 SMP(对称多处理)的实现——允许多个进程运行在核心态,通过信号量锁或自旋锁 spin-locks 对内核数据结构存取访问进行保护的原理,实现内核的可抢占调度。插入抢占点方法是在 Linux 内核插入一些抢占点,当内核执行到抢占点,就检查是否有更高优先级的实时进程正在等待,如果有,当前执行进程将被挂起,转去执行更高优先级的实时进程;如果没有更高优先级的实时进程等待,则当前进程将继续执行。这 2 种方式从实现上来讲各有特点,完全剥夺方式可以实现任意点的内核抢占[6],但需要花费更多的系统开销(需要信号量锁及自旋锁);而插入抢占点的方式尽管有比较小的系统开销,但插入点的选择非常关键,如果设置不当,会增加系统对实时任务的调度延迟。

在实践中,有两种成熟的内核补丁——抢占式补丁及低潜伏补丁分别与上述两种实现思想相吻合。抢占式补丁是由 Monta Vista 提出的,由 Robert M. love 维护,其基本思想是产生调度机会,减少事件发生到产生调度的时延,通过修改内核代码中的 spinlock 宏及中断返回代码,使内核可被安全抢占,即如果内核不是在一个中断处理程序中,并且不再 spinlock 保护的代码中,就认为可以"安全"地进行进程切换[3]。低潜伏补丁是由 Ingo Molnar 提出,由 Andraw Morton 维护,其基本思想是在内核执行时间较长的代码内增加抢占点,从而增强 Linux 内核的抢占性[7]。这两种补丁的本质都是通过修改 Linux 内核达到目的,并且实验证明,低潜伏补丁与抢占式补丁相结合可以获得的最大调度延迟达到 1.2 μs,是降低调度延迟的最好方法。

2.2 细化 Linux 时钟粒度

标准 Linux 系统的时钟粒度为 10 ms,不能满足系统实时调度的需求,因此需要细化其时钟粒度,具体可以通过两种方式加以实现:一是通过将系统的实时时钟芯片置于单次触发模式,提供十几个微秒级的调度粒度,从而细化时钟粒度;二是通过修改 Linux 内核中 Hz 宏的定义来细化时钟粒度,如标准 Linux 中 Hz 为 100,如果将其改为 100 000,则时钟粒度可以达到 10 μs,这种方式虽然会增加许多的系统开销,但在强周期性环境下,对定时器的设置只需初始化时进行一次,这样又在一定程度上保证了处理效率[9]。

2.3 屏蔽中断的处理

由于 Linux 在中断处理及虚存管理的缺页处理过程中允许屏蔽中断,这就使得 Linux 系统实时调度的粒度比较大,为此,通常用中断抽象层方法加以解决[8],其基本思想是尽量在本质上不对 Linux 内核源代码进行修改,增加的中断抽象层可以用来截获硬件发出的中断请求,从而完全控制硬件中断,并产生模拟中断信号,发送给 Linux 内核,在系统中,有独立的调度器,Linux 内核作为优先级最低的任务被调度器调度,从而增强 Linux 系统实时调度功能。这里的中断抽象层既可以是软件实现的仿真层,也可以是硬件抽象层。

RT-Linux 是新墨西哥科技大学研制的一个实时系统,其在 Linux 系统中实现一个微内核的小的实时操作系统(也称之为 RT-Linux 的实时子系统),这个 RT-Linux 内核就是由软件实现的中断抽象层,将普通 Linux 内核作为一个该操作系统中的一个低优先级的任务来运行,普通 Linux 系统中的任务可以通过 FIFO

和实时任务进行通信。其框架如图 1 所示。

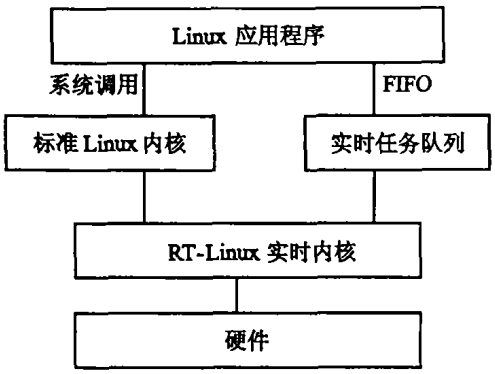

图 1　RT-Linux 结构

RT-Linux 的关键技术是通过软件来模拟硬件的中断控制器。当 Linux 系统要屏蔽 CPU 的中断时，RT-Linux 中的实时内核会截取到这个请求并记录下来，而并不真正封锁硬件中断，这样就避免了由于封中断所造成的系统在一段时间没有响应的情况，从而提高了实时性。当有硬件中断到来时，RT-Linux 截取该中断，并判断是由实时子系统中的中断例程来处理还是传递给普通的 Linux 内核进行处理[3]，不难看出，实际上 RT-Linux 实现了一种双内核机制，不过仍然需要对 Linux 内核源码进行适当的增加或修改。

RTAI 是 DIAPM 研发的由硬件抽象层实现的实时系统，它通过硬件抽象层为实时任务提供与 Linux 内核进行交互的接口，其实现思想基本与 RT-Linux 相同，当系统无实时任务执行时才允许 Linux 内核执行，只是这种方式无需对 Linux 内核进行修改，并且非常便于在实时 Linux 与标准 Linux 系统间转换[9]。

2.4　改善 Linux 内核实时调度算法及策略

传统的用于实时系统的调度算法一般有 3 种[5]：

1）TD(time driven)时间驱动的调度算法。在系统的设计阶段，在明确系统中所有的处理情况下，对于各个任务的开始、切换、以及结束时间等就事先做出明确的安排和设计，通常提供数据处理的预测性，其本身是一种设计时就确定下来的离线的静态调度算法。

2）PD(priority driven)优先级调度算法，又可分为静态及动态优先级。静态优先级算法根据应用的属性如：任务周期，用户优先级等静态地分配给进程优先级。如 RM(rate-monotonic)——频率单调调度算法，就是根据任务的执行周期长短来决定调度优先级，周期短的任务具有较高优先级。动态优先级算法根据进程的资源需求动态分配进程的优先级。许多非实时系统采用此种算法，如短作业优先调度算法。还有实时系统中常采用的 EDF(earliest deadline first)——最早时限优先算法，根据就绪队列中的各任务的 Deadline(时限)来分配优先级，Deadline 最近的任务具有最高优先级。

3）SD(shared driven)基于 CPU 使用比例的进程调度算法，让他们的执行时间和他们的权重成正比。方法有 2 种：一种是调节任务队列在调度队列队首的频率；二是根据权重分配时间片，并逐次调度队列中的进程投入运行。主要类别有：轮转法、公平共享、公平队列、彩票调度法(Lottery)等。

一般设计人员都会根据目标系统的要求选择使用相应的算法，或是以这 3 种算法为基础派生出一些新的与目标系统相适应的算法。而标准 Linux 系统中由于缺乏实时调度机制和调度算法，影响了 Linux 的实时调度性能，于是目前提出了许多较新的调度框架和调度算法用来改善 Linux 实时调度性能，如 RED-Linux 采用的通用的实时调度框架，QLinux 所采用的分层式的调度框架；一些比较新的算法如 H-SFQ 资源调度算法(Hierarchical Start-time Fair Queuing)，网络包的延迟处理技术(Lazy Receiver Processing：LRP)，Cello 磁盘调度算法等也已在现在的一些实时系统中采用[3]，这些都大大提高了基于标准 Linux 内核的实时系统的性能。

3　Linux 实时性研究的发展方向

目前关于 Linux 实时性的研究大多根据具体的目标系统进行，针对具体的应用采用一些相应的增强 Linux 实时性的方法，甚至将多种方法集中在一个系统内加以实现，如 RED-Linux 的实现基本上借鉴了以上的 4 种方法来增强 Linux 内核的实时性，并且，它还充分利用了操作系统设计中策略与机制的分离，将各种不同的调度算法集成在一起[5]，可以对调度策略灵活配置，使得该系统既适宜实时任务的调度，也可实现对非实时任务的调度。但随着基于 Linux 系统的实时应用的不断深入，除传统的工业控制及嵌入式等领域外，还被广泛应用于网络及人工智能等系统，要求系统不仅具有普通操作系统的功能，还能够在复杂环境下处理多种不同的实时任务；另外，一个实时系统的实现还与目标系统的硬件及其所要求的 QoS 密切相关，仅依靠采取前述的一些方法还远远不够，还需要对 Linux 系统实时性做进一步的研究。

关于 Linux 系统的实时性研究大致可在两个方面进行：一是在传统的单调度策略的实时系统环境下，针对具体的目标系统进一步改善系统实时调度性能，这方面已有了许多成熟的技术和调度策略及算法，实现了许多具有代表性的实时系统，如 RT-Linux、Qlinux 等。二是研究能够在多种任务或任务集共存的复杂系统中、在保证系统的 QoS 前提下，实现及增强系统的

实时调度性能,即在开放式的实时环境下研究系统的调度框架、策略、算法等。在此,开放式实时系统指的是系统中非相关的实时或非实时应用可单独进行开发和验证,并且当系统进行动态扩展时,无须做全局的实时性分析[9];也可以将开放式实时系统简单理解成能够处理各种不同的任务(实时或非实时)且在系统负载动态变化时,不影响系统对已有的实时任务的调度。目前关于开放式实时调度系统已提出了一些调度框架及调度算法,如 RED-Linux 采用的通用实时调度框架,就实现了一定程度的开放式实时调度,至于针对开放式实时系统提出的的调度算法更是多不胜举。

4 结束语

Linux 由于其源码的开放性,基于标准 Linux 内核开发具有特定应用的实时操作系统成为一个非常有效的途径,除了在嵌入式系统内的有效应用外,实时流媒体系统、一些高负载高吞吐率服务器系统,路由器节点等应用都需要实时 Linux 系统,掌握和研究增强 Linux 实时性的不同方法,有利于人们在实现一个具体的目标系统时进行有选择运用,从而达到目标系统所要求的 QoS。

参考文献:

[1] WILLIAM STALLINGS. 操作系统——内核与设计原理[M]. (第4版). 北京:电子工业出版社,2001.

[2] 李善平,陈文智. 边学边干——Linux 内核指导[M]. 杭州:浙江大学出版社,2002.

[3] 张焕强. 基本 Linux 的实时系统[EB/OL]. http://www-900. ibm. com/developerWorks/cn/linux/embed/l - real-time/index. shtml,2004 - 10 - 04.

[4] 郭玉东. Linux 操作系统机构分析[M]. 西安:西安电子科技大学出版社,2002.

[5] LIN KWEI-JAY, WANG YU-CHUNG. The Design and Implementation of Real-Time Schedulers in RED-Linux[J]. Proceedings of the IEEE, 2003,91(7):1 114 - 1 129.

[6] 徐晓磊,董兆华,吴建峰,等. Linux 抢占式内核分析[J]. 计算机工程, 2003,29(15):115 - 117.

[7] 翟鸿鸣. Linux 实时性能增强方法的研究[J]. 微机发展, 2003,13(S1):4 - 8.

[8] 齐俊生,崔杜武,黑新宏. 嵌入式 Linux 硬实时性的研究与实现[J]. 计算机应用, 2003,23(6):53 - 57.

[9] 邹勇,李明树,王青. 开放式实时系统的调度理论与方法分析[J]. 软件学报, 2003,14(1):83 - 90.

Real-time Performance of Linux System

JIANG Yi, LI Lin-hao, CHEN Long, XIONG An-ping

(Institute of Computer Science & Engineering, Chongqing University of Posts & Telecom Munication, Chongqing 400065, China)

Abstract: The standard Linux is a typical time-sharing system, it has a poor real-time performance. But according to the applications of the Linux OS are expanding on real-time domain, it becomes necessary to enhance the performance of the Linux system. This paper analyses the schedule policies and schedule algorithms of the Linux kernel, and discusses several different kinds of methods about enhancing real-time performance of the Linux system from the total Linux system. They authors summarizes the current research's developments of the Linux system.

Key words: Linux; real-time performance; schedule policy; schedule algorithm

(编辑 吕赛英)