Linux 操作系统的实时化分析*)

左天军[1] 左圆圆[2] 陈 平[1]
(西安电子科技大学软件工程研究所 西安710071)[1] (大唐电信)[2]

摘 要 随着实时操作系统的广泛应用和Linux的迅速发展,人们更加关注实时Linux的开发问题。文中,我们讨论了调度策略、内核的可重入性、中断处理以及内存管理机制等关键问题。这些问题与Linux扩展到实时操作系统密切相关。然后,我们详细分析了两个有代表性的实时Linux,即RT Linux和KURT Linux的主要实现。我们还介绍了它们自己的特性以及它们之间的基本差异。最后提出了未来的研究工作。

关键词 实时Linux,调度算法,内核的可重入性,中断处理,资源分配,可预言性

The Analysis of Extensions for Linux to a Real-time Operating System

ZUO Tian-Jun ZUO Yuan-Yuan CHEN Ping
(Institute of Software Engineering, Xidian University, Xi'an 710071)

Abstract With the wide application of real-time operating systems and the rapid development of Linux, more and more efforts are put on issues to develop real-time Linux. In this paper, we discuss the key problems, such as scheduling strategy, kernel reentry, interrupt handling, memory management etc, which are closely related to the extension of Linux to a real-time operating system. Then we analyze the main implementations of two promising real-time Linux, RT-Linux and KURT-Linux, in detail. We also present their own characteristics as well as the fundamental differences between them. The future research work is proposed at last.

Keywords Real-time Linux, Scheduling algorithm, Kernel reentry, Interrupt handling, Resource allocation, Predictability

1 引言

近年来,实时操作系统在多媒体通信、在线事务处理、生产过程控制、交通控制等各个领域得到了广泛的应用。随着不同类型的实时应用的不断增加,越来越多的开发人员开始重新思考分时多用户的通用操作系统的设计,并致力于实时操作系统的研究开发。Linux作为一个开放源码、可定制的通用操作系统,如果能够具备实时扩展功能,必将得到更加广泛的应用。

Linux操作系统的设计结构具有通用操作系统的特点,支持多用户和多进程,负责管理CPU、磁盘I/O、网络、内存等系统资源,其设计的基本原则是尽量缩短系统的平均响应时间并提高系统的吞吐量,在单位时间内尽可能多地为用户请求提供服务。用户在Linux上运行的应用程序具有不能预测的启动、阻塞和终止行为。

实时操作系统是一种能够在边界时间内提供所需级别服务的操作系统,它能够明确说明每一个系统服务运行所需的最长时间,运行在它上面的任务的行为都是可预测的。每个实时操作系统一般都是为某一类实时系统应用所设计的。实时系统可以定义为:其正确性不仅仅依赖于计算的逻辑结果而且依赖于结果产生的时间[7]。因此,对于实时系统最重要的要求就是必须具备在一个事先定义好的时间限制内对外部或内部的事件进行响应和处理的能力。实时系统可以分为硬实时(hard real-time)和软实时(soft real-time)。硬实时系统是指必须严格保证计算的时间要求。如果在截至时间之后才完成,那么计算结果没有任何价值,并且可能造成巨大的损失和灾难。例如自动化工业流水线系统、宇宙飞船和航天系统等。另一方面,软实时系统是指统计意义上的实时,能够提供高速的响应和较大的系统吞吐量,但不能保证特定的任务在指定的时间内完成。在软实时系统中,当系统在重负载的情况下允许发生错过截至时间的情况而不会造成非常大的危害。

本文讨论了Linux系统实时化存在的关键问题,分析了RT-Linux和KURT-Linux这两种有代表性的实时Linux系统,并指出进一步的研究方向。本文第2节讨论实现Linux系统实时化所必须考虑的几个关键问题,包括调度策略、内核的可重入性、中断处理以及内存管理机制。第3节详细分析RT-Linux和KURT-Linux实现实时化的主要方法、各自的特点,指出二者之间存在的基本差异。最后总结全文,提出Linux系统实时化还需要研究的问题。

2 Linux实时化的关键问题

实时操作系统的一个关键要求就是保证系统行为的可预测性。可预测性[9]意味着操作系统在运行时能够提供有界的最坏时延。具体来说,操作系统的任务(例如调度和内存管理)以及同步原语(例如信号量、信号和消息传递机制)必须具有确定的执行和阻塞时间。实时操作系统要求在系统运行的任何时刻,在任何情况下,其资源调配策略都能为争夺资源的多个实时任务合理地分配资源,使每个实时任务的实时性要求

*)基金项目:可信软件工程(413150501)。左天军 博士生,主要研究方向软件工程技术,操作系统,网络分布计算;左圆圆 硕士;陈 平 教授,博士生导师。

都能得到满足[5]。因此实时操作系统设计的一个基本原则就是要在最坏情况下，仍然能够满足每个实时任务的实时性要求[3]。

Linux 作为一个通用操作系统，其设计结构注重系统的平均性能，而不是单个任务的响应时间。实现 Linux 的实时性，需要从以下几个方面加以改进。

2.1 调度策略

在通用操作系统中，为了保证系统的整体吞吐量，任务调度策略一般采用基于优先级的抢占式调度策略，对于优先级相同的进程则采用时间片轮转调度方式，用户进程可以通过系统调用动态调整自己的优先级，操作系统也可根据情况调整某些进程的优先级。

Linux 系统的进程调度的主要依据是：当调度程序运行时，在所有处于可运行状态的进程中选择最值得运行的进程投入运行。Linux 的每一个进程都有一个 task_struct 结构，包括进程的调度策略、静态优先级、进程剩余的时间片和实时优先级，这四项是 Linux 调度程序选择进程的依据。静态优先级不随时间变化，只能由用户进行修改，它表明了在被迫和其他进程竞争 CPU 之前，该进程应该被允许执行的时间片的最大值。进程的剩余时间片相当于进程的动态优先级，只要进程占有 CPU，它将随着时间不断减小。实时优先级用于实时进程间的选择，对于非实时进程，其实时优先级为零，因此实时进程总是优先于非实时进程。

Linux 根据调度策略从整体上区分实时进程和普通进程，提供了三种调度算法。对于普通进程，Linux 采用动态优先调度，根据进程的剩余时间片进行选择。在进程运行过程中，其剩余时间片不断减少，减少为零时放弃对 CPU 的使用，当所有处于可运行状态的普通进程的时间片都用完之后才对进程剩余的时间片重新赋值。对于实时进程，Linux 采用了两种调度策略，即先来先服务调度(FIFO)和时间片轮转调度(RR)。与普通进程不同的是，衡量实时进程是否值得运行的标准不再是剩余时间片，而是实时优先级。虽然 Linux 提供了 FIFO 和 RR 这两种符合 POSIX.1b 操作系统标准所规定的实时调度程序，但这种实时是软实时，不能满足硬实时的要求。所以，必须改进 Linux 系统中以大吞吐量为目标的调度算法以适应实时应用[12]的需要，在满足实时任务的时间正确性的同时，保证系统的可预测性和可维护性。

2.2 系统内核的可重入性

在 Linux 操作系统中，核心态系统调用通常是不可重入的。当一个低优先级任务通过系统调用进入核心态运行时，在这段时间内到达的高优先级任务必须等到低优先级任务的系统调用完成回到用户态后才能得到系统调度。不可重入的内核必然带来慢速的中断响应和不可预料的操作时间，因此实时操作系统内核必须是可重入的。

2.3 中断处理

Linux 操作系统的外部中断大多是开启的，中断处理通常由设备驱动程序完成，系统将中断处理程序的优先级设定为高于任何用户进程。这种中断处理机制主要是基于通用操作系统的用户进程一般没有实时性要求，而中断处理程序与硬件设备直接交互，具有一定的实时性要求。对于实时操作系统，用户进程一般都有实时性要求，中断处理程序的优先级高于所有用户进程的设定方式将不能保证实时应用的要求，另一方面，对于发生频率不可预测的外部中断，用户进程被中断处理程序阻塞的时间开销也不可预测，从而影响了整个系统的可预测性。一种解决方法是对中断处理程序设定优先级，由调度程序对处于可运行态的进程和中断处理程序统一进行调度。另外，为了提供快速的外部中断响应时间，应尽量减少系统内核中屏蔽中断的时间，缩小内核的临界区域。

2.4 内存管理机制

在计算机系统中内存管理有两个基本的任务，第一是为系统中运行的许多实时或非实时任务提供共享的内存；第二是提供虚拟内存。虚拟内存是将系统的一部分后备的磁盘空间作为实际物理内存的扩充，在获得远大于物理内存容量的虚拟空间的同时，并未明显增大应用程序平均访问内存所需的时间。Linux 操作系统提供了虚拟内存管理机制，系统将一部分最近未用的程序代码和数据从物理内存换出到硬盘上，当需要使用时再从硬盘换回到物理内存。虚拟内存技术所带来的一个直接问题就是系统响应时间的不确定性，这在实时系统中是无法容忍的。实时操作系统要求高效的内存管理，并且内存管理开销是可预见的。一种解决方案是预先分配内存，为每个实时任务划分固定的内存区域，这种静态分配内存的方法比较适合于硬实时系统。另一种方法是在原有虚拟内存管理的基础上增加页面锁功能，可以将实时任务锁进内存。在系统管理虚拟内存时，不会将加锁的内存页面换出物理内存，这样可以提供比较好的响应速度，提高系统的可预测性。这种方法的缺点在于不能完全保障系统的可预测性，因而不适合硬实时系统。

3 两种具有代表性的实时 Linux 操作系统分析

随着 Linux 自由软件的迅速发展，实时领域的实时 Linux 操作系统开始出现。目前已有的两种具有代表性的实时 Linux 操作系统，它们分别是新墨西哥工学院的 RT-Linux 和勘萨斯大学的 KURT-Linux。这两种实时 Linux 操作系统都以稳定、高效的 Linux 内核作为基础，通过扩展而建立开放、标准、高效的多任务实时操作系统，在提供 Linux 的网络支持、图形界面和开发环境等资源的同时，也为用户提供了很好的实时能力。

3.1 RT-Linux

Real-Time Linux(RT-Linux)[2]是一个硬实时操作系统，允许实时任务、中断处理与标准的 Linux 进程运行在同一台处理机上。RT-Linux 不同于微内核和大型内核，属于实时 EXE(real-time executive)体系结构，其系统结构如图1所示。

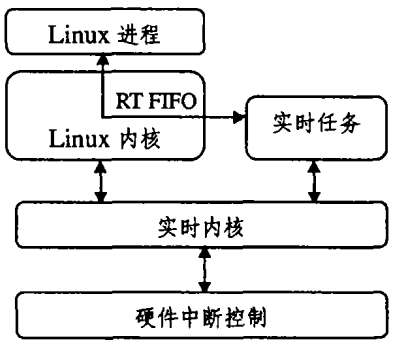

图1 RT-Linux 的系统结构

RT-Linux 实现了一个实时内核。实时内核截取所有硬件中断，并提供硬件中断在实时任务和原来的非实时内核之间的复用。原来的 Linux 内核作为一个可抢占的任务在实时内核上运行，优先级最低，随时会被高优先级任务抢占。实时任

务作为可加载的内核模块在核心地址空间运行，通过 FIFO 队列或者共享内存与非实时 Linux 任务进行通信。因此，系统在支持数据获取、系统控制等实时应用的同时，还可以作为一台标准的 Linux 工作站为用户提供服务。

RT-Linux 设计中的一个关键思想是采用软件模拟硬件中断控制。当 Linux 系统通知硬件屏蔽中断时，实时子系统截取屏蔽中断请求，记录下来之后返回 Linux。通过禁止 Linux 屏蔽硬件中断的方法，可以有效地阻止 Linux 增加实时子系统的中断响应延时。当中断到来时，RT-Linux 实时内核获取中断，既可以调用该中断的实时处理程序，也可以在中断处理程序与 Linux 共享该中断的情况下将该中断标志为挂起。当 Linux 重新打开中断时，实时内核将所有被挂起的中断发送给 Linux，调用其中断处理程序。

RT-Linux 实现的调度程序有基于优先级的调度方式和最早期限优先算法(EDF)调度[6]。基于优先级的调度使用"单调率算法"，可以直接支持周期任务。对负荷不太重的系统，RT-Linux 是一个比较实际的选择。RT-Linux 的一个不足之处在于没有对 CPU 之外的其他资源进行实时调度。在 RT-Linux 中没有一个综合的资源管理机制和 QoS 保证，这种机制在系统重负荷运行时是必不可少的[10]。

3.2 KURT-Linux

KURT-Linux[4]通过修改标准的 Linux 核心函数的方法以实现 Linux 的实时性，在这方面与 RT-Linux 有显著的不同。RT-Linux 没有对标准的 Linux 内核进行修改，而是在标准的 Linux 与硬件之间增加了一层实时内核。RT-Linux 与 KURT-Linux 之间的另一个不同之处在于 RT-Linux 设计为支持硬实时应用，而 KURT-Linux 则是为严格实时(firm real-time)应用所设计的，例如一些多媒体应用和 ATM 网络应用等。

KURT-Linux 提高了标准 Linux 的时钟分辨率。在标准 Linux 中，硬件以 10ms 为周期发出中断给 CPU。一方面，10ms 的周期频率不适合具有更细粒度的时间限制的实时应用；另一方面，仅仅依靠提高硬件时钟中断的频率将增加操作系统处理时钟中断的时间，从而导致整个系统吞吐量的下降。KURT-Linux 以一次性的方式对硬件时钟进行编程，避开了上面的这个问题。也就是说，将硬件时钟作为一个非周期的设备，只有在需要的时候，才对其进行编程，使它产生 ms 级中断。这种设计方法的代价是由于不时地对时钟芯片进行重新编程而增加了额外的开销。

KURT-Linux 核心包括两个部分：内核和实时模块。内核负责实时事件的调度，实时模块作为可加载的内核模块以内核模式运行，为用户进程提供特定的实时服务。KURT-Linux 可运行在两种状态之下，普通状态和实时状态。在普通状态下，所有进程都可以运行，但某些核心服务将带来中断屏蔽的不可预测性；在实时状态下，只允许实时进程运行。KURT-Linux 支持的调度策略包括 FIFO、轮转调度和 UNIX 分时调度。此外，KURT-Linux 实现了一种基于明确的预先定义的实时 CPU 调度策略，它要求实时应用必须明确指出实时事件发生的时间。KURT-Linux 还为基于 Linux 的 ATM 增加了对可变比特率(VBR)类型业务的支持，对磁盘子系统也进行了一定的修改以减少中断屏蔽的时间。

总结 根据以上的分析可以看出，操作系统对于 CPU 的调度、内存管理以及外设访问具有完全的控制权，而操作系统内核在操作系统中又具有独特的地位，它需要处理不同应用之间的资源竞争并进行仲裁，在预留和管理系统资源方面，操作系统内核起到了决定性的作用。因此，实现 Linux 的实时性可以根据对原有内核的不同处理分为两种方法：一种是在 Linux 内核之外，增加一个实时内核，原来的 Linux 内核作为最低级别的任务运行在实时内核上；另一种是对标准的 Linux 内核进行修改，尽可能扩充实时性，例如通过增加内核的可抢占点[11]以减少内核延时等。

对于 Linux 操作系统的实时化，需要进一步研究的问题包括在不同的系统资源之间建立一个综合的资源分配[1,8]和实时调度机制，以及系统的稳定性和容错、系统的实时性和可预测性的验证等问题。

参考文献

1 Anderson D P. Metascheduling for Continuous Media. ACM Transactions on Computer Systems, 1993, 11(3): 226～252

2 Barabanov M, Yodaiken V. Real-Time Linux. Linux Journal, Feb. 1997

3 Stankovic J A, Ramamritham K. Advances in Real-Time Systems. IEEE Computer Society Press, 1993

4 Srinivasan B, Pather S, Hill R, Ansari S, Niehaus D. A Firm Real-Time System Implementation Using Commercial Off-the-shelf Hardware and Free Software. In: Proc. of Real-Time Technology and Applications Symposium, Denver, June 1998

5 Ramamritham K, Stankovic J A. Scheduling Algorithms and Operating Systems Support for Real-Time Systems. Proceedings of the IEEE, 1994, 82(1): 55～67

6 Baruah S, Mok A, Rosier L. Preemptively scheduling hard real-time sporadic tasks on one processor. IEEE Real-Time Systems Symposium, pp. 182～190

7 Stankovic J A. Misconception about real-time computing - A serious problem for next-generation systems. IEEE Computer, 1988, 21(10)

8 Mehra A, Indiresan A, Shin K. Resource Management for Real Time Communication: Making Theory Meet Practice: [Technical Report. CSETR28196]. Computer Science and Engineering Division, The University of Michigan. Jan. 1996

9 Liedtke J, Hartig H, Hohmuth M. OS-controlled cache predictability for real-time systems. In: Third IEEE Real-time Technology and Applications Symposium (RTAS), 1997. 213～223

10 Rajkumar R, Lee C, Lehoczky J, Siewiorek D. A resource allocation model for qos management. In: Proc. of the 18th IEEE Real-Time Systems Symposium, 1997

11 Mercer C W, Tokuda H. Preemptibility in RealTime Operating Systems. In: Proc. of the 13th IEEE RealTime Systems Symposium. Dec. 1992

12 Jeffay K, Smith F D, Moorthy A, Anderson J. Proportional Share Scheduling of Operating System Services for Real Time Applications. IEEE Real Time Systems Symposium, Dec. 1998