DOI :10.19652/j.cnki.femt.2008.08.016

Linux 下高速卡驱动的高效实现

孟 芳 杨明欣 余贞侠
(成都信息工程学院 成都 610225)

摘 要:本文针对测控系统中的数据采集模块,讨论基于 Linux 操作系统的 PCI 高速实时数据采集卡设备驱动的高效实现方法。以数据交换量大且实时性要求高的视频图像采集卡为实例,提出了一种优化的数据传输机制——优化 DMA 方式和 MMAP 内存映射,该方法能在提高视频数据吞吐量的同时降低 CPU 系统资源的占用率。

关键词:数据采集;设备驱动;直接存储器存取 DMA;内存映射 MMAP
中图分类号:TP311.52 文献标识码:A

Efficient implementation of high speed data acquisition device driver for Linux

Meng Fang Yang Mingxin Yu Zhenxia
(Chengdu University of Information Technology, Chengdu 610225)

Abstract: This paper delves into the data acquisition module of measurement and control system, aim at highly efficient implementation of high speed data acquisition card of PCI device driver for Linux. An example of video capture card with an emphasis on the optimized data transfer mechanism of optimized DMA and MMAP was provided in this paper. It not only enhances the system throughput but also avoid high CPU utility.

Keywords: data acquisition; device driver; direct memory access; memory mapping

0 引 言

以虚拟仪器替代传统仪器组建自动测试系统的迅速发展,促进了集测量、控制和现场监控于一体的远程测控系统的发展。对接入对象实施远程测控需要采集现场景物视频图象以及对有形对象的图象进行采集和处理,带有 DSP 功能的数据采集系统能够满足监测系统高速和实时性的要求,而高速数据采集卡是数据采集系统的主要组成部分,其性能直接影响了整个系统的性能。

高精度、高速度、稳定可靠及实时性是数据采集技术的发展方向,而 Linux 是一种完全公开源代码,具有很强的稳定性、实时性和嵌入性的操作系统[1]。因此基于 linux 平台的高速数据采集卡能够满足数据采集系统的稳定性、实时性及低单位成本等多方面的要求。本文以数据交换量大以及实时性操作要求高的带有 DSP 的视频图像采集卡为实例,讨论了 Linux 下高速数据采集卡设备驱动的高效实现,对在 Linux 下其他总线设备的开发具有借鉴意义。

1 相关的工作和目标

Linux 设备驱动程序工作在内核空间,不但要负责内核空间和用户空间的数据传输,同时要负担内核和硬件的数据交换[2]。高速的数据采集系统在采集数据流时最常见的问题是数据丢失或采集后数据流不完整。高效的设备驱动必须确保高效的数据通信性能。在设备驱动中,获得高效数据通信性能的关键技术就是直接存储器存取 DMA 方式和 MMAP 技术[3-4]。

本文的目标是在设备驱动中设计并实现更为优化的数据传输机制——优化 DMA 方式和 MMAP 内存映射。以数据交换量大以及实时性操作要求高的视频压缩卡为实例进行了性能测试。该视频压缩卡的数据传输机制如图 1 所示。在测试 DMA 方式时,利用了 DSP 自带的 DMA 控制器,自动地从处理器来回传送数据,同时采用邮箱中断方式协调总线主控器与 PCI 设备之间的数据传送。在测试 MMAP 技术时,分别采用 read 调用和 mmap 方法完成数据在内核空间与用户空间中拷贝并比对了这 2 种方法。

作者简介:孟芳(1973-),女,山东高青人,讲师主要从事系统可靠性,嵌入式技术方向的研究工作。

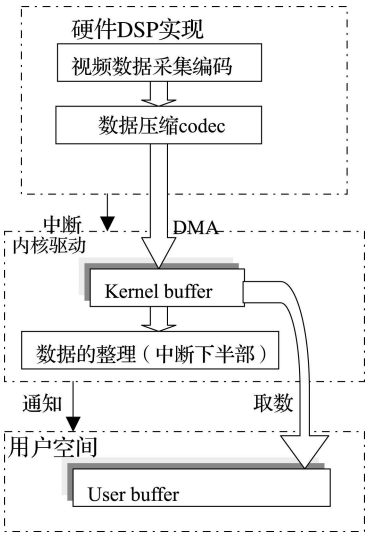

图 1　PCI 视频压缩卡的数据传输机制

2　DMA 数据传输

理论上，采用 PCI 总线的 DMA 方式可以达到 132 MB/s的数据传输率[3]。为了测试并得到实际应用中 DMA 方式的数据传输率，以图 1 中硬件 DSP 向主机传输数据为例，性能测试平台采用标准的 PC 机（Pentium Pro CPU200 MHZ，128 M 内存），主机操作系统为 linux（2.4 内核），实验所用的视频压缩卡驱动程序采用 gcc 编译。测试所需的操作步骤如下：

（1）主机内存分配一个适合 DMA 传输的缓存。

（2）驱动程序中通过写寄存器操作初始化 DMA（将 PCI 控制器所需要的参数写入 PCI 控制器特定的状态寄存器，包括主机 DMA 缓存的起始地址及长度、DMA 传输方向等）。

（3）PCI 控制器将数据从 DSP 传送到主机 DMA 缓存。

（4）主机 DMA 缓存满时会收到来自 DSP 的一个信号（例如中断）用于清空 DMA 缓存。

（5）主机将数据拷贝到适当的接收缓冲区以清空 DMA 缓存。同时 PCI 控制器再次用新数据填满主机 DMA 缓存。

接下来讨论上述操作步骤在驱动程序中的具体实现。

2.1　DMA 缓冲区的管理

要实现 DMA，设备驱动程序必须分配一个或者更多的适合 DMA 的特殊缓冲区。如果要求分配的 DMA 缓冲区大于一页，它必须占据物理内存中的连续页，这时就可能遇到系统内存碎片问题[5]，这种情况下请求会失败。在执行分配时采用轮询的方式不断请求分配将导致高的系统负荷，造成系统死锁。这时可以在代码中调用 kmalloc（GFP_ATOMIC）直到失败为止，然后等待内核释放若干页面后再一次进行分配。如果密切注意已分配的页面池，迟早会发现由连续页面组成的 DMA 缓冲区已经出现。这时候使用内核定时器释放除了被选中缓冲区之外的每一页。

在成功完成 DMA 缓冲区的分配后，需要完成 DMA 映射，就是为该缓冲区生成一个能够被设备访问的地址的组合操作。DMA 映射用 dma_addr_t 来表示总线地址，调用 pci_alloc_consistent 设置一致 DMA 映射[6]并处理缓冲区的分配和映射；调用 pci_free_consitent 将缓冲区返还给系统。注意被一致 DMA 映射的缓冲区必须同时可被 CPU 和外围设备访问，因此需要提供 CPU 地址和总线地址。2.4 内核包含了支持 PCI DMA 的总线控制，它处理缓冲区分配的细节，也能够为支持多页传送的硬件进行总线硬件设置。本例中的函数需要一个 struct pci_dev结构，例程如下：

```
#define MAX_KERNEL_BUFFER_SIZE 4 *110 *1024
struct virt_and_bus
{
unsigned long cpu_addr; //virtual address
dma_addr_t dma_handle; //physical address
};
int ReadData(struct pDev_ext * pde, char * pStream,unsigned long *pFrameLength)
{
struct virt_and_bus data_buffer;
int size_d=MAX_KERNEL_BUFFER_SIZE;
data_buffer.cpu_addr=(unsigned long) pci_alloc_consistent(pde->pdev, size_d, &data_buffer.dma_handle);
if(!data_buffer.cpu_addr)
……
else
writel(data_buffer.dma_handle, dev->PortBase+(nOffset &0x3fffff));
……
```

2.2　DMA 传输的实现及优化

DMA 传输可以通过 2 种方式触发，软件请求数据和硬件异步传输[7]。考虑到硬件异步传输的 DMA 方式的效率远远高于软件请求数据的 DMA 方式，主要讨论用硬件异步传输的 DMA 方式实现 DMA 传输。

硬件异步传输的 DMA 方式的实现过程概括为：事先在内核中分配好 DMA 缓冲区，然后将其物理地址告知设备，DSP 硬件设备将采集到的视频数据压缩编码后一帧一帧地持续 DMA 到内核分配的 DMA 缓冲区 kernel buffer 中（即使没有任何读进程），每放入一帧，就发出中断通知驱动有新的数据到达。驱动收到中断通知后，调用中断服务子程序，对 kernel buffer 中的数据进行整理和处理。这时只要通过 read 调用就能将所有累积的数据取回到用户

空间。

为了分析、评估 DMA 方式的数据传输率，我们将上述实现过程作了修正，首先修改了 DSP 代码部分，在 DSP 的 SDRAM 中开辟 2 M 堆空间用于存放发送到主机的数据；其次在驱动程序中通过 ioctl 向应用程序提供控制 DSP 的接口；最后向 DSP 特定位置写命令参数启动和终止传递数据。测试采用普通 DMA 和优化 DMA 分别从 DSP 中 SDRAM 的数据缓冲区发送 64 K 字节到 PC 中 DMA 缓冲区。

普通DMA，即所有代码使用高级语言，主机驱动收到中断后即对数据进行处理，不采用任何任务队列。从表 1 可以看出，普通 DMA 方式实现的数据传输远远小于 132 MB/s。

本文在对普通 DMA 的实现过程作进一步分析和研究后，对代码作了优化。首先 25% 的通信时间用于从 DSP 到主机的数据传输的中断服务。Pentium Pro 的 CPU 派发中断到内核中断处理程序大约需要 10μs。高效 DMA 处理依赖于中断报告，设备驱动程序在它的中断处理例程中应该做尽可能少的工作，使得 Linux 核心可以尽可能快的结束中断并返回，收到中断后需要做大量工作可以使用下半部或者任务队列实现[5]。进行代码优化时，采用中断的上半部完成任务队列排队，中断的下半部分完成对 kernel buffer 中的数据整理和处理。其次 Linux 系统对于信号实现的代价也是比较大的，高负载的时候应尽量减少信号的使用，通过互斥锁来协调多个进程的操作也能提高 DMA 效率。最后从编译器的角度来看，采用手工汇编方式优化代码的效率比通过程序编译器优化的效率高。采用汇编语言优化代码主要集中在以下方面：

(1)优化寄存器的使用。

(2)降低循环体的复杂性，将不必要的计算移出内循环，提高循环体效率。

(3)采用硬件轮询。

(4)通过指令的调度消除流水线的迟延。

表 1 为普通 DMA 和优化后的 DMA 的通信性能的测试结果。

表 1 **DMA** 方式的通信性能

传输方式	传输数据长度/ B	传输时间/ ms	传输数率 MB/ s
普通 DMA	65 536	19.72	3.245
优化 DMA	65 536	1.901	33.67

3 MMAP 技术的应用

在驱动程序的设计中，另一种提高系统性能的途径就是使用内存映射 MMAP 技术。它的优势在于用户不需使用 copy_to_user 和 copy_from_user 将数据在内核空间与用户空间中拷贝[8]。

用 MMAP 技术开发了基于 TI 公司的 TMS320DM-642 的视频数据采集卡的驱动程序，并将它同传统的 read 调用方法作了对比。

实现 MMAP 技术，必须在映射设备进程的地址空间中填充一个 VMA 结构体 vm_area_struct，VMA 虚拟内存区域是进程虚拟内存中的一个同构区间，一个具有相同许可标志的地址的连续范围[8-10]。内核需要维护 VMA 链表和树以便优化对区域的查询，而 vm_area_struct 的几个成员则用来维护这种组织形式。因此，驱动程序不能随便地生成 VMA，否则结构体会遭到破坏。构造用于映射一段物理地址的新页表的工作由 remap_page_range 完成，由于程序访问设备的动作总是依赖于设备，它知道如何使被映射的内存区域有意义，当区域大小不是页大小的整数倍时，内核可通过生成一个稍微大一些的区域来调节页面大小粒度。以图 1 所示为例，用 MMAP 方法将 kernel buffer 从内核空间映射到用户空间，然后直接在应用程序中完成压缩数据的整理。驱动例程如下：

```
int dvr_mmap(struct file *flip, struct vm_area_struct *vma)
{
stuct pDev_ext *pDe;
unsigned long offset, size;
……
offset = vma->vm_pgoff<<PAGE_SHIFT;
size = vma->vm_end - vma->vm_start;
if(offset & ~PAGE_MASK)
……
else
……
if((vma->vm_flags & VM_WRITE) &&!(vma->vm_flags & VM_SHARED))
  return -EINVAL;
vma->vm_flags |=VM_LOCKED;
if(offset ==0)
{
if(remap_page_range(vma->vm_start, pDe->data_buffer.dma_handle, size, PAGE_SHARED))
  return -ENXIO;
}
……
}
```

下述为分别采用 read 调用和 MMAP 方法将数据从内核取回到用户空间 2 种结果的比较：

(1)在使用 read 调用将数据取回到用户空间时，数据交换量突然增大时，出现 CPU 的占用率达到峰值的情况。使用 MMAP 方法后，没有出现过这种情况。

(2)MMAP 技术通过 remap_page_range 将分配给 DMA 缓存区的内核空间映射到用户空间，比传统的 read 操作减少了一次内核态到用户态的拷贝。经测试，数据交换速度最高时候提高了 5 倍左右。

(3)MMAP 系统调用同 read 系统调用的实现相比,前者在调用之前内核不需要做太多的工作。后者需要内核完成很多工作。从这一点来看,MMAP 方法除了大量的系统开销。

表 2　基于优化 **DMA** 和 **MMAP** 的驱动

视频输入路数	CPU 占用率/%	内存使用情况	延时/ms
4	1~2	稳定,无峰值	
8	1~2	稳定,无峰值	＜200
16	5~7	稳定,无峰值	
32	15~20	稳定,无峰值	＜300
64	35~40	稳定,无峰值	＜500

4　结束语

设备驱动程序作为数据采集模块的重要组成部分,其性能直接关系到数据采集模块的效率。在设备驱动中采用优化 DMA 方式和 MMAP 内存映射能在提高数据吞吐量的同时降低 CPU 系统资源的占用率。在基于 TMS320DM642 的视频数据采集卡的驱动程序设计中同时采用优化 DMA 方式和 MMAP 内存映射,在视频输入路数达到 64 路时,CPU 占用率为 35%~40%,内存占用值稳定,没有出现峰值报告。压缩编码数据解码播放时没有发现丢帧和马赛克现象。其相应测试结果如表 2 所示。实验表明,优化的 DMA 方式由于有效地提高了系统的中断服务效率,具有具有低延迟、低开销和高带宽的数据处理能力。采用 MMAP 技术实现数据交换能够在提高数据通讯吞吐量的同时有效地降低系统负载和提高 CPU 的利用率。

参 考 文 献

[1] RUBINI A. Linux device drivers[M]. 2nd Edition. USA: O'Reilly, 2001.
[2] 李善平. Linux 内核 2.4 版源代码分析大全[M]. 北京: 机械工业出版社, 2002.
[3] TOMSHANLEY, DONAERSON. PCI 系统结构[M]. 4 版. 北京: 电子工业出版社, 2000.
[4] 毛德操, 胡希明. Linux 内核源代码情景分析[M]. 杭州: 浙江大学出版社, 2001.
[5] RUBINI A, JONATHAN. Corbet Linux device drivers[M]. 2nd Edition. USA: O'REILLY, 2002.
[6] 姜明华, 周敬利, 黄晓涛. Linux 下加密卡驱动程序的开发与性能分析[J]. 计算机工程, 2004, 30(16): 185-187.
[7] 王乐, 张晓彤, 李磊, 等. Linux 下的 DDR DIMM 总线接口设备检测方法[J]. 计算机工程, 2007, 33(18): 256-258.
[8] 陈俊楷, 冯穗力, 叶梧. Linux 下 PCI 设备驱动程序研究[J]. 计算机应用研究, 2002, 19(11): 23-26.
[9] 王渊, 赵宇. 嵌入式 Linux 网络通信的实现[J]. 电子测量技术, 2006, 29(6): 94-97.
[10] 陈大庆, 高雷, 韩九强, 等. 基于 PCI 总线的雷达虚拟 A/R 显示卡研制[J]. 仪器仪表学报, 2007, 28(3): 560-564.