Windows Mobile 平台上 H. 264 解码器的优化*)

赵 鹏 周 兵

(郑州大学信息工程学院 郑州 450001) (郑州大学河南省信息网络省重点学科开放实验室 郑州 450052)

摘 要 H. 264 视频压缩标准的出现为在较低带宽上提供高质量的图像传输提供了更好的保障,而且对无线网络传输提供了更好的支持功能。但由于 H. 264 计算量较大,因此对于如何在计算和存储能力都很有限的移动设备上实现实时解码一直是一个研究的热点。本文探讨了基于 Windows Mobile(WM)平台的 H. 264 解码器的优化方法,根据解码器及 PLA270 处理器的特点,提出了 C 语言与 WMMX 优化的具体方法。实验证明,经过优化的解码器基本达到了手机视频监控应用的要求。

关键词 H. 264,解码器,Windows Mobile,WMMX

The Optimization of H. 264 Decoder on Windows Mobiles

ZHAO Peng ZHOU Bing

(School of Information Engineering, Zhengzhou University, Zhengzhou 450001)

(Henan Provincial Information & Network Key Laboratory, Zhengzhou 450052)

Abstract The emerging of a new video compressing code H. 264, which provides a better support for wireless video-transmitting, further guarantees the transmitting the enhanced picture in low bit-rate. But due to the high complexity of computation, it is a hotspot as to how to put the new decoder into applies. This paper discusses in detail several ways of optimization of h. 264 decoder on windows mobiles. According to the characteristics of the decoder and the PLA270 center processor, we suggest a way of optimization using C and WMMX. It is demonstrated after experiments that the optimized decoder could meet the basic requirements of in-time surveillance.

Keywords H. 264, Decoder, Windows mobile, WMMX

1 引言

由于移动设备性能的局限性和 H. 264 计算复杂性的增加[1],使得在移动设备上利用 H. 264 实现实时解码很困难。目前大多数解码器优化方法主要针对 PC 机平台[2~4],而对于采用嵌入式处理器的移动平台则需要探讨新的优化方案。

WMMX 是 Intel 为增强其处理器的多媒体处理能力而提供的无线多媒体扩展指令集[8]。其 SIMD(Single Instruction Multiple Data)功能使得处理器可以在 1 个时钟周期内对多条数据进行流水操作,提高了操作处理的并行性。本文介绍了几种通用的 C 语言程序的优化方法,针对嵌入式处理器 Intel ® PLA270 提出了利用 WMMX(Wireless MMX)提高程序并行性和执行效率的优化过程,最后给出了优化前后的结果对比。实验证明经优化后的解码器可基本满足视频监控应用的基本需求。

图 1 为解码器的流程图。由解码器的流程可以看出,完成熵解码和重排序以及反变换反量化之后,数据经过抗块效应滤波器进行帧重建输出,同时对参考帧还要缓存到参考帧队列。参考帧队列中的帧对解码后续帧提供运动补偿(其中包括 1/4 像素内插)或帧内预测参考。其中抗块效应滤波器是解码器中最耗时的一部份。尽管采用了各种优化算法,解码器的计算复杂度仍有近 1/3 来自于抗块效应滤波器[5],除此之外解码器耗时最长的模块主要集中在 1/4、1/8 线形内插以及反变换与反量化方面[6,9,10]。

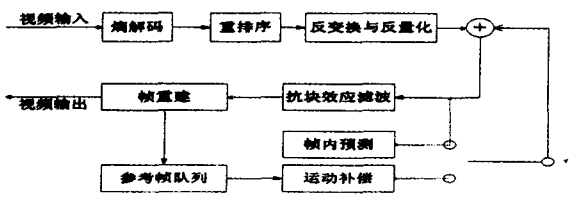

图 1 H. 264 解码器流程图

2 解码器的优化过程

2.1 解码器性能瓶颈分析

通过对 QCIF 格式的 foreman_qcif、mobile 和 akiyo_qcif 等与视频监控应用较为相似的几个典型视频序列的分析可以发现解码器主要端的效率瓶颈分别存在于去块效应(filter_mb)、运动补偿(hl_motion)和反变换反量化(idct)三个函数。因此优化的重点主要放在这些模块上。

表 1 解码器端最耗时的几个函数。单位:ms

foreman_qcif (QP=28)		Mobile (QP=28)		akiyo_qcif (QP=28)	
函数名	执行时间	函数名	执行时间	函数名	执行时间
filter_mb	1629	filter_mb	1012	hl_motion	1085
hl_motion	1007	hl_motion	582	filter_mb	433
idct	73	idct	148	idct	51

*)受河南省教育厅高校杰出人才创新工程资助(20050901)。赵 鹏 硕士研究生,研究方向:视频监控;周 兵 教授,研究方向:视频监控。

2.2 优化方法与实现

除了算法本身的效率外，本文还采取了C语言层级的优化方法，主要有[7]：

(1)循环中出现的重复计算，每次计算结果不发生变化的表达式，应当将计算移到循环之外完成以消除循环的低效率；

(2)可以适当展开循环，将多次循环的工作合并到一次中完成以减少循环次数降低循环开销；

(3)减少过程调用；

(4)尽量消除不必要的存储器引用以减少对内存的读写次数。

针对支持WMMX的处理器(比如：PLA270)，使用SIMD汇编指令及流水线优化方法以提高数据操作的并行性，从而提升程序效率[8,10]。

2.2.1 减少过程调用

在解码foreman_qicf前30帧的过程中，解码器总共产生了1560237次函数调用，因此函数调用的开销不可忽视。图2显示了调用次数比较多的几个函数及其调用次数。这些经常被调用的函数都相对比较短小，因此比较适合采用inline关键字。

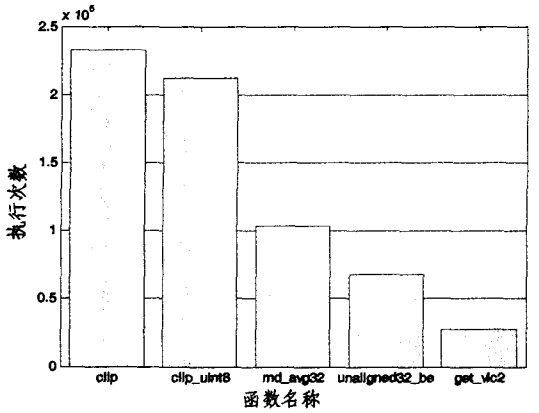

图2 执行次数最多的五个函数

2.2.2 增加数据操作的并行性

图形图像处理中的多数算法都涉及对相当数量的数据元素迭代相同的操作，这些重复的操作为并行计算提供了可能。将这些低效的每次只对1个、2个、4个字节进行处理的重复操作使用具有SIMD功能的指令(WMMX技术)进行改写及进一步的流水线优化即可简单的通过增加数据操作并行性减少迭代次数而使程序效率上升到一个新的层次。

h264_add_idct_c函数中有如下代码段：

```
static void h264_add_idct_c ( … unsigned short * block, … ){
… …
for(i=0; i<4; i++){
        const int z0= block[i + 4 * 0]+ block[i + 4 * 2];
        const int z1= block[i + 4 * 0]- block[i + 4 * 2];
        const int z2= (block[i + 4 * 1]>>1) - block[i + 4 * 3];
        const int z3= block[i + 4 * 1]+ (block[i + 4 * 3]>>1);
    … …
}
```

h264_add_idct_c函数完成对4×4大小宏块的快速反变换与反量化。上述C语言代码共产生4×8次内存访问。另外有4×2次加法和4×2次减法操作。经WMMX指令优化后4次循环操作一次即可完成。总共只需4次内存访问和2次加法、2次减法操作。H.264中类似循环取数并相加减的代码段有很多，很适合采用WMMX优化。

使用WMMX改写：

```
… …                                  @ R1: block
WLDRD   wR1, [R1]                    @ block[i+4×0] i∈{1,2,3,4}
WSRAHG  wR10, wR1, wCGR1
WLDRD   wR2, [R1, #8]                @ block[i+4×1] i∈{1,2,3,4}
WLDRD   wR3, [R1, #16]               @ block[i+4×2] i∈{1,2,3,4}
WADDH   wR5, wR1, wR3                @ z1
WLDRD   wR4, [R1, #24]               @ block[i+4×3] i∈{1,2,3,4}
WSRAHG  wR11, wR4, wCGR1
WSUBH   wR6, wR1, wR3                @ z2
WSUBH   wR7, wR10, wR4               @ z3
WADDH   wR8, wR2, wR11               @ z4
… …
```

3 实验结果

3.1 减少过程调用

通过对经常调用的函数采用inline扩展后的效果如图3所示，由此可见内联函数在此发挥的作用。采用内联扩展的函数由于没有了函数调用的开销，其执行时间最多降低了80%。

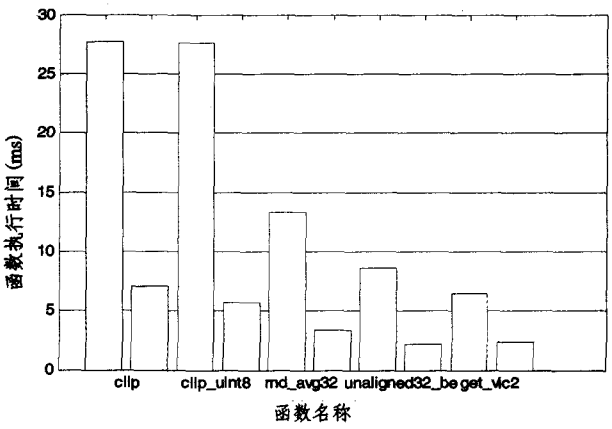

图3 优化前后各函数执行时间对比

3.2 使用WMMX

表2给出了采用WMMX优化前后系统性能对比，对于运动剧烈的foreman_qicf序列，其优化结果最为理想，优化后的解码器运行时间仅为优化前的48%，对于画面变化不大的akiyo_qicf序列性能也有了明显的提升。图4为解码器器运行效果截图。经过优化的解码器解码速率可以达到15～20帧每秒，基本上满足了视频监控的需要。

表2 WMMX优化前后解码器性能对比

foreman_qcif (QP=28)		Mobile (QP=28)		akiyo_qcif (QP=28)	
优化前	优化后	优化前	优化后	优化前	优化后
3.03s	1.44s	4.93s	2.59s	1.69s	1.17s
性能提高	2.10x	性能提高	1.90x	性能提高	1.44x
平均性能提高		1.81x			

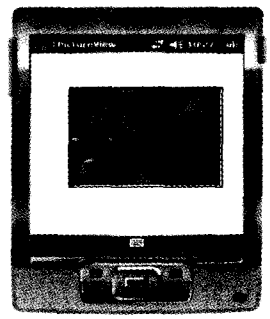

图4 解码器运行效果截图

(下转第64页)

文件系统需要管理磁盘资源，为用户提供一致的访问接口。在BWFS下，MS管理系统的元数据，而这些元数据和普通数据一样存放在SN上；标准NBD已具备了普通数据的访问协议，还不具备元数据访问协议，因此我们需要为ENBD增加元数据访问协议。

我们称标准NBD提供的数据访问协议为普通协议，普通协议与特殊协议可以通过I/O请求(request)的块号域加以区分，定义如表1。

表1 Enbd driver协议块号定义表

块号(bno)	用途
0—3	读写超级块
4	申请空闲block资源
5	申请空闲inode资源
6	读指定的inode
7—17	保留块号
18	释放空闲block资源
19	释放空闲inode资源
20	写指定的inode
21—25	保留块号
26	获取指定资源组的资源统计信息
27	获取系统所有资源组的信息
28	获取指定资源组的属性信息
29—127	保留块号
128—	普通数据的读写块号

ENBD的普通协议与标准NBD协议一致，包括所有请求块号不小于128的读写请求，以及请求块号在0-3的超级块请求，其他位于4-127的协议号均为特殊协议。

ENBD驱动和enbd_server之间网络请求enbd_request的结构如表2所示。

表2 enbd_request结构

成员	用途
magic	enbd请求幻数标识，ENBD_REQUEST_MAGIC = 0x25609513
type	enbd读写请求标识，0 = READ || 1 = WRITE
handle[8]	enbd请求的序号标识，通常为请求的request内存地址
from	读写偏移，以字节计算
len	读写长度，以字节计算

响应enbd_reply的结构如表3所示。

表3 enbd_reply结构

成员	用途
magic	enbd应答幻数标识，NBD_REPLY_MAGIC = 0x67446698
error	enbd请求结果正确标识，0 = ok，else error
handle[8]	enbd请求的序号标识，来自对应请求的handle部分

有了网络请求和响应的结构，我们就可以具体地看一个特殊协议是如何使用它们来完成元数据访问的。块号为4的请求，目的是为了向SN申请空闲的block资源。其enbd_request请求的格式为：

```
magic = NBD_REQUEST_MAGIC;
type = READ;
handle = &req;
from = 4 * 512;
len = gid(资源组标识);
```

由于request的块号域是以扇区为单位的，因此上面格式中的from的大小应为块号域的值再乘上扇区的大小。SN收到上述请求后，会返回一个响应包，如果握手正确，SN会再向ENBD返回空闲的块资源。

ENBD就是通过以上方式来实现对存储系统的元数据操作的，其他元数据访问协议实现方法类同。

总结 本文简单介绍了标准NBD，阐明了由于其功能不完全满足蓝鲸集群文件系统的需求，需要对其进行扩展，随后重点分析了如何扩展该软件以支持蓝鲸集群文件系统对多存储服务器支持、资源路径查询、元数据访问三个特性的需求。上述工作对于利用开源软件进行复杂系统集成或开发有指导意义。

下一步的工作是分析在现有NBD扩展基础上蓝鲸集群文件系统的整体性能。

参考文献

1 Machek P, Network Block Device. http://atrey.karlin.mff.cuni.cz/~machek/nbd/nbd.html

2 杨德志，黄华，张建刚，许鲁. 大容量、高性能、高扩展能力的蓝鲸分布式文件系统. 计算机研究与发展，2005(6)：1028～1033

3 Ares P T, The Network Block Device. http://www.linuxjournal.com/article/3778

4 Novell Storage Services File System Administration Guide for NetWare 6.5. http://www.novell.com/documentation/nw65/index.html? page =/documentation/nw65/nss_enu/data/hn0r5fzo.html.

(上接第61页)

结束语 本文介绍的解码器实现方案适用于运行Windows Mobile 5.0平台的PPC和Smartphone设备，WMMX优化方案仅适用于支持Intel ® WMMX指令集的PLA27x处理器系列。该方案针对嵌入式移动视频监控系统的要求而研制，同样适于手机视频点播和手机电视应用的视频解码器优化。

参考文献

1 Richardson I E G. H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia [M]. America: John Wiley & Sons, Ltd, 2003

2 刘鹏飞，刘志，安平，等. H.264/AVC解码器优化的研究[J]. 中国图象图形学报，2006，11(11)：1627～1630

3 Xi Yinglai, Hao Chongyang, Lai Changcai. Fast Sub-pixel Motion Estimation Techniques Having Lower Computational Complexity[J]. 电子科学学刊(英文版)，2006，23(6)：873～876

4 魏芳，李学明. 基于MMX技术的H.264解码器优化[J]. 计算机工程与设计，2004，25(12)：2218～2224

5 List P, Joch A, Lainema J, et al. Adaptive Deblocking Filter[J]. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, 2003, 13(7): 614～619

6 Lappalainen V, Hallapuro A, Hamalainen T D. Complexity of Optimized H.26L Video Decoder Implementation [J]. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, 2003, 13(7): 717～725

7 Bryant R E, O'Hallaron D. Computer Systems A Programmer's Perspective [M]. America: Pearson Prentice Hall, 2003

8 Intel Wireless MMX Technology Developer Guide[M]. America: Intel Corporation, 2002

9 Lindroth T, Avessta N, Teuhola J, Seceleanu T. Complexity Analysis of H.264 Decoder for FPGA Design[J]. ICME, 2006: 1253～1256

10 Lee J, Moon S, Sung W. H.264 Decoder Optimization Exploiting SIMD Instructions[C]. IEEE Asia-Pacific Conference on Circuits and Systems, 2004