采用混合时钟模式提高 Linux 时钟精度的方法

张 健,刘青昆,王异奇,周 娇
(辽宁师范大学 计算机与信息技术学院,辽宁 大连 116081)

摘 要:时钟精度直接影响到实时任务能否被及时响应和调度,数控系统要求其软件平台能提供微秒级的时钟精度。Linux 操作系统由于其开放源代码的特点,非常适合开发具有自主产权的全软件数控系统,但是其毫秒级时钟精度明显过于粗糙。结合上述方法,提出一种混合多种时钟模式的动态时钟系统,提高 Linux 的时钟精度。通过仿真测试证明能满足数控系统的要求。

关键词:数控;Linux 系统;时钟精度;定时器

中图分类号:TP316.2 **文献标识码**:A **文章编号**:1674-7720(2011)03-0054-04

Mixed-clock mode to improve Linux clock accuracy

Zhang Jian, Liu Qingkun, Wang Yiqi, Zhou Jiao
(School of Computer and Information Technology, Liaoning Normal University, Dalian 116081, China)

Abstract: Clock-precision affects directly that whether the real-time task can be responded timely and scheduled, CNC system requirements its software platform to provide clock-precision of ms-level. Because the characteristics for Linux operating system is free source, it is an ideal for development of numerical control system with independent property rights, but the ms-level clock precision is too rough. Combination of the above methods, a dynamic-clock system of hybrid multi-clock model is proposed to improve Linux clock-precision. Finally, simulation tests show that the system can meet the requirements of CNC system.

Key words: CNC; Linux system; clock precision; timer

数控技术(CNC)已经成为现代制造业的核心技术之一,开放式数控系统相对于传统数控系统在功能、灵活性、成本等方面的优势,使得开放式数控成为数控系统未来发展的主要趋势。目前,开放式数控系统主要有三种结构,即专用 CNC+PC、通用 PC+运动控制器和软数控系统。其中软数控系统采用多任务实时操作系统,将运动控制部分与管理部分集成到一个硬件平台上,满足数控系统在功能方面和非功能方面(主要表现在实时性)的要求[1]。开放式数控系统的理想软件平台是实时多任务操作系统,目前,商业实时多任务操作系统有很多,比较著名的有 VxWORKS、iRMX、QNX 等,但这些操作系统产品大多成本高、开放性差。Linux 是一种发展十分迅速的类 UNIX 系统,已被广泛地运用到服务器、桌面系统以及嵌入式应用领域。Linux 由于其开放源代码的特点,可以在此基础上开发具有自主知识产权的数控系统。但是 Linux 最初的设计目标是一个分时操作系统,追求系统效率和公平性,在对实时性要求高的领域应用受到限制。虽然 2.6 内核的 Linux 时钟粒度提高到了 1 ms,但仍远不能满足数控系统对定时精度的要求。

近年来的研究以细化时钟粒度来提高 Linux 的实时应用能力提出了一些方案和设想,主要有 KURT-Linux 系统、RT-Linux 系统。本文对 KURT_Linux、RT-Linux 提高时钟精度的方法进行分析,考虑在强周期性应用或者在某个时段内有大量高精度定时器将超时的情况下,采用一种动态的多模式时钟机制来提高 Linux 的时钟精度,并通过分析测试证明该方案确实可行。

1 Linux 时钟机制与改进

1.1 Linux 时钟机制

时钟和定时器对 Linux 系统来说是至关重要的。首先,内核要管理系统的运行时间以及墙上时间;其次,内核中大量的任务是基于时间驱动,其中有些任务是周期执行,如对调度程序中运行队列进行平衡调整或对屏幕进行刷新,而有些任务需要推后执行的 I/O 操作则需要等待一个相对时间后才运行。

系统时钟是定时器硬件和系统软件的结合，在X86体系结构中，使用最普遍的定时器硬件是Intel8254可编程定时器芯片(PIT)，它产生的中断就是时钟中断(tick)。时钟中断是特定的周期性中断，对应中断服务程序，完成更新系统时间以及任务的管理、调度等工作；系统在每次时钟中断处理中更新jiffies，维护系统定时器链表timer_list，对超时的定时器进行处理。

与系统定时器相对的是动态定时器，是用来调度事件在将来某个时刻发生的机制。它依赖于系统时钟中断，在时钟中断服务程序的下半部，系统检查是否有超时的动态定时器并进行处理。linux2.6内核的系统时钟频率为1 000 Hz，即时钟中断的触发周期为1 ms，中断服务程序最快每1 ms执行一次。动态定时器随时都可能超时，但只有在中断服务处理程序执行时才会检查、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周期。

CNC数控系统的工作过程通常是首先内建一个定时器(由操作系统完成)，然后周期性地执行控制程序，周期通常为几十微秒到十几毫秒。在每个周期内要完成状态监测、译码、刀具补偿计算、插补计算、PLC管理、位置控制等工作。可见，在加工工件过程中，CNC要求的实时性非常高，必须在很短、很精确的周期内完成一系列的计算和输入输出，否则加工精度无法得到保障。标准2.6内核Linux定时器精度远远达不到数控系统周期实时任务要求的微秒级定时精度。

1.2 提高时钟精度的办法

近年来人们对Linux进行实时化改造提出了一些方案和设想，主要有KURT-Linux、RT-Linux等[2]。下面分别进行介绍。

KURT_Linux[3]由kansas大学开发，通过对Linux内核内部进行改造来满足实时应用需求。在时钟精度方面，KURT-Linux将Linux的时钟中断固定模式改为单次触发模式(one-shot mode)，即每次给时钟芯片设置一个超时时间，然后等到该超时事件发生，在时钟中断处理过程中再次根据需要设置一个超时时间。通过这种变长时钟模式，将Linux时钟精度提高到μs级。既保证了特定实时任务的精度要求，又避免了不必要的调度负担。

RT-Linux是新墨西哥工学院研制的一个基于Linux的硬实时系统。它采用双内核方法，在原有Linux基础上设计一个专门处理实时进程的内核，然后把整个Linux作为实时内核上运行的一个低优先级进程。在时钟精度方面类似KURT_Linux，将系统实时时钟设置为单次触发状态，然后利用TSC提供高达CPU时钟频率的定时精度。

Monta Vista Linux是由James Ready领导开发的嵌入式Linux，通过对Linux内核进行内部改造，直接修改原有Linux内核的数据结构等来满足实时需要。在高精度时钟方面，抛开传统的周期中断CPU的方法，使定时器在需要的任何一个μs产生中断，但不在每个μs产生中断，将系统的定时精度提高到μs级。

Linux-SRT是剑桥大学David Ingram的博士论文项目，它简单地修改了Linux中Hz的定义，将Linux时钟频率由100 Hz提高到1 024 Hz。这种方式实现起来很简单，但是由此带来频繁的定时中断使得系统开销很大。

借鉴KURT-Linux的one-shot思想来提高时钟精度，并利用高级可编程中断控制器(APIC)[4]或通过附加的硬件资源实现一个与系统时钟并行的高精度实时时钟，在系统中维护一个高精度实时时钟和一个低精度系统时钟[5-7]，是一种普遍采用的提高时钟精度的方法。但是在缺乏附加硬件支持或APIC使用受限的应用环境下，只能利用PIT芯片作为高精度时钟源，在每次中断处理时要重新计算下一次中断时间和对PIT进行编程。由于PC的兼容性，PIT芯片位于低速的ISA总线上，频繁设置定时器硬件也需要耗费大量的时钟周期。因此one-shot模式时钟中断处理时间可能达到标准Linux时钟中断处理时间的7~15倍[8-9]。在强周期应用或有大量定时器集中在某个时段内时超时，需要采取一种不同于one-shot的时钟模式。

如果系统中没有任何实时定时器，则系统每隔1 ms会有一次周期性jiffies时钟中断，采用one-shot模式使得系统性能下降大约1.5%。如果系统中没有任何实时定时器，则需要重新将时钟设置为RTL CLOCK MODE PERIODIC工作模式，并且时钟周期和标准Linux下时钟周期一致，使Linux能在系统中不存在实时任务的情况下高效地工作。

2 动态高精度时钟设计和实现

动态高精度时钟设计方案借鉴了KURT-Linux思想，但与其不同的是提供一个与标准Linux核心时钟并行的具有精密刻度的实时时钟，并与原核心时钟区别开。采用X86体系CPU提供的TSC作为高精度的时间标度，权衡一定时间段(如一个jiffies)内高精度定时器的数量，设置Linux时钟中断模式为标准模式、one-shot模式或高频周期时钟模式。实现了μs级定时精度的同时，降低了频繁计算和设置时钟芯片的时间代价。

下面给出关键的全局变量：

(1)*time_mode*：表示当前时钟工作模式。其中-1代表高频周期时钟模式，该模式下，根据需要达到的定时精度，设置时钟芯片以较高的频率产生周期性中断；0代表标准模式，时钟芯片以标准Linux默认的频率产生周期中断；1代表one-shot模式，时钟芯片被设置为单次触发状态，即每次给时钟芯片设置一个超时时间，超时事件发生时，在时钟中断处理程序中根据需要再次给时钟芯片设置一个超时时间。系统启动时设置为默认值0。

(2)*SCALE*：时钟精度提高比。设置高频周期模式需

要的参数，用来表示所需要达到的时钟精度相对普通 Linux 时钟精度的提高倍数。

(3)*Threshold*：阈值。如果即将在某一时间段内超时的实时定时器数量大于预设值，系统设置硬件定时器工作在高频周期时钟模式。

2.1 时钟中断处理

为了加强 Linux 的实时功能，同时又要保持 Linux 的完整性，本方案的动态多模式时钟机制以模块化的方式实现有关实时部分的功能，并利用接口函数实现实时模块与 Linux 核心的联系。

(1)标准模式。标准模式下的中断处理首先查询实时定时器队列中是否有实时定时器在下一个系统时钟中断(jiffies+1)之前超时，即在(jiffies，jiffies+1)内是否有实时定时器要处理，根据实时定时器数量设置时钟芯片的工作模式，执行 do_timer_interrupt()等函数维护系统相关时间，标记下半部。

(2)one_shot 模式。one-shot 模式下的中断处理先判断 jiffies 时钟是否到期，如果到期：

①查询实时定时器队列中是否有实时定时器在下一个系统时钟中断(tick+1)之前超时，即在(jiffies，jiffies+1) 内有实时定时器要处理（其超时时间用 sub_jiffies 表示），然后根据实时定时器数量设置时钟芯片工作模式。

②执行 do_timer_interrupt()函数等维护与系统有关的时间，并标记下半部。

如果 jiffies 时钟未到期，则查询实时定时器链表，根据其最早超时实时定时器的超时时间与当前时间的差值设置时钟芯片产生下一次中断的时间。

(3)高频周期时钟模式。高频周期模式下中断处理先判断 jiffies 时钟是否到期，如果系统时钟节拍到期，执行上述①、②模式。否则，如果有实时定时器超时，标记中断下半部；如果没有实时定时器超时则直接返回。

对超时定时器的处理都留到时钟中断下半部(softirq)处理，超时的实时定时器优先得到处理，以尽可能保证实时定时器的及时处理，随后处理普通 Linux 的定时器，时钟中断处理过程如图 1 所示。

2.2 定时器组织

普通 Linux 系统原有的粗粒度定时器对于内核的稳定和不要求高精度定时的非实时应用仍是合适的，只是针对有高精度定时要求的实时应用组织一个高精度定时器队列 HRT_list，队列中的定时器按超时时间非降序排列，队列中第一个定时器的超时时间就是队列的最早超时时间。

原 Linux 内核中的定时器是通过称为 CTW (Cascading Timer Wheel)的结构管理和维护，并因此使得对定时器的插入、删除等操作的时间为 0 (1)。本文把 HRT_list 队列和 CTW 结合起来以降低定时器处理时间、

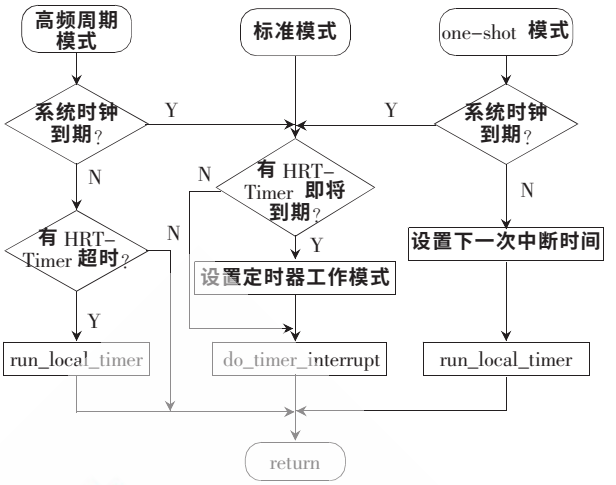

图 1 时钟中断服务程序执行流图

提高效率。把需要较长时间才超时的实时定时器仍旧插入到原定时器队列中，借助该队列维护。在每次系统时钟中断处理的下半部处理完超时的实时定时器后，把在下一次系统时钟中断前超时的高精度定时器从原队列移除，并插入到 HRT_list 队列中。因此，HRT_list 队列中所需要维护的高精度实时定时器也是有限的，避免了维护一个大规模定时器队列的开销，近似实现了 0(1)的系统开销。

3 性能分析与测试

3.1 性能分析

当系统中没有高精度定时器时，PIT 仍以 Linux 系统默认的频率触发时钟中断，在每一次系统时钟中断处理过程中，只需要判断工作模式以及下一次 jiffies 中断前有否实时定时器超时，经测试由此而带来的处理时间不超过 1us，增加系统负担<0.1%，不会影响系统的性能。当在某个时间段内系统中实时定时器不多于阈值时，系统时钟工作在类似 KURT-Linux 的 one-shot 模式，同时维持普通 Linux 系统时钟的稳定。而由此而带来的系统负担是可以接受的[3]。

当系统中存在大量实时定时器或在某个时间段内即将超时的实时定时器数量超过一定值(阈值)时，相对于 one-shot 模式需要频繁地计算下次中断时间，并重新编程在低速的 ISA 总线上的 PIT 的时间代价是可取的，证明如下：

用 T_{hw} 表示中断的硬件处理时间，T_{isr} 表示中断程序上半部执行时间，n 代表某个时段内(一个 jiffies 内)超时的定时器数量。得到两种模式下总的时钟中断处理时间关系式：

$$\sum^{n}(T_{hw}+T_{isr}^{one-shot})>SCALE\times(T_{hw}+T_{isr}^{hf}) \quad (1)$$

由式(1)得到阈值计算公式：

$$Threshold=n>SCALE\times(T_{hw}+T_{isr}^{hf})/(T_{hw}+T_{isr}^{one-shot}) \quad (2)$$

显然，当某个时段内超时的定时器数量大于

Threshold 时，采用高频周期模式的时间开销就会小于 one-shot 模式。

3.2 模拟测试

测试环境为 Pentium4 3.0 GHz CPU，1GDDR 内存的硬件平台和 2.6.15.6 版本内核的 Fedora core linux 操作系统平台。

根据数控实时任务的要求设定了周期为 0.1 ms、1 ms 和 100 ms 的进程模拟数控实时周期任务[10]，统计运行 1 000 次的数据，比较改进后的高精度定时器和原 linux 定时器的平均定时偏差，并令阈值为 30，设置周期任务数量为 4、20、40，使时钟工作在不同模式下。测试结果如表 1 所示。

表 1 定时器定时偏差比较

高精度定时器 HRT_Timer 内核（精度为 10 μs）	标准 Linux 内核（精度为 1 ms）
34	968

由测试数据对比，原 linux 系统的定时平均偏差为 968 μs，改进后系统的定时平均偏差为 34 μs。显而易见，改进后的定时器定时精度大大提高，达到 10 μs 级，能满足数控系统应用的要求。

在原 Linux 内核和改进后的高精度定时器内核上睡眠 50 μs 各 1 000 次，测试实际睡眠时间所得结果与表 1 类似，50 μs 的实际睡眠时间从(2.001~2.116) ms 级降到(57~91) μs 级。

全软件数控系统以应用软件的形式实现运动控制，是开放式数控系统的发展方向。开源的 Linux 是开发具有自主知识产权数控系统的理想平台，但是其粗糙的时钟粒度是普通 Linux 直接应用于数控系统的最大障碍，因此需要细化 Linux 的时钟粒度提高其实时性。

简单地提高系统时钟频率将引起频繁的中断处理，导致系统性能的下降。KURT-Linux 采用的 one-shot 方式将周期性的时钟中断改进为单次触发状态，实现了 μs 级的定时精度。本文分析了普通 Linux 时钟机制和几种实时 Linux 操作系统细化时钟精度的方式，提出了一种混合多种时钟模式的动态时钟机制，达到了 CNC 要求的时钟精度。最后的性能分析和模拟测试证实了新时钟机制的技术性能。

参考文献

[1] 李迪，万加富，叶峰，等.软数控系统混合任务两级调度策略[J].机械工程学报，2008，44(12)：157-162.

[2] 王霞，马忠梅，何小庆，等.提高嵌入式 linux 时钟精度的方法[J].计算机工程，2006，32(23)：70-96.

[3] 施映，何嘉.KURT-Linux 实时性研究及改进策略[J].计算机科学，2006，33(7)：417-420.

[4] 丁一，胡封林，李国宽.高级可编程中断控制系统的研究[J].计算机工程与科学，2005，27(12)：97-100.

[5] 范剑英，吴岩，贾佳，等.Linux2.6 实时性分析与改进方案[J].哈尔滨理工大学学报，2008，13(1)：24-28.

[6] 於时才，缪东升，孙华，等.Linux2.6 调度系统的分析与改进[J].微计算机信息，2007，24(5-3)：252-254.

[7] 周鹏，周明天.linux 内核中一种高精度定时器的设计与实现[J].计算机技术与发展，2006，16(4)：73-78.

[8] SRINIVASAN B, PATHER S, HILL R, et al. A firm real-time system implementation using commercial off-the-shelf hardware and free software [R]. rtas. Fourth IEEE Real-Time Technology and Applications Symposium (RTAS' 98), 1998.

[9] 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[J].计算机研究与发展，2003，40(5)：734-740.

[10] 姚鑫骅，潘雪增，傅建中，等.数控系统的混合任务模型及其最优调度算法研究[J].浙江大学学报，2006，40(8)：1315-1319.

(收稿日期：2010-07-03)

作者简介：

张健，男，1974 年生，硕士研究生，主要研究方向：嵌入式操作系统。

刘青昆，男，1971 年生，副教授，博士，主要研究方向：嵌入式操作系统，分布式系统与并行计算。

王异奇，女，1985 年生，硕士研究生，主要研究方向：嵌入式操作系统。