文章编号: 1001- 2486(2009) 01- 0099- 05

基于多线程的 CABAC 并行编码方法*

陈胜刚, 孙书为, 陈书明
(国防科技大学 计算机学院, 湖南 长沙 410073)

摘 要: CABAC(Context-based Adaptive Binary Arithmetic Coding) 是 H. 264/AVC 主要档次以上配置中推荐采用的熵编码方法。相比其他的熵编码方法, CABAC 能够节省编码码率, 但其计算串行性强, 不能较好适应片上多核环境。针对这一问题提出了基于多线程的 CABAC 并行编码方法。大量实验统计表明, 该方法负载划分较为均衡, 对序列熵编码的单独加速比最高可达 1. 78。

关键词: CABAC; 并行化; H. 264/AVC
中图分类号: TP368. 1 文献标识码: B

A CABAC Parallel Encoding Method for Multithreading

CHEN Sheng-gang, SUN Shu-wei, CHEN Shu-ming
(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: The context-based adaptive binary arithmetic coder (CABAC) is the preferred entropy encoding method in the main profile of H. 264/AVC, for it can offer more bit rate saving than other entropy coding methods. However, high computational complexity and poor parallelism make it unsuitable for chip multicore processors application. A new CABAC parallel encoding method is proposed for the troubleshooting. Results from experiments show that the proposed method can indeed strike a near balanced workload partition while achieving a peak speedup of 1. 78 when solely applied.

Key words: CABAC; parallelism; H. 164/AVC

H. 264/AVC 是国际上最新流行的视频编解码标准。为了提高视频编码压缩效率和编码图像质量, H. 264/AVC 引入和加强了一系列新技术, 这些技术组成了 H. 264 的工具集[1]。CABAC(基于上下文的自适应二进制算术编码) 是 H. 264/AVC 主要档次以上的档次中推荐采用的熵编码方式。CABAC 能够更加有效地去除语法序列中的表示冗余, 据统计, CABAC 熵编码的效率可以比 CAVLC(Context-base Adaptive Variable Length Coding) 高出 10% ~ 15%[2]。但是, CABAC 计算过程是按二进制 bit 顺序进行, 串行化要求高, 导致计算复杂度增大。

随着 VLSI 技术的发展, 单片集成晶体管数目已经超过 10 亿, Intel Tukwila 处理器的集成晶体管数目已经达到 20 亿; 单片多核(CMP, Chip Multcore Processors) 也已经成为了微体系结构主流设计方法。要充分发挥 CMP 的效能, 必须提供足够多的并行线程。因此, CABAC 编码并行化是硬件体系结构发展的必然要求。

1 相关工作

为了提高 CABAC 的计算速度, 研究者从专用硬件结构和符号并行等方面提出了很多方法。Osorio[2] 等发现, EQ(EQually Probable Symbols) 符号可以与接下来的 LPS(Least Probability Symbol) 或者 MPS (Most Probability Symbol) 同时编码, 一对 EQ 符号也可以同时编码。另外, 通过降低频率, 可以在一个周期编码两个符号, 从而提高系统吞吐率。结果发现, 系统的最大吞吐率达每周期 4 个符号。在不采用 RDO 的情况下, 可以完成 QCIF、CIF 及 HD 画面的实时编码。Tian X H[4] 通过利用流水线的方式来减少

* 收稿日期: 2008- 07- 02
基金项目: 国家 863 计划资助项目(2007AA01Z108) ; 教育部“高性能微处理器技术”创新团队研究计划项目
作者简介: 陈胜刚(1981—) , 男, 博士生。

数据相关性,并设计了一种完全支持RDO的上下文存储器管理机制,能够达到每个时钟周期处理一个符号的速度。综合结果显示,在0.13nm的情况下达到620MHz的处理速度,并且比参考设计节省85%的空间。

Bin G[5]等人把概率更新的步长N调整到大于1个符号来加速编码。根据对不同的N进行的实验发现,在$N \geqslant 2$的情况下加速比都超过了6。不过遗憾的是,编码效率和图像质量也有很大的下降。Marpe G[6]等人提出了一种快速的归一化方法,对解码和编码分别提升24%和53%。Lin在文献[7]中,称第一次提出了设计并行算术编码的系统方法,对IBM Q-Coder和H.264中采用的CABAC进行了优化。在对H.264的CABAC的并行化中,Lin采用了一种查找表技术,并提出了一种新的并行化概率模型(QT-Coder)实现同时编码多个二进制符号。这种模型是现有编码概率模型的推广,其增加的表格数目随着N的增加会急剧增加。

前面提到的算法和实现方法并没有解决CMP环境下多个线程上如何分配CABAC编码任务的并行化问题。本文研究了如何合理划分CABAC编码任务,在多核系统上进行并行编码。

2 应用背景

多核体系结构设计哲学是利用众人拾柴火焰高的设计思想,把传统的运行于单个核心的计算任务分配给多个核心协同计算,使得运行于低时钟频率的硅片能够达到甚至超过传统高速芯片的性能。这抛弃了传统设计方法对时钟频率的追求,更多地考虑实际运行性能。根据多核摩尔定律的预测[8],到2010年,单片上集成的核的数据将会超过1024。汪东等人[9]所在团队设计实现的异构多核DSP处理器QDSP(Quad DSP)就是一款典型的多核SOC。QDSP包含一个32-bit的RISC控制核,4个同构的32-bit DSP处理核。4个同构DSP之间可以通过紧耦合的共享数据缓冲池(SCG-SPM)进行快速的数据共享,也可以通过高速多通道的EDMA进行大量的数据交换。但是,CMP的加速比能否跟随核的数目呈线性增加,主要取决于软件是否能够提供足够多的并行线程任务。本文并行算法就是要解决这一背景下CABAC的运行问题。

3 基于多线程的CABAC并行编码方法

基于多线程的CABAC并行编码方法突破了CABAC难以并行化的难题,其中最重要的就是如何在多核线程之间进行任务的划分方法。

3.1 基于语法元素分类的任务划分方法

H.264/AVC编码产生的的语法元素具备不同的概率统计特性,CABAC用不同的上下文模型来表征,并用上下文模型索引值进行寻址。H.264/AVC设置了上百种不同的上下文模型[10]。编码符号根据约定的规则产生模型索引查找上下文模型,获得对应概率模型的状态,进入算术编码引擎进行编码。

H.264/AVC还把这些上下文模型大致分为4个类别,每一个类别对应H.264/AVC码流结构中的某一部分的语法结构元素,相关性较强;不同类别之间的相关性则较弱。受此启发,本文提出了基于符号划分的任务划分方法。

任务划分的基本方法是根据H.264语法结构和上下文模型的相关性,对待编码语法元素进行归类,经过二进制化后形成不同类别的编码符号。划分的依据是语法元素在H.264码流结构中的位置是否便于分类处理、数据相关性处理是否合适以及负载是否均衡。经过分析后,本文把语法元素分为以下三类,形成了三类待编码符号,如表1所示。

表 1 语法元素分类

Tab. 1 Syntax elements categories

类 别	描 述
type iv	宏块类型, 预测模式和控制信息。主要包括的语法元素有宏块类型、Skip 信息、子宏块类型、运动向量、参考帧索引、帧内预测模式、编码模式和量化参数等
type ㈦	Coded _ Block _ Flag 以及残差的 Map 信息。包括的语法元素有 coded _ block _ flag、singificant _ coff _ flag、last _ significant _ coff _ flag 等
type ㈣	残差的 Level 信息。主要包括的语法元素有 coff _ abs _ level _ minus1、coff _ sign _ flag

对于实时任务来讲, 衡量负载均衡性最重要的标准之一就是处理时间。本文统计了在不同的 QP 下三类语法元素所占用的平均编码时间之比, 如图 1 所示。其中平均编码时间指不同序列的编码时间的平均值。从图中可以看出, 随着 QP 的增加, Type ㈦ 和 Type ㈣的相对编码时间在下降, 二者下降的趋势基本保持一致。这是因为 Type ㈦和Type ㈣中的语法元素描述的对象主要与变换量化残差相关, 随着 QP 的变大, 有效的残差数据会变少, 因此产生的二进制符号也就会逐步下降, 以致编码时间也随之降低。同时可以注意到, 在 QP 为 36~ 38 时, 三类符号的编码时间负载之比达到比较均衡的状态, 并且在常用的编码量化参数范围内, 三类符号的编码时间之比最大不超过 1.3∶1。因此本文提出的任务划分方法基本满足任务的负载均衡性。进一步的均衡会破坏 H.264 本身的码流结果特征, 带来数据传递的额外开销。

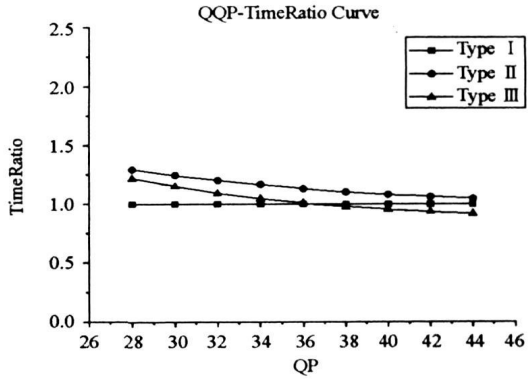

图 1 三类语法元素的平均编码时间之比随 QP 的变化趋势

Fig. 1 Average coding time vs. QP

3.2 熵解码器结构

并行 CABAC 编码结构会产生三条不同的编码码流, 因此 H.264 的标准解码器是无法对其进行正确解码的。由于每个并行的编码过程都遵循 CABAC 过程, 因此对标准的串行解码器略作改动就能够串行解码并行码流。

首先, 必须为解码器的解码引擎设置三个不同编码环境, 并且在编码不同的码流的时候进行状态的切换和保存。其次, 为了能够正确地解码, 码流中加入码流长度信息, 因此读取码流的过程需要分析此字段。

4 实验结果和分析

本文把提出的 CABAC 并行方法集成到 JM8.6[10] 编码器, 并对多个 MPEG 标准测试序列进行了测试。每个测试序列编码 120 帧, 帧率为 30fps。同时本文对 JM8.6 的解码器[10] 进行了修改, 验证了解码的正确性。编码参数设置为: 使能 RDO, QP 等于 28、30、32、34、36、38、40, 图像序列类型为 IPPP, 采用 H.264 的主要档次, 参考帧数为 1, MV 搜索范围为 16, 使能所有的帧间搜索块模式, 禁止采用快速运动估计, 开启 Hadamard 变换。

实验运行的操作系统环境为 Windows XP Pro SP2, Visual C++ 6.0(SP6) 编译生成 release 版本可执行文件。硬件平台为 Pentium 4 2.8GHz 双核 CPU, 1.5GB 的 RAM。

4.1 图像率失真结果

表 1 给出了编码质量降低的两个指标 $\Delta APSNR$ 和 ΔABR。$\Delta APSNR$ 和 ΔABR 分别表示平均峰值信噪比差值和平均码率差值, 具体计算方法可以参考文献[3]。其中所有的结果都来自 JM 参考模型的运行日志。

表 2 编码质量衡量

Tab. 2 Coding quality

Sequence(CIF)	$\Delta APSNR$(dB)	ΔABR(%)
coastguard	0.6274	-14.0215
container	-0.8475	18.1708
hall	-0.5168	12.3389
mother-daughter	0.3131	5.3610
highway	0.0677	3.1692
mobile	0.1922	-6.3236
news	-0.7577	-0.4938
paris	-0.3036	1.0811
average	-0.15315	1.9709

从表 2 可以看出并行的编码方法对图像的编码质量带来的损失。其中平均码率的增加约 1.9%，PSNR 下降了 0.15dB。由于采用多个编码引擎，就会产生多个编码码流。为了方便解码，在最后合并生成的码流中还需要添加一些控制位。这些控制位在每个宏块都会出现，对码率有一定的影响。另外，每个编码引擎在每个宏块的结束以及每个 slice 的结束都需要写入结束符，从而三个编码引擎带来的结束符的增加也会导致码率的增加。

序列 coastguard、mobile 在图像质量有所提高的前提下，还节省了码率。序列 coastguard 是背景变化较大而前景变化较小的图片，而 mobile 图像细节繁多，但是运动规律也是背景变化非常大，前景变化相对较小。序列 container 给出了最差的结果，在平均 PSNR 下降 0.8dB 的情况下，码率却有 18%的上升。

综合分析可知，本文提出的并行编码方法不会带来较大的图像编码质量的损失和码率的增加。

4.2 加速比

加速比的比较对象是 JM8.6 模型的 CABAC 编码方法编码所有的语法元素所需要的时间与本文所提出的并行编码方法编码所有的语法元素所需要的时间之比，包括 RDO 过程中的编码。

表 3 给出了不同的 QP 下的平均加速比。其中平均加速比指在同一 QP 下不同序列的加速比的平均值。从表中可以看出，能够得到的序列的加速比都在 1.66 以上，最大可达 1.78。从表 3 中也可以看出，加速比并没有超过 2，这是因为本文的分类方法中，第二类和第三类符号的熵编码过程中，对残差向量的处理工作存在冗余，从而导致了总编码时间的增加，影响了加速比的提升。

表 3 加速比

Tab. 3 Speedup

QP	Speedup
28	1.75799
30	1.69729
32	1.77685
34	1.69361
36	1.68051
38	1.67092
40	1.66166

实验结果表明，本文提出的 CABAC 行编码方法能够在不对编码图像质量和码率产生较大负面影响的情况下，达到最高单系统加速比 1.78，全系统加速比 1.22。

5 小 结

针对 CABAC 不易并行的问题提出了一种基于多线程的 CABAC 并行化方法。该方法根据 H. 264 语法元素的上下文模型分类, 把 CABAC 编码负载均衡划分为三个部分, 利用三个并行的线程资源编码。通过在 JM8. 6 上进行模拟实验, 统计结果表明, 这种划分方法是合理、有效的, 负载的划分基本均衡。在采用 RDO 的情况下, 熵编码系统加速比能够达到 1. 78。

致 谢

感谢国防科技大学计算机学院 QDSP 研究小组的所有成员, 你们的工作给予了本文平台支撑, 你们对本文提出的中肯的意见对本文的行文有极大的帮助, 对本文作者的研究方法和思路也起到了积极的引导作用。

参 考 文 献:

[1] Wiegand T, Sullivan G J, Bjontegaard G, et al. Overview of the H. 264/AVC Video Coding Standard [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2003, 13: 560- 576.

[2] Osorio R R, Bruguera J D. High-throughput Architecture for H. 264/AVC CABAC Compression System [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2006, 16: 1376- 1384.

[3] Bjontegaard G. Calculation of Average PSNR Differences between RD-curves [C]// 13th VCEG- M33 Meeting Austin, TX, 2001.

[4] Tian X H, Le T M, Ho B L, et al. A CABAC Encoder Design of H. 264/AVC with RDO Support [C]// 18th IEEE/IFIP International Workshop on Rapid System Prototyping, 2007: 167- 173.

[5] Guo B, Wang W D, Shen Y L, et al. A High Speed CABAC Algorithm Based on Probability Estimation Update[C]// Fourth International Conference on Image and Graphics, 2007: 195- 199.

[6] Marpe D, Marten G, Cycon H L. A Fast Renormalization Technique for H. 264/MPEG4-AVC Arithmetic Coding[C]// 51st International Scientific Colloquium (IWK), Ilmenau University of Technology, 2006.

[7] Lin J H, Parhi K K. Parallelization of Context-based Adaptive Binary Arithmetic Coders [J]. IEEE Transactions on Signal Processing, 2006, 54: 3702- 3711.

[8] Agarwal A. Tiled Multicore Processors: The Four Stages of Reality [R]. MIT and Tilera, 2008.

[9] Wang D, Chen X, Chen S, et al. FCC- SDP: A Fast Close-coupled Shared Data Pool for Multi-core DSPs[C]// Advances in Computer Systems Architecture, 2007: 80- 89.

[10] Joint Model (JM) —H. 264/AVC Reference Software [EB/OL]. Http://www. iphome. hhi. de/suehring/tml/download/, 2008.