

基于资源重分配的 Windows 实时性改造

杜旭东¹, 蒋泽军¹, 王丽芳¹, 刘志强²

(1 西北工业大学 计算机学院, 陕西 西安 710072; 2 西北工业大学 软件与微电子学院, 陕西 西安 710072)

摘 要: 分析了目前在 PC 机上实现实时系统的主要方案, 结合多核 PC 机的特性, 提出了基于多核系统改造 Windows 的实时性的方案. 方案充分利用多核的特性, 从三个方面改进 Windows 的实时性: ① 重新分配现有的系统资源; ② 为实时任务提供独立的实时时钟和 CPU 资源; ③ 减少中断延迟对实时任务的影响. 将所有的改造编写成 Windows 的内核驱动程序, 测试表明所提出的方案是可行的.

关键词: 实时系统; 中断; 系统资源; 多核; 驱动

中图分类号: TP316.86

文献标识码: A

文章编号: 1000-7180(2012)05-0095-04

Resconstruct Real-time of Windows Based on Assigning Computer Resource

DU Xu-dong¹, JIANG Ze-jun¹, WANG Li-fang¹, LIU Zhi-qiang²

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

2 School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an 710129, China)

Abstract: Analyzing the existing project of Real-time System based on common computer, and propose a project that reconstruct the real-time of Windows System based on multicore CPU. This project take advantage of multicore architecture. The project provide independent CPU, real-time clock and CPU, cut down the interrupt delay on real-time task. It finishes through programming driver. The result demonstrates that this project is feasible.

Key words: real-time system; interrupt; system resource; multicore; driver

1 引言

在工业领域中,随着实时嵌入式技术的飞速发展,Windows 系列操作系统也越来越多的被考虑作为实时系统的应用平台. 为了满足硬实时系统严格的响应时间要求,增加 Windows 系统的实时能力非常必要. 目前在 Windows 上改造实时性最流行的方法是尽可能不增加额外的硬件,在现有操作系统上来改造其实时性,充分利用现有操作系统所提供的强大功能,对多核的支持、丰富的图形用户界面(GUI)、数据分析和数据显示软件包等,减少开发费用. 随着 PC 机的发展,多核系统变成主流系统,基于多核的 Windows 系统是多个 CPU、大内存、多

定时器等系统资源于一体的操作系统. 文中通过分析 Windows 系统的实现机制,借鉴 Windows 内核的“机制与策略分离”的思想^[1],对系统资源重分配,再加上适当的管理策略来改进 Windows 的实时性. 这种改造方式的代价最小、兼容性好、开发周期也短,并且能够满足实时性.

2 现有的基于 PC 机的实时系统分析

目前,专用的实时操作系统大多在嵌入式应用领域内应用,其软件编程相当复杂,系统升级和维护的工作量也很大,更为重要的是这类操作系统非常昂贵,而且只能安装在一台计算机上.

在 Windows 上实现实时性效果比较好的有

收稿日期: 2011-09-10; **修回日期:** 2011-10-15

基金项目: 航空科学基金(2010ZD53, 2009ZD53044); 陕西省自然科学基金(2009JM8017, 2009JQ021, 2010JM8023); 西北工业大学种子基金(Z2011133)

RTX. 试验验证,其实时性能满足硬实时的要求,但是价格昂贵,由国外的公司开发,没有源码,做相关的系统开发以及修改都很不方便,同时也存在一些未知的安全性,但是,此方案软件开发量大,周期长,难度较大.所以,此方案只能作为测试实时性的一个标准对待,开发上必须寻找另一条路.

3 Windows 系统的非实时性分析

通过对现有的基于 PC 机的实时系统分析得知,影响实时任务的主要因素是非实时任务对实时任务的 CPU 资源的抢占、不确定的中断延迟以及 Windows 调度周期太长等.这三个因素使现有的 Windows 系统上的实时任务不能按时完成,保证不了实时性.

3.1 Windows 调度周期对实时任务的影响

众所周知,时钟是一个操作系统的脉搏,Windows 操作系统的时钟周期大约为 15ms. 创建一个实时任务后,操作系统不会立即它,而是等到下一次时钟中断到来时,引发调度器调度,然后才有可能把此任务投入运行.在这种情况下,不考虑调度机制以及其他影响因素,调度周期太长就造成了一个实时任务的平均延迟在 7.5ms 左右,所以必须提供更快的实时时钟,缩短调度周期,才有可能进一步改造 Windows 的实时性,同时提供一个更快的实时时钟并缩短调度周期是改造 Windows 实时性的必要条件.

3.2 系统中的非实时任务对实时任务的影响

目前 Windows 系统不能实现实时性的一个重要原因是由于系统中存在大量的非实时任务,它们可能由于等待、被唤醒等操作后,任务优先级会有提升或者降低,在一定程度上存在很多不确定性因素,所以任务的调度也就存在很多不确定性.由于 Windows 的优先级太少,当系统负载较大时,同一优先级的任务较多,Windows 的调度策略是采用时间片轮转调度,在这种情况下,新创建的实时任务的执行延迟时间不仅取决于系统的调度周期,还取决于同一个优先级上的任务的多少.如果同一优先级上的任务较多,那么实时任务需等到其他同优先级的线程的时间片都执行完之后才能够执行,所以实时任务的具体延迟时间无法确定,不能保证实时任务的实时性.

以双核 PC 为例,在 PC 中负载较小时,一个实时任务的完成时间可以得到保证;当系统负载较重时,由于众多非实时任务与实时任务抢占 CPU 资

源,使非实时任务在有限的时间内不能完全得到 CPU 资源,造成了实时任务不能按时完成,不能保证实时性.为了测试非实时任务对实时任务的影响,设计了以下试验.在试验中,实时任务 Task 计算 10000 次 $\sin(1.5)$,并计算所消耗的时间,为了测试具有代表性,这里运行任务 1000 次并求平均值.此试验进行 3 次,测试时分两种情况进行:① 空负载测试,② 启动一个无限循环的进程,可以看做非实时任务,然后进行测试,测试结果如表 1.

表 1 非实时任务对实时任务的影响(微秒)

测试次数/任务耗时	空负载	满负载	延迟
1	4682.722	6356.318	1673.596
2	4696.852	6398.726	1701.874
3	4706.536	6381.383	1674.847

由表 1 数据得知,在满负载的情况下,由于非实时任务的存在,会抢占实时任务的 CPU 资源,造成实时任务不能按时完成,平均延迟为 1683.439 μ s(3 次试验的结果取平均值).对于实时任务来说,当非实时任务较多时,由于 CPU 资源分配的不合理,实时任务的运行会受到影响,远远不能满足实时性要求.为了提高实时性,有必要对 CPU 资源进行重分配.

3.3 PC 中的外部中断对实时任务的影响

主流多核 PC 机的中断系统采用高级可编程中断控制器 APIC(Advanced Programmable Interrupt Controller),它是面向多处理器设计的,包括 SMP(对称式多处理机)和非对称的多处理机系统.实际上,Intel 多处理器规范的核心就是高级可编程中断控制器的使用,每个 CPU 有自己的 LocalAPIC,并且还有一个 I/O APIC 是安装在主板上.外部的设备连接到 I/O APIC,I/O APIC 再把中断分发给各个 CPU 的 LocalAPIC 处理.

每次外部硬件产生一次中断发生后,CPU 都需要保存当前的线程运行环境,然后去处理相应的中断过程,中断类型不同耗费的时间也不相同.现场保护和恢复所消耗的时间基本固定,不同的中断服务程序所消耗的时间差别巨大,并且大多数中断还会引发 DPC 处理,DPC 的处理会消耗大量的时间,CPU 必须处理完这些中断以及中断所引发的 DPC 操作之后才会继续执行实时任务^[2].所以 PC 的外部中断会占用 CPU 资源,并对实时任务造成延迟,具体的延迟时间取决于中断的类型以及发生频

率^[3]. 中断操作的耗时越多、发生的频率越高,所消耗的 CPU 资源就越多,对实时任务造成的延迟就越大.

为了测试外部中断对实时任务的影响,对于 3.2 中的实验进行改进,运行实时任务 Task,同时从外部磁盘上拷贝一个大约 4G 的文件到 PC 磁盘上来模拟短时间内产生大量中断的过程,在此过程中按照 3.2 节中的方法测试实时任务 Task 的运行时间,并与之前的结果相比较,实验结果如表 2 所示.

表 2 外部中断对实时任务的影响(微秒)

测试次数/任务耗时	外部中断较少	外部中断较多	误差
1	4673.732	4726.418	52.686
2	4672.468	4725.682	53.214
3	4676.242	4729.229	52.987

大量外部中断发生时,执行中断操作会造成实时任务的延迟,由表 2 中的数据可知,大量磁盘中断产生时,对实时任务的造成的延迟大约为 52.962 μ s (3 次结果的平均值),说明外部中断对实时任务的影响很大,必须减少这些中断的发生,才能减少实时任务因中断操作而造成的延迟.

4 Windows 实时性改造方案及实现

多核 PC 机的中断管理系统采用 APIC 结构并且存在闲置的定时器,所以方案主要从以下三个方面改造 Windows 的实时性:提供快速的实时时钟、重新分配 CPU 资源减少非实时任务对实时任务的影响,重新分配中断资源减少中断延迟对实时任务的影响.

4.1 为 Windows 提供高速的实时时钟

多核 PC 中的 LocalAPIC 定时器为闲置状态,我们利用它来实现快速的实时时钟.在 LocalAPIC 中,有相关的初始计数器、定时器向量等寄存器,根据时钟周期计算处定时器的初始计数值,写入 LocalAPIC 的初始计数寄存器即可,具体方法请参考文献[4].实际方案中我们把它设为 100 μ s 中断一次,即调度周期为 100 μ s.在此条件下,可以保证任何一个实时任务被调度的平均时间延迟在 50 μ s 左右,远小于 Windows 的 7.5ms,满足实时系统的要求,也是本文进行 CPU 资源重分配、中断资源重分配的基础.

4.2 重新分配 CPU 资源

在改造方案中,通过对系统中的 CPU 资源进

行重新分配,一部分 CPU 资源用来执行非实时任务,称之为非实时 CPU,剩下的 CPU 资源用来执行实时任务,称为实时 CPU,并且要保证两部分 CPU 的执行要保持相对独立性,执行任务时互不影响.

多核系统中,每个 CPU 都有单独的寄存器、PRCB 表、IDT 表等,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 Windows 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.

在具体的方案中,CPU 资源的分配可以通过 CPU 亲缘性来实现.Windows 的 CPU 亲缘性就是指在系统中能够将一个或多个进程或线程绑定到一个或多个处理器上运行.微软实现了一个基于 Windows 内核驱动的线程管理服务系统,它能让用户根据每一个任务线程对 CPU 资源的需要程度和对实时性的要求,在多核 CPU 上合理为线程分配 CPU 核.在实际的改造方案中,以作者的双核 PC 机为例,这里规定 CPU0 为非实时核,CPU1 为实时核.首先遍历现有系统中的所有进程(假定这些进程全部为非实时进程),通过设置亲缘性把非实时进程绑定到 CPU0 上运行,这样以来现有系统中没有任务需要在 CPU1 上运行,那么 CPU1 就可以用来执行实时任务了.关于 CPU 亲缘性以及亲缘性的设置方法可以参考文献[5]和文献[6].

4.3 重新分配中断资源

现有的 PC 的外部中断连接到 IOAPIC,IOAPIC 起到一个中断路由的作用,通过重定向表来决定把收到的外部中断发给哪个 CPU 处理.

由 Interl 手册^[7]得知,IOAPIC 单元包含一组总线输入信号,24 个 64 位的中断重定向表、可编程的控制器和一个发送、接收 APIC 总线上 APIC 信息的寄存器.I/O 设备通过在 IOAPIC 中申请一个中断信号线来向操作系统发送中断.然后 IOAPIC 在重定向表中选择相应的项并且使用项中的信息形成一条中断请求信息通过 APIC 总线发送给相应的 CPU 处理.重定向表中的每一项可以单独地被编程,可以指定它为 edge/level 敏感的中断信号、中断向量和优先级、目标处理器,以及处理器被选择的方

式(静态或者是动态)。

改造方案中只用到了 IOAPIC 中的 24 个重定向表, APIC 就是通过这 24 个重定向表实现了中断的“路由”功能。通过对重定向表重新编程, 使外部中断只发给非实时 CPU 核处理, 不发给实时 CPU 核, 从而减轻实时 CPU 的负载, 减少实时任务的延迟。

IOAPIC 中的 24 个 64 位的重定向表, 每一项都可连接外部中断, 其中第 56~63 位表示把此中断发往哪些 CPU 处理。这八位二进制位从低到高分别表示 8 个 CPU 核, 把相应比特位置为 1, 表示此中断需要发给此 CPU 核处理; 置为 0 表示此中断不发给此 CPU 处理。在 Windows 中, IOAPIC 的地址为 0xfec00000, 通过读写其一系列的寄存器来实现编程控制, 要访问 IOAPIC, 不能直接使用它的内部寄存器, 而是要通过 2 个内存映射寄存器来访问, 一个是 IOREGSEL, 一个是 IOWIN, 他们地址分别是 0xFEC00000 和 0xFEC00010, 这里所说的地址均是物理地址, 需要使用 wdk 中的 MmMapIoSpace 函数映射后才能对其操作。利用 2 个映射的寄存器遍历 24 项重定向表, 把每一个向量的第 56~63 位修改为相应的值即可(具体实现可以根据自己的 PC 而定)。在作者的 PC(双核系统)中, 只需把重定向表项中的第 56 位置为 1, 把第 57~63 位置为 0, 即可实现外部中断全部发给非实时核 CPU0, 而 CPU1 不会接收到外部中断。通过此设置即可完成中断资源的重分配, 此时实时 CPU1 不再接收外部中断, 所有的外部的中断都由非实时 CPU 核处理, 从而可以消除实时任务因为中断处理而造成的延迟。

5 资源重分配方案的实时性验证

系统资源重分配后, Windows 拥有了周期 100us 的实时时钟, 保证了能及时调度实时任务; 拥有独立的 CPU 资源并且实时任务执行时不会受到非实时任务和外部中断的影响, 消除了外部因素对实时任务执行的影响。验证方法是: 资源重分配后重新测试 3.2 节和 3.3 节中的任务, 对结果进行比较, 分析其实时性。

5.1 CPU 资源分配后的实时性验证

在测试之前, 使用 4.2 节中的方法进行 CPU 资源重分配, 创建实时任务后, 设置它在实时 CPU 核上运行, 此时实时 CPU 核只是用来执行实时任务。

设置完成后重新测试 3.2 中试验。试验条件: 实时任务 Task 计算 10000 次 $\sin(1.5)$, 并设置其在实

时 CPU 上运行, 同时系统中与用一个无限循环程序来模拟非实时任务, 设置在非实时核上运行, 计算 Task 所消耗的时间, 为了测试具有代表性, 这里通运行任务 1000 次求平均值, 此试验进行 3 次, 结果如表 3 所示。

表 3 CPU 资源分配后的实时性验证(微秒)

测试次数/任务耗时	未分配 CPU 资源	已分配 CPU 资源	误差延迟
1	6319.943	4668.934	1651.009
2	6269.766	4671.168	1598.598
3	6386.394	4668.213	1718.181

由表 3 数据和表 1 数据分析可知, 在系统满负载时, 未分配 CPU 资源的情况下, 实时任务的运行时间延迟 1655.929 μ s 左右(取 3 次试验的平均值), 而在分配 CPU 资源的情况下, 无论非实时任务的负载多大, 实时任务的运行时间基本不变。由此说明, 在对系统 CPU 资源重新分配后, 保证了实时任务可以独占实时 CPU 核, 而不受非实时任务的影响, 相对于未分配 CPU 资源的 PC 系统来说提高了实时性, 证明此方案可行并且对 Windows 的实时性有显著的提高。

5.2 中断资源的重分配后的实时性验证

CPU 资源重分配后, 实时任务的主要延迟来自于外部中断的延迟, 为了消除外部中断的影响按照 4.3 节中的方法进行中断资源重分配, 重新测试 3.3 节中的试验, 计算 Task 所耗费的时间, 测试数据如表 4 所示。

表 4 中断资源重分配后的实时性验证(微秒)

测试次数/任务耗时	中断资源未分配	中断资源已分配	误差
1	4727.700	4671.261	-56.439
2	4734.007	4669.635	-64.372
3	4731.003	4670.257	-60.746

中断资源重分配后, 从外部拷贝大文件时会不断地产生的中断不会发往实时 CPU 核, 而是发送给非实时核处理, 实时核上运行的实时任务不会受到中断的影响。表 4 数据显示, 中断资源分配后, 实时任务的运行时间比中断资源分配前减少了 60.519 μ s(取 3 次的平均值), 说明中断资源的重分配后, 实时 CPU 核受外部中断的影响减小, 实时任务能按时完成, 保证了实时性, 说明此方案可行并且对于 Windows 的实时性有显著的提高。

(下转第 103 页)

实验结果可以明显看出优化后的 k -同构算法比原算法匿名质量相对提高,有效保护用户的隐私信息。

6 结束语

本文采用 k -同构方法保护用户的节点信息和连接信息,提出一种优化的图划分算法对图进行划分,最后通过增加或者删除边使得发布图是 k -同构的,同样也是 k 安全的,从而保护用户的隐私。通过数据集实验表明算法能提高匿名质量并大大减少了信息丢失。今后的工作将处理社会网络中动态的、带标签的图数据同构问题,以满足高要求图数据发布的安全问题。

参考文献:

- [1] Liu K, Terzi E. Towards identity anonymization on graphs[C]//Proceedings of the 2008 ACM SIGMOD international Conference on Management of Data. USA, New York: SIGMOD, 2008: 93-106.
- [2] 刘恒,王铁军,周明天,等. 普适环境中基于角色的隐私保护系统设计[J]. 微电子学与计算机, 2010, 27(12): 42-46.
- [3] Zheleva E, Getoor L. Preserving the privacy of sensitive relationships in graph data[C]//Proceedings of the 1st ACM SIGKDD Workshop on Privacy, Security, and Trust in KDD (PinKDD' 07). San Jose, CA, USA: ACM, 2007: 153-171.

- [4] Ying X, Wu X. Randomizing social networks: a spectrum preserving approach[C]//Proceedings of the 2008 SIAM International Conference. on Data Mining. USA: [s. n.], 2008: 739-750.
- [5] Lei Zou, Lei Chen, Tamer Ozsu M. K-automorphism: a general framework for privacy preserving network publication[J]. VLDB, 2009, 2(1): 946-957.
- [6] James Cheng, Ada Wai-Chee Fu, Jia Liu. K-Isomorphism: privacy preserving network publication against structural attacks[C]//Proceedings of the 2010 SIGMOD international Conference on Management of Data. USA, Indianapolis, Indiana: ACM, 2010: 459-470.
- [7] Zhou B, Pei J. Preserving privacy in social networks against neighborhood attacks[C]//Proceedings of the 24th IEEE International Conference. on Data Engineering (ICDE'08). Cancun, Mexico: IEEE Computer Society, 2008: 506-515.

作者简介:

张晓琳 女, (1966—), 博士, 教授. 研究方向为数据库理论与技术、信息安全。

李玉峰 男, (1985—), 硕士研究生. 研究方向为信息安全、图数据。

刘立新 女, (1983—), 硕士研究生, 助教. 研究方向为信息安全、不确定数据管理。

郑珍珍 女, (1985—), 硕士研究生. 研究方向为连续不确定 XML、信息安全。

(上接第 98 页)

6 结束语

为了提高 Windows 的实时性, 本文提出了对系统资源重新分配来改造实时性的方案, 并设计试验说明此方案显著地提高了 Windows 的实时性. 不仅保持了 Windows 操作系统的完整性, 还能保证实时任务在规定时间内完成, 为开发完整的实时系统提供了可行性的参考方案。

参考文献:

- [1] 潘爱民. Windows 内核原理与实现[M]. 西安: 电子工业出版社, 2010.
- [2] 张帆, 史彩成. Windows 驱动开发技术详解[M]. 北京: 电子工业出版社, 2009.
- [3] 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.

- [4] 潘汉, 莫苏苏. 基于 LocalAPIC 的 Windows2000 实时化改造[J]. 电子元器件应用, 2009(10): 79-82.
- [5] 龚广, 李舟军, 忽朝俭, 等. Windows 内核级 Rootkits 隐藏技术的研究[J]. 计算机科学, 2010(4): 59-62.
- [6] 李骥, 姜守达, 邹昕光. Windows 操作系统多核 CPU 内核线程管理方法[J]. 自动化技术与应用, 2010(1): 27-30.
- [7] Intel Corporation IA-32 Intel Architecture Software Developer's Manual Volume 3: System Programming Guide.

作者简介:

杜旭东 男, (1987—), 硕士研究生, 研究方向为嵌入式计算与应用。