基于 RTHAL 的 Linux 实时性研究和实现

苏曙光 刘云生
(华中科技大学软件学院 武汉 430074)

摘 要 研究嵌入式 Linux 操作系统实时性的扩展和实现。提出通过在底层硬件和操作系统之间增加实时硬件抽象层的方式来扩充和增强 Linux 实时性能。该方法对操作系统内核只需做微小修改,就能达到既保持 Linux 现有功能和资源又兼有硬实时能力,其进程切换延迟仅为 3 微秒左右。讨论了对 Linux 和实时硬件抽象层两方面的扩展过程,同时还提出了如何有效利用两者间的缓冲区的方法。最后在嵌入式 MPEG4 流媒体系统中应用和测试了该方法,结果表明该方法实时性能优异,使用简单有效。

关键词 实时硬件抽象层,实时性,Linux,实时视频编码
中图法分类号 TP309 **文献标识码** A

Research and Implement of Realtime of Linux Based on RTHAL

SU Shu-guang LIU Yun-sheng
(School of Software Engineering, Huazhong Univ. of Sci. & Tech., Wuhan 430074, China)

Abstract The paper focused on research and implement of realtime of Linux. The paper proposed a method taking advantage of RTHAL(Real Time Hardware Abstract Layer), which run between hardware and Linux. The method made only a few of modification on Linux and had a schedule delay of only about 3μs. The paper discussed how to modify Linux and RTHAL for hard realtime. In addition the paper proposed an effective method to take advantage of buffer between them. Lastly an example of realtime MPEG4 stream system was cited, which applied the method and proved it could add to hard realtime of Linux effectively and concisely.

Keywords RTHAL, Realtime, Linux, Video stream coding

1 Linux 实时性和 RTHAL

在研究操作系统实时性时一般都定义为系统能够在指定的时间及时响应和完成任务。这要求操作系统在占先式内核、调度策略分析、任务优先级分配、程序执行时间的可确定性等多个方面对实时性进行支持。Linux 不是为实时系统设计的,因而不能够直接满足实时系统的需要,有很多不适合实时应用的特点[1]:高优先级的用户态进程不能抢占低优先级的核心态进程;临界区操作采用屏蔽中断的方式;定时器的粒度过大,任务响应延迟过大;虚拟内存管理技术等。这些特点都导致 Linux 操作系统中断和任务响应延迟过大,而不适宜实时性要求很高的领域,尤其是硬实时领域。

通常用 3 种典型方式[2]设计实时操作系统(RTOS, Realtime Operate System)。第一种方法是从零开始设计完整的 RTOS,如 RT-IX。其缺点是工作量大;第二种方法是先设计一个实时微内核,然后在此微内核上逐步增加功能,如 VxWorks。该方法优点是工作量小、系统灵活,缺点是不开源且互不兼容;第三种方法是改造通用操作系统,扩充其有实时能力,如 RTLinux。该方法优点是既能保持原操作系统的功能和资源,又兼有硬实时能力,而且编程简单。正因如此该方法在嵌入式实时系统设计中被广泛采用,本文也采用该方法在 Linux 上扩充硬实时性。

实时硬件抽象层(RTHAL, RealTime Hardware Abstract Layer)是操作系统 Linux 和底层硬件之间的一个中间层。RTHAL 对硬件完全控制,为 Linux 屏蔽了硬件细节并禁止其对硬件直接操作。和实时性能相关的中断管理、任务管理、时钟都由中间层完成。实时应用程序接口(RTAI, Realtime Application Interface)是 RTHAL 框架的一个具体实现版本。RTAI 由 5 部分组成:实时硬件抽象层 RTHAL、Linux 兼容层、实时内核、LX/RT(Linux RealTime)、扩展功能包。RTHAL 为硬件提供接口,Linux 和硬实时操作系统核运行在其之上。Linux 兼容层为 Linux 提供接口以便 Linux 任务管理器可以管理 RTAI 的任务。可见 RTAI 避免对 Linux 内核做大量的修改,这样可以尽量减少对标准 Linux 内核可能带来的不良作用。

2 Linux 硬实时性扩展

调度延迟时间和中断延迟时间是影响和评价操作系统实时性时经常考察的两个指标[1,2]。这两个指标主要受操作系统任务调度策略、外部中断管理、抢占式任务调度 3 个策略影

到稿日期:2008-08-27 返修日期:2009-01-12 本文受湖北省自科基金项目:低码率视频编码动态行为建模(2007ABA311)资助。
苏曙光(1975—),男,博士,讲师,研究方向为流媒体编码解码、嵌入式操作系统,E-mail:su_shuguang@163.com;刘云生(1940—),男,教授,博士生导师,研究方向为实时移动数据库、实时操作系统。

响。图1展示了在Linux系统中引入RTAI构建硬实时系统的结构。RTAI中的任务调度器提供硬实时性能，采用基于优先级的抢先调度策略。在RTAI中，由于只调度实时的任务，而实时任务在系统中数量不多，因而RTAI可以实现效率极高的可抢占调度算法和任务切换算法。在RTAI结构中实时任务以内核线程存在，在任务切换时不产生上下文切换，所以任务切换非常快。另外，在实时线程中设定抢占点，允许低优先级任务可被高优先级任务抢占，保证高优先级任务总能被优先调度。RTAI中断管理器获取所有的中断，如果它是Linux中断，就把它追加到Linux中断队列中，当没有实时任务时，将会被执行；如果是个实时中断，按照合适的方式处理，如把它发送到相应的中断处理函数；由于RTAI的中断处理没有采用底半处理队列，从而避免了调度底半处理队列而引起的延时。而且中断请求统一由RTAI中断管理器调度，不会被Linux系统屏蔽，使得RTAI能够提高中断响应的速度，减少中断处理的等待时间。

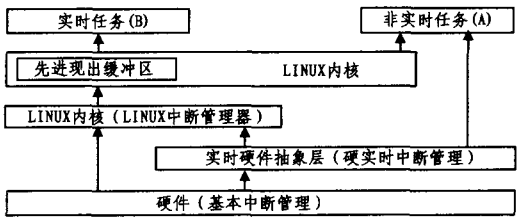

图1 增加实时硬件抽象层的实时性扩展方案

扩展Linux实时性涉及两个方面的程序修改工作：一方面对于Linux来说，需要采用模块管理方式修改和增加RTAI相应实时模块；另一方面对于RTAI来说也需要修改内核和相关的配置文件。整个来说需要采用下面5个步骤实现。

1)在Linux中应用RTAI补丁文件

在Linux主目录中运行patch命令，参数是RTAI补丁文件。补丁对几个Linux系统文件进行了修改，其中定义了一个重要变量rthal(数据类型rt_hal)供Linux及RTAI引用。

2)配置Linux内核

在Linux目录下执行make *menuconfig* 命令，选择[*]Customize Kernel Setting(New)和[*]Customize Vender/User Settings两个选项以便定制内核。定制内核时确保实现[*]insmod,[*]lsmod,[*]rmmod等3个和模块管理相关的功能，最后make编译内核。这样编译的内核就可以实现加载RTAI相应实时模块的功能。

3)配置RTAI模块

在RTAI目录下执行make命令，指定CPU硬件平台(本文实例采用摩托罗拉m68000)，以便加载相应CPU的中断管理模块。编译结果为模块形式，生成包含rtai.o,rtai_sched.o,rtai_fifos.o在内的一些RATI核心模块。

4)在Linux/romfs/dev目录下，执行命令touch建立若干个实时FIFO缓冲区设备节点。

5)通过insmod命令装载rtai相关的核心模块和用户编写的应用模块。

通过上述步骤，可以把Linux系统增强为具有硬实时性能的操作系统，从而用于不同的实时应用中。

3 MPEG4实时流媒体系统

本文作者在研究过程中设计了一个嵌入式实时音视频压缩设备和相应软件。整个系统的功能是首先实时采集来自摄像头或其它视频输出源(例如DVD影碟机)的音视频，然后进行实时压缩，最后通过网络把压缩的音视频流传递到客户端(例如PC机)被解码播放。整个系统是一个典型的实时系统[3]，支持4CIF视频格式，帧率30帧每秒。系统结构如图2所示。

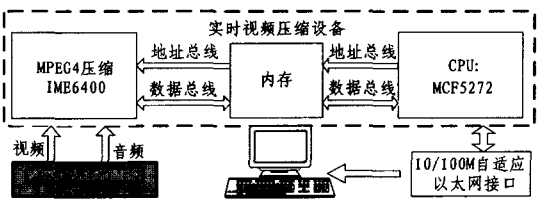

图2 嵌入式实时音视频系统的结构

其中实时音视频压缩设备的视频压缩采用IME6400芯片，处理器CPU采用摩托罗拉冷火系列MCF5272，而操作系统则采用Linux。压缩设备的数据总线16位，主工作时钟66MHz，采用异步数据传输模式，MCF5272完成一次外部数据读或写需要4个时钟周期[4]，因此最大外部数据传输率为(66/4)*(16/8) = 33MBPS。对于IME6400音视频压缩芯片，产生两次压缩缓冲区满的最短时间间隔是400us，缓冲区的大小为固定的1kBytes。即使CPU以15MBps的速率读取缓冲区的数据，读完1kBytes的数据也仅需(1*103)/(15*106) ≈67μs。因此在400us内系统要完成整个中断响应、任务切换、执行中断服务程序、中断返回等一系列任务。而真正用于传输数据的时间很短，所以系统实时响应处理能力是影响比较大的因素。在对操作系统做实时性改进之前，系统可以完成数据压缩、传输的基本功能，但会出现数据丢失的情况。原因在于处理中断时，时间开销比较多，导致压缩数据会丢失，在客户端表现为图像出现马赛克。

正是由于Linux操作系统本身并不具备实时性，故在有多个任务执行时，无法保证关键任务及时被处理。这也是本文采用RTHAL方式改造Linux获取硬实时性能的动机。

4 实时性能的实验设计

测试目的：(1)测试实时任务调度延迟时间。和中断响应延迟一样，这两个参数是评价操作系统实时性能最重要指标[5]；(2)根据本文实例(即MPEG4实时流媒体系统)测试RTAI对Linux实时性能的改进效果和测试RTAI和Linux间实时视频数据的传输是否存在异常。

测试方案：在程序中插入3个加1计数器，用来统计实时数据传递所用的相关缓冲区的溢出情况。RTF0和RTF2两个先进先出(FIFO)缓冲区是RTAI和Linux之间交换实时数据所用，因此这两个缓冲区的大小以及溢出情况关系到实时数据传输能否成功。如果它们尺寸过小，将会发生溢出从而导致实时数据丢失！因此该两个缓冲区中数据的填充情况将很大程度上反映系统实时性能的好坏。RTF0溢出计数器：当向RTF0缓冲区中写数据时，如果返回的数据和写的数据不相等，则说明RTF0没有足够写入空间出现了溢出，这时计数器加1。RTF2溢出计数器：和RTF0溢出计数器类似，

统计 RTF2 缓冲区发生溢出事件的次数。序号比较错误计数器：将 RTAI 发送数据帧的序号和来自 Linux 的接收数据帧的序号进行比较，如果不等或差值超过某一阈值，计数器就加 1。

测试环境：DVD 播放机循环播放一段音视频资料，输出模拟音频、视频信号传给嵌入式硬件设备进行实时数据压缩并通过网络传送到 PC 服务器等待转发到 PC 客户端或直接送到 PC 客户端（安装有常见媒体播放工具和解码插件）。另外设置一台 PC 通过串口和嵌入式设备通信，负责对嵌入式设备的压缩过程进行控制和观察执行结果。测试分为 5 步。

1）测试任务调度延迟时间

调度延迟时间的测试方法[5]：在实时任务执行前记下当时的时刻 t_1，让实时任务睡眠一段时间 T，再记下任务醒来后的时刻 t_2，那么 $delay=t_2-t_1-T$ 就是所要测试的调度延迟时间。测试 1000 次，延迟时间分布的测试结果如图 3 所示。

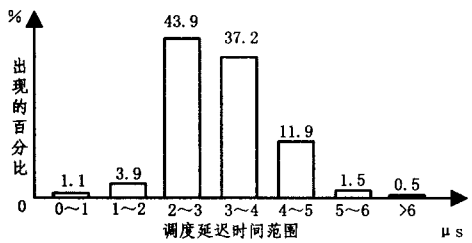

图 3　调度延迟时间分布（1 千实验）

从测试结果看，改进后的 RTAI 调度器的调度时延集中在 2μs～5μs 之间，而超过 6μs 的比例已经非常之低。可见改进后 Linux 调度时延相当小，具有优良的硬实时性能。

2）测试缓冲区大小对其溢出的影响，结果如表 1 所列。

表 1　序号比较及缓冲区溢出和 RTF0 缓冲区大小的关系

RTF0 大小	序号比较错误	RTF0 溢出	RTF2 溢出
1024	293	79	0
2536	275	56	0
2048	276	1	0
2560	267	1	0
3072	280	0	0
4096	255	0	0
5120	280	0	0

由表 1 可知，RTF0 到 2048 字节之后，RTF0 的溢出错误明显减少，到 2560 字节左右，只有 1 次溢出，但是不能完全保证没有溢出。直到为 3072 字节大小，才没有出现溢出错误。另外不管 RTF0 是否溢出，序号错误的值基本上很相近，这是因为和序号传递相关的 RTF2 没有溢出。RTF2 大小为 10 字节，而一个序号仅占用 2 个字节，这样虽然压缩数据可能会产生溢出，但它们的序号还是可以正常地被接收和处理。

3）测试在不同的统计时间长度内缓冲区溢出的情况，验证实时数据传输的稳定型。根据第一步的测试结果，把缓冲区 RTF0 的大小固定为 5120 字节，结果如表 2 所列。

表 2　缓冲区溢出和统计持续时间的关系

时间跨度	序号比较错误	RTF0 溢出	RTF2 溢出
30 秒	569	0	0
60 秒	312	0	0
90 秒	312	0	0
120 秒	310	0	0
180 秒	316	0	0

由表 2 可知即使延长了统计时间，在 RTF0 为一个合适大小（本例为 5120 字节）情况下，系统不会出现溢出现象，也即系统可以稳定可靠地传输实时数据。

4）在缓冲区 RTF0 固定大小（尽可能取大一些，本例取 5120 字节）时，改变序号错误计数器的计数方式，测试实时数据滞留在缓冲区的数量（以帧为单位）。结果如表 3 所列。

表 3　不同比较情形下序号比较错误发生次数

发送序号-接受序号	序号比较错误	RTF0 溢出	RTF2 溢出
＞1	127	0	0
＞2	0	0	0
＞3	0	0	0

由表 3 结果可知在最坏的情况下（即表第 1 栏所处的情况）接收方的序号只比发送方小 2，这意味着在 RTF0 中，最多只有 2 个完整数据帧等待处理。这个结果可以用来帮助用户确定 RTF0 的大小。这个 RTF0 可以设置得稍微大一些。考虑到至少有 2 个完整帧。本例中 1 个帧的大小是 2048 字节，再加上辅助数据所占的空间，将 RTF0 设置为 5120 字节大小是合适的。

这个测试步骤实际上提供了一个实时系统中测算先进先出（FIFO）缓冲区大小的有效方法。在 RATI 和 Linux 之间，RTF0，RTF1，RTF2 等缓冲区都是用于实时数据传输的 FIFO 类型的缓冲区。

5）测试视频实时压缩和解码播放的直观效果

经过压缩后的 MPEG 码流（4CIF 分辨率，帧率 25～30 帧/秒）通过网络传递到解码端（PC 机、软解压、realplay 或 media player 等常见媒体播放工具）能够流畅播放，没有出现明显的迟滞和马赛克现象。这表明系统内实时数据传输没有出现数据丢失和明显的延迟。

结束语　通过研究开源的 Linux 操作系统，提出了一种基于实时硬件抽象层（RTHAL）的方式增强 Linux 操作系统实时性能的方法，通过少量修改 Linux 内核而获得完全的硬实时性能，平均任务调度延迟仅为 3μs 左右。本文还提出了在 RTHAL 和 Linux 之间采用 FIFO 缓冲区方式实现实时数据传输时，设置缓冲区大小的有效方法。在一个基于 MPEG4 的实时流媒体系统中采用该方法，结果表明该方法既能充分发挥 Linux 现有功能又能有很强的实时能力。同时测试结果表明只要合理设置操作系统和 RTHAL 之间 FIFO 缓冲区的大小，就能够实现实时情况下的稳定数据传输。

参 考 文 献

[1] 廖敬萍. 基于 Linux 的实时解决方案分析[J]. 现代电子技术，2006(19)：34-39

[2] 褚敏芬. RTLinux 调度策略的研究[J]. 微计算机信息，2003，19(11)：90-92

[3] Wu Dapeng. Streaming video over the Internet：approaches and directons[J]. IEEE Transations on Circuits and Systems for Video Technology，2004，11(3)：282-300

[4] IME6400 MPEG4/2/1 Encoder Hardware Reference Manual. InTime Corp.，Korea，2002

[5] 罗蕾. 嵌入式实时操作系统及应用开发[M]. 北京：北京航空航天大学出版社，2005